# Classificação Binária

## Inteligência na Web e Big Data

Fabricio Olivetti de França e Thiago Ferreira Covões
folivetti@ufabc.edu.br, thiago.covoes@ufabc.edu.br

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

CMCC
Centro de Matemática, Computação e Cognição
Center for Mathematics, Computation and Cognition

# Classificação

# Problema de Classificação

- Temos um conjunto de dados na forma $\{(\mathbf{x}, y)\}$, com $\mathbf{x} \in \mathbb{R}^d$ um vetor de atributos e $y \in \mathbb{Y}$ uma variável alvo que define um conjunto finito de possíveis classificações, agora queremos descobrir uma função de probabilidade:

$$P(Y = y \mid \mathbf{x}).$$

Exemplos de Problemas de Classificação

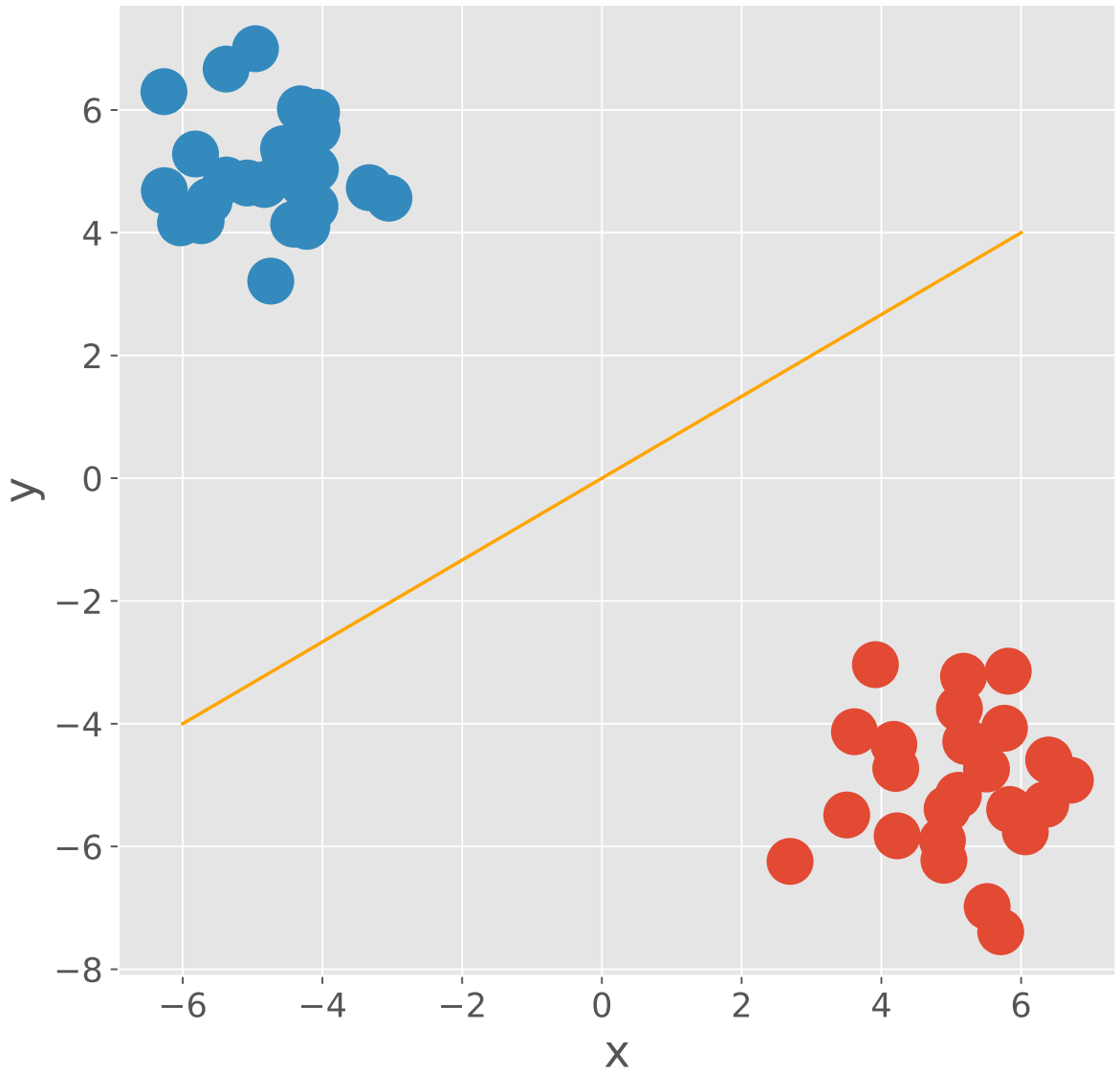
- Se um e-mail é spam ou não.
- Qual espécie é uma planta.
- Que tipo de doença um paciente tem.

# Classificadores Lineares

# Discriminador Linear

- Pensando apenas no caso de classificação binária, em que temos apenas duas classes: $-1$ e $+1$.
- Podemos definir uma função:
  - $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ tal que se $\mathbf{w} \cdot \mathbf{x} < \theta$, $\mathbf{x}$ pertence a classe $-1$; e se $\mathbf{w} \cdot \mathbf{x} > \theta$, $\mathbf{x}$ pertence a classe $+1$.
- O caso $\mathbf{w} \cdot \mathbf{x} = \theta$ resulta em uma falha de classificação.

