Review of Probability Theory

Arian Maleki and Tom Do
Stanford University

Probability theory is the study of uncertainty. Through this class, we will be relying on concepts from probability theory for deriving machine learning algorithms. These notes attempt to cover the basics of probability theory at a level appropriate for CS 229. The mathematical theory of probability is very sophisticated, and delves into a branch of analysis known as **measure theory**. In these notes, we provide a basic treatment of probability that does not address these finer details.

1 Elements of probability

In order to define a probability on a set we need a few basic elements,

- **Sample space** Ω: The set of all the outcomes of a random experiment. Here, each outcome $\omega \in \Omega$ can be thought of as a complete description of the state of the real world at the end of the experiment.
- **Set of events** (or **event space**) $\mathcal{F}$: A set whose elements $A \in \mathcal{F}$ (called **events**) are subsets of Ω (i.e., $A \subseteq \Omega$ is a collection of possible outcomes of an experiment).[1].
- **Probability measure**: A function $P : \mathcal{F} \to \mathbb{R}$ that satisfies the following properties,
  - $P(A) \geq 0$, for all $A \in \mathcal{F}$
  - $P(\Omega) = 1$
  - If $A_1, A_2, \ldots$ are disjoint events (i.e., $A_i \cap A_j = \emptyset$ whenever $i \neq j$), then
$$P(\cup_i A_i) = \sum_i P(A_i)$$

These three properties are called the **Axioms of Probability**.

Example: Consider the event of tossing a six-sided die. The sample space is $\Omega = \{1, 2, 3, 4, 5, 6\}$. We can define different event spaces on this sample space. For example, the simplest event space is the trivial event space $\mathcal{F} = \{\emptyset, \Omega\}$. Another event space is the set of all subsets of Ω. For the first event space, the unique probability measure satisfying the requirements above is given by $P(\emptyset) = 0, P(\Omega) = 1$. For the second event space, one valid probability measure is to assign the probability of each set in the event space to be $\frac{i}{6}$ where i is the number of elements of that set; for example, $P(\{1, 2, 3, 4\}) = \frac{4}{6}$ and $P(\{1, 2, 3\}) = \frac{3}{6}$.

Properties:

- If $A \subseteq B \Longrightarrow P(A) \leq P(B)$.
- $P(A \cap B) \leq \min(P(A), P(B))$.
- (Union Bound) $P(A \cup B) \leq P(A) + P(B)$.
- $P(\Omega \setminus A) = 1 - P(A)$.
- (Law of Total Probability) If $A_1, \ldots, A_k$ are a set of disjoint events such that $\cup_{i=1}^k A_i = \Omega$, then $\sum_{i=1}^k P(A_k) = 1$.

[1] $\mathcal{F}$ should satisfy three properties: (1) $\emptyset \in \mathcal{F}$; (2) $A \in \mathcal{F} \Longrightarrow \Omega \setminus A \in \mathcal{F}$; and (3) $A_1, A_2, \ldots \in \mathcal{F} \Longrightarrow \cup_i A_i \in \mathcal{F}$.

1.1 Conditional probability and independence

Let B be an event with non-zero probability. The conditional probability of any event A given B is defined as,

$$P(A|B) \triangleq \frac{P(A \cap B)}{P(B)}$$

In other words, $P(A|B)$ is the probability measure of the event A after observing the occurrence of event B. Two events are called independent if and only if $P(A \cap B) = P(A)P(B)$ (or equivalently, $P(A|B) = P(A)$). Therefore, independence is equivalent to saying that observing B does not have any effect on the probability of A.

2 Random variables

Consider an experiment in which we flip 10 coins, and we want to know the number of coins that come up heads. Here, the elements of the sample space Ω are 10-length sequences of heads and tails. For example, we might have $w_0 = \langle H, H, T, H, T, H, H, T, T, T\rangle \in \Omega$. However, in practice, we usually do not care about the probability of obtaining any particular sequence of heads and tails. Instead we usually care about real-valued functions of outcomes, such as the number of heads that appear among our 10 tosses, or the length of the longest run of tails. These functions, under some technical conditions, are known as **random variables**.

More formally, a random variable X is a function $X : \Omega \longrightarrow \mathbb{R}$.[2] Typically, we will denote random variables using upper case letters $X(\omega)$ or more simply X (where the dependence on the random outcome ω is implied). We will denote the value that a random variable may take on using lower case letters x.

Example: In our experiment above, suppose that $X(\omega)$ is the number of heads which occur in the sequence of tosses ω. Given that only 10 coins are tossed, $X(\omega)$ can take only a finite number of values, so it is known as a **discrete random variable**. Here, the probability of the set associated with a random variable X taking on some specific value k is

$$P(X = k) := P(\{\omega : X(\omega) = k\}).$$

Example: Suppose that $X(\omega)$ is a random variable indicating the amount of time it takes for a radioactive particle to decay. In this case, $X(\omega)$ takes on a infinite number of possible values, so it is called a **continuous random variable**. We denote the probability that X takes on a value between two real constants a and b (where $a < b$) as

$$P(a \leq X \leq b) := P(\{\omega : a \leq X(\omega) \leq b\}).$$

2.1 Cumulative distribution functions

In order to specify the probability measures used when dealing with random variables, it is often convenient to specify alternative functions (CDFs, PDFs, and PMFs) from which the probability measure governing an experiment immediately follows. In this section and the next two sections, we describe each of these types of functions in turn.

A **cumulative distribution function (CDF)** is a function $F_X : \mathbb{R} \rightarrow [0, 1]$ which specifies a probability measure as,

$$F_X(x) \triangleq P(X \leq x). \tag{1}$$

By using this function one can calculate the probability of any event in $\mathcal{F}$.[3] Figure **??** shows a sample CDF function.

Properties:

[2] Technically speaking, not every function is not acceptable as a random variable. From a measure-theoretic perspective, random variables must be Borel-measurable functions. Intuitively, this restriction ensures that given a random variable and its underlying outcome space, one can implicitly define the each of the events of the event space as being sets of outcomes $\omega \in \Omega$ for which $X(\omega)$ satisfies some property (e.g., the event $\{\omega : X(\omega) \geq 3\}$).

[3] This is a remarkable fact and is actually a theorem that is proved in more advanced courses.

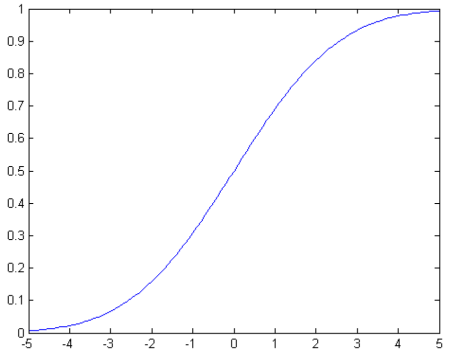


Figure 1: A cumulative distribution function (CDF).

- $0 \leq F_X(x) \leq 1$.
- $\lim_{x\to-\infty} F_X(x) = 0$.
- $\lim_{x\to\infty} F_X(x) = 1$.
- $x \leq y \Longrightarrow F_X(x) \leq F_X(y)$.

2.2 Probability mass functions

When a random variable X takes on a finite set of possible values (i.e., X is a discrete random variable), a simpler way to represent the probability measure associated with a random variable is to directly specify the probability of each value that the random variable can assume. In particular, a *probability mass function (PMF)* is a function $p_X : \Omega \to \mathbb{R}$ such that

$$p_X(x) \triangleq P(X = x).$$

In the case of discrete random variable, we use the notation $Val(X)$ for the set of possible values that the random variable X may assume. For example, if $X(\omega)$ is a random variable indicating the number of heads out of ten tosses of coin, then $Val(X) = \{0, 1, 2, \ldots, 10\}$.

Properties:

- $0 \leq p_X(x) \leq 1$.
- $\sum_{x \in Val(X)} p_X(x) = 1$.
- $\sum_{x \in A} p_X(x) = P(X \in A)$.

2.3 Probability density functions

For some continuous random variables, the cumulative distribution function $F_X(x)$ is differentiable everywhere. In these cases, we define the **Probability Density Function** or **PDF** as the derivative of the CDF, i.e.,

$$f_X(x) \triangleq \frac{dF_X(x)}{dx}. \quad (2)$$

Note here, that the PDF for a continuous random variable may not always exist (i.e., if $F_X(x)$ is not differentiable everywhere).

According to the properties of differentiation, for very small Δx,

$$P(x \leq X \leq x + \Delta x) \approx f_X(x)\Delta x. \quad (3)$$

Both CDFs and PDFs (when they exist!) can be used for calculating the probabilities of different events. But it should be emphasized that the value of PDF at any given point x is not the probability of that event, i.e., $f_X(x) \neq P(X = x)$. For example, $f_X(x)$ can take on values larger than one (but the integral of $f_X(x)$ over any subset of $\mathbb{R}$ will be at most one).

Properties:

- $f_X(x) \geq 0$.
- $\int_{-\infty}^{\infty} f_X(x) = 1$.
- $\int_{x \in A} f_X(x)dx = P(X \in A)$.

2.4 Expectation

Suppose that X is a discrete random variable with PMF $p_X(x)$ and $g : \mathbb{R} \longrightarrow \mathbb{R}$ is an arbitrary function. In this case, $g(X)$ can be considered a random variable, and we define the **expectation** or **expected value** of $g(X)$ as

$$E[g(X)] \triangleq \sum_{x \in Val(X)} g(x)p_X(x).$$

If X is a continuous random variable with PDF $f_X(x)$, then the expected value of $g(X)$ is defined as,

$$E[g(X)] \triangleq \int_{-\infty}^{\infty} g(x)f_X(x)dx.$$

Intuitively, the expectation of $g(X)$ can be thought of as a "weighted average" of the values that $g(x)$ can taken on for different values of x, where the weights are given by $p_X(x)$ or $f_X(x)$. As a special case of the above, note that the expectation, $E[X]$ of a random variable itself is found by letting $g(x) = x$; this is also known as the **mean** of the random variable X.

Properties:

- $E[a] = a$ for any constant $a \in \mathbb{R}$.
- $E[af(X)] = aE[f(X)]$ for any constant $a \in \mathbb{R}$.
- (Linearity of Expectation) $E[f(X) + g(X)] = E[f(X)] + E[g(X)]$.
- For a discrete random variable X, $E[1\{X = k\}] = P(X = k)$.

2.5 Variance

The **variance** of a random variable X is a measure of how concentrated the distribution of a random variable X is around its mean. Formally, the variance of a random variable X is defined as

$$Var[X] \triangleq E[(X - E(X))^2]$$

Using the properties in the previous section, we can derive an alternate expression for the variance:

$$\begin{aligned} E[(X - E[X])^2] &= E[X^2 - 2E[X]X + E[X]^2] \\ &= E[X^2] - 2E[X]E[X] + E[X]^2 \\ &= E[X^2] - E[X]^2, \end{aligned}$$

where the second equality follows from linearity of expectations and the fact that $E[X]$ is actually a constant with respect to the outer expectation.

Properties:

- $Var[a] = 0$ for any constant $a \in \mathbb{R}$.
- $Var[af(X)] = a^2 Var[f(X)]$ for any constant $a \in \mathbb{R}$.

Example Calculate the mean and the variance of the uniform random variable X with PDF $f_X(x) = 1, \ \forall x \in [0, 1]$, 0 elsewhere.

$$E[X] = \int_{-\infty}^{\infty} x f_X(x)dx = \int_0^1 x dx = \frac{1}{2}.$$

$$E[X^2] = \int_{-\infty}^{\infty} x^2 f_X(x)dx = \int_0^1 x^2 dx = \frac{1}{3}.$$

$$Var[X] = E[X^2] - E[X]^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}.$$

Example: Suppose that $g(x) = 1\{x \in A\}$ for some subset $A \subseteq \Omega$. What is $E[g(X)]$?

Discrete case:

$$E[g(X)] = \sum_{x \in Val(X)} 1\{x \in A\} P_X(x)dx = \sum_{x \in A} P_X(x)dx = P(x \in A).$$

Continuous case:

$$E[g(X)] = \int_{-\infty}^{\infty} 1\{x \in A\} f_X(x)dx = \int_{x \in A} f_X(x)dx = P(x \in A).$$

2.6 Some common random variables

Discrete random variables

- $X \sim Bernoulli(p)$ (where $0 \leq p \leq 1$): one if a coin with heads probability p comes up heads, zero otherwise.

$$p(x) = \begin{cases} p & \text{if } p = 1 \\ 1 - p & \text{if } p = 0 \end{cases}$$

- $X \sim Binomial(n, p)$ (where $0 \leq p \leq 1$): the number of heads in n independent flips of a coin with heads probability p.

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

- $X \sim Geometric(p)$ (where $p > 0$): the number of flips of a coin with heads probability p until the first heads.

$$p(x) = p(1-p)^{x-1}$$

- $X \sim Poisson(\lambda)$ (where $\lambda > 0$): a probability distribution over the nonnegative integers used for modeling the frequency of rare events.

$$p(x) = e^{-\lambda} \frac{\lambda^x}{x!}$$

Continuous random variables

- $X \sim Uniform(a, b)$ (where $a < b$): equal probability density to every value between a and b on the real line.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

- $X \sim Exponential(\lambda)$ (where $\lambda > 0$): decaying probability density over the nonnegative reals.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- $X \sim Normal(\mu, \sigma^2)$: also known as the Gaussian distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

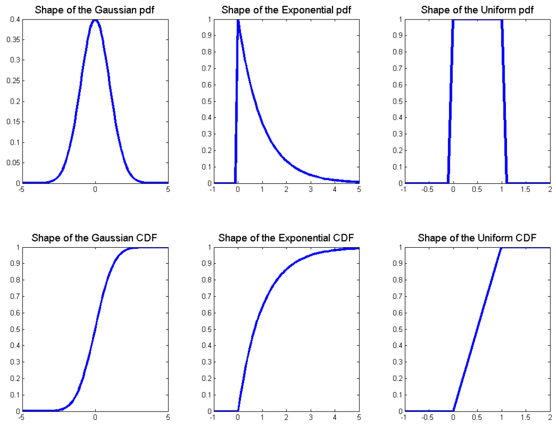


Figure 2: PDF and CDF of a couple of random variables.

The shape of the PDFs and CDFs of some of these random variables are shown in Figure **??**.

The following table is the summary of some of the properties of these distributions.

Distribution	PDF or PMF	Mean	Variance
$Bernoulli(p)$	$\begin{cases} p, & \text{if } x = 1 \\ 1-p, & \text{if } x = 0. \end{cases}$	p	$p(1-p)$
$Binomial(n, p)$	$\binom{n}{k} p^k (1-p)^{n-k}$ for $0 \leq k \leq n$	np	npq
$Geometric(p)$	$p(1-p)^{k-1}$ for $k = 1, 2, \ldots$	$\frac{1}{p}$	$\frac{1-p}{p^2}$
$Poisson(\lambda)$	$e^{-\lambda}\lambda^x/x!$ for $k = 1, 2, \ldots$	λ	λ
$Uniform(a, b)$	$\frac{1}{b-a}$ $\forall x \in (a, b)$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
$Gaussian(\mu, \sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
$Exponential(\lambda)$	$\lambda e^{-\lambda x}$ $x \geq 0, \lambda > 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$

3 Two random variables

Thus far, we have considered single random variables. In many situations, however, there may be more than one quantity that we are interested in knowing during a random experiment. For instance, in an experiment where we flip a coin ten times, we may care about both $X(\omega) = \text{the number of heads that come up}$ as well as $Y(\omega) = \text{the length of the longest run of consecutive heads}$. In this section, we consider the setting of two random variables.

3.1 Joint and marginal distributions

Suppose that we have two random variables X and Y. One way to work with these two random variables is to consider each of them separately. If we do that we will only need $F_X(x)$ and $F_Y(y)$. But if we want to know about the values that X and Y assume simultaneously during outcomes of a random experiment, we require a more complicated structure known as the **joint cumulative distribution function** of X and Y, defined by

$$F_{XY}(x, y) = P(X \leq x, Y \leq y)$$

It can be shown that by knowing the joint cumulative distribution function, the probability of any event involving X and Y can be calculated.

The joint CDF $F_{XY}(x,y)$ and the joint distribution functions $F_X(x)$ and $F_Y(y)$ of each variable separately are related by

$$\begin{aligned} F_X(x) &= \lim_{y\to\infty} F_{XY}(x,y)dy \\ F_Y(y) &= \lim_{x\to\infty} F_{XY}(x,y)dx. \end{aligned}$$

Here, we call $F_X(x)$ and $F_Y(y)$ the **marginal cumulative distribution functions** of $F_{XY}(x,y)$.

Properties:

- $0 \leq F_{XY}(x,y) \leq 1$.
- $\lim_{x,y\to\infty} F_{XY}(x,y) = 1$.
- $\lim_{x,y\to-\infty} F_{XY}(x,y) = 0$.
- $F_X(x) = \lim_{y\to\infty} F_{XY}(x,y)$.

3.2 Joint and marginal probability mass functions

If X and Y are discrete random variables, then the **joint probability mass function** $p_{XY} : \mathbb{R}\times\mathbb{R} \to [0,1]$ is defined by

$$p_{XY}(x,y) = P(X = x, Y = y).$$

Here, $0 \leq P_{XY}(x,y) \leq 1$ for all x, y, and $\sum_{x\in Val(X)} \sum_{y\in Val(Y)} P_{XY}(x,y) = 1$.

How does the joint PMF over two variables relate to the probability mass function for each variable separately? It turns out that

$$p_X(x) = \sum_y p_{XY}(x,y).$$

and similarly for $p_Y(y)$. In this case, we refer to $p_X(x)$ as the **marginal probability mass function** of X. In statistics, the process of forming the marginal distribution with respect to one variable by summing out the other variable is often known as “marginalization.”

3.3 Joint and marginal probability density functions

Let X and Y be two continuous random variables with joint distribution function F_{XY}. In the case that $F_{XY}(x,y)$ is everywhere differentiable in both x and y, then we can define the **joint probability density function**,

$$f_{XY}(x,y) = \frac{\partial^2 F_{XY}(x,y)}{\partial x \partial y}.$$

Like in the single-dimensional case, $f_{XY}(x,y) \neq P(X = x, Y = y)$, but rather

$$\iint_{x\in A} f_{XY}(x,y)dxdy = P((X,Y) \in A).$$

Note that the values of the probability density function $f_{XY}(x,y)$ are always nonnegative, but they may be greater than 1. Nonetheless, it must be the case that $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{XY}(x,y) = 1$.

Analagous to the discrete case, we define

$$f_X(x) = \int_{-\infty}^{\infty} f_{XY}(x,y)dy,$$

as the **marginal probability density function** (or **marginal density**) of X, and similarly for $f_Y(y)$.

3.4 Conditional distributions

Conditional distributions seek to answer the question, what is the probability distribution over Y, when we know that X must take on a certain value x? In the discrete case, the conditional probability mass function of X given Y is simply

$$p_{Y|X}(y|x) = \frac{p_{XY}(x,y)}{p_X(x)},$$

assuming that $p_X(x) \neq 0$.

In the continuous case, the situation is technically a little more complicated because the probability that a continuous random variable X takes on a specific value x is equal to zero[4]. Ignoring this technical point, we simply define, by analogy to the discrete case, the *conditional probability density* of Y given $X = x$ to be

$$f_{Y|X}(y|x) = \frac{f_{XY}(x,y)}{f_X(x)},$$

provided $f_X(x) \neq 0$.

3.5 Bayes's rule

A useful formula that often arises when trying to derive expression for the conditional probability of one variable given another, is **Bayes's rule**.

In the case of discrete random variables X and Y,

$$P_{Y|X}(y|x) = \frac{P_{XY}(x,y)}{P_X(x)} = \frac{P_{X|Y}(x|y)P_Y(y)}{\sum_{y' \in Val(Y)} P_{X|Y}(x|y')P_Y(y')}.$$

If the random variables X and Y are continuous,

$$f_{Y|X}(y|x) = \frac{f_{XY}(x,y)}{f_X(x)} = \frac{f_{X|Y}(x|y)f_Y(y)}{\int_{-\infty}^{\infty} f_{X|Y}(x|y')f_Y(y')dy'}.$$

3.6 Independence

Two random variables X and Y are **independent** if $F_{XY}(x,y) = F_X(x)F_Y(y)$ for all values of x and y. Equivalently,

- For discrete random variables, $p_{XY}(x,y) = p_X(x)p_Y(y)$ for all $x \in Val(X)$, $y \in Val(Y)$.
- For discrete random variables, $p_{Y|X}(y|x) = p_Y(y)$ whenever $p_X(x) \neq 0$ for all $y \in Val(Y)$.
- For continuous random variables, $f_{XY}(x,y) = f_X(x)f_Y(y)$ for all $x, y \in \mathbb{R}$.
- For continuous random variables, $f_{Y|X}(y|x) = f_Y(y)$ whenever $f_X(x) \neq 0$ for all $y \in \mathbb{R}$.

[4] To get around this, a more reasonable way to calculate the conditional CDF is,

$$F_{Y|X}(y,x) = \lim_{\Delta x \to 0} P(Y \leq y | x \leq X \leq x + \Delta x).$$

It can be easily seen that if $F(x,y)$ is differentiable in both x, y then,

$$F_{Y|X}(y,x) = \int_{-\infty}^{y} \frac{f_{X,Y}(x,\alpha)}{f_X(x)} d\alpha$$

and therefore we define the conditional PDF of Y given $X = x$ in the following way,

$$f_{Y|X}(y|x) = \frac{f_{XY}(x,y)}{f_X(x)}$$

Informally, two random variables X and Y are **independent** if "knowing" the value of one variable will never have any effect on the conditional probability distribution of the other variable, that is, you know all the information about the pair (X, Y) by just knowing $f(x)$ and $f(y)$. The following lemma formalizes this observation:

Lemma 3.1. *If X and Y are independent then for any subsets $A, B \subseteq \mathbb{R}$, we have,*

$$P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$$

By using the above lemma one can prove that if X is independent of Y then any function of X is independent of any function of Y.

3.7 Expectation and covariance

Suppose that we have two discrete random variables X, Y and $g : \mathbf{R}^2 \longrightarrow \mathbf{R}$ is a function of these two random variables. Then the expected value of g is defined in the following way,

$$E[g(X,Y)] \triangleq \sum_{x \in Val(X)} \sum_{y \in Val(Y)} g(x,y) p_{XY}(x,y).$$

For continuous random variables X, Y, the analogous expression is

$$E[g(X,Y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x,y) f_{XY}(x,y) dx dy.$$

We can use the concept of expectation to study the relationship of two random variables with each other. In particular, the **covariance** of two random variables X and Y is defined as

$$Cov[X,Y] \quad \triangleq \quad E[(X - E[X])(Y - E[Y])]$$

Using an argument similar to that for variance, we can rewrite this as,

$$\begin{aligned} Cov[X,Y] &= E[(X - E[X])(Y - E[Y])] \\ &= E[XY - XE[Y] - YE[X] + E[X]E[Y]] \\ &= E[XY] - E[X]E[Y] - E[Y]E[X] + E[X]E[Y]] \\ &= E[XY] - E[X]E[Y]. \end{aligned}$$

Here, the key step in showing the equality of the two forms of covariance is in the third equality, where we use the fact that $E[X]$ and $E[Y]$ are actually constants which can be pulled out of the expectation. When $Cov[X,Y] = 0$, we say that X and Y are **uncorrelated**[5].

Properties:

- (Linearity of expectation) $E[f(X,Y) + g(X,Y)] = E[f(X,Y)] + E[g(X,Y)]$.
- $Var[X + Y] = Var[X] + Var[Y] + 2Cov[X,Y]$.
- If X and Y are independent, then $Cov[X,Y] = 0$.
- If X and Y are independent, then $E[f(X)g(Y)] = E[f(X)]E[g(Y)]$.

4 Multiple random variables

The notions and ideas introduced in the previous section can be generalized to more than two random variables. In particular, suppose that we have n continuous random variables, $X_1(\omega), X_2(\omega), \ldots X_n(\omega)$. In this section, for simplicity of presentation, we focus only on the continuous case, but the generalization to discrete random variables works similarly.

[5] However, this is not the same thing as stating that X and Y are independent! For example, if $X \sim Uniform(-1, 1)$ and $Y = X^2$, then one can show that X and Y are uncorrelated, even though they are not independent.

4.1 Basic properties

We can define the **joint distribution function** of $X_1, X_2, \dots, X_n$, the **joint probability density function** of $X_1, X_2, \dots, X_n$, the **marginal probability density function** of X_1, and the **conditional probability density function** of X_1 given $X_2, \dots, X_n$, as

$$\begin{aligned}
F_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n) &= P(X_1 \leq x_1, X_2 \leq x_2, \dots, X_n \leq x_n) \\
f_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n) &= \frac{\partial^n F_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n)}{\partial x_1 \dots \partial x_n} \\
f_{X_1}(X_1) &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n) dx_2 \dots dx_n \\
f_{X_1|X_2,\dots,X_n}(x_1|x_2, \dots x_n) &= \frac{f_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n)}{f_{X_2,\dots,X_n}(x_1, x_2, \dots x_n)}
\end{aligned}$$

To calculate the probability of an event $A \subseteq \mathbb{R}^n$ we have,

$$P((x_1, x_2, \dots x_n) \in A) = \int_{(x_1, x_2, \dots x_n) \in A} f_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n) dx_1 dx_2 \dots dx_n \qquad (4)$$

Chain rule: From the definition of conditional probabilities for multiple random variables, one can show that

$$\begin{aligned}
f(x_1, x_2, \dots, x_n) &= f(x_n|x_1, x_2 \dots, x_{n-1}) f(x_1, x_2 \dots, x_{n-1}) \\
&= f(x_n|x_1, x_2 \dots, x_{n-1}) f(x_{n-1}|x_1, x_2 \dots, x_{n-2}) f(x_1, x_2 \dots, x_{n-2}) \\
&= \dots = f(x_1) \prod_{i=2}^{n} f(x_i|x_1, \dots, x_{i-1}).
\end{aligned}$$

Independence: For multiple events, $A_1, \dots, A_k$, we say that $A_1, \dots, A_k$ are **mutually independent** if for any subset $S \subseteq \{1, 2, \dots, k\}$, we have

$$P(\cap_{i \in S} A_i) = \prod_{i \in S} P(A_i).$$

Likewise, we say that random variables $X_1, \dots, X_n$ are independent if

$$f(x_1, \dots, x_n) = f(x_1) f(x_2) \cdots f(x_n).$$

Here, the definition of mutual independence is simply the natural generalization of independence of two random variables to multiple random variables.

Independent random variables arise often in machine learning algorithms where we assume that the training examples belonging to the training set represent independent samples from some unknown probability distribution. To make the significance of independence clear, consider a "bad" training set in which we first sample a single training example $(x^{(1)}, y^{(1)})$ from the some unknown distribution, and then add $m - 1$ copies of the exact same training example to the training set. In this case, we have (with some abuse of notation)

$$P((x^{(1)}, y^{(1)}), \dots (x^{(m)}, y^{(m)})) \neq \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}).$$

Despite the fact that the training set has size m, the examples are not independent! While clearly the procedure described here is not a sensible method for building a training set for a machine learning algorithm, it turns out that in practice, non-independence of samples does come up often, and it has the effect of reducing the "effective size" of the training set.

4.2 Random vectors

Suppose that we have n random variables. When working with all these random variables together, we will often find it convenient to put them in a vector $X = [X_1\ X_2\ \dots\ X_n]^T$. We call the resulting vector a **random vector** (more formally, a random vector is a mapping from Ω to $\mathbb{R}^n$). It should be clear that random vectors are simply an alternative notation for dealing with n random variables, so the notions of joint PDF and CDF will apply to random vectors as well.

Expectation: Consider an arbitrary function from $g : \mathbb{R}^n \to \mathbb{R}$. The expected value of this function is defined as

$$E[g(X)] = \int_{\mathbb{R}^n} g(x_1, x_2, \dots, x_n) f_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots x_n) dx_1 dx_2 \dots dx_n, \tag{5}$$

where $\int_{\mathbb{R}^n}$ is n consecutive integrations from $-\infty$ to ∞. If g is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$, then the expected value of g is the element-wise expected values of the output vector, i.e., if g is

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{bmatrix},$$

Then,

$$E[g(X)] = \begin{bmatrix} E[g_1(X)] \\ E[g_2(X)] \\ \vdots \\ E[g_m(X)] \end{bmatrix}.$$

Covariance matrix: For a given random vector $X : \Omega \to \mathbb{R}^n$, its covariance matrix Σ is the $n \times n$ square matrix whose entries are given by $\Sigma_{ij} = Cov[X_i, X_j]$.

From the definition of covariance, we have

$$\begin{aligned}
\Sigma &= \begin{bmatrix} Cov[X_1, X_1] & \cdots & Cov[X_1, X_n] \\ \vdots & \ddots & \vdots \\ Cov[X_n, X_1] & \cdots & Cov[X_n, X_n] \end{bmatrix} \\
&= \begin{bmatrix} E[X_1^2] - E[X_1]E[X_1] & \cdots & E[X_1 X_n] - E[X_1]E[X_n] \\ \vdots & \ddots & \vdots \\ E[X_n X_1] - E[X_n]E[X_1] & \cdots & E[X_n^2] - E[X_n]E[X_n] \end{bmatrix} \\
&= \begin{bmatrix} E[X_1^2] & \cdots & E[X_1 X_n] \\ \vdots & \ddots & \vdots \\ E[X_n X_1] & \cdots & E[X_n^2] \end{bmatrix} - \begin{bmatrix} E[X_1]E[X_1] & \cdots & E[X_1]E[X_n] \\ \vdots & \ddots & \vdots \\ E[X_n]E[X_1] & \cdots & E[X_n]E[X_n] \end{bmatrix} \\
&= E[XX^T] - E[X]E[X]^T = \dots = E[(X - E[X])(X - E[X])^T].
\end{aligned}$$

where the matrix expectation is defined in the obvious way.

The covariance matrix has a number of useful properties:

- $\Sigma \succeq 0$; that is, Σ is positive semidefinite.
- $\Sigma = \Sigma^T$; that is, Σ is symmetric.

4.3 The multivariate Gaussian distribution

One particularly important example of a probability distribution over random vectors X is called the **multivariate Gaussian** or **multivariate normal** distribution. A random vector $X \in \mathbb{R}^n$ is said to have a multivariate normal (or Gaussian) distribution with mean $\mu \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{S}^n_{++}$ (where $\mathbb{S}^n_{++}$ refers to the space of symmetric positive definite $n \times n$ matrices)

$$f_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots, x_n; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right).$$

We write this as $X \sim \mathcal{N}(\mu, \Sigma)$. Notice that in the case $n = 1$, this reduces the regular definition of a normal distribution with mean parameter μ_1 and variance Σ_{11}.

Generally speaking, Gaussian random variables are extremely useful in machine learning and statistics for two main reasons. First, they are extremely common when modeling "noise" in statistical algorithms. Quite often, noise can be considered to be the accumulation of a large number of small independent random perturbations affecting the measurement process; by the Central Limit Theorem, summations of independent random variables will tend to "look Gaussian." Second, Gaussian random variables are convenient for many analytical manipulations, because many of the integrals involving Gaussian distributions that arise in practice have simple closed form solutions. We will encounter this later in the course.

5 Other resources

A good textbook on probablity at the level needed for CS229 is the book, *A First Course on Probability* by Sheldon Ross.

Linear Algebra Review and Reference

Zico Kolter (updated by Chuong Do)

September 30, 2015

Contents

1 Basic Concepts and Notation

Linear algebra provides a way of compactly representing and operating on sets of linear equations. For example, consider the following system of equations:

$$\begin{array}{rcrcr} 4x_1 & - & 5x_2 & = & -13 \\ -2x_1 & + & 3x_2 & = & 9. \end{array}$$

This is two equations and two variables, so as you know from high school algebra, you can find a unique solution for x_1 and x_2 (unless the equations are somehow degenerate, for example if the second equation is simply a multiple of the first, but in the case above there is in fact a unique solution). In matrix notation, we can write the system more compactly as

$$Ax = b$$

with

$$A = \left[\begin{array}{cc} 4 & -5 \\ -2 & 3 \end{array} \right], \quad b = \left[\begin{array}{c} -13 \\ 9 \end{array} \right].$$

As we will see shortly, there are many advantages (including the obvious space savings) to analyzing linear equations in this form.

1.1 Basic Notation

We use the following notation:

- By $A \in \mathbb{R}^{m \times n}$ we denote a matrix with m rows and n columns, where the entries of A are real numbers.
- By $x \in \mathbb{R}^n$, we denote a vector with n entries. By convention, an n-dimensional vector is often thought of as a matrix with n rows and 1 column, known as a ***column vector***. If we want to explicitly represent a ***row vector*** — a matrix with 1 row and n columns — we typically write x^T (here x^T denotes the transpose of x, which we will define shortly).
- The ith element of a vector x is denoted x_i:

$$x = \left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \right].$$

- We use the notation a_{ij} (or A_{ij}, $A_{i,j}$, etc) to denote the entry of A in the ith row and jth column:
$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

- We denote the jth column of A by a_j or $A_{:,j}$:
$$A = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}.$$

- We denote the ith row of A by a_i^T or $A_{i,:}$:
$$A = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix}.$$

- Note that these definitions are ambiguous (for example, the a_1 and a_1^T in the previous two definitions are *not* the same vector). Usually the meaning of the notation should be obvious from its use.

2 Matrix Multiplication

The product of two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ is the matrix

$$C = AB \in \mathbb{R}^{m \times p},$$

where

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}.$$

Note that in order for the matrix product to exist, the number of columns in A must equal the number of rows in B. There are many ways of looking at matrix multiplication, and we'll start by examining a few special cases.

2.1 Vector-Vector Products

Given two vectors $x, y \in \mathbb{R}^n$, the quantity $x^T y$, sometimes called the ***inner product*** or ***dot product*** of the vectors, is a real number given by

$$x^T y \in \mathbb{R} = \left[\begin{array}{cccc} x_1 & x_2 & \cdots & x_n \end{array}\right] \left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \end{array}\right] = \sum_{i=1}^{n} x_i y_i .$$

Observe that inner products are really just special case of matrix multiplication. Note that it is always the case that $x^T y = y^T x$.

Given vectors $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ (not necessarily of the same size), $xy^T \in \mathbb{R}^{m\times n}$ is called the ***outer product*** of the vectors. It is a matrix whose entries are given by $(xy^T)_{ij} = x_i y_j$, i.e.,

$$xy^T \in \mathbb{R}^{m\times n} = \left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_m \end{array}\right] \left[\begin{array}{cccc} y_1 & y_2 & \cdots & y_n \end{array}\right] = \left[\begin{array}{cccc} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{array}\right].$$

As an example of how the outer product can be useful, let $\mathbf{1} \in \mathbb{R}^n$ denote an n-dimensional vector whose entries are all equal to 1. Furthermore, consider the matrix $A \in \mathbb{R}^{m\times n}$ whose columns are all equal to some vector $x \in \mathbb{R}^m$. Using outer products, we can represent A compactly as,

$$A = \left[\begin{array}{cccc} | & | & & | \\ x & x & \cdots & x \\ | & | & & | \end{array}\right] = \left[\begin{array}{cccc} x_1 & x_1 & \cdots & x_1 \\ x_2 & x_2 & \cdots & x_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_m & \cdots & x_m \end{array}\right] = \left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_m \end{array}\right] \left[\begin{array}{cccc} 1 & 1 & \cdots & 1 \end{array}\right] = x\mathbf{1}^T.$$

2.2 Matrix-Vector Products

Given a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $x \in \mathbb{R}^n$, their product is a vector $y = Ax \in \mathbb{R}^m$. There are a couple ways of looking at matrix-vector multiplication, and we will look at each of them in turn.

If we write A by rows, then we can express Ax as,

$$y = Ax = \left[\begin{array}{ccc} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{array}\right] x = \left[\begin{array}{c} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{array}\right].$$

In other words, the ith entry of y is equal to the inner product of the ith *row* of A and x, $y_i = a_i^T x$.

Alternatively, let's write A in column form. In this case we see that,

$$y = Ax = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} x_1 + \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} x_2 + \ldots + \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix} x_n \ .$$

In other words, y is a ***linear combination*** of the *columns* of A, where the coefficients of the linear combination are given by the entries of x.

So far we have been multiplying on the right by a column vector, but it is also possible to multiply on the left by a row vector. This is written, $y^T = x^T A$ for $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^m$, and $y \in \mathbb{R}^n$. As before, we can express y^T in two obvious ways, depending on whether we express A in terms on its rows or columns. In the first case we express A in terms of its columns, which gives

$$y^T = x^T A = x^T \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} x^T a_1 & x^T a_2 & \cdots & x^T a_n \end{bmatrix}$$

which demonstrates that the ith entry of y^T is equal to the inner product of x and the ith *column* of A.

Finally, expressing A in terms of rows we get the final representation of the vector-matrix product,

$$\begin{aligned} y^T &= x^T A \\ &= \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} \\ &= x_1 \begin{bmatrix} — & a_1^T & — \end{bmatrix} + x_2 \begin{bmatrix} — & a_2^T & — \end{bmatrix} + ... + x_n \begin{bmatrix} — & a_n^T & — \end{bmatrix} \end{aligned}$$

so we see that y^T is a linear combination of the *rows* of A, where the coefficients for the linear combination are given by the entries of x.

2.3 Matrix-Matrix Products

Armed with this knowledge, we can now look at four different (but, of course, equivalent) ways of viewing the matrix-matrix multiplication $C = AB$ as defined at the beginning of this section.

First, we can view matrix-matrix multiplication as a set of vector-vector products. The most obvious viewpoint, which follows immediately from the definition, is that the (i, j)th entry of C is equal to the inner product of the ith row of A and the jth column of B. Symbolically, this looks like the following,

$$C = AB = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_p \\ a_2^T b_1 & a_2^T b_2 & \cdots & a_2^T b_p \\ \vdots & \vdots & \ddots & \vdots \\ a_m^T b_1 & a_m^T b_2 & \cdots & a_m^T b_p \end{bmatrix}.$$

Remember that since $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, $a_i \in \mathbb{R}^n$ and $b_j \in \mathbb{R}^n$, so these inner products all make sense. This is the most "natural" representation when we represent A by rows and B by columns. Alternatively, we can represent A by columns, and B by rows. This representation leads to a much trickier interpretation of AB as a sum of outer products. Symbolically,

$$C = AB = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & b_1^T & — \\ — & b_2^T & — \\ & \vdots & \\ — & b_n^T & — \end{bmatrix} = \sum_{i=1}^n a_i b_i^T \ .$$

Put another way, AB is equal to the sum, over all i, of the outer product of the ith column of A and the ith row of B. Since, in this case, $a_i \in \mathbb{R}^m$ and $b_i \in \mathbb{R}^p$, the dimension of the outer product $a_i b_i^T$ is $m \times p$, which coincides with the dimension of C. Chances are, the last equality above may appear confusing to you. If so, take the time to check it for yourself!

Second, we can also view matrix-matrix multiplication as a set of matrix-vector products. Specifically, if we represent B by columns, we can view the columns of C as matrix-vector products between A and the columns of B. Symbolically,

$$C = AB = A \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ab_1 & Ab_2 & \cdots & Ab_p \\ | & | & & | \end{bmatrix}.$$

Here the ith column of C is given by the matrix-vector product with the vector on the right, $c_i = Ab_i$. These matrix-vector products can in turn be interpreted using both viewpoints given in the previous subsection. Finally, we have the analogous viewpoint, where we represent A by rows, and view the rows of C as the matrix-vector product between the rows of A and C. Symbolically,

$$C = AB = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} B = \begin{bmatrix} — & a_1^T B & — \\ — & a_2^T B & — \\ & \vdots & \\ — & a_m^T B & — \end{bmatrix}.$$

Here the ith row of C is given by the matrix-vector product with the vector on the left, $c_i^T = a_i^T B$.

It may seem like overkill to dissect matrix multiplication to such a large degree, especially when all these viewpoints follow immediately from the initial definition we gave (in about a line of math) at the beginning of this section. However, virtually all of linear algebra deals with matrix multiplications of some kind, and it is worthwhile to spend some time trying to develop an intuitive understanding of the viewpoints presented here.

In addition to this, it is useful to know a few basic properties of matrix multiplication at a higher level:

- Matrix multiplication is associative: $(AB)C = A(BC)$.
- Matrix multiplication is distributive: $A(B + C) = AB + AC$.
- Matrix multiplication is, in general, *not* commutative; that is, it can be the case that $AB \neq BA$. (For example, if $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$, the matrix product BA does not even exist if m and q are not equal!)

If you are not familiar with these properties, take the time to verify them for yourself. For example, to check the associativity of matrix multiplication, suppose that $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{p\times q}$. Note that $AB \in \mathbb{R}^{m\times p}$, so $(AB)C \in \mathbb{R}^{m\times q}$. Similarly, $BC \in \mathbb{R}^{n\times q}$, so $A(BC) \in \mathbb{R}^{m\times q}$. Thus, the dimensions of the resulting matrices agree. To show that matrix multiplication is associative, it suffices to check that the (i, j)th entry of $(AB)C$ is equal to the (i, j)th entry of $A(BC)$. We can verify this directly using the definition of matrix multiplication:

$$\begin{aligned}
((AB)C)_{ij} &= \sum_{k=1}^{p} (AB)_{ik} C_{kj} = \sum_{k=1}^{p} \left(\sum_{l=1}^{n} A_{il} B_{lk} \right) C_{kj} \\
&= \sum_{k=1}^{p} \left(\sum_{l=1}^{n} A_{il} B_{lk} C_{kj} \right) = \sum_{l=1}^{n} \left(\sum_{k=1}^{p} A_{il} B_{lk} C_{kj} \right) \\
&= \sum_{l=1}^{n} A_{il} \left(\sum_{k=1}^{p} B_{lk} C_{kj} \right) = \sum_{l=1}^{n} A_{il} (BC)_{lj} = (A(BC))_{ij}.
\end{aligned}$$

Here, the first and last two equalities simply use the definition of matrix multiplication, the third and fifth equalities use the distributive property for *scalar multiplication over addition*, and the fourth equality uses the *commutative and associativity of scalar addition*. This technique for proving matrix properties by reduction to simple scalar properties will come up often, so make sure you're familiar with it.

3 Operations and Properties

In this section we present several operations and properties of matrices and vectors. Hopefully a great deal of this will be review for you, so the notes can just serve as a reference for these topics.

