Aprendizado de Máquina

Formalização e Exemplos de Algoritmos Lineares

Enno Nagel

Instituto Federal de Alagoas, Maceió, 7 Novembro 2017

Para quê estudar o Aprendizado de Máquina?

A resposta esmagadora do guru Andrew Ng:

“I hope we can build an AI-powered society that gives everyone affordable healthcare, provides every child a personalized education, makes inexpensive self-driving cars available to all, and provides meaningful work for every man and woman.”

Traduzido para português:

Espero que possamos criar uma sociedade baseada na inteligência artificial, que

- oferece um plano de saúde acessível a todo mundo,
- proporciona uma educação personalizada a cada criança,
- disponibiliza carros auto-conduzidos baratos a todos, e
- um trabalho significativo com sentido a cada pessoa.

1. O que é um problema de aprendizado exatamente?
2. Perceptron
3. O que é aprendizado?

O que é um problema de aprendizado?

Tradicionalmente, na **especificação**, foram introduzidas regras ao computador para ele calcular as consequências.

$$\text{regras} \longrightarrow \text{computador} \longrightarrow \text{resultados}$$

No **aprendizado**, são introduzidos dados ao computador para ele calcular as regras, detectar as regularidades neles (e, em seguida, calcular as consequências destas regras).

$$\text{dados} \longrightarrow \text{computador} \longrightarrow \text{regras}$$

Com o advento de cada vez

- maiores volumes de dados e
- maior potência computacional,

o aprendizado de máquina torna-se cada vez mais proveitoso.

Por exemplo, para distinguir entre valores de moedas pelos seus pesos ($= \mathrm{X}$) e tamanhos ($= \mathrm{Y}$), estamos usando

- a *especificação* se contatamos a **cada da moeda** e classificamos as moedas por estas medidas (com uma certa margem de erro), e
- o *aprendizado* se derivamos estas medidas (com uma certa margem de erro) a partir de um grande **conjunto de moedas**.

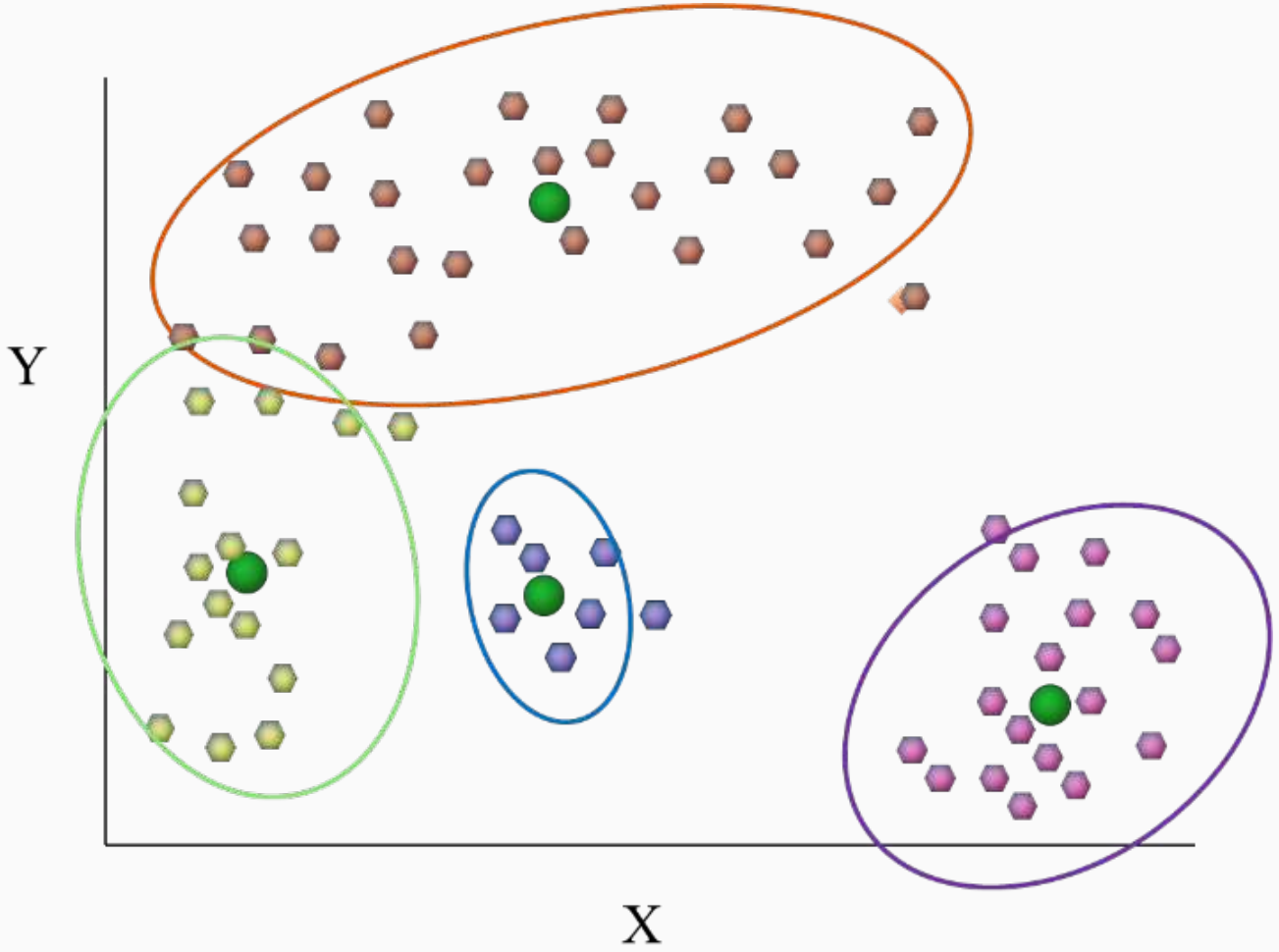


Figura 1: Agrupamento de moedas por peso e tamanho

Se os círculos foram desenhados **antes** dos pontos, trata-se de **especificação**. Se os círculos foram desenhados **depois** dos pontos, trata-se de **aprendizado**.