# Discriminador Linear



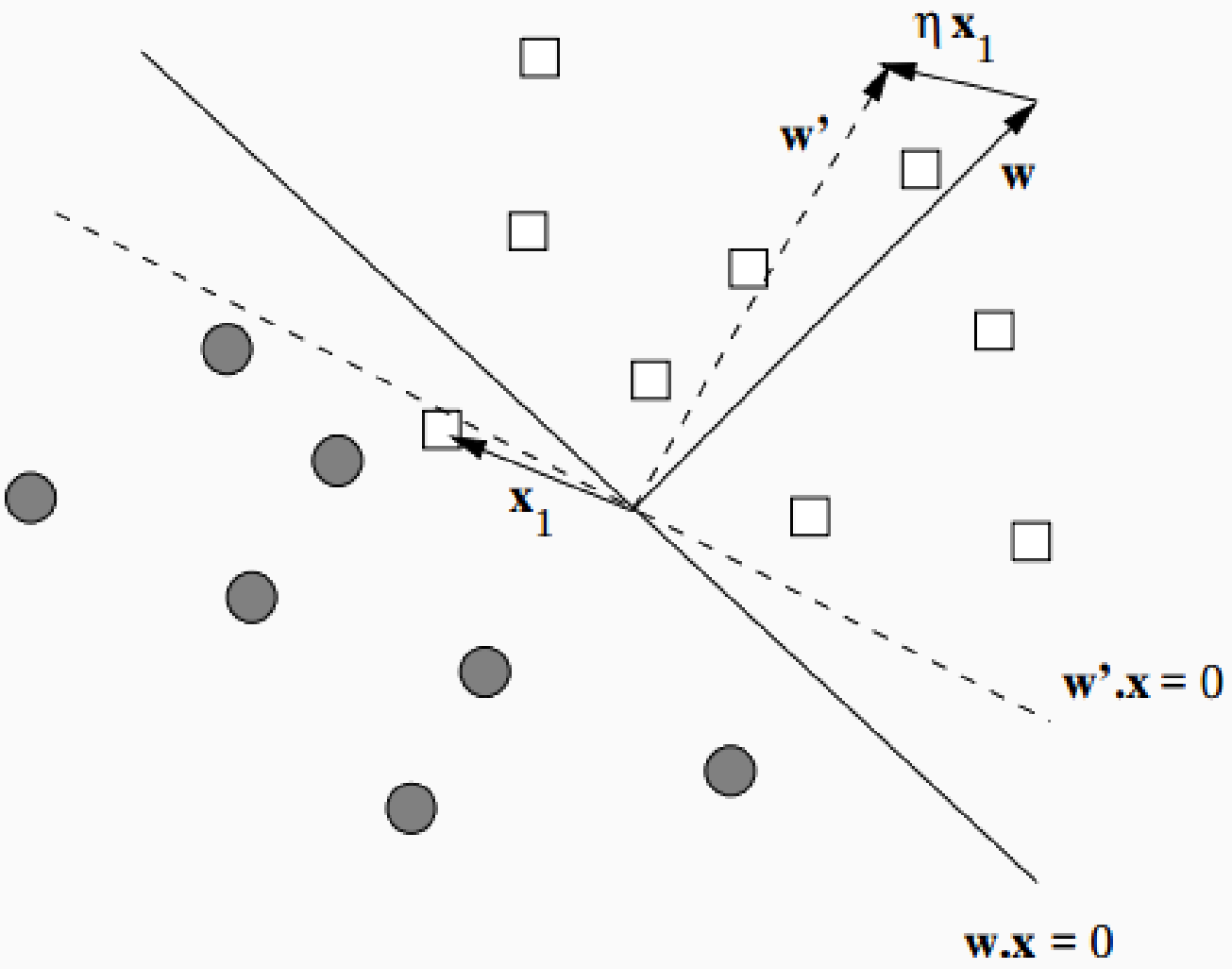
# Perceptron

- O problema se torna encontrar o vetor de pesos $\mathbf{w}$ e o parâmetro de limiar $\theta$ que maximiza a classificação correta.
  - Diferentes regras, diferentes algoritmos
- Supondo $\theta = 0$, nos limitamos em encontrar $\mathbf{w}$.

# Perceptron

- Seja $\eta$ a taxa de aprendizado: $0 \leq \eta \leq 1$ e $y \in \{-1, 1\}$
- Inicie $\mathbf{w}$ aleatoriamente
- Pegue um objeto $(\mathbf{x}, y)$ classificado incorretamente:
  1. Seja $\Delta\mathbf{w} = \eta\mathbf{x}y$
  2. Atualize $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$

# Perceptron




- retirado de Mining Massive Datasets - Ulman et al.

# Perceptron

- Embora exista garantia de convergência quando as classes são linearmente separáveis, se não for o caso o vetor $\mathbf{w}$ irá ser atualizado em um ciclo.
- O ciclo é difícil de perceber computacionalmente sem um alto custo.

# Perceptron

- Critérios de parada:
  - Após um número de iterações.
  - Quando o número de classificações incorretas não alterar (passo a classificar +1 corretamente e +1 incorretamente).
- Separe uma base de validação e pare quando não houver melhoras.

# Algoritmo de Winnow

- Para os casos de um vetor de atributos binários (ex.: texto), podemos aplicar o algoritmo de Winnow:
  - Se $\mathbf{w} \cdot \mathbf{x} \leq \theta$ e $y = +1$, para todo $x_i = 1$ faça $w_i = 2 \cdot w_i$.
  - Se $\mathbf{w} \cdot \mathbf{x} \geq \theta$ e $y = -1$, para todo $x_i = 1$ faça $w_i = w_i/2$.