3.1 The Identity Matrix and Diagonal Matrices

The ***identity matrix***, denoted $I \in \mathbb{R}^{n\times n}$, is a square matrix with ones on the diagonal and zeros everywhere else. That is,

$$I_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

It has the property that for all $A \in \mathbb{R}^{m\times n}$,

$$AI = A = IA.$$

Note that in some sense, the notation for the identity matrix is ambiguous, since it does not specify the dimension of I. Generally, the dimensions of I are inferred from context so as to make matrix multiplication possible. For example, in the equation above, the I in $AI = A$ is an $n \times n$ matrix, whereas the I in $A = IA$ is an $m \times m$ matrix.

A ***diagonal matrix*** is a matrix where all non-diagonal elements are 0. This is typically denoted $D = \text{diag}(d_1, d_2, \ldots, d_n)$, with

$$D_{ij} = \begin{cases} d_i & i = j \\ 0 & i \neq j \end{cases}$$

Clearly, $I = \text{diag}(1, 1, \ldots, 1)$.

3.2 The Transpose

The ***transpose*** of a matrix results from "flipping" the rows and columns. Given a matrix $A \in \mathbb{R}^{m\times n}$, its transpose, written $A^T \in \mathbb{R}^{n\times m}$, is the $n \times m$ matrix whose entries are given by

$$(A^T)_{ij} = A_{ji}.$$

We have in fact already been using the transpose when describing row vectors, since the transpose of a column vector is naturally a row vector.

The following properties of transposes are easily verified:

- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A + B)^T = A^T + B^T$

3.3 Symmetric Matrices

A square matrix $A \in \mathbb{R}^{n\times n}$ is ***symmetric*** if $A = A^T$. It is ***anti-symmetric*** if $A = -A^T$. It is easy to show that for any matrix $A \in \mathbb{R}^{n\times n}$, the matrix $A + A^T$ is symmetric and the matrix $A - A^T$ is anti-symmetric. From this it follows that any square matrix $A \in \mathbb{R}^{n \times n}$ can be represented as a sum of a symmetric matrix and an anti-symmetric matrix, since

$$A = \frac{1}{2}(A + A^T) + \frac{1}{2}(A - A^T)$$

and the first matrix on the right is symmetric, while the second is anti-symmetric. It turns out that symmetric matrices occur a great deal in practice, and they have many nice properties which we will look at shortly. It is common to denote the set of all symmetric matrices of size n as $\mathbb{S}^n$, so that $A \in \mathbb{S}^n$ means that A is a symmetric $n \times n$ matrix;

3.4 The Trace

The ***trace*** of a square matrix $A \in \mathbb{R}^{n \times n}$, denoted $\mathrm{tr}(A)$ (or just $\mathrm{tr}A$ if the parentheses are obviously implied), is the sum of diagonal elements in the matrix:

$$\mathrm{tr}A = \sum_{i=1}^{n} A_{ii}.$$

As described in the CS229 lecture notes, the trace has the following properties (included here for the sake of completeness):

- For $A \in \mathbb{R}^{n \times n}$, $\mathrm{tr}A = \mathrm{tr}A^T$.
- For $A, B \in \mathbb{R}^{n \times n}$, $\mathrm{tr}(A + B) = \mathrm{tr}A + \mathrm{tr}B$.
- For $A \in \mathbb{R}^{n \times n}$, $t \in \mathbb{R}$, $\mathrm{tr}(tA) = t\ \mathrm{tr}A$.
- For A, B such that AB is square, $\mathrm{tr}AB = \mathrm{tr}BA$.
- For A, B, C such that ABC is square, $\mathrm{tr}ABC = \mathrm{tr}BCA = \mathrm{tr}CAB$, and so on for the product of more matrices.

As an example of how these properties can be proven, we'll consider the fourth property given above. Suppose that $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times m}$ (so that $AB \in \mathbb{R}^{m \times m}$ is a square matrix). Observe that $BA \in \mathbb{R}^{n \times n}$ is also a square matrix, so it makes sense to apply the trace operator to it. To verify that $\mathrm{tr}AB = \mathrm{tr}BA$, note that

$$\begin{aligned}
\mathrm{tr}AB &= \sum_{i=1}^{m} (AB)_{ii} = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij} B_{ji} \right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ji} = \sum_{j=1}^{n} \sum_{i=1}^{m} B_{ji} A_{ij} \\
&= \sum_{j=1}^{n} \left(\sum_{i=1}^{m} B_{ji} A_{ij} \right) = \sum_{j=1}^{n} (BA)_{jj} = \mathrm{tr}BA.
\end{aligned}$$

Here, the first and last two equalities use the definition of the trace operator and matrix multiplication. The fourth equality, where the main work occurs, uses the commutativity of scalar multiplication in order to reverse the order of the terms in each product, and the commutativity and associativity of scalar addition in order to rearrange the order of the summation.

3.5 Norms

A ***norm*** of a vector $\|x\|$ is informally a measure of the "length" of the vector. For example, we have the commonly-used Euclidean or ℓ_2 norm,

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

Note that $\|x\|_2^2 = x^T x$.

More formally, a norm is any function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies 4 properties:

1. For all $x \in \mathbb{R}^n$, $f(x) \geq 0$ (non-negativity).
2. $f(x) = 0$ if and only if $x = 0$ (definiteness).
3. For all $x \in \mathbb{R}^n$, $t \in \mathbb{R}$, $f(tx) = |t| f(x)$ (homogeneity).
4. For all $x, y \in \mathbb{R}^n$, $f(x + y) \leq f(x) + f(y)$ (triangle inequality).

Other examples of norms are the ℓ_1 norm,

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

and the ℓ_∞ norm,

$$\|x\|_\infty = \max_i |x_i|.$$

In fact, all three norms presented so far are examples of the family of ℓ_p norms, which are parameterized by a real number $p \geq 1$, and defined as

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

Norms can also be defined for matrices, such as the Frobenius norm,

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2} = \sqrt{\operatorname{tr}(A^T A)}.$$

Many other norms exist, but they are beyond the scope of this review.

3.6 Linear Independence and Rank

A set of vectors $\{x_1, x_2, \ldots x_n\} \subset \mathbb{R}^m$ is said to be ***(linearly) independent*** if no vector can be represented as a linear combination of the remaining vectors. Conversely, if one vector belonging to the set *can* be represented as a linear combination of the remaining vectors, then the vectors are said to be ***(linearly) dependent***. That is, if

$$x_n = \sum_{i=1}^{n-1} \alpha_i x_i$$

for some scalar values $\alpha_1, \ldots, \alpha_{n-1} \in \mathbb{R}$, then we say that the vectors $x_1, \ldots, x_n$ are linearly dependent; otherwise, the vectors are linearly independent. For example, the vectors

$$x_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad x_2 = \begin{bmatrix} 4 \\ 1 \\ 5 \end{bmatrix} \quad x_3 = \begin{bmatrix} 2 \\ -3 \\ -1 \end{bmatrix}$$

are linearly dependent because $x_3 = -2x_1 + x_2$.

The ***column rank*** of a matrix $A \in \mathbb{R}^{m\times n}$ is the size of the largest subset of columns of A that constitute a linearly independent set. With some abuse of terminology, this is often referred to simply as the number of linearly independent columns of A. In the same way, the ***row rank*** is the largest number of rows of A that constitute a linearly independent set.

For any matrix $A \in \mathbb{R}^{m\times n}$, it turns out that the column rank of A is equal to the row rank of A (though we will not prove this), and so both quantities are referred to collectively as the ***rank*** of A, denoted as $\text{rank}(A)$. The following are some basic properties of the rank:

- For $A \in \mathbb{R}^{m\times n}$, $\text{rank}(A) \le \min(m, n)$. If $\text{rank}(A) = \min(m, n)$, then A is said to be ***full rank***.
- For $A \in \mathbb{R}^{m\times n}$, $\text{rank}(A) = \text{rank}(A^T)$.
- For $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $\text{rank}(AB) \le \min(\text{rank}(A), \text{rank}(B))$.
- For $A, B \in \mathbb{R}^{m\times n}$, $\text{rank}(A + B) \le \text{rank}(A) + \text{rank}(B)$.

3.7 The Inverse

The ***inverse*** of a square matrix $A \in \mathbb{R}^{n\times n}$ is denoted A^{-1}, and is the unique matrix such that

$$A^{-1}A = I = AA^{-1}.$$

Note that not all matrices have inverses. Non-square matrices, for example, do not have inverses by definition. However, for some square matrices A, it may still be the case that

A^{-1} may not exist. In particular, we say that A is ***invertible*** or ***non-singular*** if A^{-1} exists and ***non-invertible*** or ***singular*** otherwise.[1]

In order for a square matrix A to have an inverse A^{-1}, then A must be full rank. We will soon see that there are many alternative sufficient and necessary conditions, in addition to full rank, for invertibility.

The following are properties of the inverse; all assume that $A, B \in \mathbb{R}^{n\times n}$ are non-singular:

- $(A^{-1})^{-1} = A$
- $(AB)^{-1} = B^{-1}A^{-1}$
- $(A^{-1})^T = (A^T)^{-1}$. For this reason this matrix is often denoted A^{-T}.

As an example of how the inverse is used, consider the linear system of equations, $Ax = b$ where $A \in \mathbb{R}^{n\times n}$, and $x, b \in \mathbb{R}^n$. If A is nonsingular (i.e., invertible), then $x = A^{-1}b$. (What if $A \in \mathbb{R}^{m\times n}$ is not a square matrix? Does this work?)

3.8 Orthogonal Matrices

Two vectors $x, y \in \mathbb{R}^n$ are ***orthogonal*** if $x^Ty = 0$. A vector $x \in \mathbb{R}^n$ is ***normalized*** if $\|x\|_2 = 1$. A square matrix $U \in \mathbb{R}^{n\times n}$ is ***orthogonal*** (note the different meanings when talking about vectors versus matrices) if all its columns are orthogonal to each other and are normalized (the columns are then referred to as being ***orthonormal***).

It follows immediately from the definition of orthogonality and normality that

$$U^TU = I = UU^T.$$

In other words, the inverse of an orthogonal matrix is its transpose. Note that if U is not square — i.e., $U \in \mathbb{R}^{m\times n}$, $n < m$ — but its columns are still orthonormal, then $U^TU = I$, but $UU^T \neq I$. We generally only use the term orthogonal to describe the previous case, where U is square.

Another nice property of orthogonal matrices is that operating on a vector with an orthogonal matrix will not change its Euclidean norm, i.e.,

$$\|Ux\|_2 = \|x\|_2$$

for any $x \in \mathbb{R}^n$, $U \in \mathbb{R}^{n\times n}$ orthogonal.

3.9 Range and Nullspace of a Matrix

The ***span*** of a set of vectors $\{x_1, x_2, \ldots x_n\}$ is the set of all vectors that can be expressed as a linear combination of $\{x_1, \ldots, x_n\}$. That is,

$$\mathrm{span}(\{x_1, \ldots x_n\}) = \left\{ v : v = \sum_{i=1}^n \alpha_i x_i, \ \ \alpha_i \in \mathbb{R} \right\}.$$

[1] It's easy to get confused and think that non-singular means non-invertible. But in fact, it means the opposite! Watch out!

It can be shown that if $\{x_1, \ldots, x_n\}$ is a set of n linearly independent vectors, where each $x_i \in \mathbb{R}^n$, then $\text{span}(\{x_1, \ldots x_n\}) = \mathbb{R}^n$. In other words, *any* vector $v \in \mathbb{R}^n$ can be written as a linear combination of x_1 through x_n. The ***projection*** of a vector $y \in \mathbb{R}^m$ onto the span of $\{x_1, \ldots, x_n\}$ (here we assume $x_i \in \mathbb{R}^m$) is the vector $v \in \text{span}(\{x_1, \ldots x_n\})$, such that v is as close as possible to y, as measured by the Euclidean norm $\|v - y\|_2$. We denote the projection as $\text{Proj}(y; \{x_1, \ldots, x_n\})$ and can define it formally as,

$$\text{Proj}(y; \{x_1, \ldots x_n\}) = \text{argmin}_{v \in \text{span}(\{x_1, \ldots, x_n\})} \|y - v\|_2.$$

The ***range*** (sometimes also called the columnspace) of a matrix $A \in \mathbb{R}^{m \times n}$, denoted $\mathcal{R}(A)$, is the the span of the columns of A. In other words,

$$\mathcal{R}(A) = \{v \in \mathbb{R}^m : v = Ax, x \in \mathbb{R}^n\}.$$

Making a few technical assumptions (namely that A is full rank and that $n < m$), the projection of a vector $y \in \mathbb{R}^m$ onto the range of A is given by,

$$\text{Proj}(y; A) = \text{argmin}_{v \in \mathcal{R}(A)} \|v - y\|_2 = A(A^T A)^{-1} A^T y \quad .$$

This last equation should look extremely familiar, since it is almost the same formula we derived in class (and which we will soon derive again) for the least squares estimation of parameters. Looking at the definition for the projection, it should not be too hard to convince yourself that this is in fact the same objective that we minimized in our least squares problem (except for a squaring of the norm, which doesn't affect the optimal point) and so these problems are naturally very connected. When A contains only a single column, $a \in \mathbb{R}^m$, this gives the special case for a projection of a vector on to a line:

$$\text{Proj}(y; a) = \frac{aa^T}{a^T a} y \quad .$$

The ***nullspace*** of a matrix $A \in \mathbb{R}^{m \times n}$, denoted $\mathcal{N}(A)$ is the set of all vectors that equal 0 when multiplied by A, i.e.,

$$\mathcal{N}(A) = \{x \in \mathbb{R}^n : Ax = 0\}.$$

Note that vectors in $\mathcal{R}(A)$ are of size m, while vectors in the $\mathcal{N}(A)$ are of size n, so vectors in $\mathcal{R}(A^T)$ and $\mathcal{N}(A)$ are both in $\mathbb{R}^n$. In fact, we can say much more. It turns out that

$$\left\{w : w = u + v, u \in \mathcal{R}(A^T), v \in \mathcal{N}(A)\right\} = \mathbb{R}^n \text{ and } \mathcal{R}(A^T) \cap \mathcal{N}(A) = \{\mathbf{0}\} \quad .$$

In other words, $\mathcal{R}(A^T)$ and $\mathcal{N}(A)$ are disjoint subsets that together span the entire space of $\mathbb{R}^n$. Sets of this type are called ***orthogonal complements***, and we denote this $\mathcal{R}(A^T) = \mathcal{N}(A)^\perp$.

3.10 The Determinant

The ***determinant*** of a square matrix $A \in \mathbb{R}^{n\times n}$, is a function $\det : \mathbb{R}^{n\times n} \to \mathbb{R}$, and is denoted $|A|$ or $\det A$ (like the trace operator, we usually omit parentheses). Algebraically, one could write down an explicit formula for the determinant of A, but this unfortunately gives little intuition about its meaning. Instead, we'll start out by providing a geometric interpretation of the determinant and then visit some of its specific algebraic properties afterwards.

Given a matrix

$$\begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_n^T & — \end{bmatrix},$$

consider the set of points $S \subset \mathbb{R}^n$ formed by taking all possible linear combinations of the row vectors $a_1, \ldots, a_n \in \mathbb{R}^n$ of A, where the coefficients of the linear combination are all between 0 and 1; that is, the set S is the restriction of $\text{span}(\{a_1, \ldots, a_n\})$ to only those linear combinations whose coefficients $\alpha_1, \ldots, \alpha_n$ satisfy $0 \le \alpha_i \le 1$, $i = 1, \ldots, n$. Formally,

$$S = \{v \in \mathbb{R}^n : v = \sum_{i=1}^{n} \alpha_i a_i \text{ where } 0 \le \alpha_i \le 1, i = 1, \ldots, n\}.$$

The absolute value of the determinant of A, it turns out, is a measure of the "volume" of the set S.[2]

For example, consider the 2×2 matrix,

$$A = \begin{bmatrix} 1 & 3 \\ 3 & 2 \end{bmatrix}. \tag{1}$$

Here, the rows of the matrix are

$$a_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad a_2 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}.$$

The set S corresponding to these rows is shown in Figure 1. For two-dimensional matrices, S generally has the shape of a *parallelogram.* In our example, the value of the determinant is $|A| = -7$ (as can be computed using the formulas shown later in this section), so the area of the parallelogram is 7. (Verify this for yourself!)

In three dimensions, the set S corresponds to an object known as a *parallelepiped* (a three-dimensional box with skewed sides, such that every face has the shape of a parallelogram). The absolute value of the determinant of the 3×3 matrix whose rows define S give the three-dimensional volume of the parallelepiped. In even higher dimensions, the set S is an object known as an n-dimensional *parallelotope.*

[2] Admittedly, we have not actually defined what we mean by "volume" here, but hopefully the intuition should be clear enough. When $n = 2$, our notion of "volume" corresponds to the area of S in the Cartesian plane. When $n = 3$, "volume" corresponds with our usual notion of volume for a three-dimensional object.

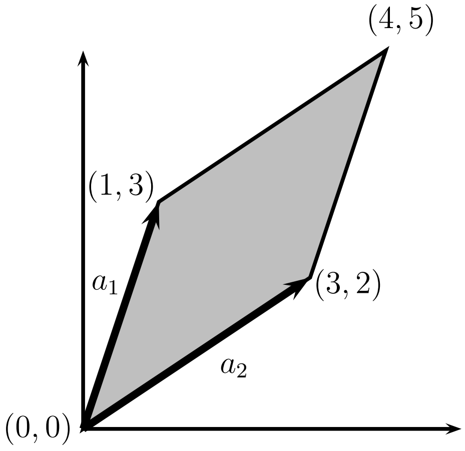


Figure 1: Illustration of the determinant for the 2×2 matrix A given in (1). Here, a_1 and a_2 are vectors corresponding to the rows of A, and the set S corresponds to the shaded region (i.e., the parallelogram). The absolute value of the determinant, $|\text{det}A| = 7$, is the area of the parallelogram.

Algebraically, the determinant satisfies the following three properties (from which all other properties follow, including the general formula):

1. The determinant of the identity is 1, $|I| = 1$. (Geometrically, the volume of a unit hypercube is 1).

2. Given a matrix $A \in \mathbb{R}^{n \times n}$, if we multiply a single row in A by a scalar $t \in \mathbb{R}$, then the determinant of the new matrix is $t|A|$,

$$\left| \begin{bmatrix} — & t\ a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} \right| = t|A|.$$

   (Geometrically, multiplying one of the sides of the set S by a factor t causes the volume to increase by a factor t.)

3. If we exchange any two rows a_i^T and a_j^T of A, then the determinant of the new matrix is $-|A|$, for example

$$\left| \begin{bmatrix} — & a_2^T & — \\ — & a_1^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} \right| = -|A|.$$

In case you are wondering, it is not immediately obvious that a function satisfying the above three properties exists. In fact, though, such a function does exist, and is unique (which we will not prove here).

Several properties that follow from the three properties above include:

- For $A \in \mathbb{R}^{n\times n}$, $|A| = |A^T|$.
- For $A, B \in \mathbb{R}^{n\times n}$, $|AB| = |A||B|$.
- For $A \in \mathbb{R}^{n\times n}$, $|A| = 0$ if and only if A is singular (i.e., non-invertible). (If A is singular then it does not have full rank, and hence its columns are linearly dependent. In this case, the set S corresponds to a "flat sheet" within the n-dimensional space and hence has zero volume.)
- For $A \in \mathbb{R}^{n\times n}$ and A non-singular, $|A^{-1}| = 1/|A|$.

Before giving the general definition for the determinant, we define, for $A \in \mathbb{R}^{n\times n}$, $A_{\backslash i,\backslash j} \in \mathbb{R}^{(n-1)\times(n-1)}$ to be the *matrix* that results from deleting the ith row and jth column from A. The general (recursive) formula for the determinant is

$$\begin{aligned}|A| &= \sum_{i=1}^{n}(-1)^{i+j}a_{ij}|A_{\backslash i,\backslash j}| \quad \text{(for any } j \in 1,\ldots,n\text{)}\\ &= \sum_{j=1}^{n}(-1)^{i+j}a_{ij}|A_{\backslash i,\backslash j}| \quad \text{(for any } i \in 1,\ldots,n\text{)}\end{aligned}$$

with the initial case that $|A| = a_{11}$ for $A \in \mathbb{R}^{1\times 1}$. If we were to expand this formula completely for $A \in \mathbb{R}^{n\times n}$, there would be a total of $n!$ (n factorial) different terms. For this reason, we hardly ever explicitly write the complete equation of the determinant for matrices bigger than 3×3. However, the equations for determinants of matrices up to size 3×3 are fairly common, and it is good to know them:

$$\begin{aligned}|[a_{11}]| &= a_{11}\\ \left|\begin{bmatrix} a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\right| &= a_{11}a_{22} - a_{12}a_{21}\\ \left|\begin{bmatrix} a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}\right| &= \begin{array}{l} a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}\\ \quad -a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31}\end{array}\end{aligned}$$

The ***classical adjoint*** (often just called the adjoint) of a matrix $A \in \mathbb{R}^{n\times n}$, is denoted $\mathrm{adj}(A)$, and defined as

$$\mathrm{adj}(A) \in \mathbb{R}^{n\times n}, \quad (\mathrm{adj}(A))_{ij} = (-1)^{i+j}|A_{\backslash j,\backslash i}|$$

(note the switch in the indices $A_{\backslash j,\backslash i}$). It can be shown that for any nonsingular $A \in \mathbb{R}^{n\times n}$,

$$A^{-1} = \frac{1}{|A|}\mathrm{adj}(A) \ .$$

While this is a nice "explicit" formula for the inverse of matrix, we should note that, numerically, there are in fact much more efficient ways of computing the inverse.

3.11 Quadratic Forms and Positive Semidefinite Matrices

Given a square matrix $A \in \mathbb{R}^{n \times n}$ and a vector $x \in \mathbb{R}^n$, the scalar value $x^T Ax$ is called a ***quadratic form***. Written explicitly, we see that

$$x^T Ax = \sum_{i=1}^{n} x_i (Ax)_i = \sum_{i=1}^{n} x_i \left(\sum_{j=1}^{n} A_{ij} x_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j \ \ .$$

Note that,

$$x^T Ax = (x^T Ax)^T = x^T A^T x = x^T \left(\frac{1}{2} A + \frac{1}{2} A^T \right) x,$$

where the first equality follows from the fact that the transpose of a scalar is equal to itself, and the second equality follows from the fact that we are averaging two quantities which are themselves equal. From this, we can conclude that only the symmetric part of A contributes to the quadratic form. For this reason, we often implicitly assume that the matrices appearing in a quadratic form are symmetric.

We give the following definitions:

- A symmetric matrix $A \in \mathbb{S}^n$ is ***positive definite*** (PD) if for all non-zero vectors $x \in \mathbb{R}^n$, $x^T Ax > 0$. This is usually denoted $A \succ 0$ (or just $A > 0$), and often times the set of all positive definite matrices is denoted $\mathbb{S}^n_{++}$.

- A symmetric matrix $A \in \mathbb{S}^n$ is ***positive semidefinite*** (PSD) if for all vectors $x^T Ax \geq 0$. This is written $A \succeq 0$ (or just $A \geq 0$), and the set of all positive semidefinite matrices is often denoted $\mathbb{S}^n_{+}$.

- Likewise, a symmetric matrix $A \in \mathbb{S}^n$ is ***negative definite*** (ND), denoted $A \prec 0$ (or just $A < 0$) if for all non-zero $x \in \mathbb{R}^n$, $x^T Ax < 0$.

- Similarly, a symmetric matrix $A \in \mathbb{S}^n$ is ***negative semidefinite*** (NSD), denoted $A \preceq 0$ (or just $A \leq 0$) if for all $x \in \mathbb{R}^n$, $x^T Ax \leq 0$.

- Finally, a symmetric matrix $A \in \mathbb{S}^n$ is ***indefinite***, if it is neither positive semidefinite nor negative semidefinite — i.e., if there exists $x_1, x_2 \in \mathbb{R}^n$ such that $x_1^T Ax_1 > 0$ and $x_2^T Ax_2 < 0$.

It should be obvious that if A is positive definite, then $-A$ is negative definite and vice versa. Likewise, if A is positive semidefinite then $-A$ is negative semidefinite and vice versa. If A is indefinite, then so is $-A$.

One important property of positive definite and negative definite matrices is that they are always full rank, and hence, invertible. To see why this is the case, suppose that some matrix $A \in \mathbb{R}^{n \times n}$ is not full rank. Then, suppose that the jth column of A is expressible as a linear combination of other $n-1$ columns:

$$a_j = \sum_{i \neq j} x_i a_i,$$

for some $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n \in \mathbb{R}$. Setting $x_j = -1$, we have

$$Ax = \sum_{i=1}^{n} x_i a_i = 0.$$

But this implies $x^T Ax = 0$ for some non-zero vector x, so A must be neither positive definite nor negative definite. Therefore, if A is either positive definite or negative definite, it must be full rank.

Finally, there is one type of positive definite matrix that comes up frequently, and so deserves some special mention. Given any matrix $A \in \mathbb{R}^{m \times n}$ (not necessarily symmetric or even square), the matrix $G = A^T A$ (sometimes called a ***Gram matrix***) is always positive semidefinite. Further, if $m \geq n$ (and we assume for convenience that A is full rank), then $G = A^T A$ is positive definite.

3.12 Eigenvalues and Eigenvectors

Given a square matrix $A \in \mathbb{R}^{n \times n}$, we say that $\lambda \in \mathbb{C}$ is an ***eigenvalue*** of A and $x \in \mathbb{C}^n$ is the corresponding ***eigenvector***[3] if

$$Ax = \lambda x, \quad x \neq 0.$$

Intuitively, this definition means that multiplying A by the vector x results in a new vector that points in the same direction as x, but scaled by a factor λ. Also note that for any eigenvector $x \in \mathbb{C}^n$, and scalar $t \in \mathbb{C}$, $A(cx) = cAx = c\lambda x = \lambda(cx)$, so cx is also an eigenvector. For this reason when we talk about "the" eigenvector associated with λ, we usually assume that the eigenvector is normalized to have length 1 (this still creates some ambiguity, since x and $-x$ will both be eigenvectors, but we will have to live with this).

We can rewrite the equation above to state that (λ, x) is an eigenvalue-eigenvector pair of A if,

$$(\lambda I - A)x = 0, \quad x \neq 0.$$

But $(\lambda I - A)x = 0$ has a non-zero solution to x if and only if $(\lambda I - A)$ has a non-empty nullspace, which is only the case if $(\lambda I - A)$ is singular, i.e.,

$$|(\lambda I - A)| = 0.$$

We can now use the previous definition of the determinant to expand this expression into a (very large) polynomial in λ, where λ will have maximum degree n. We then find the n (possibly complex) roots of this polynomial to find the n eigenvalues $\lambda_1, \ldots, \lambda_n$. To find the eigenvector corresponding to the eigenvalue λ_i, we simply solve the linear equation $(\lambda_i I - A)x = 0$. It should be noted that this is not the method which is actually used

[3] Note that λ and the entries of x are actually in $\mathbb{C}$, the set of complex numbers, not just the reals; we will see shortly why this is necessary. Don't worry about this technicality for now, you can think of complex vectors in the same way as real vectors.

in practice to numerically compute the eigenvalues and eigenvectors (remember that the complete expansion of the determinant has $n!$ terms); it is rather a mathematical argument.

The following are properties of eigenvalues and eigenvectors (in all cases assume $A \in \mathbb{R}^{n\times n}$ has eigenvalues $\lambda_i, \ldots, \lambda_n$ and associated eigenvectors $x_1, \ldots x_n$):

- The trace of a A is equal to the sum of its eigenvalues,
$$\mathrm{tr}A = \sum_{i=1}^{n} \lambda_i.$$
- The determinant of A is equal to the product of its eigenvalues,
$$|A| = \prod_{i=1}^{n} \lambda_i.$$
- The rank of A is equal to the number of non-zero eigenvalues of A.
- If A is non-singular then $1/\lambda_i$ is an eigenvalue of A^{-1} with associated eigenvector x_i, i.e., $A^{-1}x_i = (1/\lambda_i)x_i$. (To prove this, take the eigenvector equation, $Ax_i = \lambda_i x_i$ and left-multiply each side by A^{-1}.)
- The eigenvalues of a diagonal matrix $D = \mathrm{diag}(d_1, \ldots d_n)$ are just the diagonal entries $d_1, \ldots d_n$.

We can write all the eigenvector equations simultaneously as

$$AX = X\Lambda$$

where the columns of $X \in \mathbb{R}^{n\times n}$ are the eigenvectors of A and Λ is a diagonal matrix whose entries are the eigenvalues of A, i.e.,

$$X \in \mathbb{R}^{n\times n} = \left[\begin{array}{cccc} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{array} \right], \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n).$$

If the eigenvectors of A are linearly independent, then the matrix X will be invertible, so $A = X\Lambda X^{-1}$. A matrix that can be written in this form is called ***diagonalizable***.

3.13 Eigenvalues and Eigenvectors of Symmetric Matrices

Two remarkable properties come about when we look at the eigenvalues and eigenvectors of a symmetric matrix $A \in \mathbb{S}^n$. First, it can be shown that all the eigenvalues of A are real. Secondly, the eigenvectors of A are orthonormal, i.e., the matrix X defined above is an orthogonal matrix (for this reason, we denote the matrix of eigenvectors as U in this case).

We can therefore represent A as $A = U\Lambda U^T$, remembering from above that the inverse of an orthogonal matrix is just its transpose.

Using this, we can show that the definiteness of a matrix depends entirely on the sign of its eigenvalues. Suppose $A \in \mathbb{S}^n = U\Lambda U^T$. Then

$$x^T Ax = x^T U\Lambda U^T x = y^T \Lambda y = \sum_{i=1}^{n} \lambda_i y_i^2$$

where $y = U^T x$ (and since U is full rank, any vector $y \in \mathbb{R}^n$ can be represented in this form). Because y_i^2 is always positive, the sign of this expression depends entirely on the λ_i's. If all $\lambda_i > 0$, then the matrix is positive definite; if all $\lambda_i \geq 0$, it is positive semidefinite. Likewise, if all $\lambda_i < 0$ or $\lambda_i \leq 0$, then A is negative definite or negative semidefinite respectively. Finally, if A has both positive and negative eigenvalues, it is indefinite.

An application where eigenvalues and eigenvectors come up frequently is in maximizing some function of a matrix. In particular, for a matrix $A \in \mathbb{S}^n$, consider the following maximization problem,

$$\max_{x \in \mathbb{R}^n} \; x^T Ax \quad \text{subject to } \|x\|_2^2 = 1$$

i.e., we want to find the vector (of norm 1) which maximizes the quadratic form. Assuming the eigenvalues are ordered as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, the optimal x for this optimization problem is x_1, the eigenvector corresponding to λ_1. In this case the maximal value of the quadratic form is λ_1. Similarly, the optimal solution to the minimization problem,

$$\min_{x \in \mathbb{R}^n} \; x^T Ax \quad \text{subject to } \|x\|_2^2 = 1$$

is x_n, the eigenvector corresponding to λ_n, and the minimal value is λ_n. This can be proved by appealing to the eigenvector-eigenvalue form of A and the properties of orthogonal matrices. However, in the next section we will see a way of showing it directly using matrix calculus.

4 Matrix Calculus

While the topics in the previous sections are typically covered in a standard course on linear algebra, one topic that does not seem to be covered very often (and which we will use extensively) is the extension of calculus to the vector setting. Despite the fact that all the actual calculus we use is relatively trivial, the notation can often make things look much more difficult than they are. In this section we present some basic definitions of matrix calculus and provide a few examples.

4.1 The Gradient

Suppose that $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a function that takes as input a matrix A of size $m \times n$ and returns a real value. Then the ***gradient*** of f (with respect to $A \in \mathbb{R}^{m\times n}$) is the matrix of partial derivatives, defined as:

$$\nabla_A f(A) \in \mathbb{R}^{m\times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

i.e., an $m \times n$ matrix with

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}.$$

Note that the size of $\nabla_A f(A)$ is always the same as the size of A. So if, in particular, A is just a vector $x \in \mathbb{R}^n$,

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}.$$

It is very important to remember that the gradient of a function is *only* defined if the function is real-valued, that is, if it returns a scalar value. We can not, for example, take the gradient of $Ax, A \in \mathbb{R}^{n\times n}$ with respect to x, since this quantity is vector-valued.

It follows directly from the equivalent properties of partial derivatives that:

- $\nabla_x(f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x)$.
- For $t \in \mathbb{R}$, $\nabla_x(t\, f(x)) = t\nabla_x f(x)$.

In principle, gradients are a natural extension of partial derivatives to functions of multiple variables. In practice, however, working with gradients can sometimes be tricky for notational reasons. For example, suppose that $A \in \mathbb{R}^{m\times n}$ is a matrix of fixed coefficients and suppose that $b \in \mathbb{R}^m$ is a vector of fixed coefficients. Let $f : \mathbb{R}^m \to \mathbb{R}$ be the function defined by $f(z) = z^T z$, such that $\nabla_z f(z) = 2z$. But now, consider the expression,

$$\nabla f(Ax).$$

How should this expression be interpreted? There are at least two possibilities:

1. In the first interpretation, recall that $\nabla_z f(z) = 2z$. Here, we interpret $\nabla f(Ax)$ as evaluating the gradient at the point Ax, hence,

$$\nabla f(Ax) = 2(Ax) = 2Ax \in \mathbb{R}^m.$$

2. In the second interpretation, we consider the quantity $f(Ax)$ as a function of the input variables x. More formally, let $g(x) = f(Ax)$. Then in this interpretation,

$$\nabla f(Ax) = \nabla_x g(x) \in \mathbb{R}^n.$$

Here, we can see that these two interpretations are indeed different. One interpretation yields an m-dimensional vector as a result, while the other interpretation yields an n-dimensional vector as a result! How can we resolve this?

Here, the key is to make explicit the variables which we are differentiating with respect to. In the first case, we are differentiating the function f with respect to its arguments z and then substituting the argument Ax. In the second case, we are differentiating the composite function $g(x) = f(Ax)$ with respect to x directly. We denote the first case as $\nabla_z f(Ax)$ and the second case as $\nabla_x f(Ax)$.[4] Keeping the notation clear is extremely important (as you'll find out in your homework, in fact!).

4.2 The Hessian

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a function that takes a vector in $\mathbb{R}^n$ and returns a real number. Then the ***Hessian*** matrix with respect to x, written $\nabla_x^2 f(x)$ or simply as H is the $n \times n$ matrix of partial derivatives,

$$\nabla_x^2 f(x) \in \mathbb{R}^{n \times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}.$$

In other words, $\nabla_x^2 f(x) \in \mathbb{R}^{n \times n}$, with

$$(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}.$$

Note that the Hessian is always symmetric, since

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i}.$$

Similar to the gradient, the Hessian is defined only when $f(x)$ is real-valued.

It is natural to think of the gradient as the analogue of the first derivative for functions of vectors, and the Hessian as the analogue of the second derivative (and the symbols we use also suggest this relation). This intuition is generally correct, but there a few caveats to keep in mind.

[4] A drawback to this notation that we will have to live with is the fact that in the first case, $\nabla_z f(Ax)$ it appears that we are differentiating with respect to a variable that does not even appear in the expression being differentiated! For this reason, the first case is often written as $\nabla f(Ax)$, and the fact that we are differentiating with respect to the arguments of f is understood. However, the second case is *always* written as $\nabla_x f(Ax)$.

First, for real-valued functions of one variable $f : \mathbb{R} \to \mathbb{R}$, it is a basic definition that the second derivative is the derivative of the first derivative, i.e.,

$$\frac{\partial^2 f(x)}{\partial x^2} = \frac{\partial}{\partial x}\frac{\partial}{\partial x} f(x).$$

However, for functions of a vector, the gradient of the function is a vector, and we cannot take the gradient of a vector — i.e.,

$$\nabla_x \nabla_x f(x) = \nabla_x \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

and this expression is not defined. Therefore, it is *not* the case that the Hessian is the gradient of the gradient. However, this is *almost* true, in the following sense: If we look at the ith entry of the gradient $(\nabla_x f(x))_i = \partial f(x)/\partial x_i$, and take the gradient with respect to x we get

$$\nabla_x \frac{\partial f(x)}{\partial x_i} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_i \partial x_1} \\ \frac{\partial^2 f(x)}{\partial x_i \partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_i \partial x_n} \end{bmatrix}$$

which is the ith column (or row) of the Hessian. Therefore,

$$\nabla_x^2 f(x) = \left[\begin{array}{cccc} \nabla_x (\nabla_x f(x))_1 & \nabla_x (\nabla_x f(x))_2 & \cdots & \nabla_x (\nabla_x f(x))_n \end{array} \right].$$

If we don't mind being a little bit sloppy we can say that (essentially) $\nabla_x^2 f(x) = \nabla_x (\nabla_x f(x))^T$, so long as we understand that this really means taking the gradient of each entry of $(\nabla_x f(x))^T$, not the gradient of the whole vector.

Finally, note that while we can take the gradient with respect to a matrix $A \in \mathbb{R}^n$, for the purposes of this class we will only consider taking the Hessian with respect to a vector $x \in \mathbb{R}^n$. This is simply a matter of convenience (and the fact that none of the calculations we do require us to find the Hessian with respect to a matrix), since the Hessian with respect to a matrix would have to represent all the partial derivatives $\partial^2 f(A)/(\partial A_{ij} \partial A_{k\ell})$, and it is rather cumbersome to represent this as a matrix.

4.3 Gradients and Hessians of Quadratic and Linear Functions

Now let's try to determine the gradient and Hessian matrices for a few simple functions. It should be noted that all the gradients given here are special cases of the gradients given in the CS229 lecture notes.

For $x \in \mathbb{R}^n$, let $f(x) = b^T x$ for some known vector $b \in \mathbb{R}^n$. Then

$$f(x) = \sum_{i=1}^{n} b_i x_i$$

so

$$\frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{n} b_i x_i = b_k.$$

From this we can easily see that $\nabla_x b^T x = b$. This should be compared to the analogous situation in single variable calculus, where $\partial/(\partial x)\; ax = a$.

Now consider the quadratic function $f(x) = x^T A x$ for $A \in \mathbb{S}^n$. Remember that

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j.$$

To take the partial derivative, we'll consider the terms including x_k and x_k^2 factors separately:

$$\begin{aligned}
\frac{\partial f(x)}{\partial x_k} &= \frac{\partial}{\partial x_k} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j \\
&= \frac{\partial}{\partial x_k} \left[\sum_{i \neq k} \sum_{j \neq k} A_{ij} x_i x_j + \sum_{i \neq k} A_{ik} x_i x_k + \sum_{j \neq k} A_{kj} x_k x_j + A_{kk} x_k^2 \right] \\
&= \sum_{i \neq k} A_{ik} x_i + \sum_{j \neq k} A_{kj} x_j + 2 A_{kk} x_k \\
&= \sum_{i=1}^{n} A_{ik} x_i + \sum_{j=1}^{n} A_{kj} x_j = 2 \sum_{i=1}^{n} A_{ki} x_i,
\end{aligned}$$

where the last equality follows since A is symmetric (which we can safely assume, since it is appearing in a quadratic form). Note that the kth entry of $\nabla_x f(x)$ is just the inner product of the kth row of A and x. Therefore, $\nabla_x x^T A x = 2Ax$. Again, this should remind you of the analogous fact in single-variable calculus, that $\partial/(\partial x)\; ax^2 = 2ax$.

Finally, let's look at the Hessian of the quadratic function $f(x) = x^T A x$ (it should be obvious that the Hessian of a linear function $b^T x$ is zero). In this case,

$$\frac{\partial^2 f(x)}{\partial x_k \partial x_\ell} = \frac{\partial}{\partial x_k} \left[\frac{\partial f(x)}{\partial x_\ell} \right] = \frac{\partial}{\partial x_k} \left[2 \sum_{i=1}^{n} A_{\ell i} x_i \right] = 2 A_{\ell k} = 2 A_{k \ell}.$$

Therefore, it should be clear that $\nabla_x^2 x^T A x = 2A$, which should be entirely expected (and again analogous to the single-variable fact that $\partial^2/(\partial x^2)\; ax^2 = 2a$).

To recap,

- $\nabla_x b^T x = b$
- $\nabla_x x^T A x = 2Ax$ (if A symmetric)
- $\nabla_x^2 x^T A x = 2A$ (if A symmetric)

4.4 Least Squares

Let's apply the equations we obtained in the last section to derive the least squares equations. Suppose we are given matrices $A \in \mathbb{R}^{m\times n}$ (for simplicity we assume A is full rank) and a vector $b \in \mathbb{R}^m$ such that $b \notin \mathcal{R}(A)$. In this situation we will not be able to find a vector $x \in \mathbb{R}^n$, such that $Ax = b$, so instead we want to find a vector x such that Ax is as close as possible to b, as measured by the square of the Euclidean norm $\|Ax - b\|_2^2$.

Using the fact that $\|x\|_2^2 = x^T x$, we have

$$\begin{aligned} \|Ax - b\|_2^2 &= (Ax - b)^T(Ax - b) \\ &= x^T A^T Ax - 2b^T Ax + b^T b \end{aligned}$$

Taking the gradient with respect to x we have, and using the properties we derived in the previous section

$$\begin{aligned} \nabla_x(x^T A^T Ax - 2b^T Ax + b^T b) &= \nabla_x x^T A^T Ax - \nabla_x 2b^T Ax + \nabla_x b^T b \\ &= 2A^T Ax - 2A^T b \end{aligned}$$

Setting this last expression equal to zero and solving for x gives the normal equations

$$x = (A^T A)^{-1} A^T b$$

which is the same as what we derived in class.

4.5 Gradients of the Determinant

Now let's consider a situation where we find the gradient of a function with respect to a matrix, namely for $A \in \mathbb{R}^{n\times n}$, we want to find $\nabla_A |A|$. Recall from our discussion of determinants that

$$|A| = \sum_{i=1}^{n} (-1)^{i+j} A_{ij} |A_{\backslash i,\backslash j}| \qquad \text{(for any } j \in 1,\ldots,n)$$

so

$$\frac{\partial}{\partial A_{k\ell}} |A| = \frac{\partial}{\partial A_{k\ell}} \sum_{i=1}^{n} (-1)^{i+j} A_{ij} |A_{\backslash i,\backslash j}| = (-1)^{k+\ell} |A_{\backslash k,\backslash \ell}| = (\operatorname{adj}(A))_{\ell k}.$$

From this it immediately follows from the properties of the adjoint that

$$\nabla_A |A| = (\operatorname{adj}(A))^T = |A| A^{-T}.$$

Now let's consider the function $f : \mathbb{S}^n_{++} \to \mathbb{R}$, $f(A) = \log |A|$. Note that we have to restrict the domain of f to be the positive definite matrices, since this ensures that $|A| > 0$, so that the log of $|A|$ is a real number. In this case we can use the chain rule (nothing fancy, just the ordinary chain rule from single-variable calculus) to see that

$$\frac{\partial \log |A|}{\partial A_{ij}} = \frac{\partial \log |A|}{\partial |A|} \frac{\partial |A|}{\partial A_{ij}} = \frac{1}{|A|} \frac{\partial |A|}{\partial A_{ij}}.$$

From this it should be obvious that

$$\nabla_A \log |A| = \frac{1}{|A|}\nabla_A |A| = A^{-1},$$

where we can drop the transpose in the last expression because A is symmetric. Note the similarity to the single-valued case, where $\partial/(\partial x)\ \log x = 1/x$.