Aprendizado com Supervisionamento

No aprendizado com supervisionamento, os exemplos (da amostra) são **classificados**. Isto é, temos pares

$$(\text{entrada}, \text{saída})$$

Por exemplo,

- duas pastas de e-mails, a primeira do tipo `spam` e a outra do tipo `ham` (= não `spam`); aqui a entrada é o (conteúdo do) e-mail e a saída sim ou não, ou
- várias montes de moedas de valores diferentes; aqui a entrada é o (tamanho e peso) da moeda e a saída o seu valor (em centavos).

Aprendizado por Reforço

No aprendizado por reforço, os exemplos da amostra são **parcialmente avaliados**. Isto é, temos pares

$$(\text{entrada, saída e a sua avaliação})$$

Destacamos que, dada uma entrada x, **não** necessariamente **todos** os pares (x, y) para todas as possíveis saídas y são contidos na amostra.

Quer dizer, não sabemos necessariamente quão boas outras saídas poderiam ter sido. Por exemplo, ao aprender um jogo de tabuleiro, dado uma configuração x do tabuleiro, y é o lance e a sua avaliação a probabilidade de ganhar o jogo.

A avaliação

No aprendizado por reforço, a **avaliação** é uma função na entrada e saída. Por exemplo,

- no jogo Xadrez, a função poderia avaliar a configuração do tabuleiro
  - pelo número de peças de cada um,
  - pela a sua centralidade e agilidade;
- no jogo Go, a função poderia contar
  - as pedras e
  - as casas controladas por elas;
- no jogo Gamão, simplesmente avaliar uma configuração como
  - *neutra* se o partido é indeciso,
  - *positiva* $(+1)$, se o partido é ganhado, e
  - *negativa* (-1), se o partido é perdido.

Aprendizado sem Supervisionamento

No aprendizado sem supervisionamento, os exemplos são **inclassificados**.

Isto é, o algoritmo

1. Precisa de criar as possíveis saídas, ou os rótulos, e
2. *depois* classifica.

O exemplo principal é o **clustering**, ou **agrupamento**. Por exemplo, mesmo sem rótulos, as moedas podem ser agrupadas pelos seus pesos e tamanhos, vide Figura 1

Exemplos de Classificação (Supervisados)

- na medicina: aprender diagnosticar a partir de
  - dados relativos à saúde de pacientes com
  - os seus diagnósticos.
- na identificação de conteúdo: classificar e-mails em `Spam` (= lixo) e `Ham` (= não Spam) a partir de
  - um conjunto de e-mails do tipo `Spam` e
  - um conjunto de e-mails do tipo `Ham` (= não `Spam`).
- nas finanças: decidir se um credor poderá pagar as suas dívidas ou não a partir de
  - dados financeiros de credores anteriores e
  - os seus desempenhos no pagamento das dívidas
- no futebol: decidir se um jogador pertence à seleção nacional a partir das
  - capacidades (físicas) de jogadores anteriores e
  - se foram nomeados ou não.

Formalização

Formalmente, em um problema de aprendizado, temos

- a **entrada** X
- a **saída** Y
- os **dados** $\mathrm{D} = \{(x_1, y(x_1)), ..., (x_\mathrm{N}, y(x_\mathrm{N}))\}$ (= um conjunto finito em $\mathrm{X} \times \mathrm{Y}$)
- a **função alvo** $y : \mathrm{X} \to \mathrm{Y}$ (= a função ótima a ser aproximada)

e

- as **hipóteses** $\mathrm{H} = \{h : \mathrm{X} \to \mathrm{Y}\}$ (= um conjunto de funções que aproximam y), e
- um **algoritmo** A que escolhe a função h em H que "aproxima" y e em particular D.

- No diagnóstico, temos
  - X = dados médicos de um paciente
  - Y = doenças
  - D = diagnósticos históricos
- Na classificação de e-mails em $\mathtt{Spam}$ ou $\mathtt{Ham}$, temos
  - X = palavras de um e-mail
  - Y = Sim ou Não
  - D = duas caixas, uma de $\mathtt{Spam}$ e a outra de $\mathtt{Ham}$
- Na avaliação da solvência de um cliente, temos
  - X = dados financeiros de um cliente
  - Y = Sim ou Não
  - D = históricos de créditos concedidos
- Na composição da seleção nacional, temos
  - X = capacidade do jogador
  - Y = Sim ou Não
  - D = seleções anteriores.

Em mais detalhes:

- Na avaliação da solvência de um possível credor por um banco, a entrada X são os **dados financeiros** dos clientes, por exemplo
  - idade
  - salário
  - haveres
  - dívidas
- Na avaliação da aptidão de um jogador para a seleção nacional, a entrada X são os **dados físicos** dos jogadores, por exemplo a velocidade
  - do chute, e
  - do jogador

Pontuação

Nestes exemplos,

- a entrada é interpretada como **ênupla de números reais**, e
- a saída é binária: **Sim ou Não**.

Neste caso, podemos dar uma **pontuação** ao possível credor ou ao jogador relativo aos seus atributos financeiros ou físicos.

Mais explicitamente, avaliar as **importâncias** dos seus atributos e **somá-las**. Se esta soma é suficientemente alto, quer dizer, em cima de um certo **limiar**, aceitamos o credor ou jogador.

...

Por exemplo,

- para certo Neymar, a velocidade
  - do chute é 100 km por hora, e
  - do jogador é 40 km por hora;
- para certo Roberto Carlos, a velocidade do
  - do chute é 200 km por hora, e
  - do jogador é 20 km por hora;
- para certo Frederico, a velocidade do
  - do chute é 100 km por hora, e
  - do jogador é 20 km por hora.