# Algoritmo de Winnow

- Podemos inserir $\theta$ no vetor de atributos com o valor $-1$, com isso ajustamos também seu valor.
- Ao atualizar $\mathbf{w}$ na posição de $\theta$, fazemos o oposto da ação feita para as posições de $\mathbf{x}$.

# Exercício

Dado os documentos de texto com suas classes:

- voce ganhou reais, spam (+1)
- voce livre agora, ham (-1)
- agora voce ganhou, spam (+1)
- voce corrigiu provas, ham (-1)

Execute o algoritmo de Winnow manualmente, inicializando $\mathbf{w} = \mathbf{1}$ e com $\theta = 1$. Qual o vetor $\mathbf{w}$ obtido?

# Exercício

| | voce | ganhou | reais | livre | agora | corrigiu | provas |
|---|---|---|---|---|---|---|---|
| 1 (+) | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 (-) | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 3 (+) | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 (-) | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

# Exercício

| Base | voce | ganhou | reais | livre | agora | corrigiu | provas |
|---|---|---|---|---|---|---|---|
| 1 (+) | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 (-) | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 3 (+) | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 (-) | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

| w | voce | ganhou | reais | livre | agora | corrigiu | provas |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0.5 | 1 | 1 | 0.5 | 0.5 | 1 | 1 |
| 3 | 0.5 | 1 | 1 | 0.5 | 0.5 | 1 | 1 |
| 4 | 0.25 | 1 | 1 | 0.5 | 0.5 | 0.5 | 0.5 |

# Exercício

| Base | voce | ganhou | reais | livre | agora | corrigiu | provas |
|---|---|---|---|---|---|---|---|
| 1 (+) | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 (-) | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 3 (+) | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 (-) | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

| w | voce | ganhou | reais | livre | agora | corrigiu | provas |
|---|---|---|---|---|---|---|---|
| 1 | 0.25 | 1 | 1 | 0.5 | 0.5 | 0.5 | 0.5 |
| 2 | 0.125 | 1 | 1 | 0.25 | 0.25 | 0.5 | 0.5 |
| 3 | 0.125 | 1 | 1 | 0.25 | 0.25 | 0.5 | 0.5 |
| 4 | 0.0625 | 1 | 1 | 0.25 | 0.25 | 0.25 | 0.25 |

# Perceptron - MapReduce

- Como montar o algoritmo Perceptron no MapReduce?
- Principais observações:
  - Se não errou, não faz nada
  - A atualização de cada componente $w_i$ é independente das demais

# Perceptron - MapReduce

```
1  def mapper(key : int, value : tuple) -> (int, double):
2    x = first(value)
3    y = second(value)
4    score = np.dot(w, x)
5    if score * y < 0:
6      for i, xi in enumerate(x):
7        yield(i, learnRate * y * xi)
8
9  def reducer(key : int, values : [double]) -> (int,
   ↪  double):
10   delta_w = sum(values)
11   yield (i, w[key] + delta_w)
```

# Classificação certa?

# Abordagem probabilística

- A pessoa abaixo terá um ataque cardíaco?

| Idade | Sexo | Pressão Sanguínea | Colesterol | Peso |
|---|---|---|---|---|
| 40 | M | 130/85 | 240 | 70 |

# Abordagem probabilística

- A pessoa abaixo terá um ataque cardíaco?

| Idade | Sexo | Pressão Sanguínea | Colesterol | Peso |
|---|---|---|---|---|
| 40 | M | 130/85 | 240 | 70 |

- Difícil dizer sim ou não
  - Não temos todos os dados necessários
  - Função não necessariamente é determinística

# $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

- Dados ideais
  - $y_1 = 0,9 = P(+1|\mathbf{x}_1)$
  - $y_2 = 0,2 = P(+1|\mathbf{x}_1)$

    $\vdots$

  - $y_N = 0,6 = P(+1|\mathbf{x}_1)$
- Dados reais
  - $y_1 = +1 \sim P(y|\mathbf{x}_1)$
  - $y_2 = -1 \sim P(y|\mathbf{x}_1)$