4.6 Eigenvalues as Optimization

Finally, we use matrix calculus to solve an optimization problem in a way that leads directly to eigenvalue/eigenvector analysis. Consider the following, equality constrained optimization problem:

$$\max_{x\in\mathbb{R}^n}\ x^TAx \quad \text{subject to } \|x\|_2^2 = 1$$

for a symmetric matrix $A \in \mathbb{S}^n$. A standard way of solving optimization problems with equality constraints is by forming the ***Lagrangian***, an objective function that includes the equality constraints.[5] The Lagrangian in this case can be given by

$$\mathcal{L}(x, \lambda) = x^TAx - \lambda x^Tx$$

where λ is called the Lagrange multiplier associated with the equality constraint. It can be established that for x^* to be a optimal point to the problem, the gradient of the Lagrangian has to be zero at x^* (this is not the only condition, but it is required). That is,

$$\nabla_x\mathcal{L}(x, \lambda) = \nabla_x(x^TAx - \lambda x^Tx) = 2A^Tx - 2\lambda x = 0.$$

Notice that this is just the linear equation $Ax = \lambda x$. This shows that the only points which can possibly maximize (or minimize) x^TAx assuming $x^Tx = 1$ are the eigenvectors of A.

[5] Don't worry if you haven't seen Lagrangians before, as we will cover them in greater detail later in CS229.

CS229 Supplemental Lecture notes Hoeffding's inequality

John Duchi

1 Basic probability bounds

A basic question in probability, statistics, and machine learning is the following: given a random variable Z with expectation $\mathbb{E}[Z]$, how likely is Z to be close to its expectation? And more precisely, how close is it likely to be? With that in mind, these notes give a few tools for computing bounds of the form

$$\mathbb{P}(Z \geq \mathbb{E}[Z] + t) \;\text{ and }\; \mathbb{P}(Z \leq \mathbb{E}[Z] - t) \tag{1}$$

for $t \geq 0$.

Our first bound is perhaps the most basic of all probability inequalities, and it is known as Markov's inequality. Given its basic-ness, it is perhaps unsurprising that its proof is essentially only one line.

Proposition 1 (Markov's inequality)**.** *Let $Z \geq 0$ be a non-negative random variable. Then for all $t \geq 0$,*

$$\mathbb{P}(Z \geq t) \leq \frac{\mathbb{E}[Z]}{t}.$$

Proof We note that $\mathbb{P}(Z \geq t) = \mathbb{E}[\mathbf{1}\{Z \geq t\}]$, and that if $Z \geq t$, then it must be the case that $Z/t \geq 1 \geq \mathbf{1}\{Z \geq t\}$, while if $Z < t$, then we still have $Z/t \geq 0 = \mathbf{1}\{Z \geq t\}$. Thus

$$\mathbb{P}(Z \geq t) = \mathbb{E}[\mathbf{1}\{Z \geq t\}] \leq \mathbb{E}\left[\frac{Z}{t}\right] = \frac{\mathbb{E}[Z]}{t},$$

as desired. □

Essentially all other bounds on the probabilities (1) are variations on Markov's inequality. The first variation uses second moments—the variance—of a random variable rather than simply its mean, and is known as Chebyshev's inequality.

Proposition 2 (Chebyshev's inequality)**.** *Let Z be any random variable with* $\mathrm{Var}(Z) < \infty$*. Then*

$$\mathbb{P}(Z \geq \mathbb{E}[Z] + t \text{ or } Z \leq \mathbb{E}[Z] - t) \leq \frac{\mathrm{Var}(Z)}{t^2}$$

for $t \geq 0$*.*

Proof The result is an immediate consequence of Markov's inequality. We note that if $Z \geq \mathbb{E}[Z] + t$, then certainly we have $(Z - \mathbb{E}[Z])^2 \geq t^2$, and similarly if $Z \leq \mathbb{E}[Z] - t$ we have $(Z - \mathbb{E}[Z])^2 \geq t^2$. Thus

$$\begin{aligned}\mathbb{P}(Z \geq \mathbb{E}[Z] + t \text{ or } Z \leq \mathbb{E}[Z] - t) &= \mathbb{P}((Z - \mathbb{E}[Z])^2 \geq t^2) \\ &\overset{(i)}{\leq} \frac{\mathbb{E}[(Z - \mathbb{E}[Z])^2]}{t^2} = \frac{\mathrm{Var}(Z)}{t^2},\end{aligned}$$

where step (i) is Markov's inequality. □

A nice consequence of Chebyshev's inequality is that averages of random variables with finite variance converge to their mean. Let us give an example of this fact. Suppose that Z_i are i.i.d. and satisfy $\mathbb{E}[Z_i] = 0$. Then $\mathbb{E}[Z_i] = 0$, while if we define $\bar{Z} = \frac{1}{n}\sum_{i=1}^n Z_i$ then

$$\mathrm{Var}(\bar{Z}) = \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n Z_i\right)^2\right] = \frac{1}{n^2}\sum_{i,j \leq n} \mathbb{E}[Z_i Z_j] = \frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[Z_i^2] = \frac{\mathrm{Var}(Z_1)}{n}.$$

In particular, for any $t \geq 0$ we have

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^n Z_i\right| \geq t\right) \leq \frac{\mathrm{Var}(Z_1)}{nt^2},$$

so that $\mathbb{P}(|\bar{Z}| \geq t) \to 0$ for any $t > 0$.

2 Moment generating functions

Often, we would like sharper—even exponential—bounds on the probability that a random variable Z exceeds its expectation by much. With that in mind, we need a stronger condition than finite variance, for which moment generating functions are natural candidates. (Conveniently, they also play nicely with sums, as we will see.) Recall that for a random variable Z, the *moment generating function* of Z is the function

$$M_Z(\lambda) := \mathbb{E}[\exp(\lambda Z)], \tag{2}$$

which may be infinite for some λ.

2.1 Chernoff bounds

Chernoff bounds use of moment generating functions in an essential way to give exponential deviation bounds.

Proposition 3 (Chernoff bounds)**.** *Let Z be any random variable. Then for any $t \geq 0$,*

$$\mathbb{P}(Z \geq \mathbb{E}[Z] + t) \leq \min_{\lambda \geq 0} \mathbb{E}[e^{\lambda(Z - \mathbb{E}[Z])}]e^{-\lambda t} = \min_{\lambda \geq 0} M_{Z - \mathbb{E}[Z]}(\lambda)e^{-\lambda t}$$

and

$$\mathbb{P}(Z \leq \mathbb{E}[Z] - t) \leq \min_{\lambda \geq 0} \mathbb{E}[e^{\lambda(\mathbb{E}[Z] - Z)}]e^{-\lambda t} = \min_{\lambda \geq 0} M_{\mathbb{E}[Z] - Z}(\lambda)e^{-\lambda t}.$$

Proof We only prove the first inequality, as the second is completely identical. We use Markov's inequality. For any $\lambda > 0$, we have $Z \geq \mathbb{E}[Z] + t$ if and only if $e^{\lambda Z} \geq e^{\lambda \mathbb{E}[Z] + \lambda t}$, or $e^{\lambda(Z - \mathbb{E}[Z])} \geq e^{\lambda t}$. Thus, we have

$$\mathbb{P}(Z - \mathbb{E}[Z] \geq t) = \mathbb{P}(e^{\lambda(Z - \mathbb{E}[Z])} \geq e^{\lambda t}) \overset{(i)}{\leq} \mathbb{E}[e^{\lambda(Z - \mathbb{E}[Z])}]e^{-\lambda t},$$

where the inequality (i) follows from Markov's inequality. As our choice of $\lambda > 0$ did not matter, we can take the best one by minizing the right side of the bound. (And noting that certainly the bound holds at $\lambda = 0$.) □

The important result is that Chernoff bounds "play nicely" with summations, which is a consequence of the moment generating function. Let us assume that Z_i are independent. Then we have that

$$M_{Z_1+\cdots+Z_n}(\lambda) = \prod_{i=1}^n M_{Z_i}(\lambda),$$

which we see because

$$\mathbb{E}\left[\exp\left(\lambda \sum_{i=1}^n Z_i\right)\right] = \mathbb{E}\left[\prod_{i=1}^n \exp(\lambda Z_i)\right] = \prod_{i=1}^n \mathbb{E}[\exp(\lambda Z_i)],$$

by of the independence of the Z_i. This means that when we calculate a Chernoff bound of a sum of i.i.d. variables, we need only calculate the moment generating function for *one* of them. Indeed, suppose that Z_i are i.i.d. and (for simplicity) mean zero. Then

$$\begin{aligned}\mathbb{P}\left(\sum_{i=1}^n Z_i \geq t\right) &\leq \frac{\prod_{i=1}^n \mathbb{E}\left[\exp(\lambda Z_i)\right]}{e^{\lambda t}} \\ &= (\mathbb{E}[e^{\lambda Z_1}])^n e^{-\lambda t},\end{aligned}$$

by the Chernoff bound.

2.2 Moment generating function examples

Now we give several examples of moment generating functions, which enable us to give a few nice deviation inequalities as a result. For all of our examples, we will have very convienent bounds of the form

$$M_Z(\lambda) = \mathbb{E}[e^{\lambda Z}] \leq \exp\left(\frac{C^2\lambda^2}{2}\right) \quad \text{for all } \lambda \in \mathbb{R},$$

for some $C \in \mathbb{R}$ (which depends on the distribution of Z); this form is *very* nice for applying Chernoff bounds.

We begin with the classical normal distribution, where $Z \sim \mathcal{N}(0, \sigma^2)$. Then we have

$$\mathbb{E}[\exp(\lambda Z)] = \exp\left(\frac{\lambda^2\sigma^2}{2}\right),$$

which one obtains via a calculation that we omit. (You should work this out if you are curious!)

A second example is known as a Rademacher random variable, or the random sign variable. Let $S = 1$ with probability $\frac{1}{2}$ and $S = -1$ with probability $\frac{1}{2}$. Then we claim that

$$\mathbb{E}[e^{\lambda S}] \le \exp\left(\frac{\lambda^2}{2}\right) \quad \text{for all } \lambda \in \mathbb{R}. \tag{3}$$

To see inequality (3), we use the Taylor expansion of the exponential function, that is, that $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$. Note that $\mathbb{E}[S^k] = 0$ whenever k is odd, while $\mathbb{E}[S^k] = 1$ whenever k is even. Then we have

$$\begin{aligned}\mathbb{E}[e^{\lambda S}] &= \sum_{k=0}^{\infty} \frac{\lambda^k \mathbb{E}[S^k]}{k!} \\ &= \sum_{k=0,2,4,\ldots} \frac{\lambda^k}{k!} = \sum_{k=0}^{\infty} \frac{\lambda^{2k}}{(2k)!}.\end{aligned}$$

Finally, we use that $(2k)! \ge 2^k \cdot k!$ for all $k = 0, 1, 2, \ldots$, so that

$$\mathbb{E}[e^{\lambda S}] \le \sum_{k=0}^{\infty} \frac{(\lambda^2)^k}{2^k \cdot k!} = \sum_{k=0}^{\infty} \left(\frac{\lambda^2}{2}\right)^k \frac{1}{k!} = \exp\left(\frac{\lambda^2}{2}\right).$$

Let us apply inequality (3) in a Chernoff bound to see how large a sum of i.i.d. random signs is likely to be.

We have that if $Z = \sum_{i=1}^{n} S_i$, where $S_i \in \{\pm 1\}$ is a random sign, then $\mathbb{E}[Z] = 0$. By the Chernoff bound, it becomes immediately clear that

$$\mathbb{P}(Z \ge t) \le \mathbb{E}[e^{\lambda Z}] e^{-\lambda t} = \mathbb{E}[e^{\lambda S_1}]^n e^{-\lambda t} \le \exp\left(\frac{n\lambda^2}{2}\right) e^{-\lambda t}.$$

Applying the Chernoff bound technique, we may minimize this in $\lambda \ge 0$, which is equivalent to finding

$$\min_{\lambda \ge 0} \left\{ \frac{n\lambda^2}{2} - \lambda t \right\}.$$

Luckily, this is a convenient function to minimize: taking derivatives and setting to zero, we have $n\lambda - t = 0$, or $\lambda = t/n$, which gives

$$\mathbb{P}(Z \ge t) \le \exp\left(-\frac{t^2}{2n}\right).$$

In particular, taking $t = \sqrt{2n \log \frac{1}{\delta}}$, we have

$$\mathbb{P}\left(\sum_{i=1}^n S_i \geq \sqrt{2n \log \frac{1}{\delta}}\right) \leq \delta.$$

So $Z = \sum_{i=1}^n S_i = O(\sqrt{n})$ with extremely high probability—the sum of n independent random signs is essentially never larger than $O(\sqrt{n})$.

3 Hoeffding's lemma and Hoeffding's inequality

Hoeffding's inequality is a powerful technique—perhaps the most important inequality in learning theory—for bounding the probability that sums of bounded random variables are too large or too small. We will state the inequality, and then we will prove a weakened version of it based on our moment generating function calculations earlier.

Theorem 4 (Hoeffding's inequality)**.** *Let $Z_1, \ldots, Z_n$ be independent bounded random variables with $Z_i \in [a, b]$ for all i, where $-\infty < a \leq b < \infty$. Then*

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n (Z_i - \mathbb{E}[Z_i]) \geq t\right) \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right)$$

and

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n (Z_i - \mathbb{E}[Z_i]) \leq -t\right) \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right)$$

for all $t \geq 0$.

We prove Theorem 4 by using a combination of (1) Chernoff bounds and (2) a classic lemma known as Hoeffding's lemma, which we now state.

Lemma 5 (Hoeffding's lemma)**.** *Let Z be a bounded random variable with $Z \in [a, b]$. Then*

$$\mathbb{E}[\exp(\lambda(Z - \mathbb{E}[Z]))] \leq \exp\left(\frac{\lambda^2(b-a)^2}{8}\right) \quad \text{for all } \lambda \in \mathbb{R}.$$

Proof We prove a slightly weaker version of this lemma with a factor of 2 instead of 8 using our random sign moment generating bound and an inequality known as *Jensen's inequality* (we will see this very important inequality later in our derivation of the EM algorithm). Jensen's inequality states the following: if $f : \mathbb{R} \to \mathbb{R}$ is a *convex* function, meaning that f is bowl-shaped, then

$$f(\mathbb{E}[Z]) \le \mathbb{E}[f(Z)].$$

The simplest way to remember this inequality is to think of $f(t) = t^2$, and note that if $\mathbb{E}[Z] = 0$ then $f(\mathbb{E}[Z]) = 0$, while we generally have $\mathbb{E}[Z^2] > 0$. In any case, $f(t) = \exp(t)$ and $f(t) = \exp(-t)$ are convex functions.

We use a clever technique in probability theory known as *symmetrization* to give our result (you are not expected to know this, but it is a very common technique in probability theory, machine learning, and statistics, so it is good to have seen). First, let Z' be an independent copy of Z with the same distribution, so that $Z' \in [a, b]$ and $\mathbb{E}[Z'] = \mathbb{E}[Z]$, but Z and Z' are independent. Then

$$\mathbb{E}_Z[\exp(\lambda(Z - \mathbb{E}_Z[Z]))] = \mathbb{E}_Z[\exp(\lambda(Z - \mathbb{E}_{Z'}[Z']))] \overset{(i)}{\le} \mathbb{E}_Z[\mathbb{E}_{Z'} \exp(\lambda(Z - Z'))],$$

where $\mathbb{E}_Z$ and $\mathbb{E}_{Z'}$ indicate expectations taken with respect to Z and Z'. Here, step (i) uses Jensen's inequality applied to $f(x) = e^{-x}$. Now, we have

$$\mathbb{E}[\exp(\lambda(Z - \mathbb{E}[Z]))] \le \mathbb{E}\left[\exp\left(\lambda(Z - Z')\right)\right].$$

Now, we note a curious fact: the difference $Z - Z'$ is symmetric about zero, so that if $S \in \{-1, 1\}$ is a random sign variable, then $S(Z - Z')$ has exactly the same distribution as $Z - Z'$. So we have

$$\begin{aligned}\mathbb{E}_{Z,Z'}[\exp(\lambda(Z - Z'))] &= \mathbb{E}_{Z,Z',S}[\exp(\lambda S(Z - Z'))] \\ &= \mathbb{E}_{Z,Z'}\left[\mathbb{E}_S\left[\exp(\lambda S(Z - Z')) \mid Z, Z'\right]\right].\end{aligned}$$

Now we use inequality (3) on the moment generating function of the random sign, which gives that

$$\mathbb{E}_S\left[\exp(\lambda S(Z - Z')) \mid Z, Z'\right] \le \exp\left(\frac{\lambda^2 (Z - Z')^2}{2}\right).$$

But of course, by assumption we have $|Z - Z'| \le (b - a)$, so $(Z - Z')^2 \le (b - a)^2$. This gives

$$\mathbb{E}_{Z,Z'}[\exp(\lambda(Z - Z'))] \le \exp\left(\frac{\lambda^2 (b - a)^2}{2}\right).$$

This is the result (except with a factor of 2 instead of 8). □

Now we use Hoeffding's lemma to prove Theorem 4, giving only the upper tail (i.e. the probability that $\frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq t$) as the lower tail has a similar proof. We use the Chernoff bound technique, which immediately tells us that

$$\begin{aligned}
\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq t\right) &= \mathbb{P}\left(\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq nt\right) \\
&\leq \mathbb{E}\left[\exp\left(\lambda\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i])\right)\right] e^{-\lambda nt} \\
&= \left(\prod_{i=1}^{n}\mathbb{E}[e^{\lambda(Z_i - \mathbb{E}[Z_i])}]\right) e^{-\lambda nt} \overset{(i)}{\leq} \left(\prod_{i=1}^{n} e^{\frac{\lambda^2(b-a)^2}{8}}\right) e^{-\lambda nt}
\end{aligned}$$

where inequality (i) is Hoeffding's Lemma (Lemma 5). Rewriting this slightly and minimzing over $\lambda \geq 0$, we have

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq t\right) \leq \min_{\lambda\geq 0}\exp\left(\frac{n\lambda^2(b-a)^2}{8} - \lambda nt\right) = \exp\left(-\frac{2nt^2}{(b-a)^2}\right),$$

as desired.

Hidden Markov Models Fundamentals

Daniel Ramage
CS229 Section Notes

December 1, 2007

Abstract

How can we apply machine learning to data that is represented as a sequence of observations over time? For instance, we might be interested in discovering the sequence of words that someone spoke based on an audio recording of their speech. Or we might be interested in annotating a sequence of words with their part-of-speech tags. These notes provides a thorough mathematical introduction to the concept of Markov Models — a formalism for reasoning about states over time — and Hidden Markov Models — where we wish to recover a series of states from a series of observations. The final section includes some pointers to resources that present this material from other perspectives.

1 Markov Models

Given a set of states $S = \{s_1, s_2, ...s_{|S|}\}$ we can observe a series over time $\vec{z} \in S^T$. For example, we might have the states from a weather system $S = \{sun, cloud, rain\}$ with $|S| = 3$ and observe the weather over a few days $\{z_1 = s_{sun}, z_2 = s_{cloud}, z_3 = s_{cloud}, z_4 = s_{rain}, z_5 = s_{cloud}\}$ with $T = 5$.

The observed states of our weather example represent the output of a random process over time. Without some further assumptions, state s_j at time t could be a function of any number of variables, including all the states from times 1 to $t-1$ and possibly many others that we don't even model. However, we will make two MARKOV ASSUMPTIONS that will allow us to tractably reason about time series.

The LIMITED HORIZON ASSUMPTION is that the probability of being in a state at time t depends only on the state at time $t-1$. The intuition underlying this assumption is that the state at time t represents "enough" summary of the past to reasonably predict the future. Formally:

$$P(z_t|z_{t-1}, z_{t-2}, ..., z_1) = P(z_t|z_{t-1})$$

The STATIONARY PROCESS ASSUMPTION is that the conditional distribution over next state given current state does not change over time. Formally:

$$P(z_t|z_{t-1}) = P(z_2|z_1);\ t \in 2...T$$

As a convention, we will also assume that there is an initial state and initial observation $z_0 \equiv s_0$, where s_0 represents the initial probability distribution over states at time 0. This notational convenience allows us to encode our belief about the prior probability of seeing the first real state z_1 as $P(z_1|z_0)$. Note that $P(z_t|z_{t-1}, ..., z_1) = P(z_t|z_{t-1}, ..., z_1, z_0)$ because we've defined $z_0 = s_0$ for any state sequence. (Other presentations of HMMs sometimes represent these prior believes with a vector $\pi \in \mathbb{R}^{|S|}$.)

We parametrize these transitions by defining a state transition matrix $A \in \mathbb{R}^{(|S|+1)\times(|S|+1)}$. The value A_{ij} is the probability of transitioning from state i to state j at any time t. For our sun and rain example, we might have following transition matrix:

$$A = \begin{array}{c|cccc} & s_0 & s_{sun} & s_{cloud} & s_{rain} \\ s_0 & 0 & .33 & .33 & .33 \\ s_{sun} & 0 & .8 & .1 & .1 \\ s_{cloud} & 0 & .2 & .6 & .2 \\ s_{rain} & 0 & .1 & .2 & .7 \end{array}$$

Note that these numbers (which I made up) represent the intuition that the weather is self-correlated: if it's sunny it will tend to stay sunny, cloudy will stay cloudy, etc. This pattern is common in many Markov models and can be observed as a strong diagonal in the transition matrix. Note that in this example, our initial state s_0 shows uniform probability of transitioning to each of the three states in our weather system.

1.1 Two questions of a Markov Model

Combining the Markov assumptions with our state transition parametrization A, we can answer two basic questions about a sequence of states in a Markov chain. What is the probability of a particular sequence of states $\vec{z}$? And how do we estimate the parameters of our model A such to maximize the likelihood of an observed sequence $\vec{z}$?

1.1.1 Probability of a state sequence

We can compute the probability of a particular series of states $\vec{z}$ by use of the chain rule of probability:

$$\begin{aligned} P(\vec{z}) &= P(z_t, z_{t-1}, ..., z_1; A) \\ &= P(z_t, z_{t-1}, ..., z_1, z_0; A) \\ &= P(z_t|z_{t-1}, z_{t-2}, ..., z_1; A)P(z_{t-1}|z_{t-2}, ..., z_1; A)...P(z_1|z_0; A) \\ &= P(z_t|z_{t-1}; A)P(z_{t-1}|z_{t-2}; A)...P(z_2|z_1; A)P(z_1|z_0; A) \end{aligned}$$

$$
\begin{aligned}
&= \prod_{t=1}^{T} P(z_t|z_{t-1}; A) \\
&= \prod_{t=1}^{T} A_{z_{t-1}\, z_t}
\end{aligned}
$$

In the second line we introduce z_0 into our joint probability, which is allowed by the definition of z_0 above. The third line is true of any joint distribution by the chain rule of probabilities or repeated application of Bayes rule. The fourth line follows from the Markov assumptions and the last line represents these terms as their elements in our transition matrix A.

Let's compute the probability of our example time sequence from earlier. We want $P(z_1 = s_{sun}, z_2 = s_{cloud}, z_3 = s_{rain}, z_4 = s_{rain}, z_5 = s_{cloud})$ which can be factored as $P(s_{sun}|s_0)P(s_{cloud}|s_{sun})P(s_{rain}|s_{cloud})P(s_{rain}|s_{rain})P(s_{cloud}|s_{rain}) = .33 \times .1 \times .2 \times .7 \times .2$.

1.1.2 Maximum likelihood parameter assignment

From a learning perspective, we could seek to find the parameters A that maximize the log-likelihood of sequence of observations $\vec{z}$. This corresponds to finding the likelihoods of transitioning from sunny to cloudy versus sunny to sunny, etc., that make a set of observations most likely. Let's define the log-likelihood a Markov model.

$$
\begin{aligned}
l(A) &= \log P(\vec{z}; A) \\
&= \log \prod_{t=1}^{T} A_{z_{t-1}\, z_t} \\
&= \sum_{t=1}^{T} \log A_{z_{t-1}\, z_t} \\
&= \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}
\end{aligned}
$$

In the last line, we use an indicator function whose value is one when the condition holds and zero otherwise to select the observed transition at each time step. When solving this optimization problem, it's important to ensure that solved parameters A still make a valid transition matrix. In particular, we need to enforce that the outgoing probability distribution from state i always sums to 1 and all elements of A are non-negative. We can solve this optimization problem using the method of Lagrange multipliers.

$$
\max_{A} \qquad l(A)
$$

$$\begin{aligned} s.t. \qquad & \sum_{j=1}^{|S|} A_{ij} = 1, \;\; i = 1..|S| \\ & A_{ij} \geq 0, \;\; i,j = 1..|S| \end{aligned}$$

This constrained optimization problem can be solved in closed form using the method of Lagrange multipliers. We'll introduce the equality constraint into the Lagrangian, but the inequality constraint can safely be ignored — the optimal solution will produce positive values for A_{ij} anyway. Therefore we construct the Lagrangian as:

$$\mathcal{L}(A,\alpha) \;=\; \sum_{i=1}^{|S|}\sum_{j=1}^{|S|}\sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij} + \sum_{i=1}^{|S|} \alpha_i (1 - \sum_{j=1}^{|S|} A_{ij})$$

Taking partial derivatives and setting them equal to zero we get:

$$\begin{aligned} \frac{\partial \mathcal{L}(A,\alpha)}{\partial A_{ij}} &= \frac{\partial}{\partial A_{ij}} (\sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}) + \frac{\partial}{\partial A_{ij}} \alpha_i (1 - \sum_{j=1}^{|S|} A_{ij}) \\ &= \frac{1}{A_{ij}} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} - \alpha_i \equiv 0 \\ &\Rightarrow \\ A_{ij} &= \frac{1}{\alpha_i} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \end{aligned}$$

Substituting back in and setting the partial with respect to α equal to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}(A,\beta)}{\partial \alpha_i} &= 1 - \sum_{j=1}^{|S|} A_{ij} \\ &= 1 - \sum_{j=1}^{|S|} \frac{1}{\alpha_i} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \equiv 0 \\ &\Rightarrow \\ \alpha_i &= \sum_{j=1}^{|S|} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \\ &= \sum_{t=1}^{T} 1\{z_{t-1} = s_i\} \end{aligned}$$

Substituting in this value for α_i into the expression we derived for A_{ij} we obtain our final maximum likelihood parameter value for $\hat{A}_{ij}$.

$$\hat{A}_{ij} \quad = \quad \frac{\sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}}{\sum_{t=1}^{T} 1\{z_{t-1} = s_i\}}$$

This formula encodes a simple intuition: the maximum likelihood probability of transitioning from state i to state j is just the number of times we transition from i to j divided by the total number of times we are in i. In other words, the maximum likelihood parameter corresponds to the fraction of the time when we were in state i that we transitioned to j.

2 Hidden Markov Models

Markov Models are a powerful abstraction for time series data, but fail to capture a very common scenario. How can we reason about a series of states if we cannot observe the states themselves, but rather only some probabilistic function of those states? This is the scenario for part-of-speech tagging where the words are observed but the parts-of-speech tags aren't, and for speech recognition where the sound sequence is observed but not the words that generated it. For a simple example, let's borrow the setup proposed by Jason Eisner in 2002 [1], "Ice Cream Climatology."

> The situation: You are a climatologist in the year 2799, studying the history of global warming. You can't find any records of Baltimore weather, but you do find my (Jason Eisner's) diary, in which I assiduously recorded how much ice cream I ate each day. *What can you figure out from this about the weather that summer?*

A Hidden Markov Model (HMM) can be used to explore this scenario. We don't get to observe the actual sequence of states (the weather on each day). Rather, we can only observe some outcome generated by each state (how many ice creams were eaten that day).

Formally, an HMM is a Markov model for which we have a series of *observed* outputs $x = \{x_1, x_2, ..., x_T\}$ drawn from an output alphabet $V = \{v_1, v_2, ..., v_{|V|}\}$, i.e. $x_t \in V$, $t = 1..T$. As in the previous section, we also posit the existence of series of states $z = \{z_1, z_2, ..., z_T\}$ drawn from a state alphabet $S = \{s_1, s_2, ...s_{|S|}\}$, $z_t \in S$, $t = 1..T$ but in this scenario the values of the states are *unobserved*. The transition between states i and j will again be represented by the corresponding value in our state transition matrix A_{ij}.

We also model the probability of generating an output observation as a function of our hidden state. To do so, we make the OUTPUT INDEPENDENCE ASSUMPTION and define $P(x_t = v_k | z_t = s_j) = P(x_t = v_k | x_1, ..., x_T, z_1, ..., z_T) = B_{jk}$. The matrix B encodes the probability of our hidden state generating output v_k given that the state at the corresponding time was s_j.

Returning to the weather example, imagine that you have logs of ice cream consumption over a four day period: $\vec{x} = \{x_1 = v_3, x_2 = v_2, x_3 = v_1, x_4 = v_2\}$

where our alphabet just encodes the number of ice creams consumed, i.e. $V = \{v_1 = 1\, ice\, cream, v_2 = 2\, ice\, creams, v_3 = 3\, ice\, creams\}$. What questions can an HMM let us answer?

2.1 Three questions of a Hidden Markov Model

There are three fundamental questions we might ask of an HMM. What is the probability of an observed sequence (how likely were we to see $3, 2, 1, 2$ ice creams consumed)? What is the most likely series of states to generate the observations (what was the weather for those four days)? And how can we learn values for the HMM's parameters A and B given some data?

2.2 Probability of an observed sequence: Forward procedure

In an HMM, we assume that our data was generated by the following process: posit the existence of a series of states $\vec{z}$ over the length of our time series. This state sequence is generated by a Markov model parametrized by a state transition matrix A. At each time step t, we select an output x_t as a function of the state z_t. Therefore, to get the probability of a sequence of observations, we need to add up the likelihood of the data $\vec{x}$ given every possible series of states.

$$\begin{aligned} P(\vec{x}; A, B) &= \sum_{\vec{z}} P(\vec{x}, \vec{z}; A, B) \\ &= \sum_{\vec{z}} P(\vec{x}|\vec{z}; A, B) P(\vec{z}; A, B) \end{aligned}$$

The formulas above are true for any probability distribution. However, the HMM assumptions allow us to simplify the expression further:

$$\begin{aligned} P(\vec{x}; A, B) &= \sum_{\vec{z}} P(\vec{x}|\vec{z}; A, B) P(\vec{z}; A, B) \\ &= \sum_{\vec{z}} (\prod_{t=1}^{T} P(x_t|z_t; B)) (\prod_{t=1}^{T} P(z_t|z_{t-1}; A)) \\ &= \sum_{\vec{z}} (\prod_{t=1}^{T} B_{z_t\, x_t}) (\prod_{t=1}^{T} A_{z_{t-1}\, z_t}) \end{aligned}$$

The good news is that this is a simple expression in terms of our parameters. The derivation follows the HMM assumptions: the output independence assumption, Markov assumption, and stationary process assumption are all used to derive the second line. The bad news is that the sum is over every possible assignment to $\vec{z}$. Because z_t can take one of $|S|$ possible values at each time step, evaluating this sum directly will require $O(|S|^T)$ operations.

Algorithm 1 Forward Procedure for computing $\alpha_i(t)$

```
1. Base case: $\alpha_i(0) = A_{0\,i},\ i = 1..|S|$
2. Recursion: $\alpha_j(t) = \sum_{i=1}^{|S|} \alpha_i(t-1) A_{ij} B_{j\,x_t},\ \ j = 1..|S|,\ t = 1..T$
```

Fortunately, a faster means of computing $P(\vec{x}; A, B)$ is possible via a dynamic programming algorithm called the Forward Procedure. First, let's define a quantity $\alpha_i(t) = P(x_1, x_2, ..., x_t, z_t = s_i; A, B)$. $\alpha_i(t)$ represents the total probability of all the observations up through time t (by any state assignment) and that we are in state s_i at time t. If we had such a quantity, the probability of our full set of observations $P(\vec{x})$ could be represented as:

$$\begin{aligned} P(\vec{x}; A, B) &= P(x_1, x_2, ..., x_T; A, B) \\ &= \sum_{i=1}^{|S|} P(x_1, x_2, ..., x_T, z_T = s_i; A, B) \\ &= \sum_{i=1}^{|S|} \alpha_i(T) \end{aligned}$$

Algorithm 2.2 presents an efficient way to compute $\alpha_i(t)$. At each time step we must do only $O(|S|)$ operations, resulting in a final algorithm complexity of $O(|S| \cdot T)$ to compute the total probability of an observed state sequence $P(\vec{x}; A, B)$.

A similar algorithm known as the Backward Procedure can be used to compute an analogous probability $\beta_i(t) = P(x_T, x_{T-1}, .., x_{t+1}, z_t = s_i; A, B)$.

2.3 Maximum Likelihood State Assignment: The Viterbi Algorithm

One of the most common queries of a Hidden Markov Model is to ask what was the most likely series of states $\vec{z} \in S^T$ given an observed series of outputs $\vec{x} \in V^T$. Formally, we seek:

$$\arg\max_{\vec{z}} P(\vec{z}|\vec{x}; A, B) = \arg\max_{\vec{z}} \frac{P(\vec{x}, \vec{z}; A, B)}{\sum_{\vec{z}} P(\vec{x}, \vec{z}; A, B)} = \arg\max_{\vec{z}} P(\vec{x}, \vec{z}; A, B)$$

The first simplification follows from Bayes rule and the second from the observation that the denominator does not directly depend on $\vec{z}$. Naively, we might try every possible assignment to $\vec{z}$ and take the one with the highest joint probability assigned by our model. However, this would require $O(|S|^T)$ operations just to enumerate the set of possible assignments. At this point, you might think a dynamic programming solution like the Forward Algorithm might save the day, and you'd be right. Notice that if you replaced the $\arg\max_{\vec{z}}$ with $\sum_{\vec{z}}$, our current task is exactly analogous to the expression which motivated the forward procedure.

Algorithm 2 Naive application of EM to HMMs

```
Repeat until convergence {
(E-Step) For every possible labeling $\vec{z} \in S^T$, set

    $Q(\vec{z}) := p(\vec{z}|\vec{x}; A, B)$

(M-Step) Set

    $A, B := \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log \frac{P(\vec{x}, \vec{z}; A, B)}{Q(\vec{z})}$
    $s.t. \sum_{j=1}^{|S|} A_{ij} = 1,\ i = 1..|S|;\ A_{ij} \geq 0,\ \ i, j = 1..|S|$
    $\sum_{k=1}^{|V|} B_{ik} = 1,\ i = 1..|S|;\ B_{ik} \geq 0,\ \ i = 1..|S|, k = 1..|V|$
}
```

The Viterbi Algorithm is just like the forward procedure except that instead of tracking the total probability of generating the observations seen so far, we need only track the *maximum* probability and record its corresponding state sequence.

2.4 Parameter Learning: EM for HMMs

The final question to ask of an HMM is: given a set of observations, what are the values of the state transition probabilities A and the output emission probabilities B that make the data most likely? For example, solving for the maximum likelihood parameters based on a speech recognition dataset will allow us to effectively train the HMM before asking for the maximum likelihood state assignment of a candidate speech signal.

In this section, we present a derivation of the Expectation Maximization algorithm for Hidden Markov Models. This proof follows from the general formulation of EM presented in the CS229 lecture notes. Algorithm 2.4 shows the basic EM algorithm. Notice that the optimization problem in the M-Step is now constrained such that A and B contain valid probabilities. Like the maximum likelihood solution we found for (non-Hidden) Markov models, we'll be able to solve this optimization problem with Lagrange multipliers. Notice also that the E-Step and M-Step both require enumerating all $|S|^T$ possible labellings of $\vec{z}$. We'll make use of the Forward and Backward algorithms mentioned earlier to compute a set of sufficient statistics for our E-Step and M-Step tractably.

First, let's rewrite the objective function using our Markov assumptions.

$$\begin{aligned}
A, B &= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log \frac{P(\vec{x}, \vec{z}; A, B)}{Q(\vec{z})} \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log P(\vec{x}, \vec{z}; A, B) \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log(\prod_{t=1}^{T} P(x_t|z_t; B)) \, (\prod_{t=1}^{T} P(z_t|z_{t-1}; A)) \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} \log B_{z_t\, x_t} + \log A_{z_{t-1}\, z_t} \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{k=1}^{|V|} \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \log B_{jk} + 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}
\end{aligned}$$

In the first line we split the log division into a subtraction and note that the denominator's term does not depend on the parameters A, B. The Markov assumptions are applied in line 3. Line 5 uses indicator functions to index A and B by state.

Just as for the maximum likelihood parameters for a visible Markov model, it is safe to ignore the inequality constraints because the solution form naturally results in only positive solutions. Constructing the Lagrangian:

$$\begin{aligned}
\mathcal{L}(A, B, \delta, \epsilon) &= \sum_{\vec{z}} Q(\vec{z}) \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{k=1}^{|V|} \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \log B_{jk} + 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij} \\
&\quad + \sum_{j=1}^{|S|} \epsilon_j (1 - \sum_{k=1}^{|V|} B_{jk}) + \sum_{i=1}^{|S|} \delta_i (1 - \sum_{j=1}^{|S|} A_{ij})
\end{aligned}$$

Taking partial derivatives and setting them equal to zero:

$$\begin{aligned}
\frac{\partial \mathcal{L}(A, B, \delta, \epsilon)}{\partial A_{ij}} &= \sum_{\vec{z}} Q(\vec{z}) \frac{1}{A_{ij}} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} - \delta_i \equiv 0 \\
A_{ij} &= \frac{1}{\delta_i} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}(A, B, \delta, \epsilon)}{\partial B_{jk}} &= \sum_{\vec{z}} Q(\vec{z}) \frac{1}{B_{jk}} \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} - \epsilon_j \equiv 0 \\
B_{jk} &= \frac{1}{\epsilon_j} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\}
\end{aligned}$$

Taking partial derivatives with respect to the Lagrange multipliers and substituting our values of A_{ij} and B_{jk} above:

$$\begin{aligned}
\frac{\partial \mathcal{L}(A,B,\delta,\epsilon)}{\partial \delta_i} &= 1 - \sum_{j=1}^{|S|} A_{ij} \\
&= 1 - \sum_{j=1}^{|S|} \frac{1}{\delta_i} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \equiv 0 \\
\delta_i &= \sum_{j=1}^{|S|} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \\
&= \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i\}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}(A,B,\delta,\epsilon)}{\partial \epsilon_j} &= 1 - \sum_{k=1}^{|V|} B_{jk} \\
&= 1 - \sum_{k=1}^{|V|} \frac{1}{\epsilon_j} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \equiv 0 \\
\epsilon_j &= \sum_{k=1}^{|V|} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \\
&= \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j\}
\end{aligned}$$

Substituting back into our expressions above, we find that parameters $\hat{A}$ and $\hat{B}$ that maximize our predicted counts with respect to the dataset are:

$$\begin{aligned}
\hat{A}_{ij} &= \frac{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}}{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i\}} \\
\hat{B}_{jk} &= \frac{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\}}{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j\}}
\end{aligned}$$

Unfortunately, each of these sums is over all possible labellings $\vec{z} \in S^T$. But recall that $Q(\vec{z})$ was defined in the E-step as $P(\vec{z}|\vec{x}; A, B)$ for parameters A and B at the last time step. Let's consider how to represent first the numerator of $\hat{A}_{ij}$ in terms of our forward and backward probabilities, $\alpha_i(t)$ and $\beta_j(t)$.

$$\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}$$

$$
\begin{aligned}
&= \sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1} = s_i \wedge z_t = s_j\} Q(\vec{z}) \\
&= \sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1} = s_i \wedge z_t = s_j\} P(\vec{z}|\vec{x}; A, B) \\
&= \frac{1}{P(\vec{x}; A, B)} \sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1} = s_i \wedge z_t = s_j\} P(\vec{z}, \vec{x}; A, B) \\
&= \frac{1}{P(\vec{x}; A, B)} \sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)
\end{aligned}
$$

In the first two steps we rearrange terms and substitute in for our definition of Q. Then we use Bayes rule in deriving line four, followed by the definitions of α, β, A, and B, in line five. Similarly, the denominator can be represented by summing out over j the value of the numerator.

$$
\begin{aligned}
& \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i\} \\
&= \sum_{j=1}^{|S|}\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \\
&= \frac{1}{P(\vec{x}; A, B)} \sum_{j=1}^{|S|}\sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)
\end{aligned}
$$

Combining these expressions, we can fully characterize our maximum likelihood state transitions $\hat{A}_{ij}$ without needing to enumerate all possible labellings as:

$$
\hat{A}_{ij} = \frac{\sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)}{\sum_{j=1}^{|S|}\sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)}
$$

Similarly, we can represent the numerator for $\hat{B}_{jk}$ as:

$$
\begin{aligned}
& \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \\
&= \frac{1}{P(\vec{x}; A, B)} \sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_t = s_j \wedge x_t = v_k\} P(\vec{z}, \vec{x}; A, B) \\
&= \frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|}\sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1} = s_i \wedge z_t = s_j \wedge x_t = v_k\} P(\vec{z}, \vec{x}; A, B)
\end{aligned}
$$

Algorithm 3 Forward-Backward algorithm for HMM parameter learning

```
Initialization: Set $A$ and $B$ as random valid probability matrices
    where $A_{i0} = 0$ and $B_{0k} = 0$ for $i = 1..|S|$ and $k = 1..|V|$.
Repeat until convergence {
(E-Step) Run the Forward and Backward algorithms to compute $\alpha_i$ and $\beta_i$ for $i = 1..|S|$. Then set:
    $\gamma_t(i,j) \;:=\; \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)$
(M-Step) Re-estimate the maximum likelihood parameters as:
    $A_{ij} \;:=\; \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{j=1}^{|S|} \sum_{t=1}^{T} \gamma_t(i,j)}$
    $B_{jk} \;:=\; \frac{\sum_{i=1}^{|S|} \sum_{t=1}^{T} 1\{x_t = v_k\}\, \gamma_t(i,j)}{\sum_{i=1}^{|S|} \sum_{t=1}^{T} \gamma_t(i,j)}$
}
```

$$= \frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|} \sum_{t=1}^{T} 1\{x_t = v_k\} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)$$

And the denominator of $\hat{B}_{jk}$ as:

$$\begin{aligned} & \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j\} \\ = & \frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|} \sum_{t=1}^{T} \sum_{\vec{z}} 1\{z_{t-1} = s_i \wedge z_t = s_j\} P(\vec{z}, \vec{x}; A, B) \\ = & \frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|} \sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1) \end{aligned}$$

Combining these expressions, we have the following form for our maximum likelihood emission probabilities as:

$$\hat{B}_{jk} = \frac{\sum_{i=1}^{|S|} \sum_{t=1}^{T} 1\{x_t = v_k\} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)}{\sum_{i=1}^{|S|} \sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)}$$

Algorithm 2.4 shows a variant of the FORWARD-BACKWARD ALGORITHM, or the BAUM-WELCH ALGORITHM for parameter learning in HMMs. In the

E-Step, rather than explicitly evaluating $Q(\vec{z})$ for all $\vec{z} \in S^T$, we compute a sufficient statistics $\gamma_t(i,j) = \alpha_i(t)A_{ij}B_{j\,x_t}\beta_j(t+1)$ that is proportional to the probability of transitioning between sate s_i and s_j at time t given all of our observations $\vec{x}$. The derived expressions for A_{ij} and B_{jk} are intuitively appealing. A_{ij} is computed as the expected number of transitions from s_i to s_j divided by the expected number of appearances of s_i. Similarly, B_{jk} is computed as the expected number of emissions of v_k from s_j divided by the expected number of appearances of s_j.

