Classificação: Perceptron, Regressão Logística e Naive Bayes

Fabrício Olivetti de França

Universidade Federal do ABC

Tópicos

1. Classificação
2. Classificadores Lineares
3. Regressão Logística

UFABC

Classificação

Problema de Classificação

Similar ao problema de regressão 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:

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

UFABC

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

Perceptron

Pensando apenas no caso de classificação binária, em que temos apenas duas classes: -1 e $+1$.

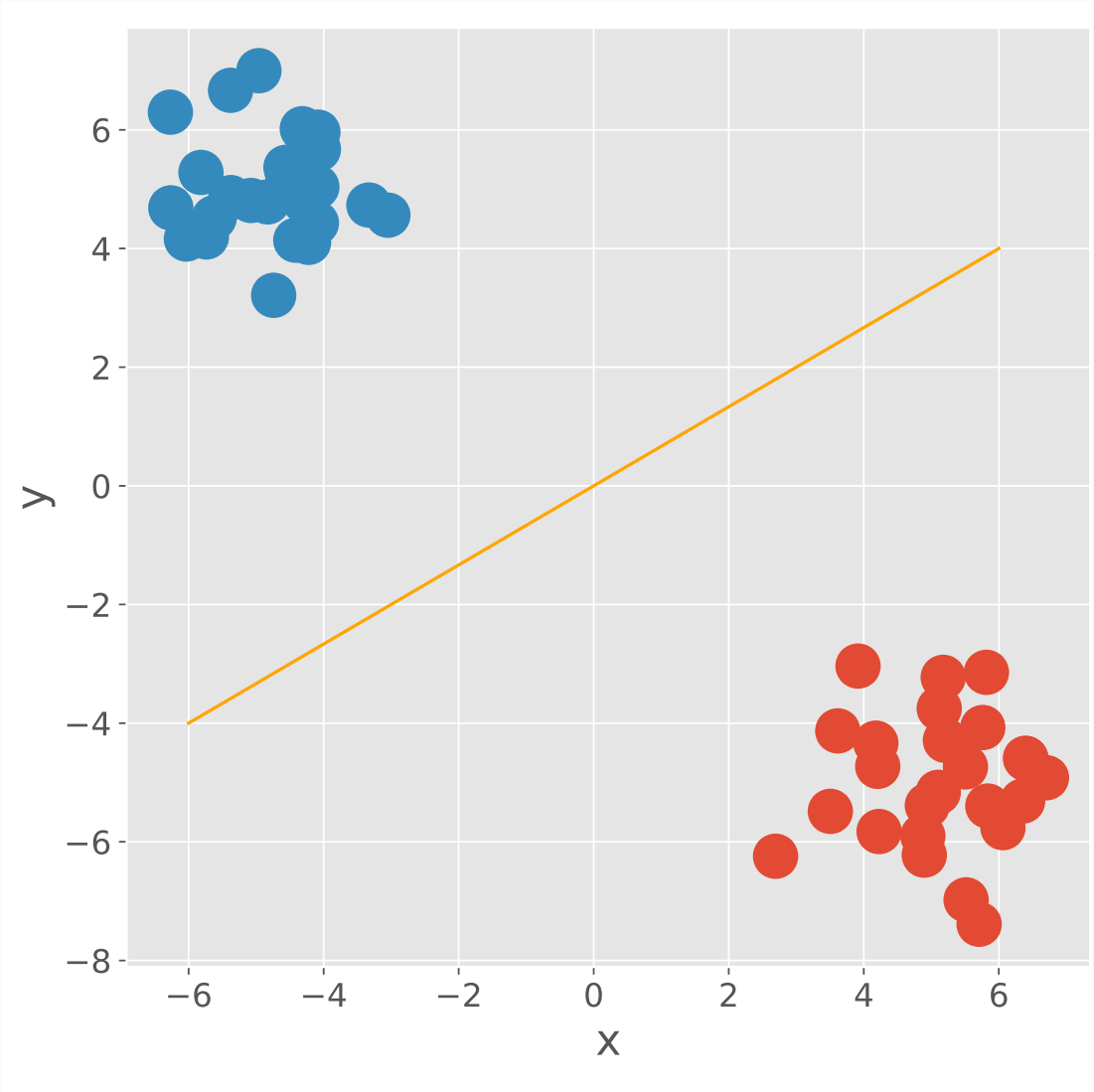
Podemos definir uma função:

$f(x) = w \cdot x$ tal que se $w \cdot x < \theta$, x pertence a classe -1; e se $w \cdot x > \theta$, x pertence a classe $+1$.

O caso $w \cdot x = \theta$ resulta em uma falha de classificação.