Queremos encontrar uma fórmula que atribua uma pontuação p,

- $p = 9$ a Neymar,
- $p = 8$ a Roberto Carlos e
- $p = 6$ a Frederico

e tal que o jogador entrou na seleção se e tão-somente se $p > 7$.

$\Longrightarrow$ `Perceptron`

Aplicaremos o método geométrico do `Perceptron` em busca de um *hiperplano* (= **reta** se $\mathbb{X}$ é o plano) que separa (se for possível) entre

- os pontos que correspondem àqueles credores anteriores que **pagaram** as suas dívidas respectivamente àqueles jogadores que **entraram** na seleção nacional, e
- os pontos que correspondem àqueles credores anteriores que **faliram** respectivamente àqueles jogadores que **ficaram fora**.

1 O que é um problema de aprendizado exatamente?

2 Perceptron

3 O que é aprendizado?

Perceptron

Se o problema de aprendizado é uma *classificação binária*, isto é

- $\mathrm{X} = \mathbb{R} \times \cdots \times \mathbb{R}$, e
- $\mathrm{Y} = \{\pm 1\}$,

podemos aplicar o método de aprendizado do **Perceptron**, em que

- $\mathrm{H} = \{h(x) = \mathrm{sinal}(w_1 x_1 + \cdots + w_d x_d - b) :$
  $w_1, ..., w_d, b$ em $\mathbb{R}\}$

com $\mathrm{sinal}(x) = 1$ se $x > 0$ e -1 se $x < 0$.

...

Isto é, dado $x_1, ..., x_d$, uma função em H é dada pelos

- seus **pesos** $w_1, ..., w_d$ e
- o seu **limiar** b (= a nossa pontuação mínima),

e avalia se

$$w_1 x_1 + \cdots + w_d x_d > b \text{ ou não.}$$

Geometricamente, esta desigualdade divide o espaço X por um hiperplano (trasladado) em dois semiespaços. Por exemplo, se X é o plano, ela define uma **reta** e cada lado dela um **semiplano**.

- Detecção de um e-mail como `Spam` dependendo de quais palavras o e-mail contém e as suas frequências em (e-mails de) `Spam` e em (e-mails de) `Ham`. São pesos
  - *positivos* palavras de propaganda e de marcas, como `promoção`, `aumento` ou `Viagra`,
  - *negativos* palavras neutros como conjunções, por exemplo, `entretanto` ou nomes raros de conhecidos, como `Vélton`.
- Concessão de um crédito dependendo dos dados financeiros de um cliente e da importância de cada critério: São pesos
  - *positivos* os bens e o salário ,
  - *negativos* as dívidas.
- Seleção de jogadores para a equipa nacional: No meu chute, a velocidade do jogador importa bem mais do que a do seu chute. Por exemplo, os pesos das velocidades são 20 (para a do jogador) e 1 (para a do chute).

Simplificação

O algoritmo *Perceptron* aproxima os pesos e o limar por iteração sobre todos os pontos no conjunto de dados exemplares D a uma solução h_0 , quer dizer, tal que

$$h_0(x) = y \quad \text{para todo } (x, y) \text{ em } \mathrm{D}.$$

Facilitamos a formulação o problema do aprendizado: Para dois vetores $v = (v_1, ..., v_d)$ e $w = (w_1, ..., w_d)$, o seu **produto escalar** é dado por

$$v^{\mathrm{T}} w := v_1 w_1 + \cdots + v_d w_d$$

Pondo $x_0 = 1$ e $w_0 = -b$, vale

$$w_1 x_1 + \cdots + w_d x_d - b = w_0 x_0 + w_1 x_1 + \cdots + w_d x_d$$

isto é, para $x = (1, x_1, ..., x_d)$ e $w = (-b, w_1, ..., w_d)$,

$$w_1 x_1 + \cdots + w_d x_d - b = w^{\mathrm{T}} x.$$

Concluímos que

- $\mathrm{X} = \{1\} \times \mathbb{R} \times \cdots \times \mathbb{R}$, e
- $\mathrm{Y} = \{\pm 1\}$,

e

- $\mathrm{H} = \{h(x) = \mathrm{sinal}(w^{\mathrm{T}} x) : w_0, w_1, ..., w_d \text{ em } \mathbb{R}\}$

com $\mathrm{sinal}(x) = 1$ se $x > 0$ e -1 se $x < 0$.

Geometricamente, para $d = 2$,

- olhamos o plano com as coordenadas x e y como o hiperplano P da equação $x_0 = z = 1$ no espaço de três coordenadas x, y e z,
- o hiperplano H no espaço dos vetores ortogonais a w, e
- a (projeção ao plano da) interseção de H e P, e a qual é a reta que visa separar os pontos devidamente.

O algoritmo de aprendizado **Perceptron** determina w iterativamente a partir de D como segue:

Seja $t = 0, 1, 2, \ldots$ a iteração e $w(t)$ o valor de w na iteração t. Seja $(x(t), y(t))$ em D um ponto erroneamente classificado, isto é, tal que $\mathrm{sinal}(w(t)^{\mathrm{T}} x(t)) \neq y(t)$.