Like many applications of EM, parameter learning for HMMs is a non-convex problem with many local maxima. EM will converge to a maximum based on its initial parameters, so multiple runs might be in order. Also, it is often important to smooth the probability distributions represented by A and B so that no transition or emission is assigned 0 probability.

2.5 Further reading

There are many good sources for learning about Hidden Markov Models. For applications in NLP, I recommend consulting Jurafsky & Martin's draft second edition of *Speech and Language Processing*[1] or Manning & Schütze's *Foundations of Statistical Natural Language Processing*. Also, Eisner's HMM-in-a-spreadsheet [1] is a light-weight interactive way to play with an HMM that requires only a spreadsheet application.

References

[1] Jason Eisner. An interactive spreadsheet for teaching the forward-backward algorithm. In Dragomir Radev and Chris Brew, editors, *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 10–18, 2002.

[1] http://www.cs.colorado.edu/~martin/slp2.html

CS229 Supplemental Lecture notes

John Duchi

1 Binary classification

In **binary classification** problems, the target y can take on at only two values. In this set of notes, we show how to model this problem by letting $y \in \{-1, +1\}$, where we say that y is a 1 if the example is a member of the positive class and $y = -1$ if the example is a member of the negative class. We assume, as usual, that we have input features $x \in \mathbb{R}^n$.

As in our standard approach to supervised learning problems, we first pick a representation for our hypothesis class (what we are trying to learn), and after that we pick a loss function that we will minimize. In binary classification problems, it is often convenient to use a hypothesis class of the form $h_\theta(x) = \theta^T x$, and, when presented with a new example x, we classify it as positive or negative depending on the sign of $\theta^T x$, that is, our predicted label is

$$\operatorname{sign}(h_\theta(x)) = \operatorname{sign}(\theta^T x) \quad \text{where} \quad \operatorname{sign}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \\ -1 & \text{if } t < 0. \end{cases}$$

In a binary classification problem, then, the hypothesis h_θ with parameter vector θ classifies a particular example (x, y) correctly if

$$\operatorname{sign}(\theta^T x) = y \quad \text{or equivalently} \quad y\theta^T x > 0. \tag{1}$$

The quantity $y\theta^T x$ in expression (1) is a very important quantity in binary classification, important enough that we call the value

$$y x^T \theta$$

the *margin* for the example (x, y). Often, though not always, one interprets the value $h_\theta(x) = x^T\theta$ as a measure of the confidence that the parameter vector θ assigns to labels for the point x: if $x^T\theta$ is very negative (or very positive), then we more strongly believe the label y is negative (or positive).

Now that we have chosen a representation for our data, we must choose a loss function. Intuitively, we would like to choose some loss function so that for our training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, the θ chosen makes the margin $y^{(i)}\theta^T x^{(i)}$ very large for each training example. Let us fix a hypothetical example (x, y), let $z = yx^T\theta$ denote the margin, and let $\varphi : \mathbb{R} \to \mathbb{R}$ be the loss function—that is, the loss for the example (x, y) with margin $z = yx^T\theta$ is $\varphi(z) = \varphi(yx^T\theta)$. For any particular loss function, the empirical risk that we minimize is then

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \varphi(y^{(i)}\theta^T x^{(i)}). \tag{2}$$

Consider our desired behavior: we wish to have $y^{(i)}\theta^T x^{(i)}$ positive for each training example $i = 1, \ldots, m$, and we should penalize those θ for which $y^{(i)}\theta^T x^{(i)} < 0$ frequently in the training data. Thus, an intuitive choice for our loss would be one with $\varphi(z)$ small if $z > 0$ (the margin is positive), while $\varphi(z)$ is large if $z < 0$ (the margin is negative). Perhaps the most natural such loss is the *zero-one* loss, given by

$$\varphi_{\text{zo}}(z) = \begin{cases} 1 & \text{if } z \le 0 \\ 0 & \text{if } z > 0. \end{cases}$$

In this case, the risk $J(\theta)$ is simply the average number of mistakes—misclassifications—the parameter θ makes on the training data. Unfortunately, the loss φ_{zo} is discontinuous, non-convex (why this matters is a bit beyond the scope of the course), and perhaps even more vexingly, NP-hard to minimize. So we prefer to choose losses that have the shape given in Figure 1. That is, we will essentially always use losses that satisfy

$$\varphi(z) \to 0 \;\text{ as } z \to \infty, \quad \text{while} \;\; \varphi(z) \to \infty \;\text{ as } z \to -\infty.$$

As a few different examples, here are three loss functions that we will see either now or later in the class, all of which are commonly used in machine learning.

(i) The *logistic loss* uses

$$\varphi_{\text{logistic}}(z) = \log(1 + e^{-z})$$

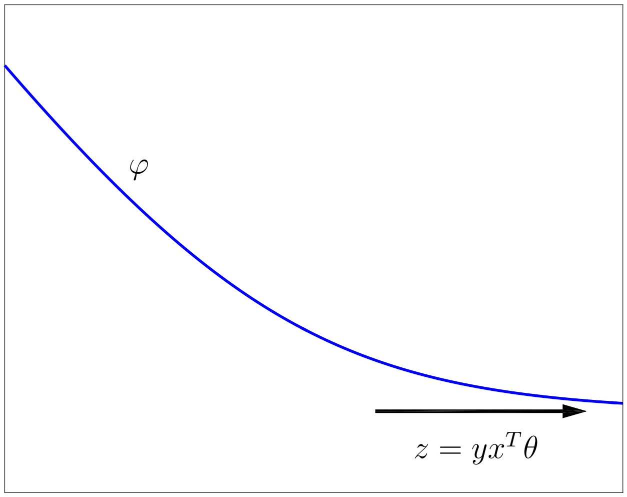


Figure 1: The rough shape of loss we desire: the loss is convex and continuous, and tends to zero as the margin $z = yx^T\theta \to \infty$.

(ii) The *hinge loss* uses

$$\varphi_{\text{hinge}}(z) = [1 - z]_+ = \max\{1 - z, 0\}$$

(iii) The *exponential loss* uses

$$\varphi_{\exp}(z) = e^{-z}.$$

In Figure 2, we plot each of these losses against the margin $z = yx^T\theta$, noting that each goes to zero as the margin grows, and each tends to $+\infty$ as the margin becomes negative. The different loss functions lead to different machine learning procedures; in particular, the logistic loss $\varphi_{\text{logistic}}$ is logistic regression, the hinge loss φ_{hinge} gives rise to so-called *support vector machines*, and the exponential loss gives rise to the classical version of *boosting*, both of which we will explore in more depth later in the class.

2 Logistic regression

With this general background in place, we now we give a complementary view of logistic regression to that in Andrew Ng's lecture notes. When we

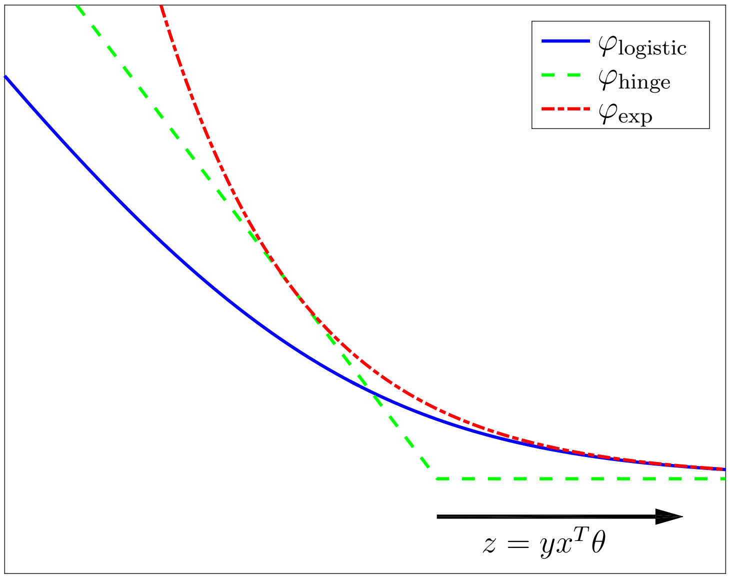


Figure 2: The three margin-based loss functions logistic loss, hinge loss, and exponential loss.

use binary labels $y \in \{-1, 1\}$, it is possible to write logistic regression more compactly. In particular, we use the logistic loss

$$\varphi_{\text{logistic}}(yx^T\theta) = \log\left(1 + \exp(-yx^T\theta)\right),$$

and the *logistic regression* algorithm corresponds to choosing θ that minimizes

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \varphi_{\text{logistic}}(y^{(i)}\theta^T x^{(i)}) = \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + \exp(-y^{(i)}\theta^T x^{(i)})\right). \quad (3)$$

Roughly, we hope that choosing θ to minimize the average logistic loss will yield a θ for which $y^{(i)}\theta^T x^{(i)} > 0$ for most (or even all!) of the training examples.

2.1 Probabilistic intrepretation

Similar to the linear regression (least-squares) case, it is possible to give a probabilistic interpretation of logistic regression. To do this, we define the

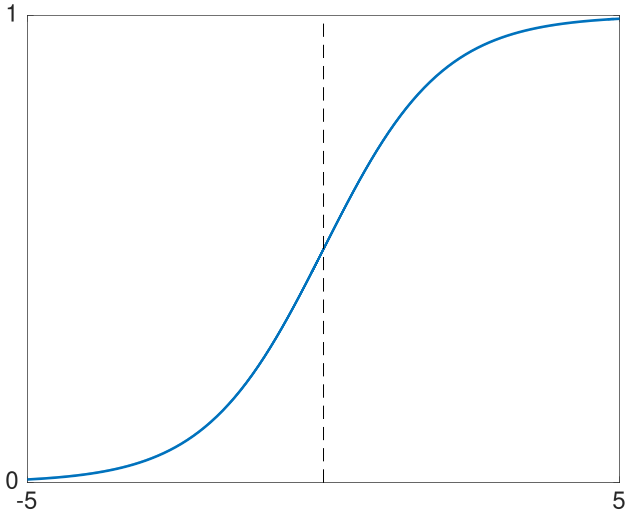


Figure 3: Sigmoid function

sigmoid function (also often called the *logistic function*)

$$g(z) = \frac{1}{1 + e^{-z}},$$

which is plotted in Fig. 3. In particular, the sigmoid function satisfies

$$g(z) + g(-z) = \frac{1}{1 + e^{-z}} + \frac{1}{1 + e^{z}} = \frac{e^{z}}{1 + e^{z}} + \frac{1}{1 + e^{z}} = 1,$$

so we can use it to define a probability model for binary classification. In particular, for $y \in \{-1, 1\}$, we define the *logistic model* for classification as

$$p(Y = y \mid x; \theta) = g(yx^T\theta) = \frac{1}{1 + e^{-yx^T\theta}}. \tag{4}$$

For intepretation, we see that if the margin $yx^T\theta$ is large—bigger than, say, 5 or so—then $p(Y = y \mid x; \theta) = g(yx^T\theta) \approx 1$, that is, we assign nearly probability 1 to the event that the label is y. Conversely, if $yx^T\theta$ is quite negative, then $p(Y = y \mid x; \theta) \approx 0$.

By redefining our hypothesis class as

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

then we see that the likelihood of the training data is

$$L(\theta) = \prod_{i=1}^{m} p(Y = y^{(i)} \mid x^{(i)}; \theta) = \prod_{i=1}^{m} h_\theta(y^{(i)} x^{(i)}),$$

and the log-likelihood is precisely

$$\ell(\theta) = \sum_{i=1}^{m} \log h_\theta(y^{(i)} x^{(i)}) = -\sum_{i=1}^{m} \log\left(1 + e^{-y^{(i)} \theta^T x^{(i)}}\right) = -mJ(\theta),$$

where $J(\theta)$ is exactly the logistic regression risk from Eq. (3). That is, maximum likelihood in the logistic model (4) is the same as minimizing the average logistic loss, and we arrive at logistic regression again.

2.2 Gradient descent methods

The final part of logistic regression is to actually fit the model. As is usually the case, we consider gradient-descent-based procedures for performing this minimization. With that in mind, we now show how to take derivatives of the logistic loss. For $\varphi_{\text{logistic}}(z) = \log(1 + e^{-z})$, we have the one-dimensional derivative

$$\frac{d}{dz}\varphi_{\text{logistic}}(z) = \varphi'_{\text{logistic}}(z) = \frac{1}{1 + e^{-z}} \cdot \frac{d}{dz} e^{-z} = -\frac{e^{-z}}{1 + e^{-z}} = -\frac{1}{1 + e^{z}} = -g(-z),$$

where g is the sigmoid function. Then we apply the chain rule to find that for a single training example (x, y), we have

$$\frac{\partial}{\partial \theta_k}\varphi_{\text{logistic}}(yx^T\theta) = -g(-yx^T\theta)\frac{\partial}{\partial \theta_k}(yx^T\theta) = -g(-yx^T\theta)yx_k.$$

Thus, a stochastic gradient procedure for minimization of $J(\theta)$ iteratively performs the following for iterations $t = 1, 2, \ldots$, where α_t is a stepsize at time t:

1. Choose an example $i \in \{1, \ldots, m\}$ uniformly at random

2. Perform the gradient update

$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - \alpha_t \cdot \nabla_\theta \varphi_{\text{logistic}}(y^{(i)} x^{(i)^T} \theta^{(t)}) \\ &= \theta^{(t)} + \alpha_t g(-y^{(i)} x^{(i)^T} \theta^{(t)}) y^{(i)} x^{(i)} = \theta^{(t)} + \alpha_t h_{\theta^{(t)}}(-y^{(i)} x^{(i)}) y^{(i)} x^{(i)}.\end{aligned}$$

This update is intuitive: if our current hypothesis $h_{\theta^{(t)}}$ assigns probability close to 1 for the *incorrect* label $-y^{(i)}$, then we try to reduce the loss by moving θ in the direction of $y^{(i)}x^{(i)}$. Conversely, if our current hypothesis $h_{\theta^{(t)}}$ assigns probability close to 0 for the incorrect label $-y^{(i)}$, the update essentially does nothing.

CS229 Supplemental Lecture notes

John Duchi

1 General loss functions

Building off of our interpretations of supervised learning as (1) choosing a representation for our problem, (2) choosing a loss function, and (3) minimizing the loss, let us consider a slightly more general formulation for supervised learning. In the supervised learning settings we have considered thus far, we have input data $x \in \mathbb{R}^n$ and targets y from a space $\mathcal{Y}$. In linear regression, this corresponded to $y \in \mathbb{R}$, that is, $\mathcal{Y} = \mathbb{R}$, for logistic regression and other binary classification problems, we had $y \in \mathcal{Y} = \{-1, 1\}$, and for multiclass classification we had $y \in \mathcal{Y} = \{1, 2, \ldots, k\}$ for some number k of classes.

For each of these problems, we made predictions based on $\theta^T x$ for some vector θ, and we constructed a loss function $\mathsf{L} : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$, where $\mathsf{L}(\theta^T x, y)$ measures the loss we suffer for predicting $\theta^T x$. For logistic regression, we use the logistic loss

$$\mathsf{L}(z, y) = \log(1 + e^{-yz}) \;\text{ or }\; \mathsf{L}(\theta^T x, y) = \log(1 + e^{-y\theta^T x}).$$

For linear regression we use the squared error

$$\mathsf{L}(z, y) = \frac{1}{2}(z - y)^2 \;\text{ or }\; \mathsf{L}(\theta^T x, y) = \frac{1}{2}(\theta^T x - y)^2.$$

For multiclass classification, we had a slight variant, where we let $\Theta = [\theta_1 \;\cdots\; \theta_k]$ for $\theta_i \in \mathbb{R}^n$, and used the loss $\mathsf{L} : \mathbb{R}^k \times \{1, \ldots, k\} \to \mathbb{R}$

$$\mathsf{L}(z, y) = \log\left(\sum_{i=1}^k \exp(z_i - z_y)\right) \;\text{ or }\; \mathsf{L}(\Theta^T x, y) = \log\left(\sum_{i=1}^k \exp(x^T(\theta_i - \theta_y))\right),$$

the idea being that we wish to have $\theta_y^T x > \theta_i^T x$ for all $i \neq k$. Given a training set of pairs $\{x^{(i)}, y^{(i)}\}$, choose θ by minimizing the empirical risk

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \mathsf{L}(\theta^T x^{(i)}, y^{(i)}). \tag{1}$$

2 The representer theorem

Let us consider a slight variant of choosing θ to minimize the risk (1). In many situations—for reasons that we will study more later in the class—it is useful to add *regularization* to the risk J. We add regularization for many reasons: often, it makes problem (1) easier to solve numerically, and also it can allow the θ we get out of minimizing the risk (1) able to generalize better to unseen data. Generally, regularization is taken to be of the form $r(\theta) = \|\theta\|$ or $r(\theta) = \|\theta\|^2$ for some norm $\|\cdot\|$ on $\mathbb{R}^n$. The most common choice is so-called ℓ_2-regularization, in which we choose

$$r(\theta) = \frac{\lambda}{2} \|\theta\|_2^2 ,$$

where we recall that $\|\theta\|_2 = \sqrt{\theta^T \theta}$ is the Euclidean norm, or length, of the vector θ. This gives rise to the *regularized risk*, which is

$$J_\lambda(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathsf{L}(\theta^T x^{(i)}, y^{(i)}) + \frac{\lambda}{2} \|\theta\|_2^2 . \tag{2}$$

Let us consider the structure of any θ that minimizes the risk (2). We assume—as we often do—that for each fixed target value $y \in \mathcal{Y}$, the function $\mathsf{L}(z, y)$ is convex in z. (This is the case for linear regression and binary and multiclass logistic regression, as well as a number of other losses we will consider.) It turns out that under these assumptions, we may always write the solutions to the problem (2) as a *linear combination* of the input variables $x^{(i)}$. More precisely, we have the following theorem, known as the *representer theorem*.

Theorem 2.1. *Suppose in the definition of the regularized risk* (2) *that* $\lambda \geq 0$. *Then there is a minimizer of the regularized risk* (2) *that can be written*

$$\theta = \sum_{i=1}^{m} \alpha_i x^{(i)}$$

for some real-valued weights α_i.

Proof For intuition, we give a proof of the result in the case that $\mathsf{L}(z, y)$, when viewed as a function of z, is differentiable and $\lambda > 0$. In Appendix A, we present a more general statement of the theorem as well as a rigorous proof.

Let $\mathsf{L}'(z, y) = \frac{\partial}{\partial z}\mathsf{L}(z, y)$ denote the derivative of the loss with respect to z. Then by the chain rule, we have the gradient identity

$$\nabla_\theta \mathsf{L}(\theta^T x, y) = \mathsf{L}'(\theta^T x, y)x \quad \text{and} \quad \nabla_\theta \frac{1}{2}\left\|\theta\right\|_2^2 = \theta,$$

where ∇_θ denotes taking a gradient with respect to θ. As the risk must have 0 gradient at all stationary points (including the minimizer), we can write

$$\nabla J_\lambda(\theta) = \frac{1}{m}\sum_{i=1}^m \mathsf{L}'(\theta^T x^{(i)}, y^{(i)})x^{(i)} + \lambda\theta = \vec{0}.$$

In particular, letting $w_i = \mathsf{L}'(\theta^T x^{(i)}, y^{(i)})$, as $\mathsf{L}'(\theta^T x^{(i)}, y^{(i)})$ is a scalar (which depends on θ, but no matter what θ is, w_i is still a real number), we have

$$\theta = -\frac{1}{\lambda}\sum_{i=1}^n w_i x^{(i)}.$$

Set $\alpha_i = -\frac{w_i}{\lambda}$ to get the result. □

3 Nonlinear features and kernels

Based on the representer theorem, Theorem 2.1, we see that we can always write the vector θ as a linear combination of the data $\{x^{(i)}\}_{i=1}^m$. Importantly, this means we can *always* make predictions

$$\theta^T x = x^T\theta = \sum_{i=1}^m \alpha_i x^T x^{(i)}.$$

That is, in *any* learning algorithm, we may can replace all appearances of $\theta^T x$ with $\sum_{i=1}^m \alpha_i {x^{(i)}}^T x$, and then minimize directly over $\alpha \in \mathbb{R}^m$.

Let us consider this idea in somewhat more generality. In our discussion of linear regression, we had a problem in which the input x was the living area of a house, and we considered performing regression using the features x, x^2 and x^3 (say) to obtain a cubic function. To distinguish between these two sets of variables, we'll call the "original" input value the input **attributes** of a problem (in this case, x, the living area). When that is mapped to some new set of quantities that are then passed to the learning algorithm, we'll call those new quantities the input **features**. (Unfortunately, different authors use different terms to describe these two things, but we'll try to use this terminology consistently in these notes.) We will also let ϕ denote the **feature mapping**, which maps from the attributes to the features. For instance, in our example, we had

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}.$$

Rather than applying a learning algorithm using the original input attributes x, we may instead want to learn using some features $\phi(x)$. To do so, we simply need to go over our previous algorithm, and replace x everywhere in it with $\phi(x)$.

Since the algorithm can be written entirely in terms of the inner products $\langle x, z\rangle$, this means that we would replace all those inner products with $\langle \phi(x), \phi(z)\rangle$. Specificically, given a feature mapping ϕ, we define the corresponding **kernel** to be

$$K(x, z) = \phi(x)^T \phi(z).$$

Then, everywhere we previously had $\langle x, z\rangle$ in our algorithm, we could simply replace it with $K(x, z)$, and our algorithm would now be learning using the features ϕ. Let us write this out more carefully. We saw by the representer theorem (Theorem 2.1) that we can write $\theta = \sum_{i=1}^m \alpha_i \phi(x^{(i)})$ for some weights α_i. Then we can write the (regularized) risk

$$\begin{aligned}
J_\lambda(\theta) &= J_\lambda(\alpha) \\
&= \frac{1}{m}\sum_{i=1}^m \mathsf{L}\left(\phi(x^{(i)})^T \sum_{j=1}^m \alpha_j \phi(x^{(j)}), y^{(i)}\right) + \frac{\lambda}{2}\left\|\sum_{i=1}^m \alpha_i \phi(x^{(i)})\right\|_2^2 \\
&= \frac{1}{m}\sum_{i=1}^m \mathsf{L}\left(\sum_{j=1}^m \alpha_j \phi(x^{(i)})^T \phi(x^{(j)}), y^{(i)}\right) + \frac{\lambda}{2}\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j \phi(x^{(i)})^T \phi(x^{(j)}) \\
&= \frac{1}{m}\sum_{i=1}^m \mathsf{L}\left(\sum_{j=1}^m \alpha_j K(x^{(i)}, x^{(j)}), y^{(i)}\right) + \frac{\lambda}{2}\sum_{i,j} \alpha_i \alpha_i K(x^{(i)}, x^{(j)}).
\end{aligned}$$

That is, we can write the entire loss function to be minimized in terms of the kernel matrix

$$K = [K(x^{(i)}, x^{(j)})]_{i,j=1}^{m} \in \mathbb{R}^{m \times m}.$$

Now, given ϕ, we could easily compute $K(x, z)$ by finding $\phi(x)$ and $\phi(z)$ and taking their inner product. But what's more interesting is that often, $K(x, z)$ may be very inexpensive to calculate, even though $\phi(x)$ itself may be very expensive to calculate (perhaps because it is an extremely high dimensional vector). In such settings, by using in our algorithm an efficient way to calculate $K(x, z)$, we can learn in the high dimensional feature space space given by ϕ, but without ever having to explicitly find or represent vectors $\phi(x)$. As a few examples, some kernels (corresponding to infinite-dimensional vectors ϕ) include

$$K(x, z) = \exp\left(-\frac{1}{2\tau^2} \|x - z\|_2^2\right),$$

known as the Gaussian or Radial Basis Function (RBF) kernel and applicable to data in any dimension, or the min-kernel (applicable when $x \in \mathbb{R}$, defined by

$$K(x, z) = \min\{x, z\}.$$

See also the lecture notes on Support Vector Machines (SVMs) for more on these kernel machines.

4 Stochastic gradient descent for kernelized machine learning

If we let $K \in \mathbb{R}^{m \times m}$ denote the kernel matrix, and for shorthand define the vectors

$$K^{(i)} = \begin{bmatrix} K(x^{(i)}, x^{(1)}) \\ K(x^{(i)}, x^{(2)}) \\ \vdots \\ K(x^{(i)}, x^{(m)}) \end{bmatrix},$$

so that $K = [K^{(1)} \; K^{(2)} \; \cdots \; K^{(m)}]$, then we may write the regularized risk in a consise form as

$$J_\lambda(\alpha) = \frac{1}{m} \sum_{i=1}^{m} \mathsf{L}(K^{(i)^T} \alpha, y^{(i)}) + \frac{\lambda}{2} \alpha^T K \alpha.$$

Now, let us consider taking a stochastic gradient of the above risk J_λ. That is, we wish to construct a (simple to compute) random vector whose expectation is $\nabla J_\lambda(\alpha)$, which does not have too much variance. To do so, we first compute the gradient of $J_\lambda(\alpha)$ on its own. We calculate the gradient of individual loss terms by noting that

$$\nabla_\alpha \mathsf{L}({K^{(i)}}^T\alpha, y^{(i)}) = \mathsf{L}'({K^{(i)}}^T\alpha, y^{(i)})K^{(i)},$$

while

$$\nabla_\alpha\left[\frac{\lambda}{2}\alpha^T K\alpha\right] = \lambda K\alpha = \lambda\sum_{i=1}^m K^{(i)}\alpha_i.$$

Thus, we have

$$\nabla_\alpha J_\lambda(\alpha) = \frac{1}{m}\sum_{i=1}^m \mathsf{L}'({K^{(i)}}^T\alpha, y^{(i)})K^{(i)} + \lambda\sum_{i=1}^m K^{(i)}\alpha_i.$$

Thus, if we choose a random index $i \in \{1, \ldots, m\}$, we have that

$$\mathsf{L}'({K^{(i)}}^T\alpha, y^{(i)})K^{(i)} + m\lambda K^{(i)}\alpha_i$$

is a stochastic gradient for $J_\lambda(\alpha)$. This gives us a stochastic gradient procedure for Kernel supervised learning problems, shown in Figure 1. One minor remark is in order regarding Algorithm 1: because we multiply the $\lambda K^{(i)}\alpha_i$ term by m to keep the gradient unbiased, it is important that $\lambda > 0$ not be too large, as the algorithm can be a bit unstable otherwise. In addition, a common choice of stepsize is to use $\eta_t = 1/\sqrt{t}$, or a constant multiple thereof.

Input: A loss function L, kernel matrix $K = [K^{(1)} \; \cdots \; K^{(m)}]$, and labels $\{y^{(i)}\}_{i=1}^m$, and sequence of positive stepsizes $\eta_1, \eta_2, \eta_3, \ldots$

Iterate for $t = 1, 2, \ldots$

(i) Choose index $i \in \{1, \ldots, m\}$ uniformly at random

(ii) Update

$$\alpha := \alpha - \eta_t\left[\mathsf{L}'({K^{(i)}}^T\alpha, y^{(i)})K^{(i)} + m\lambda K^{(i)}\alpha_i\right].$$

Figure 1: Stochastic gradient descent for kernel supervised learning problems.

5 Support vector machines

Now we discuss (one approach) to Support Vector Machines (SVM)s, which apply to binary classification problems with labels $y \in \{-1, 1\}$, and a particular choice of loss function L. In particular, for the support vector machine, we use the margin-based loss function

$$\mathsf{L}(z, y) = [1 - yz]_+ = \max\{0, 1 - yz\}. \tag{3}$$

So, in a sense, SVMs are nothing but a special case of the general theoretical results we have described above. In particular, we have the empirical regularized risk

$$J_\lambda(\alpha) = \frac{1}{m}\sum_{i=1}^{m}\left[1 - y^{(i)}K^{(i)^T}\alpha\right]_+ + \frac{\lambda}{2}\alpha^T K\alpha,$$

where the matrix $K = [K^{(1)} \ \cdots \ K^{(m)}]$ is defined by $K_{ij} = K(x^{(i)}, x^{(j)})$.

In the lecture notes, you can see another way of deriving the support vector machine and a description of why we actually call them support vector machines.

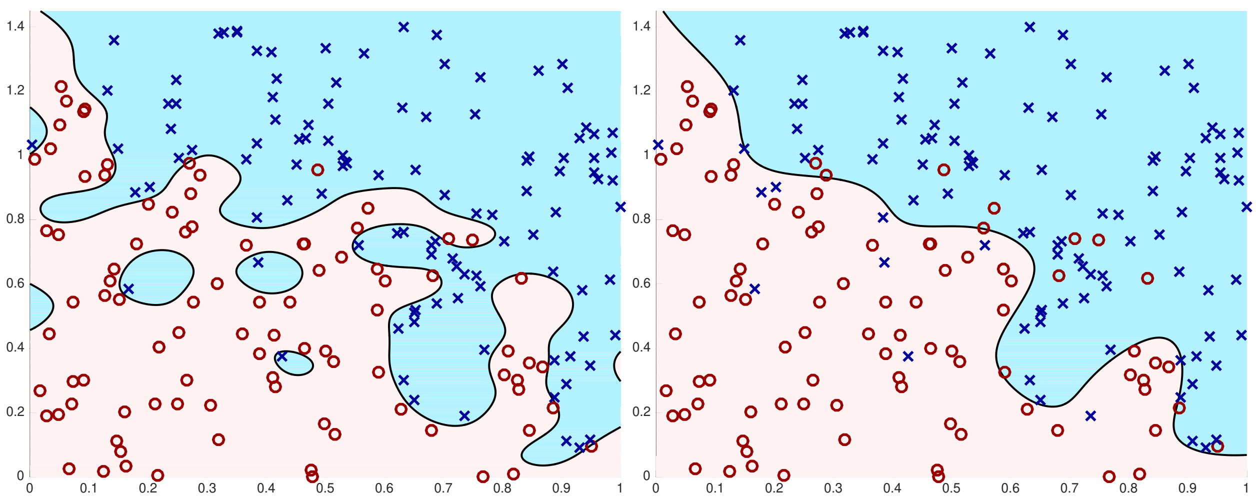
6 An example

In this section, we consider a particular example kernel, known as the Gaussian or Radial Basis Function (RBF) kernel. This kernel is defined by

$$K(x, z) = \exp\left(-\frac{1}{2\tau^2}\|x - z\|_2^2\right), \tag{4}$$

where $\tau > 0$ is a parameter controlling the *bandwidth* of the kernel. Intuitively, for τ very small, we will have $K(x, z) \approx 0$ unless $x \approx z$, that is, x and z are very close, in which case we have $K(x, z) \approx 1$. However, for τ large, then we have a much smoother kernel function K. The feature function ϕ for this kernel is infinite dimensional.[1] That said, it is possible to gain some

[1] If you have seen characteristic functions or Fourier transforms, then you might recognize the RBF kernel as the Fourier transform of the Gaussian distribution with mean zero and variance τ^2. That is, in $\mathbb{R}^n$, let $W \sim \mathsf{N}(0, \tau^2 I_{n\times n})$, so that W has density



$\tau = .1$ $\tau = .2$

Figure 2: Small bandwidths τ for Gaussian kernel $K(x,z) = \exp(-\frac{1}{2\tau^2}\left\|x-z\right\|_2^2)$.

intuition for the kernel by considering the classifications it makes on a new example x: we have prediction

$$\sum_{i=1}^m K(x^{(i)},x)\alpha_i = \sum_{i=1}^m \exp\left(-\frac{1}{2\tau^2}\left\|x^{(i)}-x\right\|_2^2\right)\alpha_i,$$

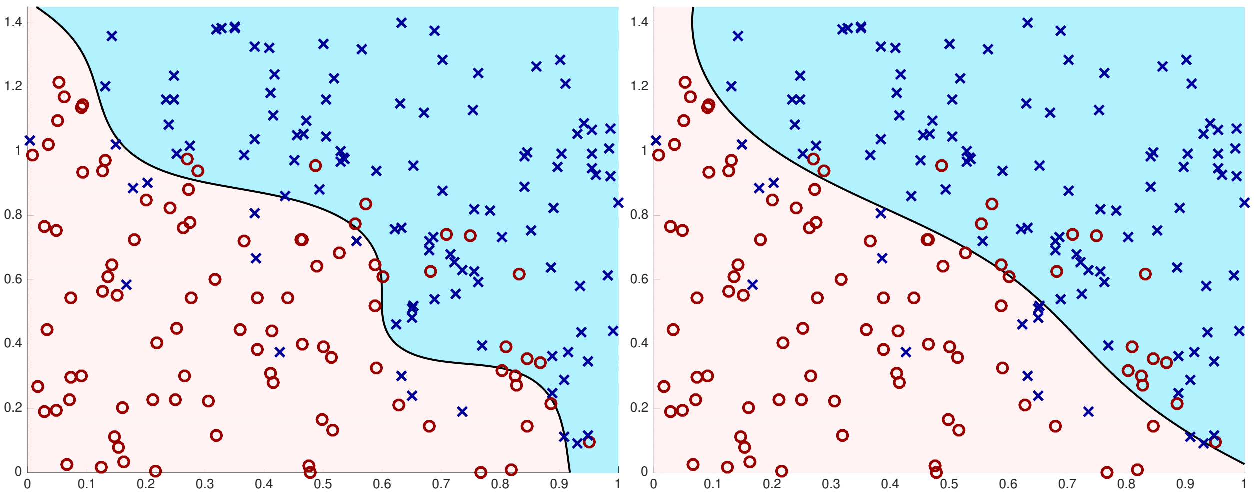
$p(w) = \frac{1}{(2\pi\tau^2)^{n/2}}\exp(-\frac{\|w\|_2^2}{2\tau^2})$. Let $i = \sqrt{-1}$ be the imaginary unit, then for any vector v we have

$$\begin{aligned}\mathbb{E}[\exp(iv^TW)] = \int \exp(iv^Tw)p(w)dw &= \int \frac{1}{(2\pi\tau^2)^{n/2}}\exp\left(iv^Tw - \frac{1}{2\tau^2}\left\|w\right\|_2^2\right)dw \\ &= \exp\left(-\frac{1}{2\tau^2}\left\|v\right\|_2^2\right).\end{aligned}$$

Thus, if we define the "vector" (really, function) $\phi(x,w) = e^{ix^Tw}$ and let a^* be the complex conjugate of $a \in \mathbb{C}$, then we have

$$\mathbb{E}[\phi(x,W)\phi(z,W)^*] = \mathbb{E}[e^{ix^TW}e^{-ix^TW}] = \mathbb{E}[\exp(iW^T(x-z))] = \exp\left(-\frac{1}{2\tau^2}\left\|x-z\right\|_2^2\right).$$

In particular, we see that $K(x,z)$ is the inner product in a space of functions that are integrable against $p(w)$.



$\tau = .4$ $\tau = .8$

Figure 3: Medium bandwidths τ for Gaussian kernel $K(x,z) = \exp(-\frac{1}{2\tau^2}\left\|x-z\right\|_2^2)$.

so that this becomes something like a weighting depending on *how close* x is to each $x^{(i)}$—that is, the contribution of weight α_i is multiplied by the similarity of x to $x^{(i)}$ as determined by the kernel function.

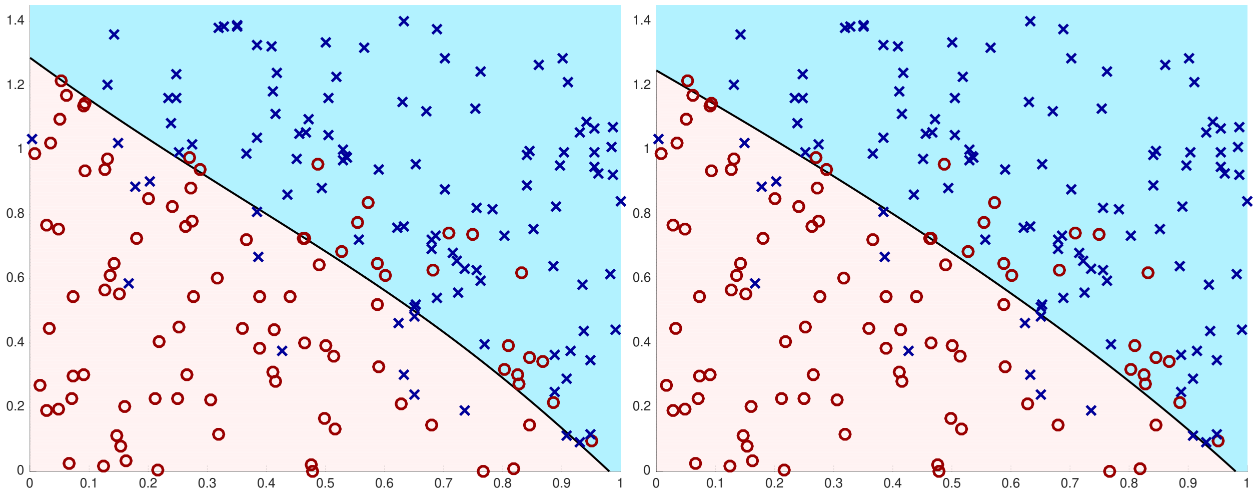
In Figures 2, 3, and 4, we show the results of training 6 different kernel classifiers by minimizing

$$J_\lambda(\alpha) = \sum_{i=1}^{m}\left[1 - y^{(i)}K^{(i)T}\alpha\right]_+ + \frac{\lambda}{2}\alpha^T K\alpha,$$

with $m = 200$ and $\lambda = 1/m$, for different values of τ in the kernel (4). We plot the training data (positive examples as blue `x`'s and negative examples as red `o`'s) as well as the decision surface of the resulting classifier. That is, we plot the lines defined by

$$\left\{x \in \mathbb{R}^2 \ : \ \sum_{i=1}^{m} K(x, x^{(i)})\alpha_i = 0\right\},$$

giving the regions where the learned classifier makes a prediction $\sum_{i=1}^m K(x,x^{(i)})\alpha_i > 0$ and $\sum_{i=1}^m K(x,x^{(i)})\alpha_i < 0$. From the figure, we see that for large τ, we have



$\tau = 1.6$ $\tau = 3.2$

Figure 4: Large bandwidths τ for Gaussian kernel $K(x, z) = \exp(-\frac{1}{2\tau^2} \|x - z\|_2^2)$.

a very simple classifier: it is nearly linear, while for $\tau = .1$, the classifier has substantial variability and is highly non-linear. For reference, in Figure 5, we plot the optimal classifier along with the training data; the optimal classifier minimizes the misclassification error given infinite training data.

A A more general representer theorem

In this section, we present a more general version of the representer theorem along with a rigorous proof. Let $r : \mathbb{R} \to \mathbb{R}$ be any non-decreasing function of its argument, and consider the regularized risk

$$J_r(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathsf{L}(x^{(i)T}\theta, y^{(i)}) + r(\|\theta\|_2). \tag{5}$$

Often, we take $r(t) = \frac{\lambda}{2}t^2$, which corresponds to the common choice of ℓ_2-regularization, but the next theorem makes clear that this is not necessary for the representer theorem. Indeed, we could simply take $r(t) = 0$ for all t, and the theorem still holds.

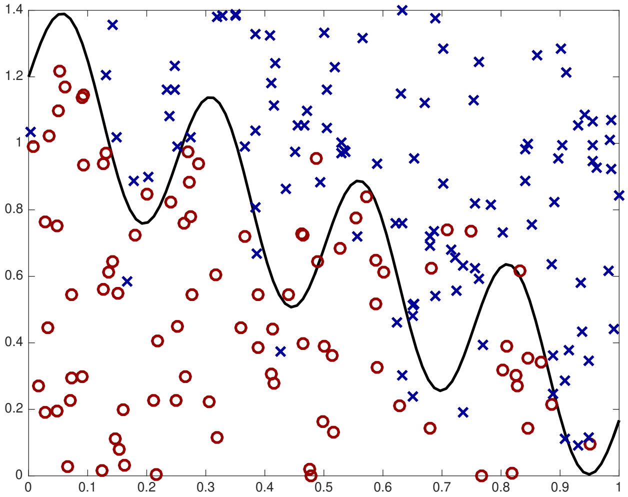


Figure 5: Optimal classifier along with training data.

Theorem A.1 (Representer theorem in $\mathbb{R}^n$)**.** *Let $\theta \in \mathbb{R}^n$ be any vector. Then there exists some $\alpha \in \mathbb{R}^m$ and $\theta^{(\alpha)} = \sum_{i=1}^m \alpha_i x^{(i)}$ such that*

$$J_r(\theta^{(\alpha)}) \leq J_r(\theta).$$

In particular, there is no loss of generality in always assuming we can write the optimization problem to minimize $J(\theta)$ by only considering θ in the span of the data.

Proof Our proof relies on some machinery from linear algebra, which allows us to keep it concise, but feel free to ask questions if it is too concise.