    $\vdots$

  - $y_N = -1 \sim P(y|\mathbf{x}_1)$

# Hipótese logística

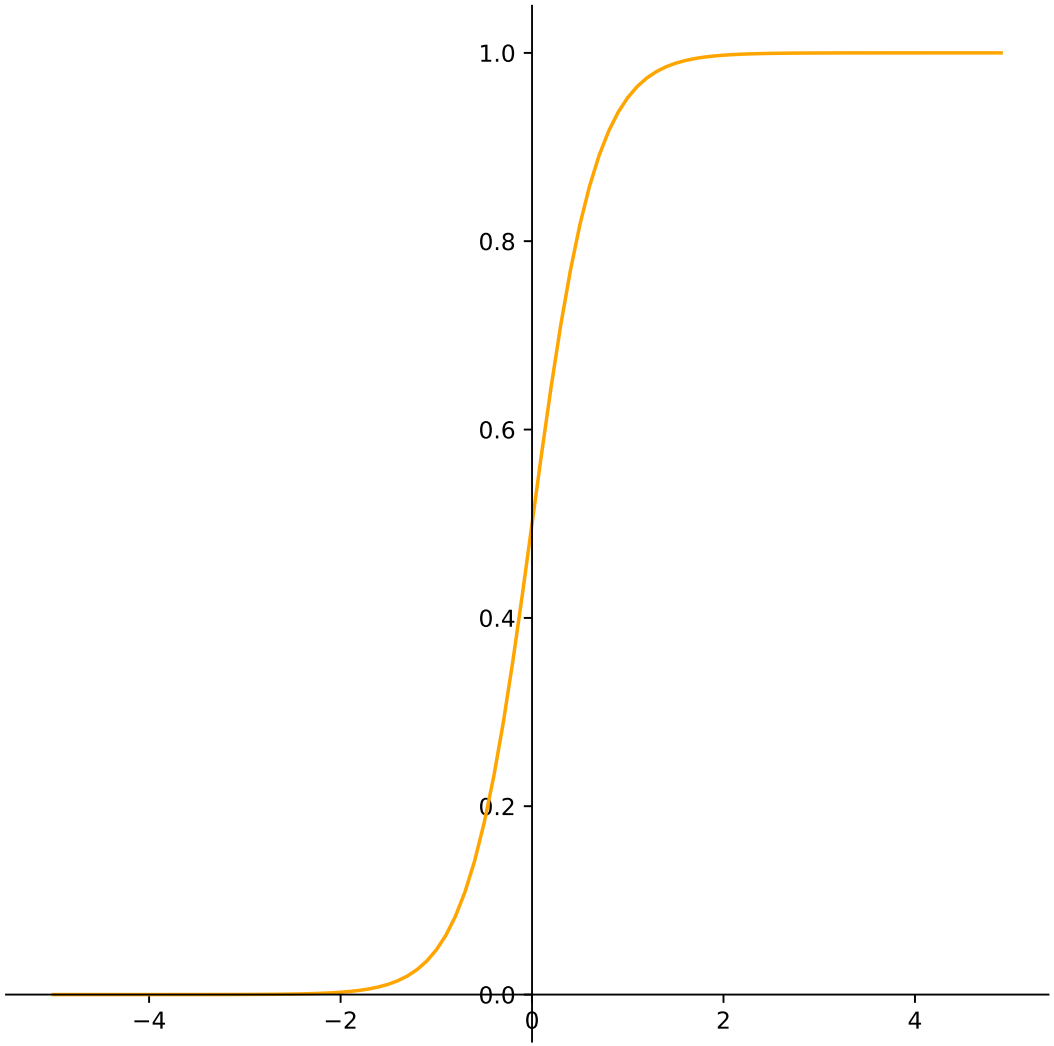
- Seja $\mathbf{x} = (x_0, x_1, \ldots x_M)$ os atributos do paciente, podemos computar um valor de risco ponderado:

$$s = \mathbf{w}^T \mathbf{x} = \sum_{m=0}^{M} w_m x_m$$

- e depois convertemos esse valor em uma probabilidade usando a **função logística**

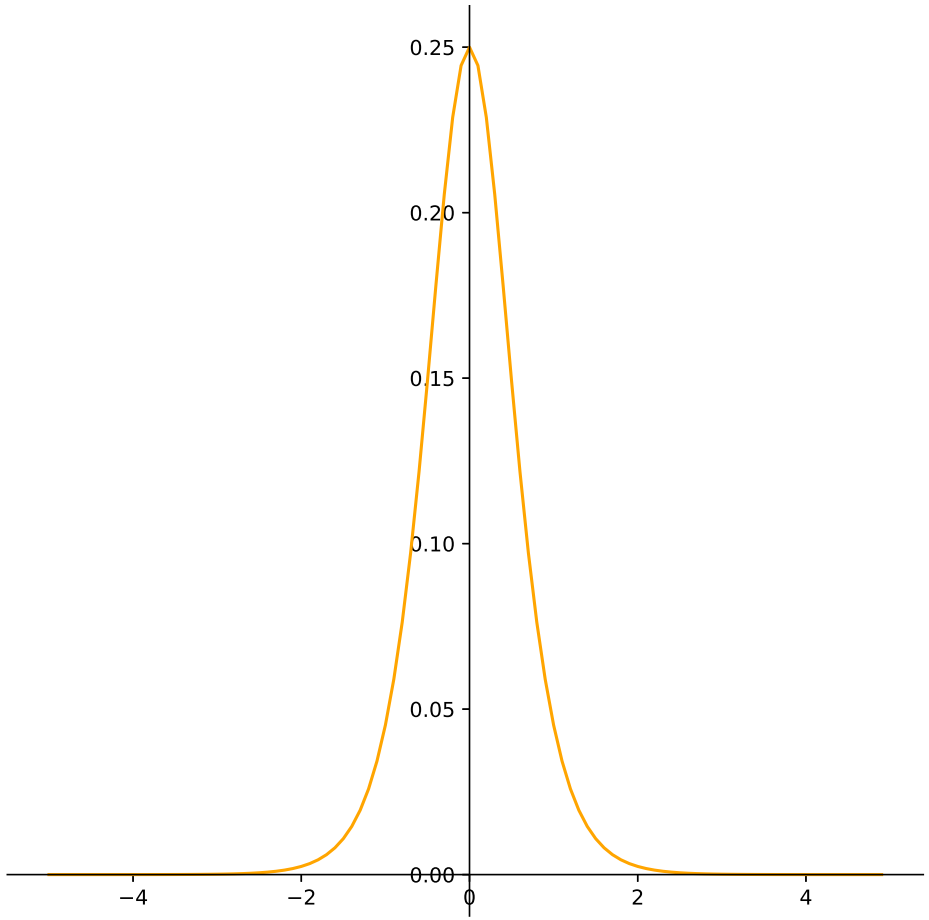
$$\theta(s) = \hat{y} = \frac{e^s}{1 + e^s} = \frac{1}{1 + e^{-s}}$$