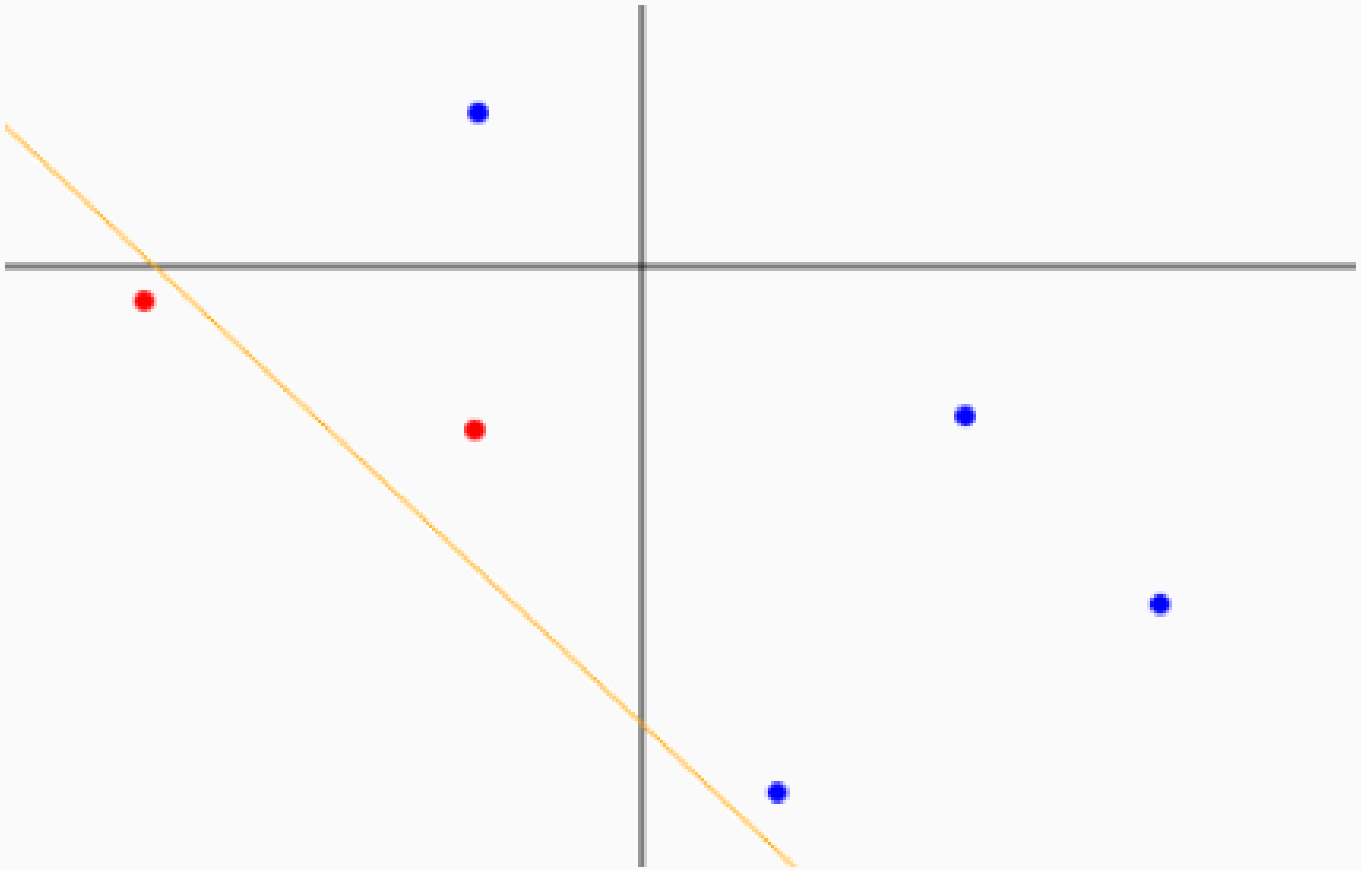


Figura 2: ponto erroneamente classificado

Algoritmo

Geometricamente, $w(t+1)$ desloca o limar na direção correta (isto é, a definida por $y(t)$) do ponto erroneamente classificado. Formalmente, há duas possibilidades:

- ou $w(t)^{\mathrm{T}} x(t) > 0$ e $y(t) = -1$,
- ou $w(t)^{\mathrm{T}} x(t) < 0$ e $y(t) = +1$.

O `Perceptron` põe

$$w(t+1) := w(t) + y(t)x(t)$$

Como

$$w(t+1)^{\mathrm{T}}x(t) = w(t)^{\mathrm{T}}x(t) + y(t)x(t)^{\mathrm{T}}x(t)$$

e $v^{\mathrm{T}}v > 0$ para qualquer vetor v, vale

- se $w(t)^{\mathrm{T}}x(t) > 0$ e $y(t) = -1$, então $w(t+1)^{\mathrm{T}}x(t) < w(t)^{\mathrm{T}}x(t)$,
- se $w(t)^{\mathrm{T}}x(t) < 0$ e $y(t) = +1$, então $w(t+1)^{\mathrm{T}}x(t) > w(t)^{\mathrm{T}}x(t)$

Isto é, o valor da (nova) função hipotecada

$$h(t+1) := w(t+1)^{\mathrm{T}}.$$

no ponto por enquanto erroneamente classificado $x(t)$ é mais próximo do valor correto (do que o da antiga função $h(t)$).

Exemplo

Seja $d = 2$, e

- consista D nos dois pontos $p = (0, 2)$ e $q = (1, 1)$ com $y(p) = 1$ e $y(q) = -1$, e
- seja $w(0) = (0, 0, 2)$.

Observamos que $x = (x_0, x_1, x_2) = (1, x, y)$ satisfaz

$$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$$

se e tão-somente se

$$y = ax + b \quad \text{com } a = -w_1/w_2 \text{ e } b = -w_0/w_2.$$

Isto é, descrevemos a projeção dos vetores ortonormais ao vetor w aos seus últimos dois coordenados pela função (traslada) linear $y = ax + b$.