The vectors $\{x^{(i)}\}_{i=1}^m$ are in $\mathbb{R}^n$, and as a consequence there is some subspace V of $\mathbb{R}^n$ such that

$$V = \left\{ \sum_{i=1}^m \beta_i x^{(i)} : \beta_i \in \mathbb{R} \right\}.$$

Then V has an orthonormal basis $\{v_1, \ldots, v_{n_0}\}$ for vectors $v_i \in \mathbb{R}^n$, where the size (dimension) of the basis is $n_0 \leq n$. Thus we can write $V = \{\sum_{i=1}^{n_0} b_i v_i : b_i \in \mathbb{R}\}$, where we recall that orthonormality means that the vectors v_i satisfy $\|v_i\|_2 = 1$ and $v_i^T v_j = 0$ for $i \neq j$. There is also an orthogonal subspace $V^\perp = \{u \in \mathbb{R}^n : u^T v = 0 \text{ for all } v \in V\}$, which has an orthonormal basis of size $n_\perp = n - n_0 \geq 0$, which we write as $\{u_1, \ldots, u_{n_\perp}\} \subset \mathbb{R}^n$. By construction they satisfy $u_i^T v_j = 0$ for all i, j.

Because $\theta \in \mathbb{R}^n$, we know that we can write it uniquely as

$$\theta = \sum_{i=1}^{n_0} \nu_i v_i + \sum_{i=1}^{n_\perp} \mu_i u_i, \quad \text{where } \ \nu_i \in \mathbb{R} \text{ and } \mu_i \in \mathbb{R},$$

and the values μ, ν are unique. Now, we know that by definition of the space V as the span of $\{x^{(i)}\}_{i=1}^m$, there exists $\alpha \in \mathbb{R}^m$ such that

$$\sum_{i=1}^{n_0} \nu_i v_i = \sum_{i=1}^{m} \alpha_i x^{(i)},$$

so that we have

$$\theta = \sum_{i=1}^{m} \alpha_i x^{(i)} + \sum_{i=1}^{n_\perp} \mu_i u_i.$$

Define $\theta^{(\alpha)} = \sum_{i=1}^m \alpha_i x^{(i)}$. Now, for any data point $x^{(j)}$, we have

$$u_i^T x^{(j)} = 0 \ \text{ for all } i = 1, \ldots, n_\perp,$$

so that $u_i^T \theta^{(\alpha)} = 0$. As a consquence, we have

$$\|\theta\|_2^2 = \left\| \theta^{(\alpha)} + \sum_{i=1}^{n_\perp} \mu_i u_i \right\|_2^2 = \left\|\theta^{(\alpha)}\right\|_2^2 + \underbrace{2 \sum_{i=1}^{n_\perp} \mu_i u_i^T \theta^{(\alpha)}}_{=0} + \left\| \sum_{i=1}^{n_\perp} \mu_i u_i \right\|_2^2 \geq \left\|\theta^{(\alpha)}\right\|_2^2, \tag{6a}$$

and we also have

$$\theta^{(\alpha)T} x^{(i)} = \theta^T x^{(i)} \tag{6b}$$

for all points $x^{(i)}$.

That is, by using $\|\theta\|_2 \geq \left\|\theta^{(\alpha)}\right\|_2$ and equality (6b), we have

$$\begin{aligned} J_r(\theta) = \frac{1}{m} \sum_{i=1}^m \mathsf{L}(\theta^T x^{(i)}, y^{(i)}) + r(\|\theta\|_2) &\overset{(6b)}{=} \frac{1}{m} \sum_{i=1}^m \mathsf{L}(\theta^{(\alpha)T} x^{(i)}, y^{(i)}) + r(\|\theta\|_2) \\ &\overset{(6a)}{\geq} \frac{1}{m} \sum_{i=1}^m \mathsf{L}(\theta^{(\alpha)T} x^{(i)}, y^{(i)}) + r(\left\|\theta^{(\alpha)}\right\|_2) \\ &= J_r(\theta^{(\alpha)}). \end{aligned}$$

This is the desired result. □

CS229 Supplemental Lecture notes

John Duchi

1 Boosting

We have seen so far how to solve classification (and other) problems when we have a data representation already chosen. We now talk about a procedure, known as *boosting*, which was originally discovered by Rob Schapire, and further developed by Schapire and Yoav Freund, that automatically chooses feature representations. We take an optimization-based perspective, which is somewhat different from the original interpretation and justification of Freund and Schapire, but which lends itself to our approach of (1) choose a representation, (2) choose a loss, and (3) minimize the loss.

Before formulating the problem, we give a little intuition for what we are going to do. Roughly, the idea of boosting is to take a *weak learning* algorithm—any learning algorithm that gives a classifier that is slightly better than random—and transforms it into a *strong* classifier, which does much much better than random. To build a bit of intuition for what this means, consider a hypothetical digit recognition experiment, where we wish to distinguish 0s from 1s, and we receive images we must classify. Then a natural weak learner might be to take the middle pixel of the image, and if it is colored, call the image a 1, and if it is blank, call the image a 0. This classifier may be far from perfect, but it is likely better than random. Boosting procedures proceed by taking a collection of such weak classifiers, and then reweighting their contributions to form a classifier with much better accuracy than any individual classifier.

With that in mind, let us formulate the problem. Our interpretation of boosting is as a coordinate descent method in an infinite dimensional space, which—while it sounds complex—is not so bad as it seems. First, we assume we have raw input examples $x \in \mathbb{R}^n$ with labels $y \in \{-1, 1\}$, as is usual in binary classification. We also assume we have an infinite collection of *feature* functions $\phi_j : \mathbb{R}^n \to \{-1, 1\}$ and an infinite vector $\theta = [\theta_1 \; \theta_2 \; \cdots]^T$, but which we assume always has only a finite number of non-zero entries. For our classifier we use

$$h_\theta(x) = \operatorname{sign}\left(\sum_{j=1}^{\infty} \theta_j \phi_j(x)\right).$$

We will abuse notation, and define $\theta^T \phi(x) = \sum_{j=1}^{\infty} \theta_j \phi_j(x)$.

In boosting, one usually calls the features ϕ_j *weak hypotheses*. Given a training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, we call a vector $p = (p^{(1)}, \ldots, p^{(m)})$ a distribution on the examples if $p^{(i)} \geq 0$ for all i and

$$\sum_{i=1}^{m} p^{(i)} = 1.$$

Then we say that there is a *weak learner with margin* $\gamma > 0$ if for any distribution p on the m training examples there exists one weak hypothesis ϕ_j such that

$$\sum_{i=1}^{m} p^{(i)} 1\left\{y^{(i)} \neq \phi_j(x^{(i)})\right\} \leq \frac{1}{2} - \gamma. \tag{1}$$

That is, we assume that there is *some* classifier that does slightly better than random guessing on the dataset. The existence of a weak learning algorithm is an assumption, but the surprising thing is that we can transform any weak learning algorithm into one with perfect accuracy.

In more generality, we assume we have access to a *weak learner*, which is an algorithm that takes as input a distribution (weights) p on the training examples and returns a classifier doing slightly better than random. We will

(i) **Input:** A distribution $p^{(1)}, \ldots, p^{(m)}$ and training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ with $\sum_{i=1}^{m} p^{(i)} = 1$ and $p^{(i)} \geq 0$

(ii) **Return:** A weak classifier $\phi_j : \mathbb{R}^n \to \{-1, 1\}$ such that

$$\sum_{i=1}^{m} p^{(i)} 1\left\{y^{(i)} \neq \phi_j(x^{(i)})\right\} \leq \frac{1}{2} - \gamma.$$

Figure 1: Weak learning algorithm

show how, given access to a weak learning algorithm, boosting can return a classifier with perfect accuracy on the training data. (Admittedly, we would like the classifer to generalize well to unseen data, but for now, we ignore this issue.)

1.1 The boosting algorithm

Roughly, boosting begins by assigning each training example equal weight in the dataset. It then receives a weak-hypothesis that does well according to the current weights on training examples, which it incorporates into its current classification model. It then reweights the training examples so that examples on which it makes mistakes receive higher weight—so that the weak learning algorithm focuses on a classifier doing well on those examples—while examples with no mistakes receive lower weight. This repeated reweighting of the training data coupled with a weak learner doing well on examples for which the classifier currently does poorly yields classifiers with good performance.

The boosting algorithm specifically performs *coordinate descent* on the exponential loss for classification problems, where the objective is

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \exp(-y^{(i)} \theta^T \phi(x^{(i)})).$$

We first show how to compute the exact form of the coordinate descent update for the risk $J(\theta)$. Coordinate descent iterates as follows:

(i) Choose a coordinate $j \in \mathbb{N}$

(ii) Update θ_j to

$$\theta_j = \arg\min_{\theta_j} J(\theta)$$

while leaving θ_k identical for all $k \neq j$.

We iterate the above procedure until convergence.

In the case of boosting, the coordinate updates are not too challenging to derive because of the analytic convenience of the exp function. We now show how to derive the update. Suppose we wish to update coordinate k. Define

$$w^{(i)} = \exp\left(-y^{(i)} \sum_{j \neq k} \theta_j \phi_j(x^{(i)}) \right)$$

to be a weight, and note that optimizing coordinate k corresponds to minimizing

$$\sum_{i=1}^{m} w^{(i)} \exp(-y^{(i)}\phi_k(x^{(i)})\alpha)$$

in $\alpha = \theta_k$. Now, define

$$W^+ := \sum_{i:y^{(i)}\phi_k(x^{(i)})=1} w^{(i)} \quad \text{and} \quad W^- := \sum_{i:y^{(i)}\phi_k(x^{(i)})=-1} w^{(i)}$$

to be the sums of the weights of examples that ϕ_k classifies correctly and incorrectly, respectively. Then finding θ_k is the same as choosing

$$\alpha = \arg\min_{\alpha} \left\{ W^+ e^{-\alpha} + W^- e^{\alpha} \right\} = \frac{1}{2}\log\frac{W^+}{W^-}.$$

To see the final equality, take derivatives and set the resulting equation to zero, so we have $-W^+e^{-\alpha} + W^-e^{\alpha} = 0$. That is, $W^-e^{2\alpha} = W^+$, or $\alpha = \frac{1}{2}\log\frac{W^+}{W^-}$.

What remains is to choose the particular coordinate to perform coordinate descent on. We assume we have access to a weak-learning algorithm as in Figure 1, which at iteration t takes as input a distribution p on the training set and returns a weak hypothesis ϕ_t satisfying the margin condition (1). We present the full boosting algorithm in Figure 2. It proceeds in iterations $t = 1, 2, 3, \ldots$. We represent the set of hypotheses returned by the weak learning algorithm at time t by $\{\phi_1, \ldots, \phi_t\}$.

2 The convergence of Boosting

We now argue that the boosting procedure achieves 0 training error, and we also provide a rate of convergence to zero. To do so, we present a lemma that guarantees progress is made.

Lemma 2.1. *Let*

$$J(\theta^{(t)}) = \frac{1}{m}\sum_{i=1}^{m} \exp\Bigg(- y^{(i)} \sum_{\tau=1}^{t} \theta_\tau \phi_\tau(x^{(i)})\Bigg).$$

Then

$$J(\theta^{(t)}) \leq \sqrt{1 - 4\gamma^2}\, J(\theta^{(t-1)}).$$

```
For each iteration $t = 1, 2, \ldots$:
    (i) Define weights
        $$w^{(i)} = \exp\Big( - y^{(i)} \sum_{\tau=1}^{t-1} \theta_\tau \phi_\tau(x^{(i)}) \Big)$$
        and distribution $p^{(i)} = w^{(i)} / \sum_{j=1}^m w^{(j)}$
    (ii) Construct a weak hypothesis $\phi_t : \mathbb{R}^n \to \{-1, 1\}$ from the distribution $p = (p^{(1)}, \ldots, p^{(m)})$ on the training set
    (iii) Compute $W_t^+ = \sum_{i : y^{(i)} \phi_t(x^{(i)}) = 1} w^{(i)}$ and $W_t^- = \sum_{i : y^{(i)} \phi_t(x^{(i)}) = -1} w^{(i)}$ and set
        $$\theta_t = \frac{1}{2} \log \frac{W_t^+}{W_t^-}.$$
```

Figure 2: Boosting algorithm

As the proof of the lemma is somewhat involved and not the central focus of these notes—though it is important to know one's algorithm will converge!—we defer the proof to Appendix A.1. Let us describe how it guarantees convergence of the boosting procedure to a classifier with zero training error.

We initialize the procedure at $\theta^{(0)} = \vec{0}$, so that the initial empirical risk $J(\theta^{(0)}) = 1$. Now, we note that for any θ, the misclassification error satisfies

$$1\left\{\text{sign}(\theta^T \phi(x)) \neq y\right\} = 1\left\{y\theta^T \phi(x) \leq 0\right\} \leq \exp\left(-y\theta^T \phi(x)\right)$$

because $e^z \geq 1$ for all $z \geq 0$. Thus, we have that the misclassification error rate has upper bound

$$\frac{1}{m} \sum_{i=1}^m 1\left\{\text{sign}(\theta^T \phi(x^{(i)})) \neq y^{(i)}\right\} \leq J(\theta),$$

and so if $J(\theta) < \frac{1}{m}$ then the vector θ makes *no* mistakes on the training data. After t iterations of boosting, we find that the empirical risk satisfies

$$J(\theta^{(t)}) \leq (1 - 4\gamma^2)^{\frac{t}{2}} J(\theta^{(0)}) = (1 - 4\gamma^2)^{\frac{t}{2}}.$$

To find how many iterations are required to guarantee $J(\theta^{(t)}) < \frac{1}{m}$, we take logarithms to find that $J(\theta^{(t)}) < 1/m$ if

$$\frac{t}{2}\log(1-4\gamma^2) < \log\frac{1}{m}, \quad \text{or} \quad t > \frac{2\log m}{-\log(1-4\gamma^2)}.$$

Using a first order Taylor expansion, that is, that $\log(1-4\gamma^2) \le -4\gamma^2$, we see that if the number of rounds of boosting—the number of weak classifiers we use—satisfies

$$t > \frac{\log m}{2\gamma^2} \ge \frac{2\log m}{-\log(1-4\gamma^2)},$$

then $J(\theta^{(t)}) < \frac{1}{m}$.

3 Implementing weak-learners

One of the major advantages of boosting algorithms is that they automatically generate features from raw data for us. Moreover, because the weak hypotheses always return values in $\{-1, 1\}$, there is no need to normalize features to have similar scales when using learning algorithms, which in practice can make a large difference. Additionally, and while this is not theoretically well-understood, many types of weak-learning procedures introduce non-linearities intelligently into our classifiers, which can yield much more expressive models than the simpler linear models of the form $\theta^T x$ that we have seen so far.

3.1 Decision stumps

There are a number of strategies for weak learners, and here we focus on one, known as *decision stumps*. For concreteness in this description, let us suppose that the input variables $x \in \mathbb{R}^n$ are real-valued. A decision stump is a function f, which is parameterized by a threshold s and index $j \in \{1, 2, \ldots, n\}$, and returns

$$\phi_{j,s}(x) = \operatorname{sign}(x_j - s) = \begin{cases} 1 & \text{if } x_j \ge s \\ -1 & \text{otherwise.} \end{cases} \tag{2}$$

These classifiers are simple enough that we can fit them efficiently even to a weighted dataset, as we now describe.

Indeed, a decision stump weak learner proceeds as follows. We begin with a distribution—set of weights $p^{(1)}, \ldots, p^{(m)}$ summing to 1—on the training set, and we wish to choose a decision stump of the form (2) to minimize the error on the training set. That is, we wish to find a threshold $s \in \mathbb{R}$ and index j such that

$$\widehat{\mathrm{Err}}(\phi_{j,s}, p) = \sum_{i=1}^{m} p^{(i)} 1\left\{\phi_{j,s}(x^{(i)}) \neq y^{(i)}\right\} = \sum_{i=1}^{m} p^{(i)} 1\left\{y^{(i)}(x_j^{(i)} - s) \leq 0\right\} \quad (3)$$

is minimized. Naively, this could be an inefficient calculation, but a more intelligent procedure allows us to solve this problem in roughly $O(nm \log m)$ time. For each feature $j = 1, 2, \ldots, n$, we sort the raw input features so that

$$x_j^{(i_1)} \geq x_j^{(i_2)} \geq \cdots \geq x_j^{(i_m)}.$$

As the only values s for which the error of the decision stump can change are the values $x_j^{(i)}$, a bit of clever book-keeping allows us to compute

$$\sum_{i=1}^{m} p^{(i)} 1\left\{y^{(i)}(x_j^{(i)} - s) \leq 0\right\} = \sum_{k=1}^{m} p^{(i_k)} 1\left\{y^{(i_k)}(x_j^{(i_k)} - s) \leq 0\right\}$$

efficiently by incrementally modifying the sum in sorted order, which takes time $O(m)$ after we have already sorted the values $x_j^{(i)}$. (We do not describe the algorithm in detail here, leaving that to the interested reader.) Thus, performing this calcuation for each of the n input features takes total time $O(nm \log m)$, and we may choose the index j and threshold s that give the best decision stump for the error (3).

One *very* important issue to note is that by flipping the sign of the thresholded decision stump $\phi_{j,s}$, we achieve error $1 - \widehat{\mathrm{Err}}(\phi_{j,s}, p)$, that is, the error of

$$\widehat{\mathrm{Err}}(-\phi_{j,s}, p) = 1 - \widehat{\mathrm{Err}}(\phi_{j,s}, p).$$

(You should convince yourself that this is true.) Thus, it is important to also track the smallest value of $1 - \widehat{\mathrm{Err}}(\phi_{j,s}, p)$ over all thresholds, because this may be smaller than $\widehat{\mathrm{Err}}(\phi_{j,s}, p)$, which gives a better weak learner. Using this procedure for our weak learner (Fig. 1) gives the basic, but extremely useful, boosting classifier.

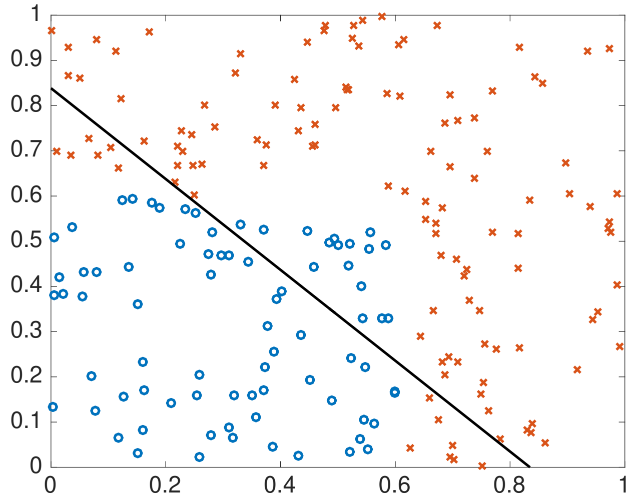


Figure 3: Best logistic regression classifier using the raw features $x \in \mathbb{R}^2$ (and a bias term $x_0 = 1$) for the example considered here.

3.2 Example

We now give an example showing the behavior of boosting on a simple dataset. In particular, we consider a problem with data points $x \in \mathbb{R}^2$, where the optimal classifier is

$$y = \begin{cases} 1 & \text{if } x_1 < .6 \text{ and } x_2 < .6 \\ -1 & \text{otherwise.} \end{cases} \tag{4}$$

This is a simple non-linear decision rule, but it is impossible for standard linear classifiers, such as logistic regression, to learn. In Figure 3, we show the best decision line that logistic regression learns, where positive examples are circles and negative examples are x's. It is clear that logistic regression is not fitting the data particularly well.

With boosted decision stumps, however, we can achieve a much better fit for the simple nonlinear classification problem (4). Figure 4 shows the boosted classifiers we have learned after different numbers of iterations of boosting, using a training set of size $m = 150$. From the figure, we see that the first decision stump is to threshold the feature x_1 at the value $s \approx .23$, that is, $\phi(x) = \text{sign}(x_1 - s)$ for $s \approx .23$.

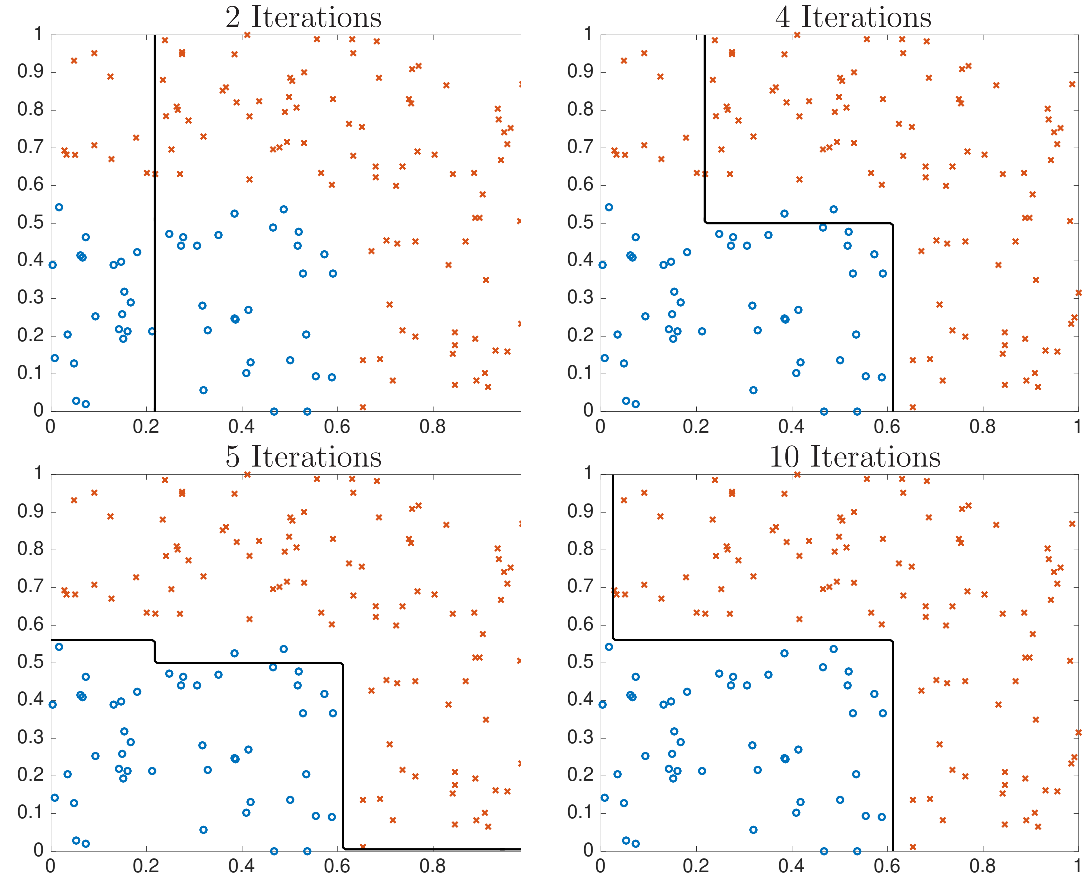


Figure 4: Boosted decision stumps after $t = 2, 4, 5,$ and 10 iterations of boosting, respectively.

3.3 Other strategies

There are a huge number of variations on the basic boosted decision stumps idea. First, we do not require that the input features x_j be real-valued. Some of them may be categorical, meaning that $x_j \in \{1, 2, \ldots, k\}$ for some k, in which case natural decision stumps are of the form

$$\phi_j(x) = \begin{cases} 1 & \text{if } x_j = l \\ -1 & \text{otherwise,} \end{cases}$$

as well as variants setting $\phi_j(x) = 1$ if $x_j \in C$ for some set $C \subset \{1, \ldots, k\}$ of categories.

Another natural variation is the *boosted decision tree*, in which instead of a single level decision for the weak learners, we consider conjuctions of features or trees of decisions. Google can help you find examples and information on these types of problems.

A Appendices

A.1 Proof of Lemma 2.1

We now return to prove the progress lemma. We prove this result by directly showing the relationship of the weights at time t to those at time $t - 1$. In particular, we note by inspection that

$$J(\theta^{(t)}) = \min_{\alpha}\{W_t^+ e^{-\alpha} + W_t^- e^{\alpha}\} = 2\sqrt{W_t^+ W_t^-}$$

while

$$J(\theta^{(t-1)}) = \frac{1}{m}\sum_{i=1}^{m} \exp\left(- y^{(i)} \sum_{\tau=1}^{t-1} \theta_\tau \phi_\tau(x^{(i)})\right) = W_t^+ + W_t^-.$$

We know by the weak-learning assumption that

$$\sum_{i=1}^{m} p^{(i)} 1\left\{y^{(i)} \neq \phi_t(x^{(i)})\right\} \leq \frac{1}{2} - \gamma, \quad \text{or} \quad \frac{1}{W_t^+ + W_t^-} \underbrace{\sum_{i:y^{(i)}\phi_t(x^{(i)})=-1} w^{(i)}}_{=W_t^-} \leq \frac{1}{2} - \gamma.$$

Rewriting this expression by noting that the sum on the right is nothing but W_t^-, we have

$$W_t^- \leq \left(\frac{1}{2} - \gamma\right)(W_t^+ + W_t^-), \quad \text{or} \quad W_t^+ \geq \frac{1+2\gamma}{1-2\gamma} W_t^-.$$

By substituting $\alpha = \frac{1}{2}\log\frac{1+2\gamma}{1-2\gamma}$ in the minimum defining $J(\theta^{(t)})$, we obtain

$$\begin{aligned}
J(\theta^{(t)}) &\leq W_t^+ \sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^- \sqrt{\frac{1+2\gamma}{1-2\gamma}} \\
&= W_t^+ \sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^- (1 - 2\gamma + 2\gamma) \sqrt{\frac{1+2\gamma}{1-2\gamma}} \\
&\leq W_t^+ \sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^- (1 - 2\gamma) \sqrt{\frac{1+2\gamma}{1-2\gamma}} + 2\gamma \frac{1-2\gamma}{1+2\gamma} \sqrt{\frac{1+2\gamma}{1-2\gamma}} W_t^+ \\
&= W_t^+ \left[\sqrt{\frac{1-2\gamma}{1+2\gamma}} + 2\gamma \sqrt{\frac{1-2\gamma}{1+2\gamma}} \right] + W_t^- \sqrt{1-4\gamma^2},
\end{aligned}$$

where we used that $W_t^- \leq \frac{1-2\gamma}{1+2\gamma} W_t^+$. Performing a few algebraic manipulations, we see that the final expression is equal to

$$\sqrt{1-4\gamma^2}(W_t^+ + W_t^-).$$

That is, $J(\theta^{(t)}) \leq \sqrt{1-4\gamma^2} J(\theta^{(t-1)})$. □

Convex Optimization Overview (cnt'd)

Chuong B. Do

November 29, 2009

During last week's section, we began our study of ***convex optimization***, the study of mathematical optimization problems of the form,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in C. \end{aligned} \tag{1}$$

In a convex optimization problem, $x \in \mathbb{R}^n$ is a vector known as the ***optimization variable***, $f : \mathbb{R}^n \to \mathbb{R}$ is a ***convex function*** that we want to minimize, and $C \subseteq \mathbb{R}^n$ is a ***convex set*** describing the set of feasible solutions. From a computational perspective, convex optimization problems are interesting in the sense that any locally optimal solution will always be guaranteed to be globally optimal. Over the last several decades, general purpose methods for solving convex optimization problems have become increasingly reliable and efficient.

In these lecture notes, we continue our foray into the field of convex optimization. In particular, we explore a powerful concept in convex optimization theory known as ***Lagrange duality***. We focus on the main intuitions and mechanics of Lagrange duality; in particular, we describe the concept of the Lagrangian, its relation to primal and dual problems, and the role of the Karush-Kuhn-Tucker (KKT) conditions in providing necessary and sufficient conditions for optimality of a convex optimization problem.

1 Lagrange duality

Generally speaking, the theory of Lagrange duality is the study of optimal solutions to convex optimization problems. As we saw previously in lecture, when minimizing a differentiable convex function $f(x)$ with respect to $x \in \mathbb{R}^n$, a necessary and sufficient condition for $x^* \in \mathbb{R}^n$ to be globally optimal is that $\nabla_x f(x^*) = \mathbf{0}$. In the more general setting of convex optimization problem with constraints, however, this simple optimality condition does not work. One primary goal of duality theory is to characterize the optimal points of convex programs in a mathematically rigorous way.

In these notes, we provide a brief introduction to Lagrange duality and its applications to generic differentiable convex optimization problems of the form,

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & h_i(x) = 0, \quad i = 1, \ldots, p, \end{array} \tag{OPT}$$

where $x \in \mathbb{R}^n$ is the ***optimization variable***, $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ are ***differentiable convex functions***[1], and $h_i : \mathbb{R}^n \to \mathbb{R}$ are ***affine functions***.[2]

1.1 The Lagrangian

In this section, we introduce an artificial-looking construct called the "Lagrangian" which is the basis of Lagrange duality theory. Given a convex constrained minimization problem of the form (OPT), the (generalized) ***Lagrangian*** is a function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, defined as

$$\mathcal{L}(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x). \tag{2}$$

Here, the first argument of the Lagrangian is a vector $x \in \mathbb{R}^n$, whose dimensionality matches that of the optimization variable in the original optimization problem; by convention, we refer to x as the ***primal variables*** of the Lagrangian. The second argument of the Lagrangian is a vector $\alpha \in \mathbb{R}^m$ with one variable α_i for each of the m convex inequality constraints in the original optimization problem. The third argument of the Lagrangian is a vector $\beta \in \mathbb{R}^p$, with one variable β_i for each of the p affine equality constraints in the original optimization problem. These elements of α and β are collectively known as the ***dual variables*** of the Lagrangian or ***Lagrange multipliers***.

Intuitively, the Lagrangian can be thought of as a modified version of the objective function to the original convex optimization problem (OPT) which accounts for each of the constraints. The Lagrange multipliers α_i and β_i can be thought of "costs" associated with violating different constraints. The key intuition behind the theory of Lagrange duality is the following:

> *For any convex optimization problem, there always exist settings of the dual variables such that the unconstrained minimum of the Lagrangian with respect to the primal variables (keeping the dual variables fixed) coincides with the solution of the original constrained minimization problem.*

We formalize this intuition when we describe the KKT conditions in Section 1.6.

[1] Recall that a function $f : S \to \mathbb{R}$ is convex if S is a convex set, and for any $x, y \in S$ and $\theta \in [0, 1]$, we have $f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$. A function f is concave if $-f$ is convex.

[2] Recall that an affine function is a function of the form $f(x) = a^T x + b$ for some $a \in \mathbb{R}^n, b \in \mathbb{R}$. Since the Hessian of an affine function is equal to the zero matrix (i.e., it is both positive semidefinite and negative semidefinite), an affine function is both convex and concave.

1.2 Primal and dual problems

To show the relationship between the Lagrangian and the original convex optimization problem (OPT), we introduce the notions of the "primal"and "dual problems" associated with a Lagrangian:

The primal problem

Consider the optimization problem,

$$\min_x \underbrace{\left[\max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \mathcal{L}(x,\alpha,\beta)\right]}_{\text{call this } \theta_{\mathcal{P}}(x)} = \min_x \theta_{\mathcal{P}}(x). \tag{P}$$

In the equation above, the function $\theta_{\mathcal{P}} : \mathbb{R}^n \to \mathbb{R}$ is called the ***primal objective***, and the unconstrained minimization problem on the right hand side is known as the ***primal problem***. Generally, we say that a point $x \in \mathbb{R}^n$ is ***primal feasible*** if $g_i(x) \leq 0, i = 1, \ldots, m$ and $h_i(x) = 0, i = 1, \ldots, p$. We typically use the vector $x^* \in \mathbb{R}^n$ to denote the solution of (P), and we let $p^* = \theta_{\mathcal{P}}(x^*)$ denote the optimal value of the primal objective.

The dual problem

By switching the order of the minimization and maximization above, we obtain an entirely *different* optimization problem,

$$\max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \underbrace{\left[\min_x \mathcal{L}(x,\alpha,\beta)\right]}_{\text{call this } \theta_{\mathcal{D}}(\alpha,\beta)} = \max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \theta_{\mathcal{D}}(\alpha,\beta). \tag{D}$$

Here, the function $\theta_{\mathcal{D}} : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is called the ***dual objective***, and the constrained maximization problem on the right hand side is known as the ***dual problem***. Generally, we say that (α, β) are ***dual feasible*** if $\alpha_i \geq 0, i = 1, \ldots, m$. We typically use the pair of vectors $(\alpha^*, \beta^*) \in \mathbb{R}^m \times \mathbb{R}^p$ to denote the solution of (D), and we let $d^* = \theta_{\mathcal{D}}(\alpha^*, \beta^*)$ denote the optimal value of the dual objective.

1.3 Interpreting the primal problem

First, observe that the primal objective, $\theta_{\mathcal{P}}(x)$, is a convex function of x.[3] To interpret the primal problem, note that

$$\begin{aligned}
\theta_{\mathcal{P}}(x) &= \max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \mathcal{L}(x,\alpha,\beta) && (4)\\
&= \max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \left[f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \right] && (5)\\
&= f(x) + \max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \left[\sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \right] && (6)
\end{aligned}$$

which follows from the fact that $f(x)$ does not depend on α or β. Considering only the bracketed term, notice that

- If any $g_i(x) > 0$, then maximizing the bracketed expression involves making the corresponding α_i an arbitrarily large positive number; however, if $g_i(x) \leq 0$, then the requirement that α_i be nonnegative means that the optimal setting of α_i to achieve the maximum is $\alpha_i = 0$, so that the maximum value is 0.

- Similarly, if any $h_i(x) \neq 0$, then maximizing the bracketed expression involves choosing the corresponding β_i to have the same sign as $h_i(x)$ and arbitrarily large magnitude; however, if $h_i(x) = 0$, then the maximum value is 0, independent of β_i.

Putting these two cases together, we see that if x is primal feasible (i.e., $g_i(x) \leq 0, i = 1,\ldots,m$ and $h_i(x) = 0, i = 1,\ldots,p$), then the maximum value of the bracketed expression is 0, but if any of the constraints are violated, then the maximum value is ∞. From this, we can write,

$$\theta_{\mathcal{P}}(x) = \underbrace{f(x)}_{\text{original objective}} + \underbrace{\begin{cases} 0 & \text{if } x \text{ is primal feasible} \\ \infty & \text{if } x \text{ is primal infeasible} \end{cases}}_{\text{barrier function for ``carving away'' infeasible solutions}} \tag{7}$$

Therefore, we can interpret the primal objective $\theta_{\mathcal{P}}(x)$ as a modified version of the convex objective function of the original problem (OPT), with the difference being that infeasible

[3] To see why, note that

$$\theta_{\mathcal{P}}(x) = \max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \mathcal{L}(x,\alpha,\beta) = \max_{\alpha,\beta:\alpha_i\geq 0,\forall i} \left[f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \right]. \tag{3}$$

Observe that each of the $g_i(x)$'s are convex functions in x, and since the α_i's are constrained to be nonnegative, then $\alpha_i g_i(x)$ is convex in x for each i. Similarly, each $\beta_i h_i(x)$ is convex in x (regardless of the sign of β_i) since $h_i(x)$ is linear. Since the sum of convex functions is always convex, we see that the quantity inside the brackets is a convex function of x. Finally, the maximum of a collection of convex functions is again a convex function (prove this for yourself!), so we can conclude that $\theta_{\mathcal{P}}(x)$ is a convex function of x.

solutions (i.e., x's for which some constraint is violated) have objective value ∞. Intuitively, we can consider

$$\max_{\alpha,\beta:\alpha_i\geq 0,\forall i}\left[\sum_{i=1}^m \alpha_i g_i(x) + \sum_{i=1}^p \beta_i h_i(x)\right] = \begin{cases} 0 & \text{if } x \text{ is feasible for (OPT)} \\ \infty & \text{if } x \text{ is infeasible for (OPT).} \end{cases} . \tag{8}$$

as a type of "barrier" function which prevents us from considering infeasible points as candidate solutions for the optimization problem.

1.4 Interpreting the dual problem

The dual objective, $\theta_{\mathcal{D}}(\alpha, \beta)$, is a concave function of α and β.[4] To interpret the dual problem, first we make the following observation:

Lemma 1. *If (α, β) are dual feasible, then $\theta_{\mathcal{D}}(\alpha, \beta) \leq p^*$*

Proof. Observe that

$$\begin{aligned} \theta_{\mathcal{D}}(\alpha,\beta) &= \min_x \mathcal{L}(x,\alpha,\beta) && (10) \\ &\leq \mathcal{L}(x^*,\alpha,\beta) && (11) \\ &= f(x^*) + \sum_{i=1}^m \alpha_i g_i(x^*) + \sum_{i=1}^p \beta_i h_i(x^*) && (12) \\ &\leq f(x^*) = p^*. && (13) \end{aligned}$$

Here, the first and third steps follow directly from the definitions of the dual objective function and the Lagrangian, respectively. The second step follows from the fact that the preceding expression minimized over possible values of x. The last step follows from the fact that x^* is primal feasible, (α, β) are dual feasible, and hence equation (8) implies that the latter two terms of (12) must be nonpositive. □

The lemma shows that that given any dual feasible (α, β), the dual objective $\theta_{\mathcal{D}}(\alpha, \beta)$ provides a lower bound on the optimal value p^* of the primal problem. Since the dual problem involves maximizing the dual objective over the space of all dual feasible (α, β), it follows that the dual problem can be seen as a search for the tightest possible lower bound on p^*. This gives rise to a property of any primal and dual optimization problem pairs known as ***weak duality***:

[4] To see why, note that

$$\theta_{\mathcal{D}}(\alpha,\beta) = \min_x \mathcal{L}(x,\alpha,\beta) = \min_x \left[f(x) + \sum_{i=1}^m \alpha_i g_i(x) + \sum_{i=1}^p \beta_i h_i(x)\right]. \tag{9}$$

Observe that for any fixed value of x, the quantity inside the brackets is an affine function of α and β, and hence concave. Since the minimum of a collection of concave functions is also concave, we can conclude that $\theta_{\mathcal{D}}(\alpha, \beta)$ is a concave function of α and β.

Lemma 2 (Weak Duality)**.** *For any pair of primal and dual problems, $d^* \leq p^*$.*

Clearly, weak duality is a consequence of Lemma 1 using (α^*, β^*) as the dual feasible point. For some primal/dual optimization problems, an even stronger result holds, known as ***strong duality***:

Lemma 3 (Strong Duality)**.** *For any pair of primal and dual problems which satisfy certain technical conditions called* ***constraint qualifications****, then $d^* = p^*$.*

A number of different "constraint qualifications" exist, of which the most commonly invoked constraint qualification is known as ***Slater's condition***: a primal/dual problem pair satisfy Slater's condition if there exists some feasible primal solution x for which all inequality constraints are strictly satisfied (i.e., $g_i(x) < 0, i = 1, \ldots, m$). In practice, nearly all convex problems satisfy some type of constraint qualification, and hence the primal and dual problems have the same optimal value.

1.5 Complementary slackness

One particularly interesting consequence of strong duality for convex optimization problems is a property known as ***complementary slackness*** (or KKT complementarity):

Lemma 4 (Complementary Slackness)**.** *If strong duality holds, then $\alpha_i^* g(x_i^*) = 0$ for each $i = 1, \ldots, m$.*

Proof. Suppose that strong duality holds. Largely copying the proof from the last section, observe that

$$\begin{aligned} p^* = d^* = \theta_{\mathcal{D}}(\alpha^*, \beta^*) &= \min_x \mathcal{L}(x, \alpha^*, \beta^*) && (14) \\ &\leq \mathcal{L}(x^*, \alpha^*, \beta^*) && (15) \\ &= f(x^*) + \sum_{i=1}^{m} \alpha_i^* g_i(x^*) + \sum_{i=1}^{p} \beta_i^* h_i(x^*) && (16) \\ &\leq f(x^*) = p^*. && (17) \end{aligned}$$

Since the first and last expressions in this sequence are equal, it follows that every intermediate expression is also equal. Subtracting the left half of (17) from (16), we see that

$$\sum_{i=1}^{m} \alpha_i^* g_i(x^*) + \sum_{i=1}^{p} \beta_i^* h_i(x^*) = 0. \qquad (18)$$

Recall, however, that each α_i^* is nonnegative, each $g_i(x^*)$ is nonpositive, and each $h_i(x^*)$ is zero due to the primal and dual feasibility of x^* and (α^*, β^*), respectively. As a consequence, (18) is a summation of all nonpositive terms which equals to zero. It readily follows that all individual terms in the summation must themselves be zero (for if not, there are no compensating positive terms in the summation which would allow the overall sum to remain zero). □

Complementary slackness can be written in many equivalent ways. One way, in particular, is the pair of conditions

$$\alpha_i^* > 0 \quad \Longrightarrow \quad g_i(x^*) = 0 \tag{19}$$

$$g_i(x^*) < 0 \quad \Longrightarrow \quad \alpha_i^* = 0. \tag{20}$$

In this form, we can see that whenever any α_i^* is strictly greater than zero, then this implies that the corresponding inequality constraint must hold with equality. We refer to this as an ***active constraint***. In the case of support vector machines (SVMs), active constraints are also known as ***support vectors***.

1.6 The KKT conditions

Finally, given everything so far, we can now characterize the optimal conditions for a primal dual optimization pair. We have the following theorem:

Theorem 1.1. *Suppose that $x^* \in \mathbb{R}^n$, $\alpha^* \in \mathbb{R}^m$ and $\beta^* \in \mathbb{R}^p$ satisfy the following conditions:*

1. *(Primal feasibility) $g_i(x^*) \leq 0, i = 1, \ldots, m$ and $h_i(x^*) = 0, i = 1, \ldots, p$,*

2. *(Dual feasibility) $\alpha_i^* \geq 0, i = 1, \ldots, m$,*

3. *(Complementary slackness) $\alpha_i^* g_i(x^*) = 0, i = 1, \ldots, m$, and*

4. *(Lagrangian stationarity) $\nabla_x \mathcal{L}(x^*, \alpha^*, \beta^*) = \mathbf{0}$.*

Then x^* is primal optimal and (α^*, β^*) are dual optimal. Furthermore, if strong duality holds, then any primal optimal x^* and dual optimal (α^*, β^*) must satisfy the conditions 1 through 4.

These conditions are known as the ***Karush-Kuhn-Tucker (KKT) conditions***.[5]

2 A simple duality example

As a simple application of duality, in this section, we will show how to form the dual problem for a simple convex optimization problem. Consider the convex optimization problem,

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & x_1^2 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_2 \geq 1. \end{aligned}$$

[5] Incidentally, the KKT theorem has an interesting history. The result was originally derived by Karush in his 1939 master's thesis but did not catch any attention until it was rediscovered in 1950 by two mathematicians Kuhn and Tucker. A variant of essentially the same result was also derived by John in 1948. For an interesting historical account of why so many iterations of this result went unnoticed for nearly a decade, see the paper,

> Kjeldsen, T.H. (2000) A contextualized historical analysis of the Kuhn-Tucker Theorem in nonlinear programming: the impact of World War II. *Historica Mathematics* **27**: 331-361.