# Regressão Logística



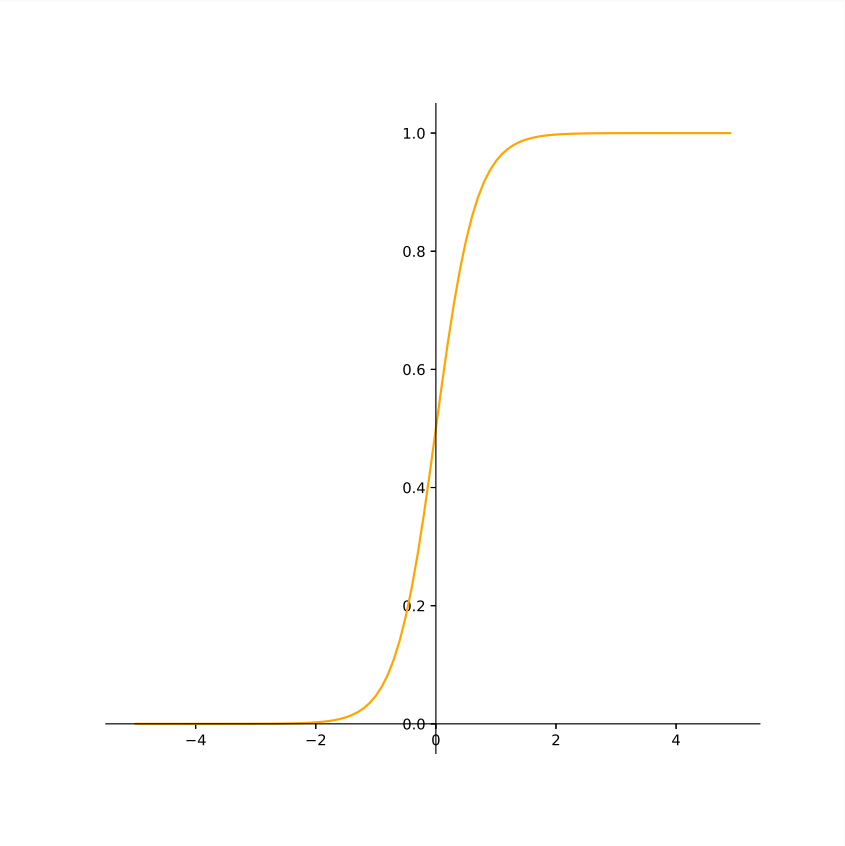
# Regressão Logística

- E a derivada $\hat{y} \cdot (1 - \hat{y})$:



# Regressão Logística

- Note que devemos alterar os valores representantes da classe para $y_i \in \{0, 1\}$.



# Regressão Logística

- A regressão logística tenta aproximar a função alvo $f(\mathbf{x}) = P(+1|\mathbf{x})$, usando

$$h(\mathbf{x}) = \frac{1}{1 + e^{(-\mathbf{w}^T \mathbf{x})}}$$

- Note como, apesar do nome, é um modelo de **classificação**

# Quiz

**Considere a hipótese logística que aproxima $P(+1|\mathbf{x})$.**

- Podemos fazer uma classificação binária usando $\text{sign}(h(\mathbf{x}) - \frac{1}{2})$. Qual seria uma fórmula alternativa para a classificação binária?
  1. $\text{sign}(\mathbf{w}^T\mathbf{x} - \frac{1}{2})$
  2. $\text{sign}(\mathbf{w}^T\mathbf{x})$
  3. $\text{sign}(\mathbf{w}^T\mathbf{x} + \frac{1}{2})$
  4. nenhuma das anteriores

# Quiz

**Considere a hipótese logística que aproxima $P(+1|\mathbf{x})$.**

- Podemos fazer uma classificação binário usando $\text{sign}(h(\mathbf{x}) - \frac{1}{2})$. Qual seria uma fórmula alternativa para a classificação binária?
  - (2) $\text{sign}(\mathbf{w}^T\mathbf{x})$
  - $\mathbf{w}^T\mathbf{x} = 0 \rightarrow h(\mathbf{x}) = \frac{1}{2}$

# Regressão Logística

- A função de erro deve ser definida como:

$$e(y, \hat{y}) = \begin{cases} -\log(\hat{y}), & \text{se } y = 1 \\ -\log(1 - \hat{y}), & \text{se } y = 0 \end{cases}$$

# Regressão Logística

- A função de erro deve ser definida como:

$$e(y, \hat{y}) = \begin{cases} -\log(\hat{y}), & \text{se } y = 1 \\ -\log(1 - \hat{y}), & \text{se } y = 0 \end{cases}$$

- Para $y = 1$:
  - Se $\hat{y} \to 1$, temos que $-\log(\hat{y}) \to 0$.
  - Se $\hat{y} \to 0$, temos que $-\log(\hat{y}) \to \infty$.

# Regressão Logística

- A função de erro deve ser definida como:

$$e(y, \hat{y}) = \begin{cases} -\log(\hat{y}), & \text{se } y = 1 \\ -\log(1 - \hat{y}), & \text{se } y = 0 \end{cases}$$

- Para $y = 0$:
  - Se $\hat{y} \to 1$, temos que $-\log(1 - \hat{y}) \to \infty$.
  - Se $\hat{y} \to 0$, temos que $-\log(1 - \hat{y}) \to 0$.