1. Como $w(0)^{\mathrm{T}} q = 2 > 0$, mas $y(q) < 0$, constatamos que q é erroneamente classificado e determinamos

   $$w(1) = w(0) + y(q)q = (0, 0, 2) - (1, 1, 1) = (-1, -1, 1).$$

   Isto é, $a = 1$ e $b = 1$.
2. Como $w(1)q = -1 > 0$ e $y(q) < 0$, constatamos que q é bem classificado (e p também) e o algoritmo termina.

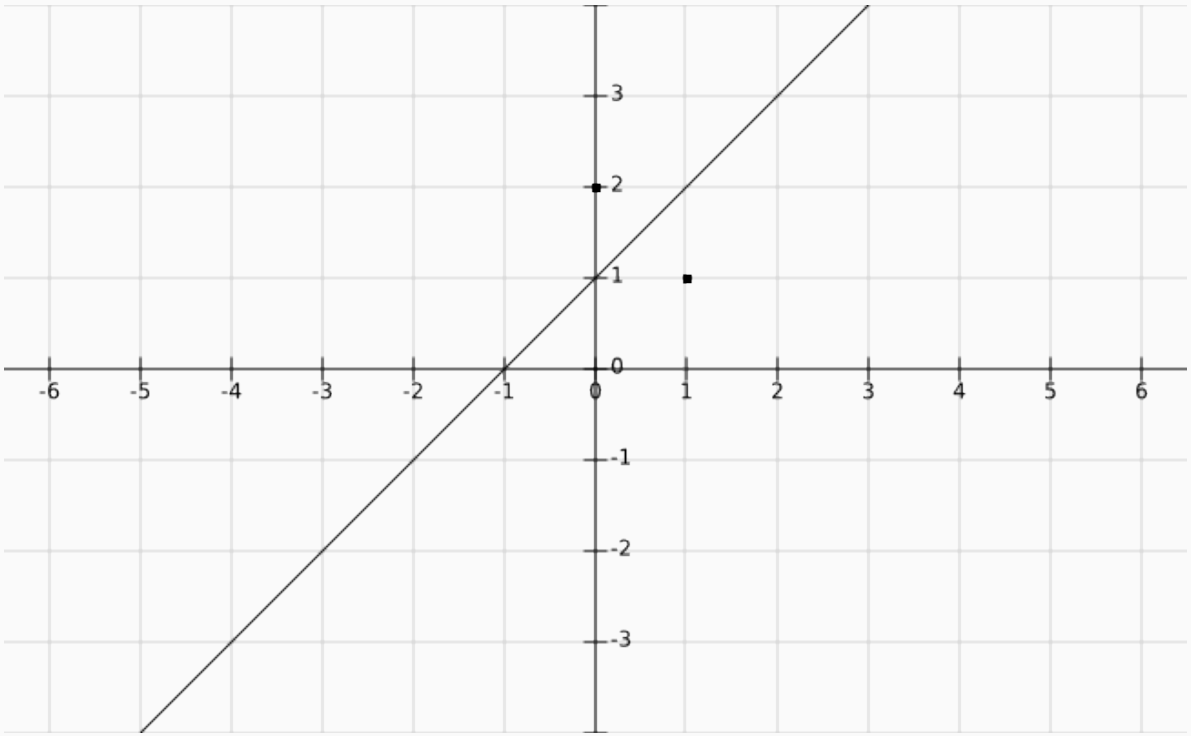
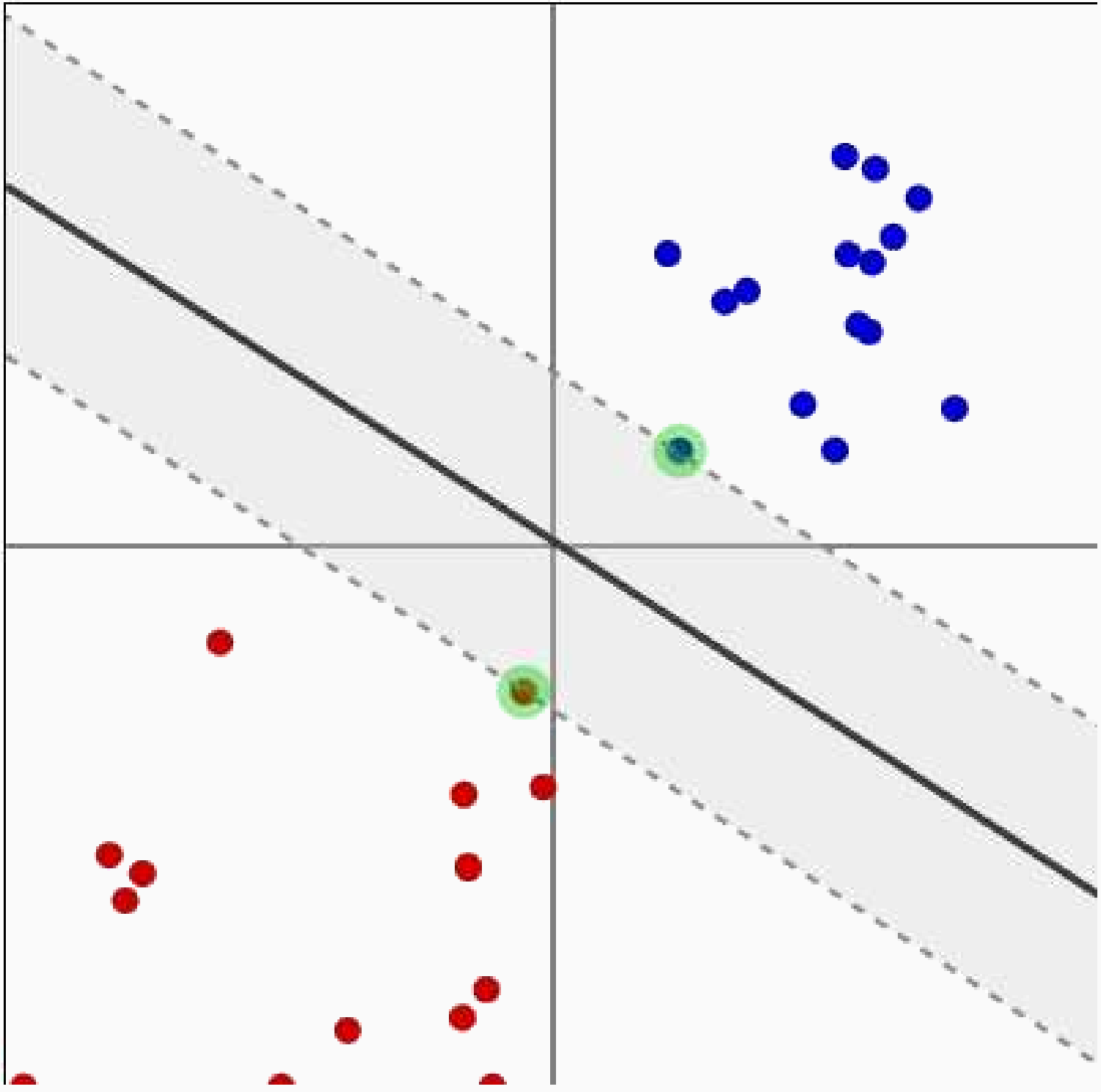