First, we rewrite our optimization problem in standard form as

$$\begin{array}{rl} \underset{x\in\mathbb{R}^2}{\text{minimize}} & x_1^2 + x_2 \\ \text{subject to} & 4 - 2x_1 - x_2 \leq 0 \\ & 1 - x_2 \leq 0. \end{array}$$

The Lagrangian is then

$$\mathcal{L}(x,\alpha) = x_1^2 + x_2 + \alpha_1(4 - 2x_1 - x_2) + \alpha_2(1 - x_2), \tag{21}$$

and the objective of the dual problem is defined to be

$$\theta_{\mathcal{D}}(\alpha) = \min_x \mathcal{L}(x,\alpha)$$

To express the dual objective in a form which depends only on α (but not x), we first observe that the the Lagrangian is differentiable in x, and in fact, is separable in the two components x_1 and x_2 (i.e., we can minimize with respect to each separately).

To minimize with respect to x_1, observe that the Lagrangian is a strictly convex quadratic function of x_1 and hence the minimum with respect to x_1 can be found by setting the derivative to zero:

$$\frac{\partial}{\partial x_1}\mathcal{L}(x,\alpha) = 2x_1 - 2\alpha_1 = 0 \quad \Longrightarrow \quad x_1 = \alpha_1. \tag{22}$$

To minimize with respect to x_2, observe that the Lagrangian is an affine function of x_2, for which the linear coefficient is precisely the derivative of the Lagrangian coefficient with respect to x_2,

$$\frac{\partial}{\partial x_2}\mathcal{L}(x,\alpha) = 1 - \alpha_1 - \alpha_2 \tag{23}$$

If the linear coefficient is non-zero, then the objective function can be made arbitrarily small by choosing the x_2 to have the opposite sign of the linear coefficient and arbitrarily large magnitude. However, if the linear coefficient is zero, then the objective function does not depend on x_2.

Putting these observations together, we have

$$\begin{aligned} \theta_{\mathcal{D}}(\alpha) &= \min_x \mathcal{L}(x,\alpha) \\ &= \min_{x_2} \left[\alpha_1^2 + x_2 + \alpha_1(4 - 2\alpha_1 - x_2) + \alpha_2(1 - x_2)\right] \\ &= \min_{x_2} \left[-\alpha_1^2 + 4\alpha_1 + \alpha_2 + x_2(1 - \alpha_1 - \alpha_2)\right] \\ &= \begin{cases} -\alpha_1^2 + 4\alpha_1 + \alpha_2 & \text{if } 1 - \alpha_1 - \alpha_2 = 0 \\ -\infty & \text{otherwise} \end{cases} \end{aligned}$$

so the dual problem is given by:

$$\begin{aligned}\max_{\alpha \in \mathbb{R}^2}\text{imize} \quad & \theta_{\mathcal{D}}(\alpha) \\ \text{subject to} \quad & \alpha_1 \geq 0 \\ & \alpha_2 \geq 0.\end{aligned}$$

Finally, we can simplify the dual problem by observing making the dual constraints explicit[6]:

$$\begin{aligned}\underset{\alpha \in \mathbb{R}^2}{\text{maximize}} \quad & -\alpha_1^2 + 4\alpha_1 + \alpha_2 \\ \text{subject to} \quad & \alpha_1 \geq 0 \\ & \alpha_2 \geq 0 \\ & 1 - \alpha_1 - \alpha_2 = 0.\end{aligned}$$

Notice that the dual problem is a concave quadratic program in the variables α.

3 The L_1-norm soft margin SVM

To see a more complex example of Lagrange duality in action, we derive the dual of the L_1-norm soft-margin SVM primal presented in class, as well as the corresponding KKT complementarity (i.e., complementary slackness) conditions. We have,

$$\begin{aligned}\underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m, \\ & \xi_i \geq 0, \quad i = 1, \ldots, m.\end{aligned}$$

First, we put this into standard form, with "≤ 0" inequality constraints:

$$\begin{aligned}\underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & 1 - \xi_i - y^{(i)}(w^T x^{(i)} + b) \leq 0, \quad i = 1, \ldots, m, \\ & -\xi_i \leq 0, \quad i = 1, \ldots, m.\end{aligned}$$

Next, we form the generalized Lagrangian,[7]

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - \xi_i - y^{(i)}(w^T x^{(i)} + b)) - \sum_{i=1}^{m}\beta_i\xi_i,$$

[6] By this, we mean that we are moving the condition which causes $\theta_D(\alpha)$ to be $-\infty$ into the set of constraints of the dual optimization problem.

[7] Here, it is important to note that (w, b, ξ) collectively play the role of the "x" primal variables. Similarly, (α, β) collectively play the role of the "α" dual variables normally used for inequality constraints. There are no "β" dual variables here since there are no affine equality constraints in this problem.

which gives the primal and dual optimization problems:

$$\max_{\alpha,\beta:\alpha_i\geq 0,\beta_i\geq 0} \theta_{\mathcal{D}}(\alpha,\beta) \quad \text{where} \quad \theta_{\mathcal{D}}(\alpha,\beta) := \min_{w,b,\xi} \mathcal{L}(w,b,\xi,\alpha,\beta), \tag{SVM-D}$$

$$\min_{w,b,\xi} \theta_{\mathcal{P}}(w,b,\xi) \quad \text{where} \quad \theta_{\mathcal{P}}(w,b,\xi) := \max_{\alpha,\beta:\alpha_i\geq 0,\beta_i\geq 0} \mathcal{L}(w,b,\xi,\alpha,\beta). \tag{SVM-P}$$

To get the dual problem in the form shown in the lecture notes, however, we still have a little more work to do. In particular,

1. **Eliminating the primal variables.** To eliminate the primal variables from the dual problem, we compute $\theta_{\mathcal{D}}(\alpha,\beta)$ by noticing that

$$\theta_{\mathcal{D}}(\alpha,\beta) = \min\nolimits_{w,b,\xi} \quad \mathcal{L}(w,b,\xi,\alpha,\beta)$$

is an unconstrained optimization problem, where the objective function $\mathcal{L}(w,b,\xi,\alpha,\beta)$ is differentiable. The Lagrangian is a strictly convex quadratic function of w, so for any fixed (α,β), if $(\hat{w},\hat{b},\hat{\xi})$ minimize the Lagrangian, it must be the case that

$$\nabla_w \mathcal{L}(\hat{w},\hat{b},\hat{\xi},\alpha,\beta) = \hat{w} - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0. \tag{24}$$

Furthermore, the Lagrangian is linear in b and ξ; by reasoning analogous to that described in the simple duality example from the previous section, we can set the derivatives with respect to b and ξ to zero, and add the resulting conditions as explicit constraints in the dual optimization problem:

$$\frac{\partial}{\partial b}\mathcal{L}(\hat{w},\hat{b},\hat{\xi},\alpha,\beta) = -\sum_{i=1}^m \alpha_i y^{(i)} = 0 \tag{25}$$

$$\frac{\partial}{\partial \xi_i}\mathcal{L}(\hat{w},\hat{b},\hat{\xi},\alpha,\beta) = C - \alpha_i - \beta_i = 0. \tag{26}$$

We can use these conditions to compute the dual objective as

$$\begin{aligned}
\theta_{\mathcal{D}}(\alpha,\beta) &= \mathcal{L}(\hat{w},\hat{b},\hat{\xi}) \\
&= \frac{1}{2}\|\hat{w}\|^2 + C\sum_{i=1}^m \hat{\xi}_i + \sum_{i=1}^m \alpha_i(1-\hat{\xi}_i - y^{(i)}(\hat{w}^T x^{(i)} + \hat{b})) - \sum_{i=1}^m \beta_i\hat{\xi}_i \\
&= \frac{1}{2}\|\hat{w}\|^2 + C\sum_{i=1}^m \hat{\xi}_i + \sum_{i=1}^m \alpha_i(1-\hat{\xi}_i - y^{(i)}(\hat{w}^T x^{(i)})) - \sum_{i=1}^m \beta_i\hat{\xi}_i \\
&= \frac{1}{2}\|\hat{w}\|^2 + \sum_{i=1}^m \alpha_i(1 - y^{(i)}(\hat{w}^T x^{(i)})),
\end{aligned}$$

where the first equality follows from the optimality of $(\hat{w},\hat{b},\hat{\xi})$ for fixed (α,β), the second equality uses the definition of the generalized Lagrangian, and the third and fourth equalities follow from (25) and (26), respectively. Finally, to use (24), observe that

$$\begin{aligned}\frac{1}{2}\|\hat{w}\|^2 + \sum_{i=1}^m \alpha_i(1 - y^{(i)}(\hat{w}^T x^{(i)})) &= \sum_{i=1}^m \alpha_i + \frac{1}{2}\|\hat{w}\|^2 - \hat{w}^T \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \\ &= \sum_{i=1}^m \alpha_i + \frac{1}{2}\|\hat{w}\|^2 - \|\hat{w}\|^2 \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2}\|\hat{w}\|^2 \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle.\end{aligned}$$

Therefore, our dual problem (with no more primal variables and all constraints made explicit) is simply

$$\begin{aligned}\underset{\alpha,\beta}{\text{maximize}} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{subject to} \quad & \alpha_i \geq 0, && i = 1, \dots, m, \\ & \beta_i \geq 0, && i = 1, \dots, m, \\ & \alpha_i + \beta_i = C, && i = 1, \dots, m, \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0.\end{aligned}$$

2. **KKT complementary.** KKT complementarity requires that for any primal optimal (w^*, b^*, ξ^*) and dual optimal (α^*, β^*),

$$\begin{aligned}\alpha_i^*(1 - \xi_i^* - y^{(i)}(w^{*T} x^{(i)} + b^*)) &= 0 \\ \beta_i^* \xi_i^* &= 0\end{aligned}$$

for $i = 1, \dots, m$. From the first condition, we see that if $\alpha_i^* > 0$, then in order for the product to be zero, then $1 - \xi_i^* - y^{(i)}(w^{*T} x^{(i)} + b^*) = 0$. It follows that

$$y^{(i)}(w^{*T} x^{(i)} + b^*) \leq 1$$

since $\xi^* \geq 0$ by primal feasibility. Similarly, if $\beta_i^* > 0$, then $\xi_i^* = 0$ to ensure complementarity. From the primal constraint, $y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$, it follows that

$$y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1.$$

Finally, since $\beta_i^* > 0$ is equivalent to $\alpha_i^* < C$ (since $\alpha^* + \beta_i^* = C$), we can summarize the KKT conditions as follows:

$$\begin{aligned}\alpha_i^* < C \quad &\Rightarrow \quad y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1, \\ \alpha_i^* > 0 \quad &\Rightarrow \quad y^{(i)}(w^{*T} x^{(i)} + b^*) \leq 1.\end{aligned}$$

or equivalently,

$$\begin{aligned} \alpha_i^* = 0 &\quad\Rightarrow\quad y^{(i)}(w^{*T}x^{(i)} + b^*) \geq 1, \\ 0 < \alpha_i^* < C &\quad\Rightarrow\quad y^{(i)}(w^{*T}x^{(i)} + b^*) = 1, \\ \alpha_i^* = C &\quad\Rightarrow\quad y^{(i)}(w^{*T}x^{(i)} + b^*) \leq 1. \end{aligned}$$

3. **Simplification.** We can tidy up our dual problem slightly by observing that each pair of constraints of the form

$$\beta_i \geq 0 \qquad\qquad\qquad \alpha_i + \beta_i = C$$

is equivalent to the single constraint, $\alpha_i \leq C$; that is, if we solve the optimization problem

$$\begin{aligned} \underset{\alpha,\beta}{\text{maximize}} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_i y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \qquad\qquad\qquad i = 1, \ldots, m, \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0. \end{aligned} \tag{27}$$

and subsequently set $\beta_i = C - \alpha_i$, then it follows that (α, β) will be optimal for the previous dual problem above. This last form, indeed, is the form of the soft-margin SVM dual given in the lecture notes.

4 Directions for further exploration

In many real-world tasks, 90% of the challenge involves figuring out how to write an optimization problem in a convex form. Once the correct form has been found, a number of pre-existing software packages for convex optimization have been well-tuned to handle different specific types of optimization problems. The following constitute a small sample of the available tools:

- commerical packages: CPLEX, MOSEK
- MATLAB-based: CVX, Optimization Toolbox (linprog, quadprog), SeDuMi
- libraries: CVXOPT (Python), GLPK (C), COIN-OR (C)
- SVMs: LIBSVM, SVM-light
- machine learning: Weka (Java)

In particular, we specifically point out CVX as an easy-to-use generic tool for solving convex optimization problems easily using MATLAB, and CVXOPT as a powerful Python-based library which runs independently of MATLAB.[8] If you're interested in looking at some of the other packages listed above, they are easy to find with a web search. In short, if you need a specific convex optimization algorithm, pre-existing software packages provide a rapid way to prototype your idea without having to deal with the numerical trickiness of implementing your own complete convex optimization routines.

Also, if you find this material fascinating, make sure to check out Stephen Boyd's class, EE364: Convex Optimization I, which will be offered during the Winter Quarter. The textbook for the class (listed as [1] in the References) has a wealth of information about convex optimization and is available for browsing online.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge UP, 2004. Online: `http://www.stanford.edu/~boyd/cvxbook/`

[8] CVX is available at `http://www.stanford.edu/~boyd/cvx/` and CVXOPT is available at `http://www.ee.ucla.edu/~vandenbe/cvxopt/`.

Convex Optimization Overview

Zico Kolter (updated by Honglak Lee)

October 17, 2008

1 Introduction

Many situations arise in machine learning where we would like to ***optimize*** the value of some function. That is, given a function $f : \mathbb{R}^n \to \mathbb{R}$, we want to find $x \in \mathbb{R}^n$ that minimizes (or maximizes) $f(x)$. We have already seen several examples of optimization problems in class: least-squares, logistic regression, and support vector machines can all be framed as optimization problems.

It turns out that, in the general case, finding the global optimum of a function can be a very difficult task. However, for a special class of optimization problems known as ***convex optimization problems***, we can efficiently find the global solution in many cases. Here, "efficiently" has both practical and theoretical connotations: it means that we can solve many real-world problems in a reasonable amount of time, and it means that theoretically we can solve problems in time that depends only *polynomially* on the problem size.

The goal of these section notes and the accompanying lecture is to give a very brief overview of the field of convex optimization. Much of the material here (including some of the figures) is heavily based on the book *Convex Optimization* [1] by Stephen Boyd and Lieven Vandenberghe (available for free online), and EE364, a class taught here at Stanford by Stephen Boyd. If you are interested in pursuing convex optimization further, these are both excellent resources.

2 Convex Sets

We begin our look at convex optimization with the notion of a ***convex set***.

Definition 2.1 *A set C is convex if, for any $x, y \in C$ and $\theta \in \mathbb{R}$ with $0 \le \theta \le 1$,*

$$\theta x + (1 - \theta) y \in C.$$

Intuitively, this means that if we take any two elements in C, and draw a line segment between these two elements, then every point on that line segment also belongs to C. Figure 1 shows an example of one convex and one non-convex set. The point $\theta x + (1 - \theta) y$ is called a ***convex combination*** of the points x and y.

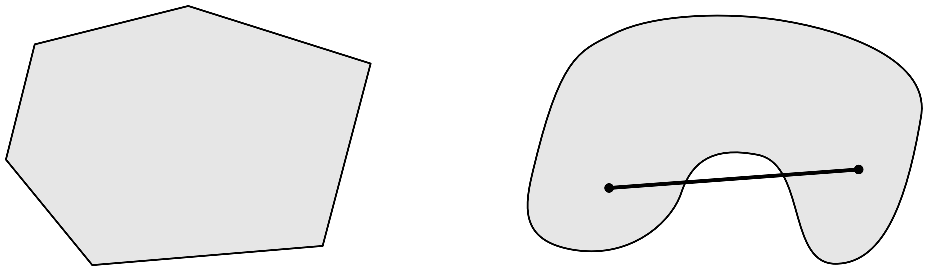


Figure 1: Examples of a convex set (a) and a non-convex set (b).

2.1 Examples

- **All of $\mathbb{R}^n$.** It should be fairly obvious that given any $x, y \in \mathbb{R}^n$, $\theta x + (1-\theta)y \in \mathbb{R}^n$.

- **The non-negative orthant, $\mathbb{R}^n_+$.** The non-negative orthant consists of all vectors in $\mathbb{R}^n$ whose elements are all non-negative: $\mathbb{R}^n_+ = \{x : x_i \geq 0 \;\; \forall i = 1, \ldots, n\}$. To show that this is a convex set, simply note that given any $x, y \in \mathbb{R}^n_+$ and $0 \leq \theta \leq 1$,

$$(\theta x + (1-\theta)y)_i = \theta x_i + (1-\theta)y_i \geq 0 \;\; \forall i.$$

- **Norm balls.** Let $\|\cdot\|$ be some norm on $\mathbb{R}^n$ (e.g., the Euclidean norm, $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$). Then the set $\{x : \|x\| \leq 1\}$ is a convex set. To see this, suppose $x, y \in \mathbb{R}^n$, with $\|x\| \leq 1, \|y\| \leq 1$, and $0 \leq \theta \leq 1$. Then

$$\|\theta x + (1-\theta)y\| \leq \|\theta x\| + \|(1-\theta)y\| = \theta\|x\| + (1-\theta)\|y\| \leq 1$$

where we used the triangle inequality and the positive homogeneity of norms.

- **Affine subspaces and polyhedra.** Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, an affine subspace is the set $\{x \in \mathbb{R}^n : Ax = b\}$ (note that this could possibly be empty if b is not in the range of A). Similarly, a polyhedron is the (again, possibly empty) set $\{x \in \mathbb{R}^n : Ax \preceq b\}$, where '$\preceq$' here denotes componentwise inequality (i.e., all the entries of Ax are less than or equal to their corresponding element in b).[1] To prove this, first consider $x, y \in \mathbb{R}^n$ such that $Ax = Ay = b$. Then for $0 \leq \theta \leq 1$,

$$A(\theta x + (1-\theta)y) = \theta Ax + (1-\theta)Ay = \theta b + (1-\theta)b = b.$$

Similarly, for $x, y \in \mathbb{R}^n$ that satisfy $Ax \leq b$ and $Ay \leq b$ and $0 \leq \theta \leq 1$,

$$A(\theta x + (1-\theta)y) = \theta Ax + (1-\theta)Ay \leq \theta b + (1-\theta)b = b.$$

[1] Similarly, for two vectors $x, y \in \mathbb{R}^n$, $x \succeq y$ denotes that each element of x is greater than or equal to the corresponding element in y. Note that sometimes '$\leq$' and '$\geq$' are used in place of '$\preceq$' and '$\succeq$'; the meaning must be determined contextually (i.e., both sides of the inequality will be vectors).

- **Intersections of convex sets.** Suppose $C_1, C_2, \ldots, C_k$ are convex sets. Then their intersection

$$\bigcap_{i=1}^{k} C_i = \{x : x \in C_i \ \ \forall i = 1, \ldots, k\}$$

  is also a convex set. To see this, consider $x, y \in \bigcap_{i=1}^{k} C_i$ and $0 \leq \theta \leq 1$. Then,

$$\theta x + (1-\theta)y \in C_i \ \ \forall i = 1, \ldots, k$$

  by the definition of a convex set. Therefore

$$\theta x + (1-\theta)y \in \bigcap_{i=1}^{k} C_i.$$

  Note, however, that the *union* of convex sets in general will not be convex.

- **Positive semidefinite matrices.** The set of all symmetric positive semidefinite matrices, often times called the *positive semidefinite cone* and denoted $\mathbb{S}^n_+$, is a convex set (in general, $\mathbb{S}^n \subset \mathbb{R}^{n \times n}$ denotes the set of symmetric $n \times n$ matrices). Recall that a matrix $A \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite if and only if $A = A^T$ and for all $x \in \mathbb{R}^n$, $x^T A x \geq 0$. Now consider two symmetric positive semidefinite matrices $A, B \in \mathbb{S}^n_+$ and $0 \leq \theta \leq 1$. Then for any $x \in \mathbb{R}^n$,

$$x^T(\theta A + (1-\theta)B)x = \theta x^T A x + (1-\theta) x^T B x \geq 0.$$

  The same logic can be used to show that the sets of all positive definite, negative definite, and negative semidefinite matrices are each also convex.

3 Convex Functions

A central element in convex optimization is the notion of a ***convex function***.

Definition 3.1 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain (denoted $\mathcal{D}(f)$) is a convex set, and if, for all $x, y \in \mathcal{D}(f)$ and $\theta \in \mathbb{R}$, $0 \leq \theta \leq 1$,*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y).$$

Intuitively, the way to think about this definition is that if we pick any two points on the graph of a convex function and draw a straight line between then, then the portion of the function between these two points will lie below this straight line. This situation is pictured in Figure 2.[2]

We say a function is ***strictly convex*** if Definition 3.1 holds with strict inequality for $x \neq y$ and $0 < \theta < 1$. We say that f is ***concave*** if $-f$ is convex, and likewise that f is ***strictly concave*** if $-f$ is strictly convex.

[2] Don't worry too much about the requirement that the domain of f be a convex set. This is just a technicality to ensure that $f(\theta x + (1-\theta)y)$ is actually defined (if $\mathcal{D}(f)$ were not convex, then it could be that $f(\theta x + (1-\theta)y)$ is undefined even though $x, y \in \mathcal{D}(f)$).

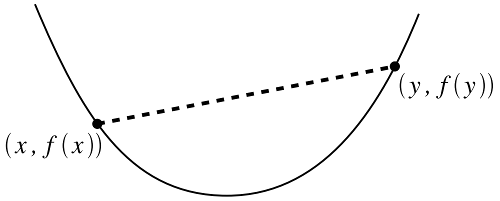


Figure 2: Graph of a convex function. By the definition of convex functions, the line connecting two points on the graph must lie above the function.

3.1 First Order Condition for Convexity

Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable (i.e., the gradient[3] $\nabla_x f(x)$ exists at all points x in the domain of f). Then f is convex if and only if $\mathcal{D}(f)$ is a convex set and for all $x, y \in \mathcal{D}(f)$,

$$f(y) \geq f(x) + \nabla_x f(x)^T (y - x).$$

The function $f(x) + \nabla_x f(x)^T (y - x)$ is called the ***first-order approximation*** to the function f at the point x. Intuitively, this can be thought of as approximating f with its tangent line at the point x. The first order condition for convexity says that f is convex if and only if the tangent line is a global underestimator of the function f. In other words, if we take our function and draw a tangent line at any point, then every point on this line will lie below the corresponding point on f.

Similar to the definition of convexity, f will be strictly convex if this holds with strict inequality, concave if the inequality is reversed, and strictly concave if the reverse inequality is strict.

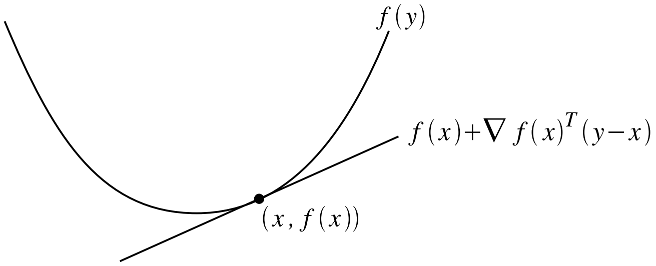


Figure 3: Illustration of the first-order condition for convexity.

[3] Recall that the gradient is defined as $\nabla_x f(x) \in \mathbb{R}^n, (\nabla_x f(x))_i = \frac{\partial f(x)}{\partial x_i}$. For a review on gradients and Hessians, see the previous section notes on linear algebra.

3.2 Second Order Condition for Convexity

Suppose a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is twice differentiable (i.e., the Hessian[4] $\nabla_x^2 f(x)$ is defined for all points x in the domain of f). Then f is convex if and only if $\mathcal{D}(f)$ is a convex set and its Hessian is positive semidefinite: i.e., for any $x \in \mathcal{D}(f)$,

$$\nabla_x^2 f(x) \succeq 0.$$

Here, the notation '$\succeq$' when used in conjunction with matrices refers to positive semidefiniteness, rather than componentwise inequality. [5] In one dimension, this is equivalent to the condition that the second derivative $f''(x)$ always be non-negative (i.e., the function always has positive non-negative).

Again analogous to both the definition and the first order conditions for convexity, f is strictly convex if its Hessian is positive definite, concave if the Hessian is negative semidefinite, and strictly concave if the Hessian is negative definite.

3.3 Jensen's Inequality

Suppose we start with the inequality in the basic definition of a convex function

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \quad \text{for} \quad 0 \leq \theta \leq 1.$$

Using induction, this can be fairly easily extended to convex combinations of more than one point,

$$f\left(\sum_{i=1}^{k} \theta_i x_i\right) \leq \sum_{i=1}^{k} \theta_i f(x_i) \quad \text{for} \quad \sum_{i=1}^{k} \theta_i = 1,\ \theta_i \geq 0 \ \ \forall i.$$

In fact, this can also be extended to infinite sums or integrals. In the latter case, the inequality can be written as

$$f\left(\int p(x) x dx\right) \leq \int p(x) f(x) dx \quad \text{for} \quad \int p(x) dx = 1,\ p(x) \geq 0 \ \ \forall x.$$

Because $p(x)$ integrates to 1, it is common to consider it as a probability density, in which case the previous equation can be written in terms of expectations,

$$f(\mathbf{E}[x]) \leq \mathbf{E}[f(x)].$$

This last inequality is known as *Jensen's inequality*, and it will come up later in class.[6]

[4] Recall the Hessian is defined as $\nabla_x^2 f(x) \in \mathbb{R}^{n\times n}$, $(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$

[5] Similarly, for a symmetric matrix $X \in \mathbb{S}^n$, $X \preceq 0$ denotes that X is negative semidefinite. As with vector inequalities, '$\leq$' and '$\geq$' are sometimes used in place of '$\preceq$' and '$\succeq$'. Despite their notational similarity to vector inequalities, these concepts are very different; in particular, $X \succeq 0$ does not imply that $X_{ij} \geq 0$ for all i and j.

[6] In fact, all four of these equations are sometimes referred to as Jensen's inequality, due to the fact that they are all equivalent. However, for this class we will use the term to refer specifically to the last inequality presented here.

3.4 Sublevel Sets

Convex functions give rise to a particularly important type of convex set called an ***α-sublevel set***. Given a convex function $f : \mathbb{R}^n \rightarrow R$ and a real number $\alpha \in \mathbb{R}$, the α-sublevel set is defined as

$$\{x \in \mathcal{D}(f) : f(x) \leq \alpha\}.$$

In other words, the α-sublevel set is the set of all points x such that $f(x) \leq \alpha$.

To show that this is a convex set, consider any $x, y \in \mathcal{D}(f)$ such that $f(x) \leq \alpha$ and $f(y) \leq \alpha$. Then

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \leq \theta\alpha + (1-\theta)\alpha = \alpha.$$

3.5 Examples

We begin with a few simple examples of convex functions of one variable, then move on to multivariate functions.

- **Exponential.** Let $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = e^{ax}$ for any $a \in \mathbb{R}$. To show f is convex, we can simply take the second derivative $f''(x) = a^2 e^{ax}$, which is positive for all x.

- **Negative logarithm.** Let $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = -\log x$ with domain $\mathcal{D}(f) = \mathbb{R}_{++}$ (here, $\mathbb{R}_{++}$ denotes the set of strictly positive real numbers, $\{x : x > 0\}$). Then $f''(x) = 1/x^2 > 0$ for all x.

- **Affine functions.** Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $f(x) = b^T x + c$ for some $b \in \mathbb{R}^n$, $c \in \mathbb{R}$. In this case the Hessian, $\nabla_x^2 f(x) = 0$ for all x. Because the zero matrix is both positive semidefinite and negative semidefinite, f is both convex and concave. In fact, affine functions of this form are the *only* functions that are both convex and concave.

- **Quadratic functions.** Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $f(x) = \frac{1}{2}x^T A x + b^T x + c$ for a symmetric matrix $A \in \mathbb{S}^n$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. In our previous section notes on linear algebra, we showed the Hessian for this function is given by

  $$\nabla_x^2 f(x) = A.$$

  Therefore, the convexity or non-convexity of f is determined entirely by whether or not A is positive semidefinite: if A is positive semidefinite then the function is convex (and analogously for strictly convex, concave, strictly concave); if A is indefinite then f is neither convex nor concave.

  Note that the squared Euclidean norm $f(x) = \|x\|_2^2 = x^T x$ is a special case of quadratic functions where $A = I$, $b = 0$, $c = 0$, so it is therefore a strictly convex function.

- **Norms.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be some norm on $\mathbb{R}^n$. Then by the triangle inequality and positive homogeneity of norms, for $x, y \in \mathbb{R}^n$, $0 \le \theta \le 1$,

  $$f(\theta x + (1-\theta)y) \le f(\theta x) + f((1-\theta)y) = \theta f(x) + (1-\theta)f(y).$$

  This is an example of a convex function where it is *not* possible to prove convexity based on the second-order or first-order conditions because norms are not generally differentiable everywhere (e.g., the 1-norm, $||x||_1 = \sum_{i=1}^n |x_i|$, is non-differentiable at all points where any x_i is equal to zero).

- **Nonnegative weighted sums of convex functions.** Let $f_1, f_2, \ldots, f_k$ be convex functions and $w_1, w_2, \ldots, w_k$ be nonnegative real numbers. Then

  $$f(x) = \sum_{i=1}^k w_i f_i(x)$$

  is a convex function, since

  $$\begin{aligned} f(\theta x + (1-\theta)y) &= \sum_{i=1}^k w_i f_i(\theta x + (1-\theta)y) \\ &\le \sum_{i=1}^k w_i(\theta f_i(x) + (1-\theta)f_i(y)) \\ &= \theta \sum_{i=1}^k w_i f_i(x) + (1-\theta)\sum_{i=1}^k w_i f_i(y) \\ &= \theta f(x) + (1-\theta)f(x). \end{aligned}$$

4 Convex Optimization Problems

Armed with the definitions of convex functions and sets, we are now equipped to consider ***convex optimization problems***. Formally, a convex optimization problem in an optimization problem of the form

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

where f is a convex function, C is a convex set, and x is the optimization variable. However, since this can be a little bit vague, we often write it as

$$\begin{array}{rll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \le 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array}$$

where f is a convex function, g_i are convex functions, and h_i are affine functions, and x is the optimization variable.

Is it imporant to note the direction of these inequalities: a convex function g_i must be *less* than zero. This is because the 0-sublevel set of g_i is a convex set, so the feasible region, which is the intersection of many convex sets, is also convex (recall that affine subspaces are convex sets as well). If we were to require that $g_i \geq 0$ for some convex g_i, the feasible region would no longer be a convex set, and the algorithms we apply for solving these problems would no longer be guaranteed to find the global optimum. Also notice that only affine functions are allowed to be equality constraints. Intuitively, you can think of this as being due to the fact that an equality constraint is equivalent to the two inequalities $h_i \leq 0$ and $h_i \geq 0$. However, these will both be valid constraints if and only if h_i is both convex and concave, i.e., h_i must be affine.

The ***optimal value*** of an optimization problem is denoted $p^\star$ (or sometimes $f^\star$) and is equal to the minimum possible value of the objective function in the feasible region[7]

$$p^\star = \min\{f(x) : g_i(x) \leq 0, i = 1, \ldots, m, h_i(x) = 0, i = 1, \ldots, p\}.$$

We allow $p^\star$ to take on the values $+\infty$ and $-\infty$ when the problem is either *infeasible* (the feasible region is empty) or *unbounded below* (there exists feasible points such that $f(x) \rightarrow -\infty$), respectively. We say that $x^\star$ is an ***optimal point*** if $f(x^\star) = p^\star$. Note that there can be more than one optimal point, even when the optimal value is finite.

4.1 Global Optimality in Convex Problems

Before stating the result of global optimality in convex problems, let us formally define the concepts of local optima and global optima. Intuitively, a feasible point is called ***locally optimal*** if there are no "nearby" feasible points that have a lower objective value. Similarly, a feasible point is called ***globally optimal*** if there are no feasible points at all that have a lower objective value. To formalize this a little bit more, we give the following two definitions.

Definition 4.1 *A point x is locally optimal if it is feasible (i.e., it satisfies the constraints of the optimization problem) and if there exists some $R > 0$ such that all feasible points z with $\|x - z\|_2 \leq R$, satisfy $f(x) \leq f(z)$.*

Definition 4.2 *A point x is globally optimal if it is feasible and for all feasible points z, $f(x) \leq f(z)$.*

We now come to the crucial element of convex optimization problems, from which they derive most of their utility. The key idea is that ***for a convex optimization problem all locally optimal points are globally optimal***.

Let's give a quick proof of this property by contradiction. Suppose that x is a locally optimal point which is not globally optimal, i.e., there exists a feasible point y such that

[7] Math majors might note that the min appearing below should more correctly be an inf. We won't worry about such technicalities here, and use min for simplicity.

$f(x) > f(y)$. By the definition of local optimality, there exist no feasible points z such that $\|x - z\|_2 \leq R$ and $f(z) < f(x)$. But now suppose we choose the point

$$z = \theta y + (1-\theta)x \quad \text{with} \quad \theta = \frac{R}{2\|x-y\|_2}.$$

Then

$$\begin{aligned}
\|x - z\|_2 &= \left\| x - \left(\frac{R}{2\|x-y\|_2} y + \left(1 - \frac{R}{2\|x-y\|_2}\right) x \right) \right\|_2 \\
&= \left\| \frac{R}{2\|x-y\|_2} (x-y) \right\|_2 \\
&= R/2 \leq R.
\end{aligned}$$

In addition, by the convexity of f we have

$$f(z) = f(\theta y + (1-\theta)x) \leq \theta f(y) + (1-\theta) f(x) < f(x).$$

Furthermore, since the feasible set is a convex set, and since x and y are both feasible $z = \theta y + (1-\theta)$ will be feasible as well. Therefore, z is a feasible point, with $\|x - z\|_2 < R$ and $f(z) < f(x)$. This contradicts our assumption, showing that x cannot be locally optimal.

4.2 Special Cases of Convex Problems

For a variety of reasons, it is oftentimes convenient to consider special cases of the general convex programming formulation. For these special cases we can often devise extremely efficient algorithms that can solve very large problems, and because of this you will probably see these special cases referred to any time people use convex optimization techniques.

- **Linear Programming.** We say that a convex optimization problem is a ***linear program*** (LP) if both the objective function f and inequality constraints g_i are affine functions. In other words, these problems have the form

$$\begin{array}{rl} \text{minimize} & c^T x + d \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

  where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$ are defined by the problem, and '$\preceq$' denotes elementwise inequality.

- **Quadratic Programming.** We say that a convex optimization problem is a ***quadratic program*** (QP) if the inequality constraints g_i are still all affine, but if the objective function f is a convex quadratic function. In other words, these problems have the form,

$$\begin{array}{rl} \text{minimize} & \frac{1}{2} x^T P x + c^T x + d \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

where again $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$ are defined by the problem, but we also have $P \in \mathbb{S}^n_+$, a symmetric positive semidefinite matrix.

- **Quadratically Constrained Quadratic Programming.** We say that a convex optimization problem is a ***quadratically constrained quadratic program*** (QCQP) if both the objective f and the inequality constraints g_i are convex quadratic functions,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + c^T x + d \\ \text{subject to} & \frac{1}{2}x^T Q_i x + r_i^T x + s_i \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

  where, as before, $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$, $P \in \mathbb{S}^n_+$, but we also have $Q_i \in \mathbb{S}^n_+$, $r_i \in \mathbb{R}^n$, $s_i \in \mathbb{R}$, for $i = 1, \ldots, m$.

- **Semidefinite Programming.** This last example is more complex than the previous ones, so don't worry if it doesn't make much sense at first. However, semidefinite programming is becoming more prevalent in many areas of machine learning research, so you might encounter these at some point, and it is good to have an idea of what they are. We say that a convex optimization problem is a ***semidefinite program*** (SDP) if it is of the form

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(CX) \\ \text{subject to} & \mathrm{tr}(A_i X) = b_i, \quad i = 1, \ldots, p \\ & X \succeq 0 \end{array}$$

  where the symmetric matrix $X \in \mathbb{S}^n$ is the optimization variable, the symmetric matrices $C, A_1, \ldots, A_p \in \mathbb{S}^n$ are defined by the problem, and the constraint $X \succeq 0$ means that we are constraining X to be positive semidefinite. This looks a bit different than the problems we have seen previously, since the optimization variable is now a matrix instead of a vector. If you are curious as to why such a formulation might be useful, you should look into a more advanced course or book on convex optimization.

It should be obvious from the definitions that quadratic programs are more general than linear programs (since a linear program is just a special case of a quadratic program where $P = 0$), and likewise that quadratically constrained quadratic programs are more general than quadratic programs. However, what is not obvious is that semidefinite programs are in fact more general than all the previous types, that is, any quadratically constrained quadratic program (and hence any quadratic program or linear program) can be expressed as a semidefinte program. We won't discuss this relationship further in this document, but this might give you just a small idea as to why semidefinite programming could be useful.

4.3 Examples

Now that we've covered plenty of the boring math and formalisms behind convex optimization, we can finally get to the fun part: using these techniques to solve actual problems.

We've already encountered a few such optimization problems in class, and in nearly every field, there is a good chance that someone has applied convex optimization to solve some problem.

- **Support Vector Machines (SVM).** One of the most prevalent applications of convex optimization methods in machine learning is the support vector machine classifier. As discussed in class, finding the support vector classifier (in the case with slack variables) can be formulated as the optimization problem

$$\begin{array}{rll} \text{minimize} & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^m \xi_i & \\ \text{subject to} & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, & i = 1, \dots, m \\ & \xi_i \geq 0, & i = 1, \dots, m \end{array}$$

with optimization variables $w \in \mathbb{R}^n$, $\xi \in \mathbb{R}^m$, $b \in \mathbb{R}$, and where $C \in \mathbb{R}$ and $x^{(i)}, y^{(i)}, i = 1, \dots m$ are defined by the problem. This is an example of a quadratic program, which we shall show by putting the problem into the form described in the previous section. In particular, if we define $k = m + n + 1$, let the optimization variable be

$$x \in \mathbb{R}^k \equiv \begin{bmatrix} w \\ \xi \\ b \end{bmatrix}$$

and define the matrices

$$P \in \mathbb{R}^{k \times k} = \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad c \in \mathbb{R}^k = \begin{bmatrix} 0 \\ C \cdot \mathbf{1} \\ 0 \end{bmatrix},$$

$$G \in \mathbb{R}^{2m \times k} = \begin{bmatrix} -\text{diag}(y)X & -I & -y \\ 0 & -I & 0 \end{bmatrix}, \quad h \in \mathbb{R}^{2m} = \begin{bmatrix} -\mathbf{1} \\ 0 \end{bmatrix}$$

where I is the identity, $\mathbf{1}$ is the vector of all ones, and X and y are defined as in class,

$$X \in \mathbb{R}^{m \times n} = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ \vdots \\ x^{(m)T} \end{bmatrix}, \quad y \in \mathbb{R}^m = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

You should convince yourself that the quadratic program described in the previous section, when using these matrices defined above, is equivalent to the SVM optimization problem. In reality, it is fairly easy to see that there the SVM optimization problem has a quadratic objective and linear constraints, so we typically don't need to put it into standard form to "prove" that it is a QP, and we would only do so if we are using an off-the-shelf solver that requires the input to be in standard form.

- **Constrained least squares.** In class we have also considered the least squares problem, where we want to minimize $\|Ax - b\|_2^2$ for some matrix $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. As we saw, this particular problem can be solved analytically via the normal equations. However, suppose that we also want to constrain the entries in the solution x to lie within some predefined ranges. In other words, suppose we wanted to solve the optimization problem,

  $$\begin{array}{rl} \text{minimize} & \frac{1}{2}\|Ax - b\|_2^2 \\ \text{subject to} & l \preceq x \preceq u \end{array}$$

  with optimization variable x and problem data $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $l \in \mathbb{R}^n$, and $u \in \mathbb{R}^n$. This might seem like a simple additional constraint, but it turns out that there will no longer be an analytical solution. However, you should convince yourself that this optimization problem is a quadratic program, with matrices defined by

  $$P \in \mathbb{R}^{n\times n} = \frac{1}{2}A^TA, \;\; c \in \mathbb{R}^n = -b^TA, \;\; d \in \mathbb{R} = \frac{1}{2}b^Tb,$$

  $$G \in \mathbb{R}^{2n\times 2n} = \left[\begin{array}{cc} -I & 0 \\ 0 & I \end{array}\right], \;\; h \in \mathbb{R}^{2n} = \left[\begin{array}{c} -l \\ u \end{array}\right].$$

- **Maximum Likelihood for Logistic Regression.** For homework one, you were required to show that the log-likelihood of the data in a logistic model was concave. The log likehood under such a model is

  $$\ell(\theta) = \sum_{i=1}^{n}\left\{y^{(i)}\ln g(\theta^Tx^{(i)}) + (1 - y^{(i)})\ln(1 - g(\theta^Tx^{(i)}))\right\}$$

  where $g(z)$ denotes the logistic function $g(z) = 1/(1 + e^{-z})$. Finding the maximum likelihood estimate is then a task of maximizing the log-likelihood (or equivalently, minimizing the negative log-likelihood, a convex function), i.e.,

  $$\text{minimize} \quad -\ell(\theta)$$

  with optimization variable $\theta \in \mathbb{R}^n$ and no constraints.

  Unlike the previous two examples, it is not so easy to put this problem into a "standard" form optimization problem. Nevertheless, you have seen on the homework that the fact that ℓ is a concave function means that you can very efficiently find the global solution using an algorithm such as Newton's method.