# Regressão Logística

- Como $y \in \{0, 1\}$, podemos reescrever a função como:

$$e(y, \hat{y}) = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

# Regressão Logística

- E o erro médio:

$$E(e(y, \hat{y})) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

# Exercício

- Mostre que

$$e(y, \hat{y}) = \begin{cases} -\log(\hat{y}), & \text{se } y = 1 \\ -\log(1 - \hat{y}), & \text{se } y = 0 \end{cases}$$

e

$$e(y, \hat{y}) = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

são equivalentes.

# Regressão Logística

- A derivada parcial em função de $w_j$ fica:

$$\frac{\partial e(y, \hat{y})}{\partial w_j} = -\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot x_{i,j}$$

# Regressão Logística

- E o gradiente:

$$\nabla e(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot x_i$$

- Em posse do gradiente, podemos aplicar gradiente descendente

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla e(y, \hat{y})$$

# Regressão Logística

- O processamento via MR pode ser feito de forma similar ao visto anteriormente

```
1  def mapper(key : int, value : tuple) -> (int, double):
2    x = first(value)
3    y = second(value)
4    score = sigmoid(np.dot(w, x))
5    for i, xi in enumerate(x):
6      yield(i, learnRate * (y - score) * xi)
7
8  def reducer(key : int, values : [double]) -> (int,
   ↪  double):
9    delta_w = sum(values)
10   yield (i, w[key] - delta_w)
```

# Regressão Logística

- Outras adaptações do algoritmo são comuns:
  - Regularização.
  - Batch, Stochastic.
  - etc.

# Regressão Logística: Problemas multi-classe

- Para $K$ classes, precisamos de $K$ $\mathbf{w}$'s, conhecido como classificador softmax

$$P(y_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k \cdot \mathbf{x}}}{\sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x}}}, \forall k \in \{1, \ldots, K\}$$

- Estratégia One vs All (outra possibilidade seria One vs One)
- Possível fazer usando apenas $K-1$ $\mathbf{w}$'s, usando uma classe como referência e o fato da probabilidade somar 1:

$$P(y_K|\mathbf{x}) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\mathbf{w}_k \cdot \mathbf{x}}}$$

# Regressão Logística: Softmax

$$P(y_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k \cdot \mathbf{x}}}{\sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x}}}, \forall k \in \{1, \ldots, K\}$$

- Embora simples, essa função pode trazer problemas numéricos, em especial quando $\mathbf{w}_k \cdot \mathbf{x}$ é grande
- Normalmente usa-se o truque `logsumexp`:

$$m = \max \mathbf{w}_k \cdot \mathbf{x}$$

$$\ln P(y_k|\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x} - (m + \ln \sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x} - m})$$

# LogSumExp

$$S = \ln \sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x}} \rightarrow e^S = \sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x}}$$

$$e^{-m} e^S = e^{-m} \sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x}}$$

$$e^{S-m} = \sum_{k=1}^{K} e^{-m} e^{\mathbf{w}_k \cdot \mathbf{x}}$$

$$S - m = \ln \sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x} - m}$$

$$S = m + \ln \sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x} - m}$$

# Naïve Bayes

# Aprendizado Probabilístico

- Foi mencionado anteriormente que o uso da função logística remetia a probabilidade de certo exemplo pertencer a uma classe.
- Por que então não estimar diretamente uma função de probabilidade?
- A ideia é estimar $P(y = y_i \mid x_i)$

# Naïve Bayes

- O algoritmo Naïve Bayes (que pode ser traduzido como Bayes ”Ingênuo”) é um classificador probabilístico que utiliza o teorema de Bayes assumindo que os atributos são condicionalmente independentes dado à classe.

# Probabilidade Condicional

- Digamos que nossos objetos representam animais, e estamos analisando 4 características:
  - Tamanho: pequeno ou grande,
  - Pele: lisa ou peluda,
  - Cor: marrom, verde ou vermelho,
  - Carne: macia ou dura

# Base de Dados

| Pele | Cor | Tamanho | Carne |
|---|---|---|---|
| peluda | marrom | grande | dura |
| peluda | verde | grande | dura |
| lisa | vermelho | grande | macia |
| peluda | verde | grande | macia |
| peluda | vermelho | pequeno | dura |
| lisa | vermelho | pequeno | dura |
| lisa | marrom | pequeno | dura |
| peluda | verde | pequeno | macia |
| lisa | verde | pequeno | dura |
| peluda | vermelho | grande | dura |
| lisa | marrom | grande | macia |
| lisa | verde | pequeno | macia |
| peluda | vermelho | pequeno | macia |
| lisa | vermelho | grande | dura |
| lisa | vermelho | pequeno | dura |
| peluda | verde | pequeno | dura |

# Probabilidade Condicional

- Digamos que queremos saber se é seguro ou perigoso comer certo animal, baseado em suas características. O primeiro passo é pré-classificar uma pequena amostra dos dados.

# Base de Dados

| Pele | Cor | Tamanho | Carne | Classe |
|---|---|---|---|---|
| peluda | marrom | grande | dura | seguro |
| peluda | verde | grande | dura | seguro |
| lisa | vermelho | grande | macia | perigoso |
| peluda | verde | grande | macia | seguro |
| peluda | vermelho | pequeno | dura | seguro |
| lisa | vermelho | pequeno | dura | seguro |
| lisa | marrom | pequeno | dura | seguro |
| peluda | verde | pequeno | macia | perigoso |
| lisa | verde | pequeno | dura | perigoso |
| peluda | vermelho | grande | dura | seguro |
| lisa | marrom | grande | macia | seguro |
| lisa | verde | pequeno | macia | perigoso |
| peluda | vermelho | pequeno | macia | seguro |
| lisa | vermelho | grande | dura | perigoso |
| lisa | vermelho | pequeno | dura | |
| peluda | verde | pequeno | dura | |

# Probabilidade Condicional

- Com uma base pré-classificada, devemos formular a seguinte questão:
  - Dado um pequeno animal que tem pele lisa de cor vermelha e carne dura. É seguro come-lo?