Perceptron

Graficamente, essa função pode ser interpretada da mesma forma que a regressão linear:



UFABC

Perceptron

O problema se torna encontrar o vetor de pesos **w** e o parâmetro de limiar θ que maximiza a classificação correta.

Supondo $\theta = 0$, nos limitamos em encontrar **w**.

UFABC

Perceptron

```
def perceptron(x, y, alpha, max_it):
  (n,d) = X.shape
  w     = np.random.normal(size=(1,d))
  for it in range(max_it):
    yhat  = X @ w.T
    erro  = yhat == y
    w = alpha * erro * yhat
  return w
```

UFABC

Perceptron - Scikit-Learn

```
from sklearn.linear_model import Perceptron

clf = Perceptron()
clf.fit(X, y)
yi = clf.predict(xi)
```

UFABC

Perceptron

Embora exista garantia de convergência quando as classes são linearmente separáveis, se não for o caso o vetor **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 θ no vetor de atributos com o valor -1, com isso ajustamos também seu valor.

Ao atualizar w na posição de θ, fazemos o oposto da ação feita para as posições de x.

Regressão Logística

Regressão Logística

O algoritmo perceptron é muito similar a Regressão Linear, pois busca por um vetor $\mathbf{w}$ e um escalar θ que minimize o erro de predição.

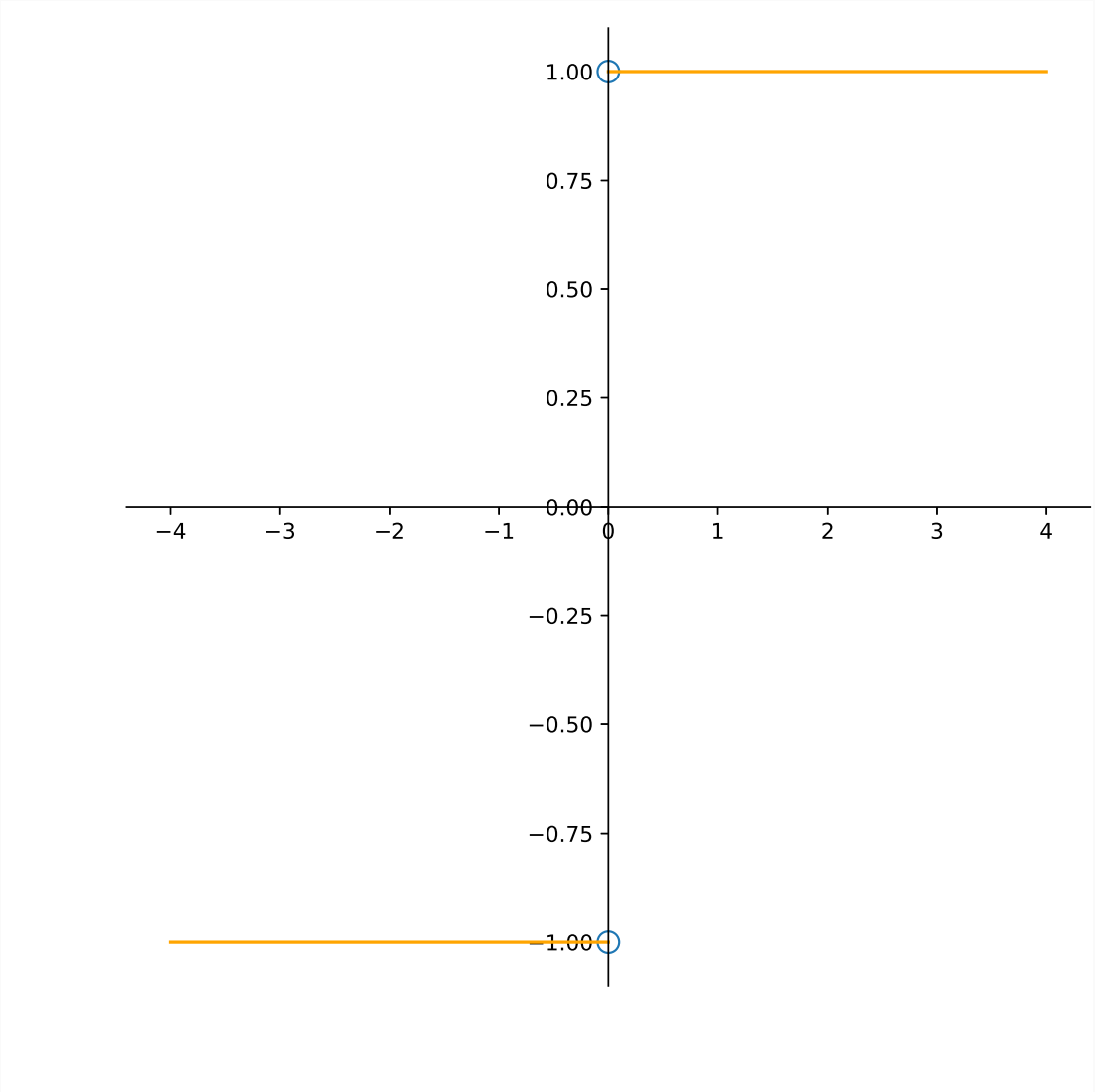
Podemos então adaptar o algoritmo do Gradiente Descendente para podermos encontrar um ótimo local para o problema de classificação. Para tanto, podemos utilizar a função sinal:

$$\hat{y} = sign(\mathbf{w} \cdot \mathbf{x}),$$

com a função *sign* retornando -1 se o resultado for negativo e $+1$, caso contrário.

Perceptron

Mas assim como a função valor absoluto, a função sinal também apresenta descontinuidade no ponto $x = 0$:



UFABC

Regressão Logística

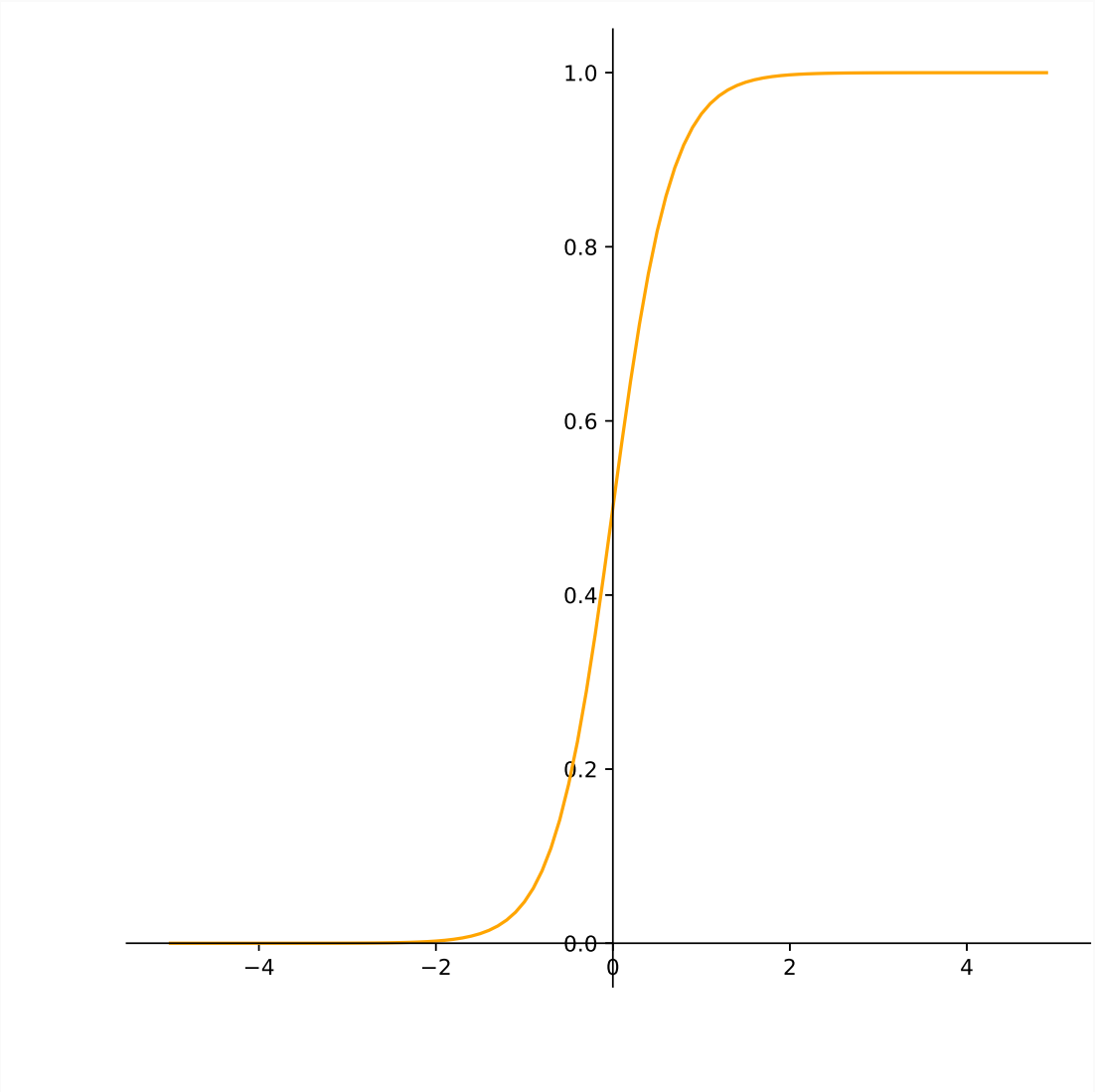
Algumas alternativas viáveis são as funções logísticas, um exemplo de tal função é a sigmoid:

$$f(\mathbf{x}) = \hat{y} = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}.$$