4.4 Implementation: Linear SVM using CVX

Many convex optimization problems can be solved by several off-the-shelf software packages including CVX, Sedumi, CPLEX, MOSEK, etc. Thus, in many cases, once you identify the convex optimization problem, you can solve it without worrying about how to implement the algorithm yourself. This is particularly useful for a rapid prototyping.[8]

Among these software packages, we introduce CVX [2] as an example. CVX is a free MATLAB-based software package for solving generic convex optimzation problems; it can solve a wide variety of convex optimization problems such as LP, QP, QCQP, SDP, etc. As an illustration, we conclude this section by implementing a linear SVM classifier for the binary classification problem using the data given in the Problem Set #1. For more general setting using other non-linear kernels, the dual formulation can be solved using CVX as well.

```
% load data
load q1x.dat
load q1y.dat

% define variables
X = q1x;
y = 2*(q1y-0.5);

C = 1;
m = size(q1x,1);
n = size(q1x,2);

% train svm using cvx
cvx_begin
    variables w(n) b xi(m)
    minimize 1/2*sum(w.*w) + C*sum(xi)
    y.*(X*w + b) >= 1 - xi;
    xi >= 0;
cvx_end

% visualize
xp = linspace(min(X(:,1)), max(X(:,1)), 100);
yp = - (w(1)*xp + b)/w(2);
yp1 = - (w(1)*xp + b - 1)/w(2); % margin boundary for support vectors for y=1
yp0 = - (w(1)*xp + b + 1)/w(2); % margin boundary for support vectors for y=0

idx0 = find(q1y==0);
idx1 = find(q1y==1);

plot(q1x(idx0, 1), q1x(idx0, 2), 'rx'); hold on
```

[8] However, depending on the optimization problem, these off-the-shelf convex optimization solvers can be much slower compared to the best possible implementation; therefore, sometimes you may have to use more customized solvers or implement your own.

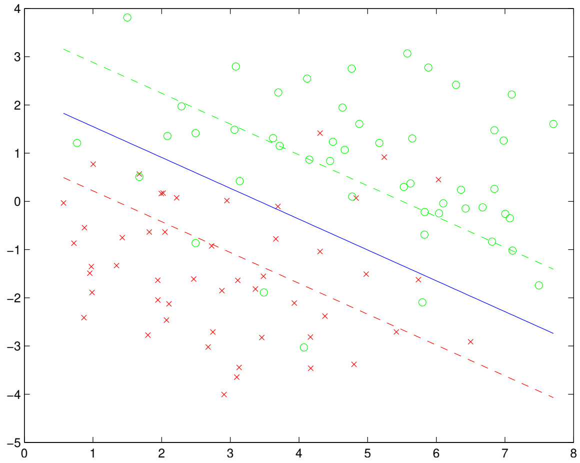


Figure 4: Decision boundary for a linear SVM classifier with $C = 1$.

```
plot(q1x(idx1, 1), q1x(idx1, 2), 'go');
plot(xp, yp, '-b', xp, yp1, '--g', xp, yp0, '--r');
hold off
title(sprintf('decision boundary for a linear SVM classifier with C=%g', C));
```

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge UP, 2004. Online: `http://www.stanford.edu/~boyd/cvxbook/`

[2] M. Grant and S. Boyd. *CVX: Matlab software for disciplined convex programming* (web page and software). `http://stanford.edu/~boyd/cvx/`, September 2008.

Gaussian processes

Chuong B. Do (updated by Honglak Lee)

November 22, 2008

Many of the classical machine learning algorithms that we talked about during the first half of this course fit the following pattern: given a training set of i.i.d. examples sampled from some unknown distribution,

1. solve a convex optimization problem in order to identify the single "best fit" model for the data, and

2. use this estimated model to make "best guess" predictions for future test input points.

In these notes, we will talk about a different flavor of learning algorithms, known as **Bayesian methods**. Unlike classical learning algorithm, Bayesian algorithms do not attempt to identify "best-fit" models of the data (or similarly, make "best guess" predictions for new test inputs). Instead, they compute a posterior distribution over models (or similarly, compute posterior predictive distributions for new test inputs). These distributions provide a useful way to quantify our uncertainty in model estimates, and to exploit our knowledge of this uncertainty in order to make more robust predictions on new test points.

We focus on **regression** problems, where the goal is to learn a mapping from some input space $\mathcal{X} = \mathbf{R}^n$ of n-dimensional vectors to an output space $\mathcal{Y} = \mathbf{R}$ of real-valued targets. In particular, we will talk about a kernel-based fully Bayesian regression algorithm, known as Gaussian process regression. The material covered in these notes draws heavily on many different topics that we discussed previously in class (namely, the probabilistic interpretation of linear regression[1], Bayesian methods[2], kernels[3], and properties of multivariate Gaussians[4]).

The organization of these notes is as follows. In Section 1, we provide a brief review of multivariate Gaussian distributions and their properties. In Section 2, we briefly review Bayesian methods in the context of probabilistic linear regression. The central ideas underlying Gaussian processes are presented in Section 3, and we derive the full Gaussian process regression model in Section 4.

[1] See course lecture notes on "Supervised Learning, Discriminative Algorithms."

[2] See course lecture notes on "Regularization and Model Selection."

[3] See course lecture notes on "Support Vector Machines."

[4] See course lecture notes on "Factor Analysis."

1 Multivariate Gaussians

A vector-valued random variable $x \in \mathbf{R}^n$ is said to have a **multivariate normal (or Gaussian) distribution** with mean $\mu \in \mathbf{R}^n$ and covariance matrix $\Sigma \in \mathbf{S}^n_{++}$ if

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right). \tag{1}$$

We write this as $x \sim \mathcal{N}(\mu,\Sigma)$. Here, recall from the section notes on linear algebra that $\mathbf{S}^n_{++}$ refers to the space of symmetric positive definite $n \times n$ matrices.[5]

Generally speaking, Gaussian random variables are extremely useful in machine learning and statistics for two main reasons. First, they are extremely common when modeling "noise" in statistical algorithms. Quite often, noise can be considered to be the accumulation of a large number of small independent random perturbations affecting the measurement process; by the Central Limit Theorem, summations of independent random variables will tend to "look Gaussian." Second, Gaussian random variables are convenient for many analytical manipulations, because many of the integrals involving Gaussian distributions that arise in practice have simple closed form solutions. In the remainder of this section, we will review a number of useful properties of multivariate Gaussians.

Consider a random vector $x \in \mathbf{R}^n$ with $x \sim \mathcal{N}(\mu,\Sigma)$. Suppose also that the variables in x have been partitioned into two sets $x_A = [x_1 \;\cdots\; x_r]^T \in \mathbf{R}^r$ and $x_B = [x_{r+1} \;\cdots\; x_n]^T \in \mathbf{R}^{n-r}$ (and similarly for μ and Σ), such that

$$x = \begin{bmatrix} x_A \\ x_B \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}.$$

Here, $\Sigma_{AB} = \Sigma_{BA}^T$ since $\Sigma = E[(x-\mu)(x-\mu)^T] = \Sigma^T$. The following properties hold:

1. **Normalization.** The density function normalizes, i.e.,

   $$\int_x p(x;\mu,\Sigma)dx = 1.$$

   This property, though seemingly trivial at first glance, turns out to be immensely useful for evaluating all sorts of integrals, even ones which appear to have no relation to probability distributions at all (see Appendix A.1)!

2. **Marginalization.** The marginal densities,

   $$\begin{aligned} p(x_A) &= \int_{x_B} p(x_A, x_B; \mu, \Sigma)dx_B \\ p(x_B) &= \int_{x_A} p(x_A, x_B; \mu, \Sigma)dx_A \end{aligned}$$

[5] There are actually cases in which we would want to deal with multivariate Gaussian distributions where Σ is positive semidefinite but not positive definite (i.e., Σ is not full rank). In such cases, Σ^{-1} does not exist, so the definition of the Gaussian density given in (1) does not apply. For instance, see the course lecture notes on "Factor Analysis."

are Gaussian:

$$\begin{aligned}
x_A &\sim \mathcal{N}(\mu_A, \Sigma_{AA}) \\
x_B &\sim \mathcal{N}(\mu_B, \Sigma_{BB}).
\end{aligned}$$

3. **Conditioning.** The conditional densities

$$\begin{aligned}
p(x_A \mid x_B) &= \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_A} p(x_A, x_B; \mu, \Sigma) dx_A} \\
p(x_B \mid x_A) &= \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_B} p(x_A, x_B; \mu, \Sigma) dx_B}
\end{aligned}$$

are also Gaussian:

$$\begin{aligned}
x_A \mid x_B &\sim \mathcal{N}\big(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\big) \\
x_B \mid x_A &\sim \mathcal{N}\big(\mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A), \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}\big).
\end{aligned}$$

A proof of this property is given in Appendix A.2. (See also Appendix A.3 for an easier version of the derivation.)

4. **Summation.** The sum of independent Gaussian random variables (with the same dimensionality), $y \sim \mathcal{N}(\mu, \Sigma)$ and $z \sim \mathcal{N}(\mu', \Sigma')$, is also Gaussian:

$$y + z \sim \mathcal{N}(\mu + \mu', \Sigma + \Sigma').$$

2 Bayesian linear regression

Let $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$ be a training set of i.i.d. examples from some unknown distribution. The standard probabilistic interpretation of linear regression states that

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}, \qquad i = 1, \dots, m$$

where the $\varepsilon^{(i)}$ are i.i.d. "noise" variables with independent $\mathcal{N}(0, \sigma^2)$ distributions. It follows that $y^{(i)} - \theta^T x^{(i)} \sim \mathcal{N}(0, \sigma^2)$, or equivalently,

$$P(y^{(i)} \mid x^{(i)}, \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right).$$

For notational convenience, we define

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix} \in \mathbf{R}^{m \times n} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbf{R}^m \qquad \vec{\varepsilon} = \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(m)} \end{bmatrix} \in \mathbf{R}^m.$$

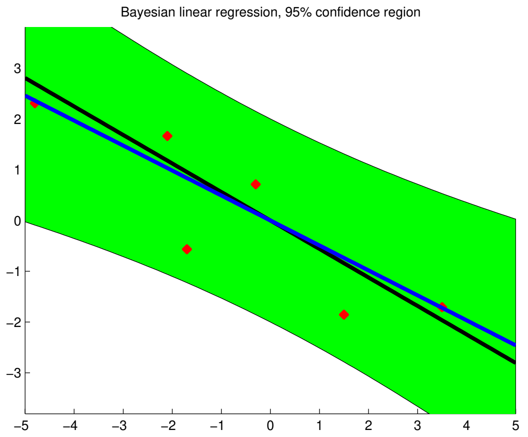


Figure 1: Bayesian linear regression for a one-dimensional linear regression problem, $y^{(i)} = \theta x^{(i)} + \epsilon^{(i)}$, with $\epsilon^{(i)} \sim \mathcal{N}(0,1)$ i.i.d. noise. The green region denotes the 95% confidence region for predictions of the model. Note that the (vertical) width of the green region is largest at the ends but narrowest in the middle. This region reflects the uncertain in the estimates for the parameter θ. In contrast, a classical linear regression model would display a confidence region of constant width, reflecting only the $\mathcal{N}(0,\sigma^2)$ noise in the outputs.

In Bayesian linear regression, we assume that a **prior distribution** over parameters is also given; a typical choice, for instance, is $\theta \sim \mathcal{N}(0,\tau^2 I)$. Using Bayes's rule, we obtain the **parameter posterior**,

$$p(\theta \mid S) = \frac{p(\theta)p(S \mid \theta)}{\int_{\theta'} p(\theta')p(S \mid \theta')d\theta'} = \frac{p(\theta)\prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)},\theta)}{\int_{\theta'} p(\theta')\prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)},\theta')d\theta'}. \tag{2}$$

Assuming the same noise model on testing points as on our training points, the "output" of Bayesian linear regression on a new test point x_* is not just a single guess "y_*", but rather an entire probability distribution over possible outputs, known as the **posterior predictive distribution**:

$$p(y_* \mid x_*, S) = \int_\theta p(y_* \mid x_*,\theta)p(\theta \mid S)d\theta. \tag{3}$$

For many types of models, the integrals in (2) and (3) are difficult to compute, and hence, we often resort to approximations, such as MAP estimation (see course lecture notes on "Regularization and Model Selection").

In the case of Bayesian linear regression, however, the integrals actually are tractable! In particular, for Bayesian linear regression, one can show (after much work!) that

$$\theta \mid S \sim \mathcal{N}\left(\frac{1}{\sigma^2}A^{-1}X^T\vec{y}, A^{-1}\right)$$
$$y_* \mid x_*, S \sim \mathcal{N}\left(\frac{1}{\sigma^2}x_*^T A^{-1}X^T\vec{y}, x_*^T A^{-1}x_* + \sigma^2\right)$$

where $A = \frac{1}{\sigma^2} X^T X + \frac{1}{\tau^2} I$. The derivation of these formulas is somewhat involved.[6] Nonetheless, from these equations, we get at least a flavor of what Bayesian methods are all about: the posterior distribution over the test output y_* for a test input x_* is a Gaussian distribution—this distribution reflects the uncertainty in our predictions $y_* = \theta^T x_* + \varepsilon_*$ arising from both the randomness in ε_* and the uncertainty in our choice of parameters θ. In contrast, classical probabilistic linear regression models estimate parameters θ directly from the training data but provide no estimate of how reliable these learned parameters may be (see Figure 1).

3 Gaussian processes

As described in Section 1, multivariate Gaussian distributions are useful for modeling finite collections of real-valued variables because of their nice analytical properties. **Gaussian processes** are the extension of multivariate Gaussians to infinite-sized collections of real-valued variables. In particular, this extension will allow us to think of Gaussian processes as distributions not just over random vectors but in fact distributions over **random functions**.[7]

3.1 Probability distributions over functions with finite domains

To understand how one might paramterize probability distributions over functions, consider the following simple example. Let $\mathcal{X} = \{x_1, \dots, x_m\}$ be any finite set of elements. Now, consider the set $\mathcal{H}$ of all possible functions mapping from $\mathcal{X}$ to $\mathbf{R}$. For instance, one example of a function $f_0(\cdot) \in \mathcal{H}$ is given by

$$f_0(x_1) = 5, \quad f_0(x_2) = 2.3, \quad f_0(x_2) = -7, \quad \dots, \quad f_0(x_{m-1}) = -\pi, \quad f_0(x_m) = 8.$$

Since the domain of any $f(\cdot) \in \mathcal{H}$ has only m elements, we can always represent $f(\cdot)$ compactly as an m-dimensional vector, $\vec{f} = \begin{bmatrix} f(x_1) & f(x_2) & \cdots & f(x_m) \end{bmatrix}^T$. In order to specify a probability distribution over functions $f(\cdot) \in \mathcal{H}$, we must associate some "probability density" with each function in $\mathcal{H}$. One natural way to do this is to exploit the one-to-one correspondence between functions $f(\cdot) \in \mathcal{H}$ and their vector representations, $\vec{f}$. In particular, if we specify that $\vec{f} \sim \mathcal{N}(\vec{\mu}, \sigma^2 I)$, then this in turn implies a probability distribution over functions $f(\cdot)$, whose probability density function is given by

$$p(h) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2} (f(x_i) - \mu_i)^2 \right).$$

[6] For the complete derivation, see, for instance, [1]. Alternatively, read the Appendices, which gives a number of arguments based on the "completion-of-squares" trick, and derive this formula yourself!

[7] Let $\mathcal{H}$ be a class of functions mapping from $\mathcal{X} \to \mathcal{Y}$. A random function $f(\cdot)$ from $\mathcal{H}$ is a function which is randomly drawn from $\mathcal{H}$, according to some probability distribution over $\mathcal{H}$. One potential source of confusion is that you may be tempted to think of random functions as functions whose outputs are in some way stochastic; this is not the case. Instead, a random function $f(\cdot)$, once selected from $\mathcal{H}$ probabilistically, implies a deterministic mapping from inputs in $\mathcal{X}$ to outputs in $\mathcal{Y}$.

In the example above, we showed that probability distributions over functions with finite domains can be represented using a finite-dimensional multivariate Gaussian distribution over function outputs $f(x_1), \ldots, f(x_m)$ at a finite number of input points $x_1, \ldots, x_m$. How can we specify probability distributions over functions when the domain size may be infinite? For this, we turn to a fancier type of probability distribution known as a Gaussian process.

3.2 Probability distributions over functions with infinite domains

A stochastic process is a collection of random variables, $\{f(x) : x \in \mathcal{X}\}$, indexed by elements from some set $\mathcal{X}$, known as the index set.[8] A **Gaussian process** is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution.

In particular, a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ is said to be drawn from a Gaussian process with **mean function** $m(\cdot)$ and **covariance function** $k(\cdot, \cdot)$ if for any finite set of elements $x_1, \ldots, x_m \in \mathcal{X}$, the associated finite set of random variables $f(x_1), \ldots, f(x_m)$ have distribution,

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_m) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_m) \end{bmatrix} \right).$$

We denote this using the notation,

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)).$$

Observe that the mean function and covariance function are aptly named since the above properties imply that

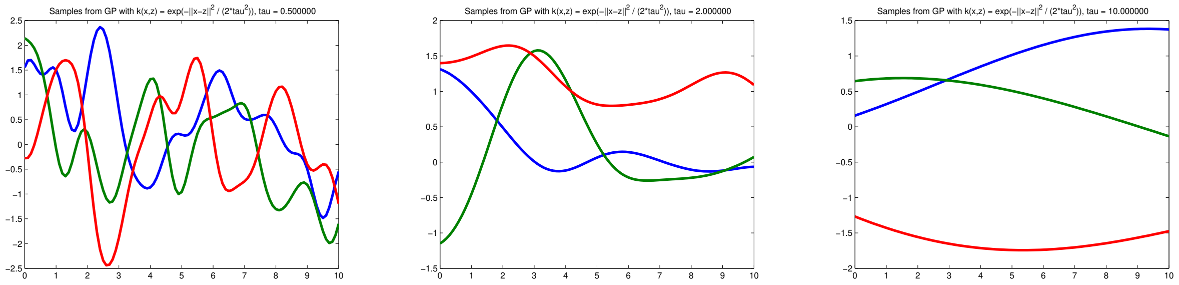
$$\begin{aligned} m(x) &= E[x] \\ k(x, x') &= E[(x - m(x))(x' - m(x')). \end{aligned}$$

for any $x, x' \in \mathcal{X}$.

Intuitively, one can think of a function $f(\cdot)$ drawn from a Gaussian process prior as an extremely high-dimensional vector drawn from an extremely high-dimensional multivariate Gaussian. Here, each dimension of the Gaussian corresponds to an element x from the index set $\mathcal{X}$, and the corresponding component of the random vector represents the value of $f(x)$. Using the marginalization property for multivariate Gaussians, we can obtain the marginal multivariate Gaussian density corresponding to any finite subcollection of variables.

What sort of functions $m(\cdot)$ and $k(\cdot, \cdot)$ give rise to valid Gaussian processes? In general, any real-valued function $m(\cdot)$ is acceptable, but for $k(\cdot, \cdot)$, it must be the case that for any

[8] Often, when $\mathcal{X} = \mathbf{R}$, one can interpret the indices $x \in \mathcal{X}$ as representing times, and hence the variables $f(x)$ represent the temporal evolution of some random quantity over time. In the models that are used for Gaussian process regression, however, the index set is taken to be the input space of our regression problem.



(a) (b) (c)

Figure 2: Samples from a zero-mean Gaussian process prior with $k_{SE}(\cdot, \cdot)$ covariance function, using (a) $\tau = 0.5$, (b) $\tau = 2$, and (c) $\tau = 10$. Note that as the bandwidth parameter τ increases, then points which are farther away will have higher correlations than before, and hence the sampled functions tend to be smoother overall.

set of elements $x_1, \ldots, x_m \in \mathcal{X}$, the resulting matrix

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_m) \end{bmatrix}$$

is a valid covariance matrix corresponding to some multivariate Gaussian distribution. A standard result in probability theory states that this is true provided that K is positive semidefinite. Sound familiar?

The positive semidefiniteness requirement for covariance matrices computed based on arbitrary input points is, in fact, identical to Mercer's condition for kernels! A function $k(\cdot, \cdot)$ is a valid kernel provided the resulting kernel matrix K defined as above is always positive semidefinite for any set of input points $x_1, \ldots, x_m \in \mathcal{X}$. Gaussian processes, therefore, are kernel-based probability distributions in the sense that any valid kernel function can be used as a covariance function!

3.3 The squared exponential kernel

In order to get an intuition for how Gaussian processes work, consider a simple zero-mean Gaussian process,

$$f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot)).$$

defined for functions $h : \mathcal{X} \to \mathbf{R}$ where we take $\mathcal{X} = \mathbf{R}$. Here, we choose the kernel function $k(\cdot, \cdot)$ to be the **squared exponential**[9] kernel function, defined as

$$k_{SE}(x, x') = \exp\left(-\frac{1}{2\tau^2}||x - x'||^2\right)$$

[9] In the context of SVMs, we called this the Gaussian kernel; to avoid confusion with "Gaussian" processes, we refer to this kernel here as the squared exponential kernel, even though the two are formally identical.

for some $\tau > 0$. What do random functions sampled from this Gaussian process look like?

In our example, since we use a zero-mean Gaussian process, we would expect that for the function values from our Gaussian process will tend to be distributed around zero. Furthermore, for any pair of elements $x, x' \in \mathcal{X}$.

- $f(x)$ and $f(x')$ will tend to have high covariance x and x' are "nearby" in the input space (i.e., $||x - x'|| = |x - x'| \approx 0$, so $\exp(-\frac{1}{2\tau^2}||x - x'||^2) \approx 1$).
- $f(x)$ and $f(x')$ will tend to have low covariance when x and x' are "far apart" (i.e., $||x - x'|| \gg 0$, so $\exp(-\frac{1}{2\tau^2}||x - x'||^2) \approx 0$).

More simply stated, functions drawn from a zero-mean Gaussian process prior with the squared exponential kernel will tend to be "locally smooth" with high probability; i.e., nearby function values are highly correlated, and the correlation drops off as a function of distance in the input space (see Figure 2).

4 Gaussian process regression

As discussed in the last section, Gaussian processes provide a method for modelling probability distributions over functions. Here, we discuss how probability distributions over functions can be used in the framework of Bayesian regression.

4.1 The Gaussian process regression model

Let $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ be a training set of i.i.d. examples from some unknown distribution. In the Gaussian process regression model,

$$y^{(i)} = f(x^{(i)}) + \varepsilon^{(i)}, \qquad i = 1, \ldots, m$$

where the $\varepsilon^{(i)}$ are i.i.d. "noise" variables with independent $\mathcal{N}(0, \sigma^2)$ distributions. Like in Bayesian linear regression, we also assume a **prior distribution** over functions $f(\cdot)$; in particular, we assume a zero-mean Gaussian process prior,

$$f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))$$

for some valid covariance function $k(\cdot, \cdot)$.

Now, let $T = \{(x_*^{(i)}, y_*^{(i)})\}_{i=1}^{m_*}$ be a set of i.i.d. testing points drawn from the same unknown distribution as S.[10] For notational convenience, we define

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix} \in \mathbf{R}^{m\times n} \qquad \vec{f} = \begin{bmatrix} f(x^{(1)}) \\ f(x^{(2)}) \\ \vdots \\ f(x^{(m)}) \end{bmatrix}, \quad \vec{\varepsilon} = \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(m)} \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbf{R}^{m},$$

$$X_* = \begin{bmatrix} — & (x_*^{(1)})^T & — \\ — & (x_*^{(2)})^T & — \\ & \vdots & \\ — & (x_*^{(m_*)})^T & — \end{bmatrix} \in \mathbf{R}^{m_*\times n} \quad \vec{f}_* = \begin{bmatrix} f(x_*^{(1)}) \\ f(x_*^{(2)}) \\ \vdots \\ f(x_*^{(m_*)}) \end{bmatrix}, \quad \vec{\varepsilon}_* = \begin{bmatrix} \varepsilon_*^{(1)} \\ \varepsilon_*^{(2)} \\ \vdots \\ \varepsilon_*^{(m_*)} \end{bmatrix}, \quad \vec{y}_* = \begin{bmatrix} y_*^{(1)} \\ y_*^{(2)} \\ \vdots \\ y_*^{(m_*)} \end{bmatrix} \in \mathbf{R}^{m_*}.$$

Given the training data S, the prior $p(h)$, and the testing inputs X_*, how can we compute the posterior predictive distribution over the testing outputs $\vec{y}_*$? For Bayesian linear regression in Section 2, we used Bayes's rule in order to compute the paramter posterior, which we then used to compute posterior predictive distribution $p(y_* \mid x_*, S)$ for a new test point x_*. For Gaussian process regression, however, it turns out that an even simpler solution exists!

4.2 Prediction

Recall that for any function $f(\cdot)$ drawn from our zero-mean Gaussian process prior with covariance function $k(\cdot,\cdot)$, the marginal distribution over any set of input points belonging to $\mathcal{X}$ must have a joint multivariate Gaussian distribution. In particular, this must hold for the training and test points, so we have

$$\left.\begin{bmatrix} \vec{f} \\ \vec{f}_* \end{bmatrix} \right| X, X_* \sim \mathcal{N}\left(\vec{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right),$$

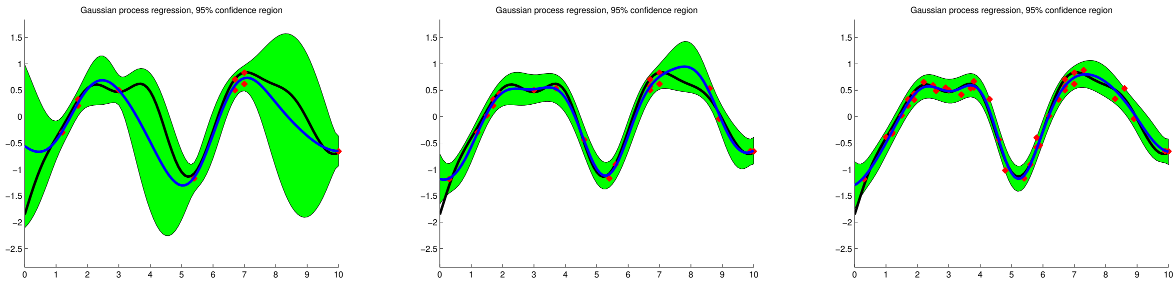
where

$$\begin{aligned} \vec{f} \in \mathbf{R}^m & \text{ such that } \vec{f} = \begin{bmatrix} f(x^{(1)}) & \cdots & f(x^{(m)}) \end{bmatrix}^T \\ \vec{f}_* \in \mathbf{R}^{m_*} & \text{ such that } \vec{f}_* = \begin{bmatrix} f(x_*^{(1)}) & \cdots & f(x_*^{(m)}) \end{bmatrix}^T \\ K(X,X) \in \mathbf{R}^{m\times m} & \text{ such that } (K(X,X))_{ij} = k(x^{(i)}, x^{(j)}) \\ K(X,X_*) \in \mathbf{R}^{m\times m_*} & \text{ such that } (K(X,X_*))_{ij} = k(x^{(i)}, x_*^{(j)}) \\ K(X_*,X) \in \mathbf{R}^{m_*\times m} & \text{ such that } (K(X_*,X))_{ij} = k(x_*^{(i)}, x^{(j)}) \\ K(X_*,X_*) \in \mathbf{R}^{m_*\times m_*} & \text{ such that } (K(X_*,X_*))_{ij} = k(x_*^{(i)}, x_*^{(j)}). \end{aligned}$$

From our i.i.d. noise assumption, we have that

$$\begin{bmatrix} \vec{\varepsilon} \\ \vec{\varepsilon}_* \end{bmatrix} \sim \mathcal{N}\left(\vec{0}, \begin{bmatrix} \sigma^2 I & \vec{0} \\ \vec{0}^T & \sigma^2 I \end{bmatrix}\right).$$

[10] We assume also that T are S are mutually independent.



(a) (b) (c)

Figure 3: Gaussian process regression using a zero-mean Gaussian process prior with $k_{SE}(\cdot,\cdot)$ covariance function (where $\tau = 0.1$), with noise level $\sigma = 1$, and (a) $m = 10$, (b) $m = 20$, and (c) $m = 40$ training examples. The blue line denotes the mean of the posterior predictive distribution, and the green shaded region denotes the 95% confidence region based on the model's variance estimates. As the number of training examples increases, the size of the confidence region shrinks to reflect the diminishing uncertainty in the model estimates. Note also that in panel (a), the 95% confidence region shrinks near training points but is much larger far away from training points, as one would expect.

The sums of independent Gaussian random variables is also Gaussian, so

$$\left.\begin{bmatrix}\vec{y}\\ \vec{y_*}\end{bmatrix}\right| X, X_* = \begin{bmatrix}\vec{f}\\ \vec{f_*}\end{bmatrix} + \begin{bmatrix}\vec{\varepsilon}\\ \vec{\varepsilon_*}\end{bmatrix} \sim \mathcal{N}\left(\vec{0}, \begin{bmatrix}K(X,X)+\sigma^2 I & K(X,X_*)\\ K(X_*,X) & K(X_*,X_*)+\sigma^2 I\end{bmatrix}\right).$$

Now, using the rules for conditioning Gaussians, it follows that

$$\vec{y_*} \mid \vec{y}, X, X_* \sim \mathcal{N}(\mu^*, \Sigma^*)$$

where

$$\begin{aligned}\mu^* &= K(X_*,X)\left(K(X,X)+\sigma^2 I\right)^{-1}\vec{y}\\ \Sigma^* &= K(X_*,X_*)+\sigma^2 I - K(X_*,X)\left(K(X,X)+\sigma^2 I\right)^{-1}K(X,X_*).\end{aligned}$$

And that's it! Remarkably, performing prediction in a Gaussian process regression model is very simple, despite the fact that Gaussian processes in themselves are fairly complicated![11]

5 Summary

We close our discussion of our Gaussian processes by pointing out some reasons why Gaussian processes are an attractive model for use in regression problems and in some cases may be preferable to alternative models (such as linear and locally-weighted linear regression):

[11] Interestingly, it turns out that Bayesian linear regression, when "kernelized" in the proper way, turns out to be exactly equivalent to Gaussian process regression! But the derivation of the posterior predictive distribution is far more complicated for Bayesian linear regression, and the effort needed to kernelize the algorithm is even greater. The Gaussian process perspective is certainly much easier!

1. As Bayesian methods, Gaussian process models allow one to quantify uncertainty in predictions resulting not just from intrinsic noise in the problem but also the errors in the parameter estimation procedure. Furthermore, many methods for model selection and hyperparameter selection in Bayesian methods are immediately applicable to Gaussian processes (though we did not address any of these advanced topics here).

2. Like locally-weighted linear regression, Gaussian process regression is non-parametric and hence can model essentially arbitrary functions of the input points.

3. Gaussian process regression models provide a natural way to introduce kernels into a regression modeling framework. By careful choice of kernels, Gaussian process regression models can sometimes take advantage of structure in the data (though, we also did not examine this issue here).

4. Gaussian process regression models, though perhaps somewhat tricky to understand conceptually, nonetheless lead to simple and straightforward linear algebra implementations.

References

[1] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006. Online: `http://www.gaussianprocess.org/gpml/`

Appendix A.1

In this example, we show how the normalization property for multivariate Gaussians can be used to compute rather intimidating multidimensional integrals without performing any real calculus! Suppose you wanted to compute the following multidimensional integral,

$$I(A,b,c) = \int_x \exp\left(-\frac{1}{2}x^TAx - x^Tb - c\right)dx,$$

for some $A \in \mathbf{S}^m_{++}$, $b \in \mathbf{R}^m$, and $c \in \mathbf{R}$. Although one could conceivably perform the multidimensional integration directly (good luck!), a much simpler line of reasoning is based on a mathematical trick known as "completion-of-squares." In particular,

$$\begin{aligned}
I(A,b,c) &= \exp(-c) \cdot \int_x \exp\left(-\frac{1}{2}x^TAx - x^TAA^{-1}b\right)dx \\
&= \exp(-c) \cdot \int_x \exp\left(-\frac{1}{2}(x - A^{-1}b)^TA(x - A^{-1}b) - b^TA^{-1}b\right)dx \\
&= \exp\left(-c - b^TA^{-1}b\right) \cdot \int_x \exp\left(-\frac{1}{2}(x - A^{-1}b)^TA(x - A^{-1}b)\right)dx.
\end{aligned}$$

Defining $\mu = A^{-1}b$ and $\Sigma = A^{-1}$, it follows that $I(A,b,c)$ is equal to

$$\frac{(2\pi)^{m/2}|\Sigma|^{1/2}}{\exp(c + b^TA^{-1}b)} \cdot \left[\frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}}\int_x \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)dx\right].$$

However, the term in brackets is identical in form to the integral of a multivariate Gaussian! Since we know that a Gaussian density normalizes, it follows that the term in brackets is equal to 1. Therefore,

$$I(A,b,c) = \frac{(2\pi)^{m/2}|A^{-1}|^{1/2}}{\exp(c + b^TA^{-1}b)}.$$

Appendix A.2

We derive the form of the distribution of x_A given x_B; the other result follows immediately by symmetry. Note that

$$\begin{aligned}
p(x_A \mid x_B) &= \frac{1}{\int_{x_A} p(x_A, x_B; \mu, \Sigma)dx_A} \cdot \left[\frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)\right] \\
&= \frac{1}{Z_1}\exp\left\{-\frac{1}{2}\left(\begin{bmatrix}x_A \\ x_B\end{bmatrix} - \begin{bmatrix}\mu_A \\ \mu_B\end{bmatrix}\right)^T \begin{bmatrix}V_{AA} & V_{AB} \\ V_{BA} & V_{BB}\end{bmatrix} \left(\begin{bmatrix}x_A \\ x_B\end{bmatrix} - \begin{bmatrix}\mu_A \\ \mu_B\end{bmatrix}\right)\right\}
\end{aligned}$$

where Z_1 is a proportionality constant which does not depend on x_A, and

$$\Sigma^{-1} = V = \begin{bmatrix}V_{AA} & V_{AB} \\ V_{BA} & V_{BB}\end{bmatrix}.$$

To simplify this expression, observe that

$$\left(\begin{bmatrix}x_A\\x_B\end{bmatrix}-\begin{bmatrix}\mu_A\\\mu_B\end{bmatrix}\right)^T\begin{bmatrix}V_{AA} & V_{AB}\\V_{BA} & V_{BB}\end{bmatrix}\left(\begin{bmatrix}x_A\\x_B\end{bmatrix}-\begin{bmatrix}\mu_A\\\mu_B\end{bmatrix}\right)$$
$$= (x_A-\mu_A)^TV_{AA}(x_A-\mu_A)+(x_A-\mu_A)^TV_{AB}(x_B-\mu_B)$$
$$+(x_B-\mu_B)^TV_{BA}(x_A-\mu_A)+(x_B-\mu_B)^TV_{BB}(x_B-\mu_B).$$

Retaining only terms dependent on x_A (and using the fact that $V_{AB}=V_{BA}^T$), we have

$$p(x_A \mid x_B)=\frac{1}{Z_2}\exp\left(-\frac{1}{2}\left[x_A^TV_{AA}x_A-2x_A^TV_{AA}\mu_A+2x_A^TV_{AB}(x_B-\mu_B)\right]\right)$$

where Z_2 is a new proportionality constant which again does not depend on x_A. Finally, using the "completion-of-squares" argument (see Appendix A.1), we have

$$p(x_A \mid x_B)=\frac{1}{Z_3}\exp\left(-\frac{1}{2}(x_A-\mu')^TV_{AA}(x_A-\mu')\right)$$

where Z_3 is again a new proportionality constant not depending on x_A, and where $\mu'=\mu_A-V_{AA}^{-1}V_{AB}(x_B-\mu_B)$. This last statement shows that the distribution of x_A, conditioned on x_B, again has the form of a multivariate Gaussian. In fact, from the normalization property, it follows immediately that

$$x_A \mid x_B \sim \mathcal{N}(\mu_A-V_{AA}^{-1}V_{AB}(x_B-\mu_B), V_{AA}^{-1}).$$

To complete the proof, we simply note that

$$\begin{bmatrix}V_{AA} & V_{AB}\\V_{BA} & V_{BB}\end{bmatrix}=\begin{bmatrix}(\Sigma_{AA}-\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})^{-1} & -(\Sigma_{AA}-\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})^{-1}\Sigma_{AB}\Sigma_{BB}^{-1}\\-\Sigma_{BB}^{-1}\Sigma_{BA}(\Sigma_{AA}-\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})^{-1} & (\Sigma_{BB}-\Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB})^{-1}\end{bmatrix}$$

follows from standard formulas for the inverse of a partitioned matrix. Substituting the relevant blocks into the previous expression gives the desired result. □

Appendix A.3

In this section, we present an alternative (and easier) derivation of the conditional distribution of multivariate Gaussian distribution. Note that, as in Appendix A.2, we can write $p(x_A \mid x_B)$ as following:

$$p(x_A \mid x_B)=\frac{1}{\int_{x_A}p(x_A,x_B;\mu,\Sigma)dx_A}\cdot\left[\frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)\right] \quad (4)$$
$$=\frac{1}{Z_1}\exp\left\{-\frac{1}{2}\left(\begin{bmatrix}x_A-\mu_A\\x_B-\mu_B\end{bmatrix}\right)^T\begin{bmatrix}V_{AA} & V_{AB}\\V_{BA} & V_{BB}\end{bmatrix}\begin{bmatrix}x_A-\mu_A\\x_B-\mu_B\end{bmatrix}\right\} \quad (5)$$

where Z_1 is a proportionality constant which does not depend on x_A.

This derivation uses an additional assumption that the conditional distribution is a multivariate Gaussian distribution; in other words, we assume that $p(x_A \mid x_B) \sim \mathcal{N}(\mu^*, \Sigma^*)$ for some μ^*, Σ^*. (Alternatively, you can think about this derivation as another way of finding "completion-of-squares".)

The key intuition in this derivation is that $p(x_A \mid x_B)$ will be maximized when $x_A = \mu^* \triangleq x_A^*$. To maximize $p(x_A \mid x_B)$, we compute the gradient of $\log p(x_A \mid x_B)$ w.r.t. x_A and set it to zero. Using Equation (5), we have

$$\nabla_{x_A} \log p(x_A \mid x_B)|_{x_A = x_A^*} \tag{6}$$
$$= -V_{AA}(x_A^* - \mu_A) - V_{AB}(x_B - \mu_B) \tag{7}$$
$$= 0. \tag{8}$$

This implies that

$$\mu^* = x_A^* = \mu_A - V_{AA}^{-1} V_{AB}(x_B - \mu_B). \tag{9}$$

Similarly, we use the fact that the inverse covariance matrix of a Gaussian distribution $p(\cdot)$ is a negative Hessian of $\log p(\cdot)$. In other words, the inverse covariance matrix of a Gaussian distribution $p(x_A|x_B)$ is a negative Hessian of $\log p(x_A|x_B)$. Using Equation (5), we have

$$\Sigma^{*-1} = -\nabla_{x_A} \nabla_{x_A}^T \log p(x_A \mid x_B) \tag{10}$$
$$= V_{AA}. \tag{11}$$

Therefore, we get

$$\Sigma^* = V_{AA}^{-1}. \tag{12}$$

□

The Multivariate Gaussian Distribution

Chuong B. Do

October 10, 2008

A vector-valued random variable $X = \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}^T$ is said to have a **multivariate normal (or Gaussian) distribution** with mean $\mu \in \mathbf{R}^n$ and covariance matrix $\Sigma \in \mathbf{S}^n_{++}$[1] if its probability density function[2] is given by

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

We write this as $X \sim \mathcal{N}(\mu, \Sigma)$. In these notes, we describe multivariate Gaussians and some of their basic properties.

1 Relationship to univariate Gaussians

Recall that the density function of a **univariate normal (or Gaussian) distribution** is given by

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right).$$

Here, the argument of the exponential function, $-\frac{1}{2\sigma^2}(x-\mu)^2$, is a quadratic function of the variable x. Furthermore, the parabola points downwards, as the coefficient of the quadratic term is negative. The coefficient in front, $\frac{1}{\sqrt{2\pi}\sigma}$, is a constant that does not depend on x; hence, we can think of it as simply a "normalization factor" used to ensure that

$$\frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) = 1.$$

[1] Recall from the section notes on linear algebra that $\mathbf{S}^n_{++}$ is the space of symmetric positive definite $n \times n$ matrices, defined as

$$\mathbf{S}^n_{++} = \left\{A \in \mathbf{R}^{n\times n} : A = A^T \text{ and } x^TAx > 0 \text{ for all } x \in \mathbf{R}^n \text{ such that } x \neq 0\right\}.$$

[2] In these notes, we use the notation $p(\bullet)$ to denote density functions, instead of $f_X(\bullet)$ (as in the section notes on probability theory).

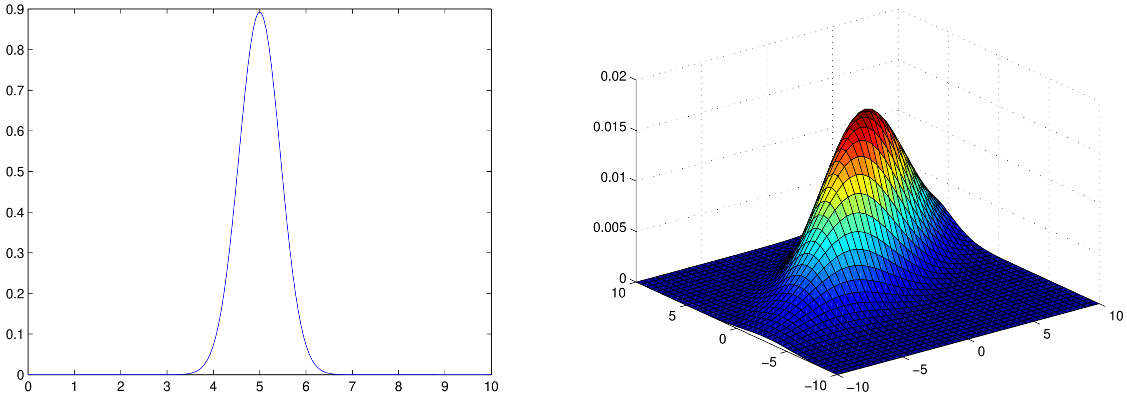


Figure 1: The figure on the left shows a univariate Gaussian density for a single variable X. The figure on the right shows a multivariate Gaussian density over two variables X_1 and X_2.