# Probabilidade Condicional

- Nosso animal pode ser representado por:

$$x = [\text{pequeno}, \texttt{lisa}, \texttt{vermelho}, \text{dura}]$$

- E a pergunta, em forma de probabilidade condicional:

$$P(y = \text{seguro} \mid x)$$

- Essa probabilidade deve ser lida como:

Qual a probabilidade de um animal ser seguro para comer dado que ele é: pequeno, lisa, vermelho, dura?

# Teorema de Bayes

- Pelo Teorema de Bayes, temos que:

$$P(y = \text{seguro} \mid x) = \frac{P(x \mid y = \text{seguro}) \cdot P(y = \text{seguro})}{P(x)},$$

com $P(x \mid y = \text{seguro})$ sendo denominado likelihood, $P(y = \text{seguro})$ é o prior e $P(x)$ a evidência.

# Naïve Bayes

- Com essa fórmula, e a tabela de exemplos podemos calcular a probabilidade de cada animal não classificado pertencer a uma classe ou outra.

# Naïve Bayes

- Expandindo temos:

$$P(y = \text{seguro} \mid x) = \frac{P(y = \text{seguro}) \prod_i P(x_i \mid y = \text{seguro})}{\sum_{v \in \{\text{seguro,perigoso}\}} P(y = v) \prod_i P(x_i | y = v)}$$

# Naïve Bayes

- Da nossa tabela, vamos calcular:

$$P(y = \texttt{seguro}) =$$
$$P(y = \texttt{perigoso}) =$$

# Base de Dados

| Pele | Cor | Tamanho | | Classe |
|---|---|---|---|---|
| peluda | marrom | grande | dura | **seguro** |
| peluda | verde | grande | dura | **seguro** |
| lisa | vermelho | grande | macia | perigoso |
| peluda | verde | grande | macia | **seguro** |
| peluda | vermelho | pequeno | dura | **seguro** |
| lisa | vermelho | pequeno | dura | **seguro** |
| lisa | marrom | pequeno | dura | **seguro** |
| peluda | verde | pequeno | macia | perigoso |
| lisa | verde | pequeno | dura | perigoso |
| peluda | vermelho | grande | dura | **seguro** |
| lisa | marrom | grande | macia | **seguro** |
| lisa | verde | pequeno | macia | perigoso |
| peluda | vermelho | pequeno | macia | **seguro** |
| lisa | vermelho | grande | dura | perigoso |
| lisa | vermelho | pequeno | dura | |
| peluda | verde | pequeno | dura | |

# Naïve Bayes

- Da nossa tabela, vamos calcular:

$$P(y = \texttt{seguro}) = 9/14$$
$$P(y = \texttt{perigoso}) = 5/14$$

# Naïve Bayes

- Dado $x = [\texttt{pequeno}, \texttt{lisa}, \texttt{vermelho}, \texttt{dura}]$:

$$
\begin{aligned}
P(\texttt{pequeno} \mid y = \texttt{seguro}) = \\
P(\texttt{lisa} \mid y = \texttt{seguro}) = \\
P(\texttt{vermelho} \mid y = \texttt{seguro}) = \\
P(\texttt{dura} \mid y = \texttt{seguro}) =
\end{aligned}
$$

# Base de Dados

| Pele | Cor | Tamanho | | Classe |
|---|---|---|---|---|
| peluda | marrom | grande | **dura** | seguro |
| peluda | verde | grande | **dura** | seguro |
| lisa | vermelho | grande | macia | perigoso |
| peluda | verde | grande | macia | seguro |
| peluda | **vermelho** | **pequeno** | **dura** | seguro |
| **lisa** | **vermelho** | **pequeno** | **dura** | seguro |
| **lisa** | marrom | **pequeno** | **dura** | seguro |
| peluda | verde | pequeno | macia | perigoso |
| lisa | verde | pequeno | dura | perigoso |
| peluda | **vermelho** | grande | **dura** | seguro |
| **lisa** | marrom | grande | macia | seguro |
| lisa | verde | pequeno | macia | perigoso |
| peluda | **vermelho** | **pequeno** | macia | seguro |
| lisa | vermelho | grande | dura | perigoso |
| lisa | vermelho | pequeno | dura | |
| peluda | verde | pequeno | dura | |

# Naïve Bayes

- E, finalmente:

$$
\begin{aligned}
P(\text{pequeno} \mid y = \text{seguro}) &= 4/9 \\
P(\text{lisa} \mid y = \text{seguro}) &= 3/9 \\
P(\text{vermelho} \mid y = \text{seguro}) &= 4/9 \\
P(\text{dura} \mid y = \text{seguro}) &= 6/9 \\
P(x \mid y = \text{seguro}) &= 288/6561 \approx 0.0438
\end{aligned}
$$

# Base de Dados

| Pele | Cor | Tamanho | | Classe |
|---|---|---|---|---|
| peluda | marrom | grande | dura | seguro |
| peluda | verde | grande | dura | seguro |
| **lisa** | **vermelho** | grande | macia | perigoso |
| peluda | verde | grande | macia | seguro |
| peluda | vermelho | pequeno | dura | seguro |
| lisa | vermelho | pequeno | dura | seguro |
| lisa | marrom | pequeno | dura | seguro |
| peluda | verde | **pequeno** | macia | perigoso |
| **lisa** | verde | **pequeno** | **dura** | perigoso |
| peluda | vermelho | grande | dura | seguro |
| lisa | marrom | grande | macia | seguro |
| **lisa** | verde | **pequeno** | macia | perigoso |
| peluda | vermelho | pequeno | macia | seguro |
| **lisa** | **vermelho** | grande | **dura** | perigoso |
| lisa | vermelho | pequeno | dura | |
| peluda | verde | pequeno | dura | |