Figura 3: O exemplo dado pelos pontos p e q

Observações

- Se D não é linearmente separável, isto é, não há uma solução em H, este algoritmo não converge a uma solução aproximativa (quer dizer, que minimiza as classificações errôneas).
- Se D é linearmente separável, quanto maior a margem entre os dois conjuntos de pontos, tanto mais fácil a sua separação, isto é, tanto mais rápido o Perceptron converge. Mais exatamente, este algoritmo termina em $\leq \mathrm{R}^2/\gamma^2$ passos onde
  - R é o *raio* de D, e
  - γ é a *margem*, a mínima distância do hiperplano (= uma reta quando X é o plano) separador ao conjunto dos pontos.



Definição *Raio* e *Margem*

- O **raio** R de um conjunto de pontos D é definido por

$$\mathrm{R} := \max\{|x| \text{ em } \mathrm{D}\},$$

- Para um hiperplano H (dado por $\mathrm{H} = \{x \text{ em } \mathrm{X} \text{ ortogonais a } h\}$ onde *ortogonal* significa $x^{\mathrm{T}}h = 0$) e dados D, a **margem** entre H e D é definida por

$$\gamma := \min\{\mathrm{dist}(d, \mathrm{H}) : d \text{ em } \mathrm{D}\}$$

onde, para um hiperplano $\mathrm{H} = \{x \in \mathrm{X} : h^{\mathrm{T}}x = 0\}$, a *distância* entre ele e um ponto x é definida por

$$\mathrm{dist}(x, \mathrm{H}) := |h^{\mathrm{T}}x|/|h|^2$$

(Recordemo-nos de que, para dois vetores x e y no plano,

$$xy = \eta|x||y|$$

onde η é o angulo entre eles, e $|x|$ e $|y|$ são os seus comprimentos.)

O Algoritmo Pocket (= bolso)

Se o problema de aprendizado é um problema de classificação binária, isto é, $\mathrm{X} = \{1\} \times \mathbb{R} \times \cdots \times \mathbb{R}$ (com $1 + d$ fatores) e $\mathrm{Y} = \{\pm 1\}$, então as hipóteses H têm a forma

$$h\colon x \mapsto \operatorname{sign}(w^{\mathrm{T}} x)$$

para vetores w em $\mathbb{R}^{d+1}$. Seja $\mathrm{D} = \{x_1, \ldots, x_n\}$ a amostra e

$$\mathrm{E}_{\mathrm{in}}(h) = \#\{x_n \text{ em } \mathrm{D} \text{ tal que } \operatorname{sign}(w^{\mathrm{T}} x_n) \neq y_n\}/\#\mathrm{D}$$

a **probabilidade do erro** que h faz na amostra. Isto é, $\mathrm{E}_{\mathrm{in}}(h)$ mede quão próximo h é da função verdadeira (= y) em D.

O Algoritmo Perceptron

Se a amostra D **é linearmente separável**, isto é, há h em H tal que

$$\mathrm{E}_{\mathrm{in}}(h) = 0,$$

então o algoritmo `Perceptron`, ou `Perceptron`, encontrará tal h. Recordemo-nos de que, para cada passo $t \geq 0$, seleciona um ponto erroneamente classificado $(x(t), y(t))$ e atualiza $w(t)$ por

$$w(t+1) := w(t) + y(t)x(t).$$

Mas se D **não é linearmente separável**, isto é, não há uma solução em H, este algoritmo não converge a uma solução aproximativa (quer dizer, que minimiza $\mathrm{E}_{\mathrm{in}}(h)$, as classificações errôneas).

O Algoritmo Pocket

Se D não é linearmente separável, isto é, é impossível achar h tal que $\mathrm{E}_{in}(h) = 0$, queremos pelo menos minimizá-lo, isto é, encontrar o (melhor) vetor de pesos w (em $\mathbf{R}^{d+1}$) que minimiza

$$\mathrm{E}_{in}(h) = \#\{x_n \text{ em } \mathrm{D} \text{ tal que } \mathrm{sign}(w^{\mathrm{T}} x_n) \neq y_n\}/\#\mathrm{D}$$

O algoritmo `Pocket` (= bolso) aplica o `Perceptron` repetidamente (por exemplo, em T repetições) e guarda no seu bolso o melhor vetor de pesos w até então encontrado. Isto é,

```
Põe w := w(0)
Para t = 0, ..., T:
    Roda Perceptron para obter w(t + 1)
    Calcula E(w(t + 1))
    Se E(w(t + 1)) < E(w), então põe w = w(t + 1)
Devolve w
```

Comparação Perceptron e Pocket

Isto é, em comparação ao `Perceptron`, o `Pocket` avalia após a atualização $w(t) \mapsto w(t+1)$ se o novo vetor é melhor (quanto o error sobre a amostra) do que o melhor até então encontrado w, e, caso sim, substitui w por $w(t+1)$.