In the case of the multivariate Gaussian density, the argument of the exponential function, $-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$, is a **quadratic form** in the vector variable x. Since Σ is positive definite, and since the inverse of any positive definite matrix is also positive definite, then for any non-zero vector z, $z^T\Sigma^{-1}z > 0$. This implies that for any vector $x \neq \mu$,

$$\begin{aligned}(x-\mu)^T\Sigma^{-1}(x-\mu) &> 0 \\ -\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) &< 0.\end{aligned}$$

Like in the univariate case, you can think of the argument of the exponential function as being a downward opening quadratic bowl. The coefficient in front (i.e., $\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}$) has an even more complicated form than in the univariate case. However, it still does not depend on x, and hence it is again simply a normalization factor used to ensure that

$$\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)dx_1dx_2\cdots dx_n = 1.$$

2 The covariance matrix

The concept of the **covariance matrix** is vital to understanding multivariate Gaussian distributions. Recall that for a pair of random variables X and Y, their **covariance** is defined as

$$Cov[X,Y] = E[(X-E[X])(Y-E[Y])] = E[XY] - E[X]E[Y].$$

When working with multiple variables, the covariance matrix provides a succinct way to summarize the covariances of all pairs of variables. In particular, the covariance matrix, which we usually denote as Σ, is the $n \times n$ matrix whose (i,j)th entry is $Cov[X_i, X_j]$.

The following proposition (whose proof is provided in the Appendix A.1) gives an alternative way to characterize the covariance matrix of a random vector X:

Proposition 1. *For any random vector X with mean μ and covariance matrix Σ,*

$$\Sigma = E[(X-\mu)(X-\mu)^T] = E[XX^T] - \mu\mu^T. \tag{1}$$

In the definition of multivariate Gaussians, we required that the covariance matrix Σ be symmetric positive definite (i.e., $\Sigma \in \mathbf{S}^n_{++}$). Why does this restriction exist? As seen in the following proposition, the covariance matrix of *any* random vector must always be symmetric positive semidefinite:

Proposition 2. *Suppose that Σ is the covariance matrix corresponding to some random vector X. Then Σ is symmetric positive semidefinite.*

Proof. The symmetry of Σ follows immediately from its definition. Next, for any vector $z \in \mathbf{R}^n$, observe that

$$\begin{aligned}
z^T\Sigma z &= \sum_{i=1}^n \sum_{j=1}^n (\Sigma_{ij} z_i z_j) && (2)\\
&= \sum_{i=1}^n \sum_{j=1}^n (Cov[X_i, X_j] \cdot z_i z_j)\\
&= \sum_{i=1}^n \sum_{j=1}^n (E[(X_i - E[X_i])(X_j - E[X_j])] \cdot z_i z_j)\\
&= E\left[\sum_{i=1}^n \sum_{j=1}^n (X_i - E[X_i])(X_j - E[X_j]) \cdot z_i z_j\right]. && (3)
\end{aligned}$$

Here, (2) follows from the formula for expanding a quadratic form (see section notes on linear algebra), and (3) follows by linearity of expectations (see probability notes).

To complete the proof, observe that the quantity inside the brackets is of the form $\sum_i \sum_j x_i x_j z_i z_j = (x^T z)^2 \geq 0$ (see problem set #1). Therefore, the quantity inside the expectation is always nonnegative, and hence the expectation itself must be nonnegative. We conclude that $z^T\Sigma z \geq 0$. □

From the above proposition it follows that Σ must be symmetric positive semidefinite in order for it to be a valid covariance matrix. However, in order for Σ^{-1} to exist (as required in the definition of the multivariate Gaussian density), then Σ must be invertible and hence full rank. Since any full rank symmetric positive semidefinite matrix is necessarily symmetric positive definite, it follows that Σ must be symmetric positive definite.

3 The diagonal covariance matrix case

To get an intuition for what a multivariate Gaussian is, consider the simple case where $n = 2$, and where the covariance matrix Σ is diagonal, i.e.,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

In this case, the multivariate Gaussian density has the form,

$$\begin{aligned} p(x; \mu, \Sigma) &= \frac{1}{2\pi \begin{vmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{vmatrix}^{1/2}} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}^{-1} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \right) \\ &= \frac{1}{2\pi (\sigma_1^2 \cdot \sigma_2^2 - 0 \cdot 0)^{1/2}} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \right), \end{aligned}$$

where we have relied on the explicit formula for the determinant of a 2 × 2 matrix[3], and the fact that the inverse of a diagonal matrix is simply found by taking the reciprocal of each diagonal entry. Continuing,

$$\begin{aligned} p(x; \mu, \Sigma) &= \frac{1}{2\pi \sigma_1 \sigma_2} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sigma_1^2}(x_1 - \mu_1) \\ \frac{1}{\sigma_2^2}(x_2 - \mu_2) \end{bmatrix} \right) \\ &= \frac{1}{2\pi \sigma_1 \sigma_2} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right) \\ &= \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 \right) \cdot \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right). \end{aligned}$$

The last equation we recognize to simply be the product of two independent Gaussian densities, one with mean μ_1 and variance σ_1^2, and the other with mean μ_2 and variance σ_2^2.

More generally, one can show that an n-dimensional Gaussian with mean $\mu \in \mathbf{R}^n$ and diagonal covariance matrix $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$ is the same as a collection of n independent Gaussian random variables with mean μ_i and variance σ_i^2, respectively.

4 Isocontours

Another way to understand a multivariate Gaussian conceptually is to understand the shape of its **isocontours**. For a function $f : \mathbf{R}^2 \to \mathbf{R}$, an isocontour is a set of the form

$$\left\{x \in \mathbf{R}^2 : f(x) = c\right\}.$$

for some $c \in \mathbf{R}$.[4]

[3] Namely, $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$.

[4] Isocontours are often also known as **level curves**. More generally, a **level set** of a function $f : \mathbf{R}^n \to \mathbf{R}$, is a set of the form $\left\{x \in \mathbf{R}^2 : f(x) = c\right\}$ for some $c \in \mathbf{R}$.

4.1 Shape of isocontours

What do the isocontours of a multivariate Gaussian look like? As before, let's consider the case where $n = 2$, and Σ is diagonal, i.e.,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

As we showed in the last section,

$$p(x; \mu, \Sigma) = \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right). \tag{4}$$

Now, let's consider the level set consisting of all points where $p(x; \mu, \Sigma) = c$ for some constant $c \in \mathbf{R}$. In particular, consider the set of all $x_1, x_2 \in \mathbf{R}$ such that

$$\begin{aligned}
c &= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right) \\
2\pi c\sigma_1\sigma_2 &= \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right) \\
\log(2\pi c\sigma_1\sigma_2) &= -\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \\
\log\left(\frac{1}{2\pi c\sigma_1\sigma_2} \right) &= \frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 + \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \\
1 &= \frac{(x_1 - \mu_1)^2}{2\sigma_1^2 \log\left(\frac{1}{2\pi c\sigma_1\sigma_2} \right)} + \frac{(x_2 - \mu_2)^2}{2\sigma_2^2 \log\left(\frac{1}{2\pi c\sigma_1\sigma_2} \right)}.
\end{aligned}$$

Defining

$$r_1 = \sqrt{2\sigma_1^2 \log\left(\frac{1}{2\pi c\sigma_1\sigma_2} \right)} \qquad r_2 = \sqrt{2\sigma_2^2 \log\left(\frac{1}{2\pi c\sigma_1\sigma_2} \right)},$$

it follows that

$$1 = \left(\frac{x_1 - \mu_1}{r_1} \right)^2 + \left(\frac{x_2 - \mu_2}{r_2} \right)^2. \tag{5}$$

Equation (5) should be familiar to you from high school analytic geometry: it is the equation of an **axis-aligned ellipse**, with center (μ_1, μ_2), where the x_1 axis has length $2r_1$ and the x_2 axis has length $2r_2$!

4.2 Length of axes

To get a better understanding of how the shape of the level curves vary as a function of the variances of the multivariate Gaussian distribution, suppose that we are interested in

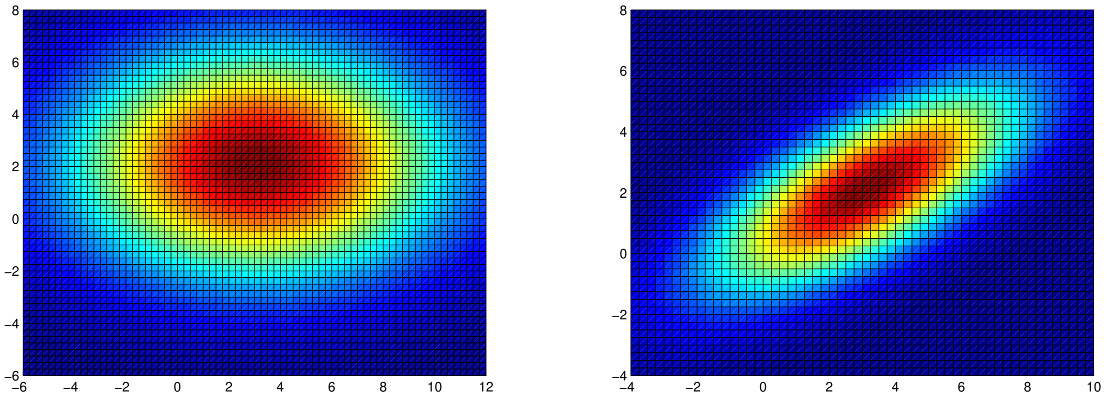


Figure 2:
The figure on the left shows a heatmap indicating values of the density function for an axis-aligned multivariate Gaussian with mean $\mu = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$ and diagonal covariance matrix $\Sigma = \begin{bmatrix} 25 & 0 \\ 0 & 9 \end{bmatrix}$. Notice that the Gaussian is centered at $(3, 2)$, and that the isocontours are all elliptically shaped with major/minor axis lengths in a 5:3 ratio. The figure on the right shows a heatmap indicating values of the density function for a non axis-aligned multivariate Gaussian with mean $\mu = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$ and covariance matrix $\Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$. Here, the ellipses are again centered at $(3, 2)$, but now the major and minor axes have been rotated via a linear transformation.

the values of r_1 and r_2 at which c is equal to a fraction $1/e$ of the peak height of Gaussian density.

First, observe that maximum of Equation (4) occurs where $x_1 = \mu_1$ and $x_2 = \mu_2$. Substituting these values into Equation (4), we see that the peak height of the Gaussian density is $\frac{1}{2\pi\sigma_1\sigma_2}$.

Second, we substitute $c = \frac{1}{e}\left(\frac{1}{2\pi\sigma_1\sigma_2}\right)$ into the equations for r_1 and r_2 to obtain

$$r_1 = \sqrt{2\sigma_1^2 \log\left(\frac{1}{2\pi\sigma_1\sigma_2 \cdot \frac{1}{e}\left(\frac{1}{2\pi\sigma_1\sigma_2}\right)}\right)} = \sigma_1\sqrt{2}$$

$$r_2 = \sqrt{2\sigma_2^2 \log\left(\frac{1}{2\pi\sigma_1\sigma_2 \cdot \frac{1}{e}\left(\frac{1}{2\pi\sigma_1\sigma_2}\right)}\right)} = \sigma_2\sqrt{2}.$$

From this, it follows that the axis length needed to reach a fraction $1/e$ of the peak height of the Gaussian density in the ith dimension grows in proportion to the standard deviation σ_i. Intuitively, this again makes sense: the smaller the variance of some random variable x_i, the more "tightly" peaked the Gaussian distribution in that dimension, and hence the smaller the radius r_i.

4.3 Non-diagonal case, higher dimensions

Clearly, the above derivations rely on the assumption that Σ is a diagonal matrix. However, in the non-diagonal case, it turns out that the picture is not all that different. Instead of being an axis-aligned ellipse, the isocontours turn out to be simply **rotated ellipses**. Furthermore, in the n-dimensional case, the level sets form geometrical structures known as **ellipsoids** in $\mathbf{R}^n$.

5 Linear transformation interpretation

In the last few sections, we focused primarily on providing an intuition for how multivariate Gaussians with diagonal covariance matrices behaved. In particular, we found that an n-dimensional multivariate Gaussian with diagonal covariance matrix could be viewed simply as a collection of n independent Gaussian-distributed random variables with means and variances μ_i and σ_i^2, respectvely. In this section, we dig a little deeper and provide a quantitative interpretation of multivariate Gaussians when the covariance matrix is not diagonal.

The key result of this section is the following theorem (see proof in Appendix A.2).

Theorem 1. *Let $X \sim \mathcal{N}(\mu, \Sigma)$ for some $\mu \in \mathbf{R}^n$ and $\Sigma \in \mathbf{S}^n_{++}$. Then, there exists a matrix $B \in \mathbf{R}^{n\times n}$ such that if we define $Z = B^{-1}(X - \mu)$, then $Z \sim \mathcal{N}(0, I)$.*

To understand the meaning of this theorem, note that if $Z \sim \mathcal{N}(0, I)$, then using the analysis from Section 4, Z can be thought of as a collection of n independent standard normal random variables (i.e., $Z_i \sim \mathcal{N}(0, 1)$). Furthermore, if $Z = B^{-1}(X - \mu)$ then $X = BZ + \mu$ follows from simple algebra.

Consequently, the theorem states that any random variable X with a multivariate Gaussian distribution can be interpreted as the result of applying a linear transformation ($X = BZ + \mu$) to some collection of n independent standard normal random variables (Z).

Appendix A.1

Proof. We prove the first of the two equalities in (1); the proof of the other equality is similar.

$$\begin{aligned}
\Sigma &= \begin{bmatrix} Cov[X_1, X_1] & \cdots & Cov[X_1, X_n] \\ \vdots & \ddots & \vdots \\ Cov[X_n, X_1] & \cdots & Cov[X_n, X_n] \end{bmatrix} \\
&= \begin{bmatrix} E[(X_1 - \mu_1)^2] & \cdots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ \vdots & \ddots & \vdots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & \cdots & E[(X_n - \mu_n)^2] \end{bmatrix} \\
&= E \begin{bmatrix} (X_1 - \mu_1)^2 & \cdots & (X_1 - \mu_1)(X_n - \mu_n) \\ \vdots & \ddots & \vdots \\ (X_n - \mu_n)(X_1 - \mu_1) & \cdots & (X_n - \mu_n)^2 \end{bmatrix} && (6) \\
&= E \left[\begin{bmatrix} X_1 - \mu_1 \\ \vdots \\ X_n - \mu_n \end{bmatrix} \begin{bmatrix} X_1 - \mu_1 & \cdots & X_n - \mu_n \end{bmatrix} \right] && (7) \\
&= E\left[(X - \mu)(X - \mu)^T \right].
\end{aligned}$$

Here, (6) follows from the fact that the expectation of a matrix is simply the matrix found by taking the componentwise expectation of each entry. Also, (7) follows from the fact that for any vector $z \in \mathbf{R}^n$,

$$zz^T = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \begin{bmatrix} z_1 & z_2 & \cdots z_n \end{bmatrix} = \begin{bmatrix} z_1 z_1 & z_1 z_2 & \cdots & z_1 z_n \\ z_2 z_1 & z_2 z_2 & \cdots & z_2 z_n \\ \vdots & \vdots & \ddots & \vdots \\ z_n z_1 & z_n z_2 & \cdots & z_n z_n \end{bmatrix}.$$

□

Appendix A.2

We restate the theorem below:

Theorem 1. *Let $X \sim \mathcal{N}(\mu, \Sigma)$ for some $\mu \in \mathbf{R}^n$ and $\Sigma \in \mathbf{S}^n_{++}$. Then, there exists a matrix $B \in \mathbf{R}^{n \times n}$ such that if we define $Z = B^{-1}(X - \mu)$, then $Z \sim \mathcal{N}(0, I)$.*

The derivation of this theorem requires some advanced linear algebra and probability theory and can be skipped for the purposes of this class. Our argument will consist of two parts. First, we will show that the covariance matrix Σ can be factorized as $\Sigma = BB^T$ for some invertible matrix B. Second, we will perform a "change-of-variable" from X to a different vector valued random variable Z using the relation $Z = B^{-1}(X - \mu)$.

Step 1: Factorizing the covariance matrix. Recall the following two properties of symmetric matrices from the notes on linear algebra[5]:

1. Any real symmetric matrix $A \in \mathbf{R}^{n\times n}$ can always be represented as $A = U\Lambda U^T$, where U is a full rank orthogonal matrix containing of the eigenvectors of A as its columns, and Λ is a diagonal matrix containing A's eigenvalues.

2. If A is symmetric positive definite, all its eigenvalues are positive.

Since the covariance matrix Σ is positive definite, using the first fact, we can write $\Sigma = U\Lambda U^T$ for some appropriately defined matrices U and Λ. Using the second fact, we can define $\Lambda^{1/2} \in \mathbf{R}^{n\times n}$ to be the diagonal matrix whose entries are the square roots of the corresponding entries from Λ. Since $\Lambda = \Lambda^{1/2}(\Lambda^{1/2})^T$, we have

$$\Sigma = U\Lambda U^T = U\Lambda^{1/2}(\Lambda^{1/2})^T U^T = U\Lambda^{1/2}(U\Lambda^{1/2})^T = BB^T,$$

where $B = U\Lambda^{1/2}$.[6] In this case, then $\Sigma^{-1} = B^{-T}B^{-1}$, so we can rewrite the standard formula for the density of a multivariate Gaussian as

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|BB^T|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T B^{-T}B^{-1}(x-\mu)\right). \tag{8}$$

Step 2: Change of variables. Now, define the vector-valued random variable $Z = B^{-1}(X-\mu)$. A basic formula of probability theory, which we did not introduce in the section notes on probability theory, is the "change-of-variables" formula for relating vector-valued random variables:

> Suppose that $X = \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}^T \in \mathbf{R}^n$ is a vector-valued random variable with joint density function $f_X : \mathbf{R}^n \to \mathbf{R}$. If $Z = H(X) \in \mathbf{R}^n$ where H is a bijective, differentiable function, then Z has joint density $f_Z : \mathbf{R}^n \to \mathbf{R}$, where
>
> $$f_Z(z) = f_X(x) \cdot \left| \det\left(\begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n}{\partial z_1} & \cdots & \frac{\partial x_n}{\partial z_n} \end{bmatrix} \right) \right|.$$

Using the change-of-variable formula, one can show (after some algebra, which we'll skip) that the vector variable Z has the following joint density:

$$p_Z(z) = \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}z^T z\right). \tag{9}$$

The claim follows immediately. □

[5] See section on "Eigenvalues and Eigenvectors of Symmetric Matrices."

[6] To show that B is invertible, it suffices to observe that U is an invertible matrix, and right-multiplying U by a diagonal matrix (with no zero diagonal entries) will rescale its columns but will not change its rank.

More on Multivariate Gaussians

Chuong B. Do

November 21, 2008

Up to this point in class, you have seen multivariate Gaussians arise in a number of applications, such as the probabilistic interpretation of linear regression, Gaussian discriminant analysis, mixture of Gaussians clustering, and most recently, factor analysis. In these lecture notes, we attempt to demystify some of the fancier properties of multivariate Gaussians that were introduced in the recent factor analysis lecture. The goal of these notes is to give you some intuition into where these properties come from, so that you can use them with confidence on your homework (hint hint!) and beyond.

1 Definition

A vector-valued random variable $x \in \mathbf{R}^n$ is said to have a **multivariate normal (or Gaussian) distribution** with mean $\mu \in \mathbf{R}^n$ and covariance matrix $\Sigma \in \mathbf{S}^n_{++}$[1] if its probability density function is given by

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right).$$

We write this as $x \sim \mathcal{N}(\mu, \Sigma)$.

2 Gaussian facts

Multivariate Gaussians turn out to be extremely handy in practice due to the following facts:

- **Fact #1:** If you know the mean μ and covariance matrix Σ of a Gaussian random variable x, you can write down the probability density function for x directly.

[1] Recall from the section notes on linear algebra that $\mathbf{S}^n_{++}$ is the space of symmetric positive definite $n \times n$ matrices, defined as

$$\mathbf{S}^n_{++} = \left\{A \in \mathbf{R}^{n\times n} : A = A^T \text{ and } x^T A x > 0 \text{ for all } x \in \mathbf{R}^n \text{ such that } x \neq 0\right\}.$$

- **Fact #2:** The following Gaussian integrals have closed-form solutions:

$$\int_{x \in \mathbf{R}^n} p(x;\mu,\Sigma)dx = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(x;\mu,\Sigma)dx_1 \ldots dx_n = 1$$

$$\int_{x \in \mathbf{R}^n} x_i p(x;\mu,\sigma^2)dx = \mu_i$$

$$\int_{x \in \mathbf{R}^n} (x_i - \mu_i)(x_j - \mu_j)p(x;\mu,\sigma^2)dx = \Sigma_{ij}.$$

- **Fact #3:** Gaussians obey a number of *closure* properties:
  - The sum of independent Gaussian random variables is Gaussian.
  - The marginal of a joint Gaussian distribution is Gaussian.
  - The conditional of a joint Gaussian distribution is Gaussian.

At first glance, some of these facts, in particular facts #1 and #2, may seem either intuitively obvious or at least plausible. What is probably not so clear, however, is why these facts are so powerful. In this document, we'll provide some intuition for how these facts can be used when performing day-to-day manipulations dealing with multivariate Gaussian random variables.

3 Closure properties

In this section, we'll go through each of the closure properties described earlier, and we'll either prove the property using facts #1 and #2, or we'll at least give some type of intuition as to why the property is true.

The following is a quick roadmap of what we'll cover:

	sums	marginals	conditionals
why is it Gaussian?	no	yes	yes
resulting density function	yes	yes	yes

3.1 Sum of independent Gaussians is Gaussian

The formal statement of this rule is:

> Suppose that $y \sim \mathcal{N}(\mu, \Sigma)$ and $z \sim \mathcal{N}(\mu', \Sigma')$ are independent Gaussian distributed random variables, where $\mu, \mu' \in \mathbf{R}^n$ and $\Sigma, \Sigma' \in \mathbf{S}^n_{++}$. Then, their sum is also Gaussian:
>
> $$y + z \sim \mathcal{N}(\mu + \mu', \Sigma + \Sigma').$$

Before we prove anything, here are some observations:

1. The first thing to point out is that the importance of the independence assumption in the above rule. To see why this matters, suppose that $y \sim \mathcal{N}(\mu, \Sigma)$ for some mean vector μ and covariance matrix Σ, and suppose that $z = -y$. Clearly, z also has a Gaussian distribution (in fact, $z \sim \mathcal{N}(-\mu, \Sigma)$, but $y + z$ is identically zero!

2. The second thing to point out is a point of confusion for many students: if we add together two Gaussian densities ("bumps" in multidimensional space), wouldn't we get back some bimodal (i.e., "two-humped" density)? Here, the thing to realize is that the density of the random variable $y + z$ in this rule is NOT found by simply adding the densities of the individual random variables y and z. Rather, the density of $y + z$ will actually turn out to be a *convolution* of the densities for y and z.[2] To show that the convolution of two Gaussian densities gives a Gaussian density, however, is beyond the scope of this class.

Instead, let's just use the observation that the convolution does give some type of Gaussian density, along with Fact #1, to figure out what the density, $p(y + z|\mu, \Sigma)$ would be, if we were to actually compute the convolution. How can we do this? Recall that from Fact #1, a Gaussian distribution is fully specified by its mean vector and covariance matrix. If we can determine what these are, then we're done.

But this is easy! For the mean, we have

$$E[y_i + z_i] = E[y_i] + E[z_i] = \mu_i + \mu'_i$$

from linearity of expectations. Therefore, the mean of $y + z$ is simply $\mu + \mu'$. Also, the (i, j)th entry of the covariance matrix is given by

$$\begin{aligned}
&E[(y_i + z_i)(y_j + z_j)] - E[y_i + z_i]E[y_j + z_j] \\
&\quad = E[y_i y_j + z_i y_j + y_i z_j + z_i z_j] - (E[y_i] + E[z_i])(E[y_j] + E[z_j]) \\
&\quad = E[y_i y_j] + E[z_i y_j] + E[y_i z_j] + E[z_i z_j] - E[y_i]E[y_j] - E[z_i]E[y_j] - E[y_i]E[z_j] - E[z_i][z_j] \\
&\quad = (E[y_i y_j] - E[y_i]E[y_j]) + (E[z_i z_j] - E[z_i]E[z_j]) \\
&\qquad + (E[z_i y_j] - E[z_i]E[y_j]) + (E[y_i z_j] - E[y_i]E[z_j]).
\end{aligned}$$

Using the fact that y and z are independent, we have $E[z_i y_j] = E[z_i]E[y_j]$ and $E[y_i z_j] = E[y_i]E[z_j]$. Therefore, the last two terms drop out, and we are left with,

$$\begin{aligned}
&E[(y_i + z_i)(y_j + z_j)] - E[y_i + z_i]E[y_j + z_j] \\
&\quad = (E[y_i y_j] - E[y_i]E[y_j]) + (E[z_i z_j] - E[z_i]E[z_j]) \\
&\quad = \Sigma_{ij} + \Sigma'_{ij}.
\end{aligned}$$

[2] For example, if y and z were univariate Gaussians (i.e., $y \sim \mathcal{N}(\mu, \sigma^2)$, $z \sim \mathcal{N}(\mu', \sigma'^2)$), then the convolution of their probability densities is given by

$$\begin{aligned}
p(y + z; \mu, \mu', \sigma^2, \sigma'^2) &= \int_{-\infty}^{\infty} p(w; \mu, \sigma^2) p(y + z - w; \mu', \sigma'^2) dw \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(w - \mu)^2\right) \cdot \frac{1}{\sqrt{2\pi}\sigma'} \exp\left(-\frac{1}{2\sigma'^2}(y + z - w - \mu')^2\right) dw
\end{aligned}$$

From this, we can conclude that the covariance matrix of $y + z$ is simply $\Sigma + \Sigma'$.

At this point, take a step back and think about what we have just done. Using some simple properties of expectations and independence, we have computed the mean and covariance matrix of $y + z$. Because of Fact #1, we can thus write down the density for $y + z$ immediately, without the need to perform a convolution![3]

3.2 Marginal of a joint Gaussian is Gaussian

The formal statement of this rule is:

> Suppose that
>
> $$\begin{bmatrix} x_A \\ x_B \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix}, \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} \right),$$
>
> where $x_A \in \mathbf{R}^m$, $x_B \in \mathbf{R}^n$, and the dimensions of the mean vectors and covariance matrix subblocks are chosen to match x_A and x_B. Then, the marginal densities,
>
> $$\begin{aligned} p(x_A) &= \int_{x_B \in \mathbf{R}^n} p(x_A, x_B; \mu, \Sigma) dx_B \\ p(x_B) &= \int_{x_A \in \mathbf{R}^m} p(x_A, x_B; \mu, \Sigma) dx_A \end{aligned}$$
>
> are Gaussian:
>
> $$\begin{aligned} x_A &\sim \mathcal{N}(\mu_A, \Sigma_{AA}) \\ x_B &\sim \mathcal{N}(\mu_B, \Sigma_{BB}). \end{aligned}$$

To justify this rule, let's just focus on the marginal distribution with respect to the variables x_A.[4]

First, note that computing the mean and covariance matrix for a marginal distribution is easy: simply take the corresponding subblocks from the mean and covariance matrix of the joint density. To make sure this is absolutely clear, let's look at the covariance between $x_{A,i}$ and $x_{A,j}$ (the ith component of x_A and the jth component of x_A). Note that $x_{A,i}$ and $x_{A,j}$ are also the ith and jth components of

$$\begin{bmatrix} x_A \\ x_B \end{bmatrix}$$

[3] Of course, we needed to know that $y + z$ had a Gaussian distribution in the first place.

[4] In general, for a random vector x which has a Gaussian distribution, we can always permute entries of x so long as we permute the entries of the mean vector and the rows/columns of the covariance matrix in the corresponding way. As a result, it suffices to look only at x_A, and the result for x_B follows immediately.

(since x_A appears at the top of this vector). To find their covariance, we need to simply look at the (i, j)th element of the covariance matrix,

$$\begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}.$$

The (i, j)th element is found in the Σ_{AA} subblock, and in fact, is precisely $\Sigma_{AA,ij}$. Using this argument for all $i, j \in \{1, \dots, m\}$, we see that the covariance matrix for x_A is simply Σ_{AA}. A similar argument can be used to find that the mean of x_A is simply μ_A. Thus, the above argument tells us that if we knew that the marginal distribution over x_A is Gaussian, then we could immediately write down a density function for x_A in terms of the appropriate submatrices of the mean and covariance matrices for the joint density!

The above argument, though simple, however, is somewhat unsatisfying: how can we actually be sure that x_A has a multivariate Gaussian distribution? The argument for this is slightly long-winded, so rather than saving up the punchline, here's our plan of attack up front:

1. Write the integral form of the marginal density explicitly.
2. Rewrite the integral by partitioning the inverse covariance matrix.
3. Use a "completion-of-squares" argument to evaluate the integral over x_B.
4. Argue that the resulting density is Gaussian.

Let's see each of these steps in action.

3.2.1 The marginal density in integral form

Suppose that we wanted to compute the density function of x_A directly. Then, we would need to compute the integral,

$$\begin{aligned} p(x_A) &= \int_{x_B \in \mathbf{R}^n} p(x_A, x_B; \mu, \Sigma) dx_B \\ &= \frac{1}{(2\pi)^{\frac{m+n}{2}} \begin{vmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{vmatrix}^{1/2}} \int_{x_B \in \mathbf{R}^n} \exp\left(-\frac{1}{2} \begin{bmatrix} x_A - \mu_A \\ x_B - \mu_B \end{bmatrix}^T \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}^{-1} \begin{bmatrix} x_A - \mu_A \\ x_B - \mu_B \end{bmatrix} \right) dx_B. \end{aligned}$$

3.2.2 Partitioning the inverse covariance matrix

To make any sort of progress, we'll need to write the matrix product in the exponent in a slightly different form. In particular, let us define the matrix $V \in \mathbf{R}^{(m+n)\times(m+n)}$ as[5]

$$V = \begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix} = \Sigma^{-1}.$$

[5] Sometimes, V is called the "precision" matrix.

It might be tempting to think that

$$V = \begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix} = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}^{-1} \text{“}=\text{”} \begin{bmatrix} \Sigma_{AA}^{-1} & \Sigma_{AB}^{-1} \\ \Sigma_{BA}^{-1} & \Sigma_{BB}^{-1} \end{bmatrix}$$

However, the rightmost equality does not hold! We'll return to this issue in a later step; for now, though, it suffices to define V as above without worrying what actual contents of each submatrix are.

Using this definition of V, the integral expands to

$$p(x_A) = \frac{1}{Z} \int_{x_B \in \mathbf{R}^n} \exp\Big(- \Big[\frac{1}{2}(x_A - \mu_A)^T V_{AA} (x_A - \mu_A) + \frac{1}{2}(x_A - \mu_A)^T V_{AB} (x_B - \mu_B) \\ + \frac{1}{2}(x_B - \mu_B)^T V_{BA} (x_A - \mu_A) + \frac{1}{2}(x_B - \mu_B)^T V_{BB} (x_B - \mu_B) \Big] \Big) dx_B,$$

where Z is some constant not depending on either x_A or x_B that we'll choose to ignore for the moment. If you haven't worked with partitioned matrices before, then the expansion above may seem a little magical to you. It is analogous to the idea that when defining a quadratic form based on some 2×2 matrix A, then

$$x^T A x = \sum_i \sum_j A_{ij} x_i x_j = x_1 A_{11} x_1 + x_1 A_{12} x_2 + x_2 A_{21} x_1 + x_2 A_{22} x_2.$$

Take some time to convince yourself that the matrix generalization above also holds.

3.2.3 Integrating out x_B

To evaluate the integral, we'll somehow want to integrate out x_B. In general, however, Gaussian integrals are hard to compute by hand. Is there anything we can do to save time? There are, in fact, a number of Gaussian integrals for which the answer is already known (see Fact #2). The basic idea in this section, then, will be to transform the integral we had in the last section into a form where we can apply one of the results from Fact #2 in order to perform the required integration easily.

The key to this is a mathematical trick known as "completion of squares." Consider the quadratic function $z^T A z + b^T z + c$ where A is a symmetric, nonsingular matrix. Then, one can verify directly that

$$\frac{1}{2} z^T A z + b^T z + c = \frac{1}{2} \big(z + A^{-1} b \big)^T A \big(z + A^{-1} b \big) + c - \frac{1}{2} b^T A^{-1} b.$$

This is the multivariate generalization of the "completion of squares" argument used in single variable algebra:

$$\frac{1}{2} a z^2 + bz + c = \frac{1}{2} a \left(z + \frac{b}{a} \right)^2 + c - \frac{b^2}{2a}$$

To apply the completion of squares in our situation above, let

$$\begin{aligned} z &= x_B - \mu_B \\ A &= V_{BB} \\ b &= V_{BA}(x_A - \mu_A) \\ c &= \frac{1}{2}(x_A - \mu_A)^T V_{AA}(x_A - \mu_A). \end{aligned}$$

Then, it follows that the integral can be rewritten as

$$\begin{aligned} p(x_A) = \frac{1}{Z}\int_{x_B \in \mathbf{R}^n} \exp\Bigg(-\Bigg[\frac{1}{2}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big)^T V_{BB}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big) \\ + \frac{1}{2}(x_A - \mu_A)^T V_{AA}(x_A - \mu_A) - \frac{1}{2}(x_A - \mu_A)^T V_{AB}V_{BB}^{-1}V_{BA}(x_A - \mu_A)\Bigg]\Bigg) dx_B \end{aligned}$$

We can factor out the terms not including x_B to obtain,

$$\begin{aligned} p(x_A) = \exp\left(-\frac{1}{2}(x_A - \mu_A)^T V_{AA}(x_A - \mu_A) + \frac{1}{2}(x_A - \mu_A)^T V_{AB}V_{BB}^{-1}V_{BA}(x_A - \mu_A)\right) \\ \cdot \frac{1}{Z}\int_{x_B \in \mathbf{R}^n} \exp\Bigg(-\frac{1}{2}\Bigg[\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big)^T V_{BB}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big)\Bigg]\Bigg) dx_B \end{aligned}$$

At this point, we can now apply Fact #2. In particular, we know that generically speaking, for a multivariate Gaussian distributed random variable x with mean μ and covariance matrix Σ, the density function normalizes, i.e.,

$$\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\int_{\mathbf{R}^n} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) = 1,$$

or equivalently,

$$\int_{\mathbf{R}^n} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) = (2\pi)^{n/2}|\Sigma|^{1/2}.$$

We use this fact to get rid of the remaining integral in our expression for $p(x_A)$:

$$p(x_A) = \frac{1}{Z}\cdot(2\pi)^{n/2}|V_{BB}|^{1/2}\cdot\exp\left(-\frac{1}{2}(x_A - \mu_A)^T(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})(x_A - \mu_A)\right).$$

3.2.4 Arguing that resulting density is Gaussian

At this point, we are almost done! Ignoring the normalization constant in front, we see that the density of x_A is the exponential of a quadratic form in x_A. We can quickly recognize that our density is none other than a Gaussian with mean vector μ_A and covariance matrix $(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1}$. Although the form of the covariance matrix may seem a bit complex, we have already achieved what we set out to show in the first place—namely, that x_A has a marginal Gaussian distribution. Using the logic before, we can conclude that this covariance matrix must somehow reduce to Σ_{AA}.

But, in case you are curious, it's also possible to show that our derivation is consistent with this earlier justification. To do this, we use the following result for partitioned matrices:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M^{-1} & -M^{-1}BD^{-1} \\ -D^{-1}CM^{-1} & D^{-1} + D^{-1}CM^{-1}BD^{-1} \end{bmatrix}.$$

where $M = A - BD^{-1}C$. This formula can be thought of as the multivariable generalization of the explicit inverse for a 2×2 matrix,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

Using the formula, it follows that

$$\begin{aligned} \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} &= \begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} (V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1} & -(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1}V_{AB}V_{BB}^{-1} \\ -V_{BB}^{-1}V_{BA}(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1} & (V_{BB} - V_{BA}V_{AA}^{-1}V_{AB})^{-1} \end{bmatrix} \end{aligned}$$

We immediately see that $(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1} = \Sigma_{AA}$, just as we expected!

3.3 Conditional of a joint Gaussian is Gaussian

The formal statement of this rule is:

> Suppose that
>
> $$\begin{bmatrix} x_A \\ x_B \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix}, \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} \right),$$
>
> where $x_A \in \mathbf{R}^m$, $x_B \in \mathbf{R}^n$, and the dimensions of the mean vectors and covariance matrix subblocks are chosen to match x_A and x_B. Then, the conditional densities
>
> $$\begin{aligned} p(x_A \mid x_B) &= \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_A \in \mathbf{R}^m} p(x_A, x_B; \mu, \Sigma) dx_A} \\ p(x_B \mid x_A) &= \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_B \in \mathbf{R}^n} p(x_A, x_B; \mu, \Sigma) dx_B} \end{aligned}$$
>
> are also Gaussian:
>
> $$\begin{aligned} x_A \mid x_B &\sim \mathcal{N}\big(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\big) \\ x_B \mid x_A &\sim \mathcal{N}\big(\mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A), \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}\big). \end{aligned}$$

As before, we'll just examine the conditional distribution $x_B \mid x_A$, and the other result will hold by symmetry. Our plan of attack will be as follows:

1. Write the form of the conditional density explicitly.
2. Rewrite the expression by partitioning the inverse covariance matrix.
3. Use a "completion-of-squares" argument.
4. Argue that the resulting density is Gaussian.

Let's see each of these steps in action.

3.3.1 The conditional density written explicitly

Suppose that we wanted to compute the density function of x_B given x_A directly. Then, we would need to compute

$$\begin{aligned} p(x_B \mid x_A) &= \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_B \in \mathbf{R}^m} p(x_A, x_B; \mu, \Sigma) dx_A} \\ &= \frac{1}{Z'} \exp\left(-\frac{1}{2} \begin{bmatrix} x_A - \mu_A \\ x_B - \mu_B \end{bmatrix}^T \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}^{-1} \begin{bmatrix} x_A - \mu_A \\ x_B - \mu_B \end{bmatrix} \right) \end{aligned}$$

where Z' is a normalization constant that we used to absorb factors not depending on x_B. Note that this time, we don't even need to compute any integrals – the value of the integral does not depend on x_B, and hence the integral can be folded into the normalization constant Z'.

3.3.2 Partitioning the inverse covariance matrix

As before, we reparameterize our density using the matrix V, to obtain

$$\begin{aligned} p(x_B \mid x_A) &= \frac{1}{Z'} \exp\left(-\frac{1}{2} \begin{bmatrix} x_A - \mu_A \\ x_B - \mu_B \end{bmatrix}^T \begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix} \begin{bmatrix} x_A - \mu_A \\ x_B - \mu_B \end{bmatrix} \right) \\ &= \frac{1}{Z'} \exp\Big(- \Big[\frac{1}{2}(x_A - \mu_A)^T V_{AA} (x_A - \mu_A) + \frac{1}{2}(x_A - \mu_A)^T V_{AB} (x_B - \mu_B) \\ &\qquad\qquad + \frac{1}{2}(x_B - \mu_B)^T V_{BA} (x_A - \mu_A) + \frac{1}{2}(x_B - \mu_B)^T V_{BB} (x_B - \mu_B) \Big] \Big). \end{aligned}$$

3.3.3 Use a "completion of squares" argument

Recall that

$$\frac{1}{2} z^T A z + b^T z + c = \frac{1}{2} \big(z + A^{-1} b\big)^T A \big(z + A^{-1} b\big) + c - \frac{1}{2} b^T A^{-1} b$$

provided A is a symmetric, nonsingular matrix. As before, to apply the completion of squares in our situation above, let

$$\begin{aligned} z &= x_B - \mu_B \\ A &= V_{BB} \\ b &= V_{BA}(x_A - \mu_A) \\ c &= \frac{1}{2}(x_A - \mu_A)^T V_{AA}(x_A - \mu_A). \end{aligned}$$

Then, it follows that the expression for $p(x_B \mid x_A)$ can be rewritten as

$$p(x_B \mid x_A) = \frac{1}{Z'} \exp\Bigg(-\bigg[\frac{1}{2}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big)^T V_{BB}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big) \\ + \frac{1}{2}(x_A - \mu_A)^T V_{AA}(x_A - \mu_A) - \frac{1}{2}(x_A - \mu_A)^T V_{AB}V_{BB}^{-1}V_{BA}(x_A - \mu_A) \bigg] \Bigg)$$

Absorbing the portion of the exponent which does not depend on x_B into the normalization constant, we have

$$p(x_B \mid x_A) = \frac{1}{Z''} \exp\Bigg(-\frac{1}{2}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big)^T V_{BB}\big(x_B - \mu_B + V_{BB}^{-1}V_{BA}(x_A - \mu_A)\big) \Bigg)$$

3.3.4 Arguing that resulting density is Gaussian

Looking at the last form, $p(x_B \mid x_A)$ has the form of a Gaussian density with mean $\mu_B - V_{BB}^{-1}V_{BA}(x_A - \mu_A)$ and covariance matrix V_{BB}^{-1}. As before, recall our matrix identity,

$$\begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} = \begin{bmatrix} (V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1} & -(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1}V_{AB}V_{BB}^{-1} \\ -V_{BB}^{-1}V_{BA}(V_{AA} - V_{AB}V_{BB}^{-1}V_{BA})^{-1} & (V_{BB} - V_{BA}V_{AA}^{-1}V_{AB})^{-1} \end{bmatrix}.$$

From this, it follows that

$$\mu_{B|A} = \mu_B - V_{BB}^{-1}V_{BA}(x_A - \mu_A) = \mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A).$$

Conversely, we can also apply our matrix identity to obtain:

$$\begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix} = \begin{bmatrix} (\Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})^{-1} & -(\Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})^{-1}\Sigma_{AB}\Sigma_{BB}^{-1} \\ -\Sigma_{BB}^{-1}\Sigma_{BA}(\Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})^{-1} & (\Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB})^{-1} \end{bmatrix},$$

from which it follows that

$$\Sigma_{B|A} = V_{BB}^{-1} = \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}.$$

And, we're done!

4 Summary

In these notes, we used a few simple properties of multivariate Gaussians (plus a couple matrix algebra tricks) in order to argue that multivariate Gaussians satisfy a number of closure properties. In general, multivariate Gaussians are exceedingly useful representations of probability distributions because the closure properties ensure that most of the types of operations we would ever want to perform using a multivariate Gaussian can be done in closed form. Analytically, integrals involving multivariate Gaussians are often nice in practice since we can rely on known Gaussian integrals to avoid having to ever perform the integration ourselves.

5 Exercise

Test your understanding! Let $A \in \mathbf{R}^{n \times n}$ be a symmetric nonsingular square matrix, $b \in \mathbf{R}^n$, and c. Prove that

$$\int_{x \in \mathbf{R}^n} \exp\left(-\frac{1}{2}x^T A x - x^T b - c\right) dx = \frac{(2\pi)^{n/2}}{|A|^{1/2} \exp(c - b^T A^{-1} b)}.$$

References

For more information on multivariate Gaussians, see

Bishop, Christopher M. *Pattern Recognition and Machine Learning.* Springer, 2006.