# Naïve Bayes

- E, finalmente:

$$
\begin{aligned}
P(\texttt{pequeno} \mid y = \texttt{perigoso}) &= 3/5 \\
P(\texttt{lisa} \mid y = \texttt{perigoso}) &= 4/5 \\
P(\texttt{vermelho} \mid y = \texttt{perigoso}) &= 2/5 \\
P(\texttt{dura} \mid y = \texttt{perigoso}) &= 2/5 \\
P(x \mid y = \texttt{perigoso}) &= 48/625 \approx 0.0768
\end{aligned}
$$

# Naïve Bayes

- Com isso podemos calcular:

$$P(y = \texttt{seguro} \mid x) \propto 0.0438 \cdot 0.642 \approx 0.0281$$
$$P(y = \texttt{perigoso} \mid x) \propto 0.0768 \cdot 0.358 \approx 0.0275$$

$$P(y = \texttt{seguro} \mid x) \approx 0.0281/(0.0281 + 0.0275) \approx 0.505$$
$$P(y = \texttt{perigoso} \mid x) \approx 0.0275/(0.0281 + 0.0275) \approx 0.494$$

# Naïve Bayes

- Existem alguns pontos nesse algoritmo a serem observados:
  - $P(x)$ é um denominador comum para todas as probabilidades, como queremos obter a maior probabilidade, podemos omitir.
  - Se um certo atributo categórico nunca apareceu na base de dados, sua probabilidade será $0$ e teremos um resultado final zero independente das demais evidências.
  - Se nosso vetor de atributos é muito grande, o termo $P(x_i \mid y = y_i)$ tende a zero, resultado da multiplicação de muitos valores pequenos.

# Valor Novo

- Para lidarmos com um novo valor para um atributo utilizamos o add-one smoothing, e toda estimativa de probabilidade é calculada como:

$$P(x) = \frac{f(x) + 1}{n + d},$$

- com $f(x)$ sendo a frequência de $x$ na base, $n$ o número de amostras/objetos na base, $d$ é o número de valores distintos daquele atributo.

# Valor Novo

- Dessa forma um novo valor para um atributo terá sempre a probabilidade de $1/(n+d)$.
- Isso indica uma pequena probabilidade de um valor novo aparecer em uma nova amostra.

# Underflow

- Para evitar que alguma produtória se torne zero, também conhecido como underflow, podemos aplicar uma transformação na nossa fórmula da probabilidade de tal forma que a relação entre duas probabilidade se mantenha, ou seja:

$$P(A) < P(B) \rightarrow g(P(A)) < g(P(B))$$
$$P(A) > P(B) \rightarrow g(P(A)) > g(P(B))$$
$$P(A) = P(B) \rightarrow g(P(A)) = g(P(B))$$

Uma escolha para a função $g(.)$ é o logaritmo.

# Underflow

- Aplicando o logaritmo em nossa função de probabilidade, temos:

$$\log P(y = \text{seguro} \mid x) = \log \left( P(y = \text{seguro}) \right) + \sum_i \log \left( P(x_i \mid y = \text{seguro}) \right)$$

# Naïve Bayes

- O algoritmo Naïve Bayes deve ser implementado primeiro pré-calculando a tabela de frequências dos atributos condicionados a classe e das classes. A ideia é gerar tabelas de hash com a chave sendo o atributo desejado e o valor a frequência.

# Naïve Bayes: exercício

- Desenvolva o algoritmo MR para classificar um objeto pelo Naïve Bayes
  - Para a tabela de classes, a chave será cada uma das classes. Ex.: (”seguro”, 10.0).
  - Para a tabela de atributos, a chave será composta por índice do atributo, valor do atributo e classe. Ex.: ((2, ”pequeno”, ”perigoso”), 15.0).
  - Já sabemos calcular a frequência de valores categóricos. Basta implementarmos uma função para gerar as transformações da base na forma que desejamos.

# k-Vizinhos mais próximos

# Vizinhos mais próximos

- Existe outro processo de aprendizado supervisionado que ainda não foi explorado.
- Uma técnica bem simples e que apresenta um desempenho razoável em muitas aplicações.

# Vizinhos mais próximos

- Digamos que queremos descobrir se um certo $\mathbf{x}_i$ pertence a classe $y_i = 1$ ou $y_i = 2$.
- Se tivermos uma coleção de exemplos $X$ já classificados, podemos dizer que $\mathbf{x}_i$ pertence a mesma classe do $\mathbf{x}' \in X$ mais similar a ele.

# Vizinhos mais próximos

- Esse processo pode ser falho pois:
  - O mais similar pode ter sido classificado incorretamente.
  - A amostra está na fronteira entre as classes 1 e 2 tendo chance do mais próximos ser da classe errada.

# k-Vizinhos mais próximos

- Para resolver, escolhemos os $k$ pontos mais similares e calculamos a moda das classes desse ponto.
- Dessa forma reduzimos a possibilidade de erros.

# k-Vizinhos mais próximos: Exercício

- Desenvolva um algoritmo MR para os k-Vizinhos Mais Próximos