Pela computação de $\mathrm{E}_{\mathrm{in}}(w(t+1))$, que percorre todos os pontos da amostra, o algoritmo `Pocket` é **mais lento** do que o `Perceptron`.

Observamos que sabemos

- nem *qual* é o melhor vetor de pesos,
- nem *quando* se encontrará.

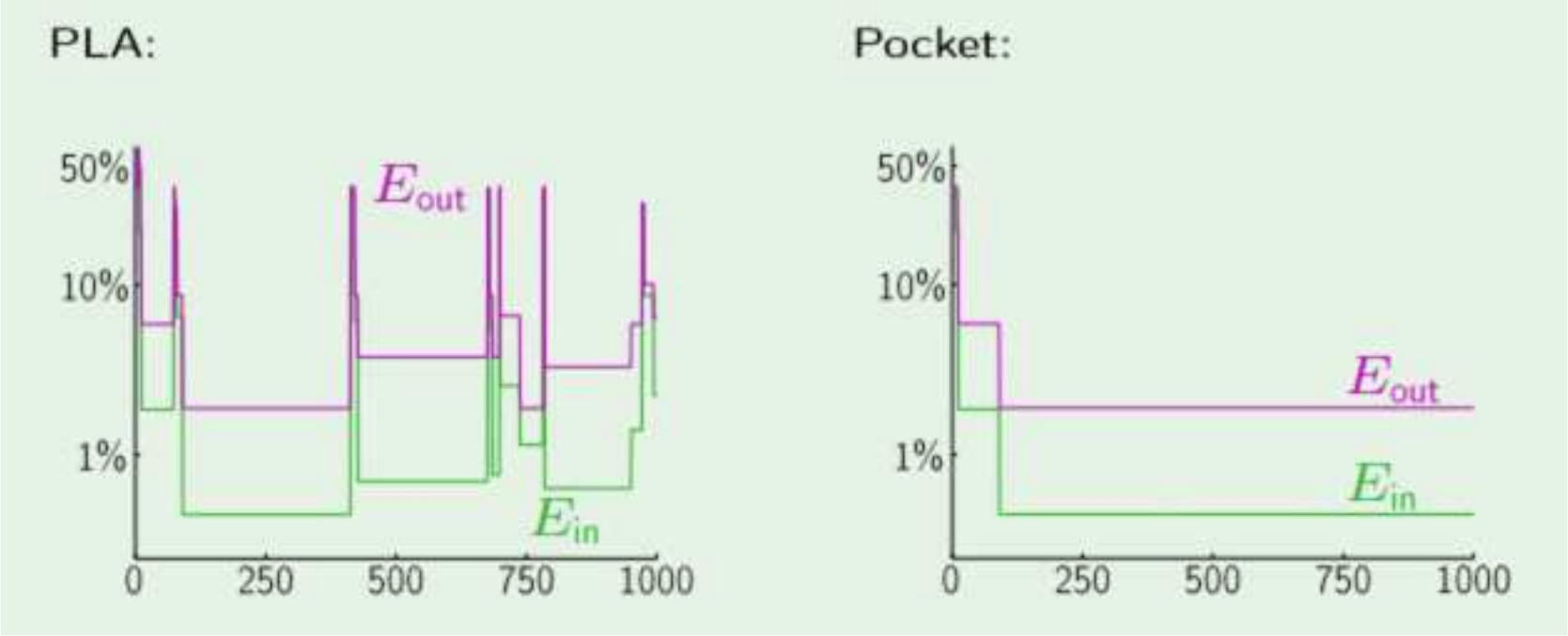


Figura 5: A evolução do erro do `Perceptron` versus o do `Pocket`

1 O que é um problema de aprendizado exatamente?

2 Perceptron

3 O que é aprendizado?

Recado Formalização

Recordemo-nos de que formalmente, em um problema de aprendizado (com supervisionamento), temos

- a **entrada** X
- a **saída** Y
- os **dados** $\mathrm{D} = \{(x_1, y_1), ..., (x_\mathrm{N}, y_\mathrm{N})\}$ (= um conjunto finito em $\mathrm{X} \times \mathrm{Y}$)
- a **função alvo** $y : \mathrm{X} \to \mathrm{Y}$ (= a função ótima a ser aproximada)

e

- as **hipóteses** $\mathrm{H} = \{h : \mathrm{X} \to \mathrm{Y}\}$ (= um conjunto de funções que aproximam y), e
- um **algoritmo** A que escolhe a função h em H que "aproxima" y (e em particular D) o mais possível.

O que significa **a mais próxima** de y?

Se, por exemplo,

- $\mathrm{X} = \{1, 2, 3, ..., 8\}$ e $\mathrm{Y} = \{\pm 1\}$ (classificação binária), e
- D tem 5 pares,

então **qualquer** função que coincide com estes 5 pares é uma possível solução; sobram 3 valores a definir. Qual destas $\#\mathrm{Y}^3 = 2^3$ extensões dos 5 aos 8 pontos é a mais próxima da função alvo? Não há resposta **definitiva**, mas **provável:**

Provavelmente Aproximadamente Correto

O problema é **apreensível por PAC** (= Provavelmente Aproximadamente Correto) se há um algoritmo A, que calcula para toda amostra $\mathrm{D} = \{x_1, ..., x_n\}$ um $h = h(\mathrm{D})$ em H tal que:

Para todo $\epsilon > 0$ e $\delta > 0$, para toda distribuição P e toda amostra, isto é, $x_1, \ldots, x_n$ em X independentes e identicamente distribuídos para P, vale

$$\mathrm{P}(\mathrm{P}(h(\mathrm{D}) \neq y) < \epsilon) > 1 - \delta;$$

onde

- a probabilidade $\mathrm{P}(h(\mathrm{D}) \neq y)$ mede o conjunto (em X) em que $h(\mathrm{D})$ e y diferem ($< \epsilon$ = Aproximadamente Correto), e
- a probabilidade $\mathrm{P}(\mathrm{P}(h(\mathrm{D}) \neq f) < \epsilon)$ mede o conjunto (em $\mathrm{X} \times \cdots \times \mathrm{X}$) em que a ênupla $\mathrm{D} = (x_1, ..., x_n)$ implica uma escolha $h(\mathrm{D})$ pelo algoritmo A que é próxima de y, isto é, tal que $\mathrm{P}(h(\mathrm{D}) \neq y) < \epsilon$ (e $> 1 - \delta$ = Provavelmente certo).

...

Isto é, **provavelmente** a amostra leva a uma hipótese **aproximadamente correta**. Quer dizer, se a amostra é suficientemente típica (o que vale com uma probabilidade $> 1 - \delta$, isto é, **provavelmente**), então a probabilidade do erro é pequena, $< \epsilon$, isto é, a função é **aproximadamente** correto.

Observamos que não supomos uma distribuição particular.

Desigualdade de Hoeffding

Dada a amostra $\mathrm{D} = \{x_1, ..., x_\mathrm{N}\}$ e uma função h, podemos calcular o erro da **amostra**,

$$\mathrm{E}_{\mathsf{in}}(h) := \#\{ \text{ os pontos } x \text{ em } \mathrm{D} \text{ em que } h(x) \neq f(x)\}/\mathrm{N},$$

isto é, a probabilidade que h está errada sobre D. Por exemplo, o algoritmo `Perceptron`, se termina, consegue $\mathrm{E}_{\mathsf{in}}(h(\mathrm{D})) = 0$.

Mas, como f é desconhecida, *não* sabemos o erro **geral**,

$$\mathrm{E}_{\mathsf{out}}(h) := \mathrm{P}(\{x \in \mathrm{X} \text{ tal que } h(x) \neq f(x)\}),$$

isto é, a probabilidade que h está errada sobre X.

Para limitar $\mathrm{E}_{\mathsf{out}}(h)$, precisamos, pela desigualdade triangular

$$|\mathrm{E}_{\mathsf{out}}(h)| \leq |\mathrm{E}_{\mathsf{out}}(h) - \mathrm{E}_{\mathsf{in}}(h)| + |\mathrm{E}_{\mathsf{in}}(h)|,$$

limitar

$$|\mathrm{E}_{\mathsf{out}}(h) - \mathrm{E}_{\mathsf{in}}(h)| \quad \text{e} \quad |\mathrm{E}_{\mathsf{in}}(h)|.$$

Viabilidade do Aprendizado

Por isso, a viabilidade do aprendizado depende de que se podemos garantir

- que o erro $\mathrm{E}_{\mathrm{out}}(h)$ seja próximo do erro $\mathrm{E}_{\mathrm{in}}(h)$, e
- que o erro $\mathrm{E}_{\mathrm{in}}(h)$ seja pequeno?

A **desigualdade de Hoeffding** estima, para qualquer $\epsilon > 0$ e $\mathrm{N} = \#\mathrm{D}$ e $\mathrm{M} = \#\mathrm{H}$ em $\mathbb{N}$,

$$\mathrm{P}(|\mathrm{E}_{\mathrm{in}}(h) - \mathrm{E}_{\mathrm{out}}(h)| > \epsilon) \leq \mathrm{M} \cdot 2e^{-2\epsilon^2 \mathrm{N}}$$

onde P mede o conjunto (em $\mathrm{X} \times \cdots \times \mathrm{X}$) das amostras $\mathrm{D} = (x_1, ..., x_{\mathrm{N}})$ que levam a uma hipótese $h(\mathrm{D})$ tal que o erro da amostra difere do erro geral no máximo por ϵ. Isto é, **não** sabemos o erro geral $\mathrm{E}_{\mathrm{out}}(h)$, mas pela desigualdade de Hoeffding sabemos **aproximá-lo** pelo erro da amostra $\mathrm{E}_{\mathrm{in}}(h)$.

Reciprocidade entre o Erro Especifico e Geral

Quanto maior $\#\mathrm{H} = \mathrm{M}$, tanto maior o limite da desigualdade

$$\mathrm{P}(|\mathrm{E}_{\text{in}}(h(\mathrm{D})) - \mathrm{E}_{\text{out}}(h(\mathrm{D}))| > \epsilon) \leq \mathrm{M} \cdot 2e^{-2\epsilon^2 \mathrm{N}},$$

mas o algoritmo consegue mais facilmente diminuir

$$\mathrm{P}(|\mathrm{E}_{\text{in}}(h(\mathrm{D}))| > \epsilon).$$

Quanto mais complexo f, tanto mais provável que

$$\mathrm{P}(|\mathrm{E}_{\text{in}}(h(\mathrm{D}))| > \epsilon)$$

aumenta. Se tentamos diminuir este valor pelo aumento de $\mathrm{M} = \#\mathrm{H}$, então o valor

$$\mathrm{P}(|\mathrm{E}_{\text{in}}(h(\mathrm{D})) - \mathrm{E}_{\text{out}}(h(\mathrm{D}))| > \epsilon) \leq \mathrm{M} \cdot 2e^{-2\epsilon^2 \mathrm{N}}$$

aumenta. Não tem saída.

1. O que é um problema de aprendizado exatamente?
2. Perceptron
3. O que é aprendizado?