UFABC

Regressão Logística

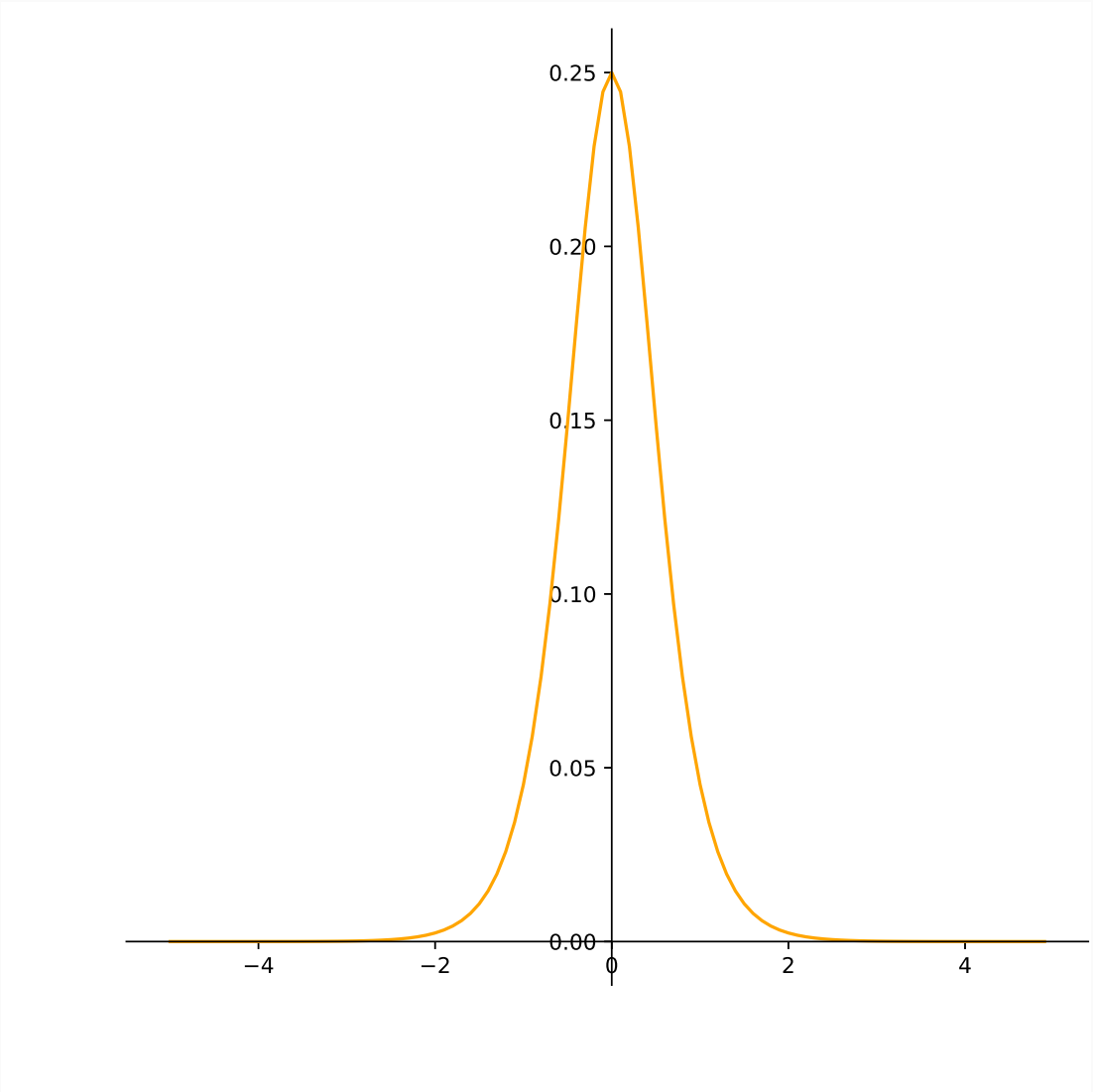
Que graficamente apresenta o seguinte comportamento:



UFABC

Regressão Logística

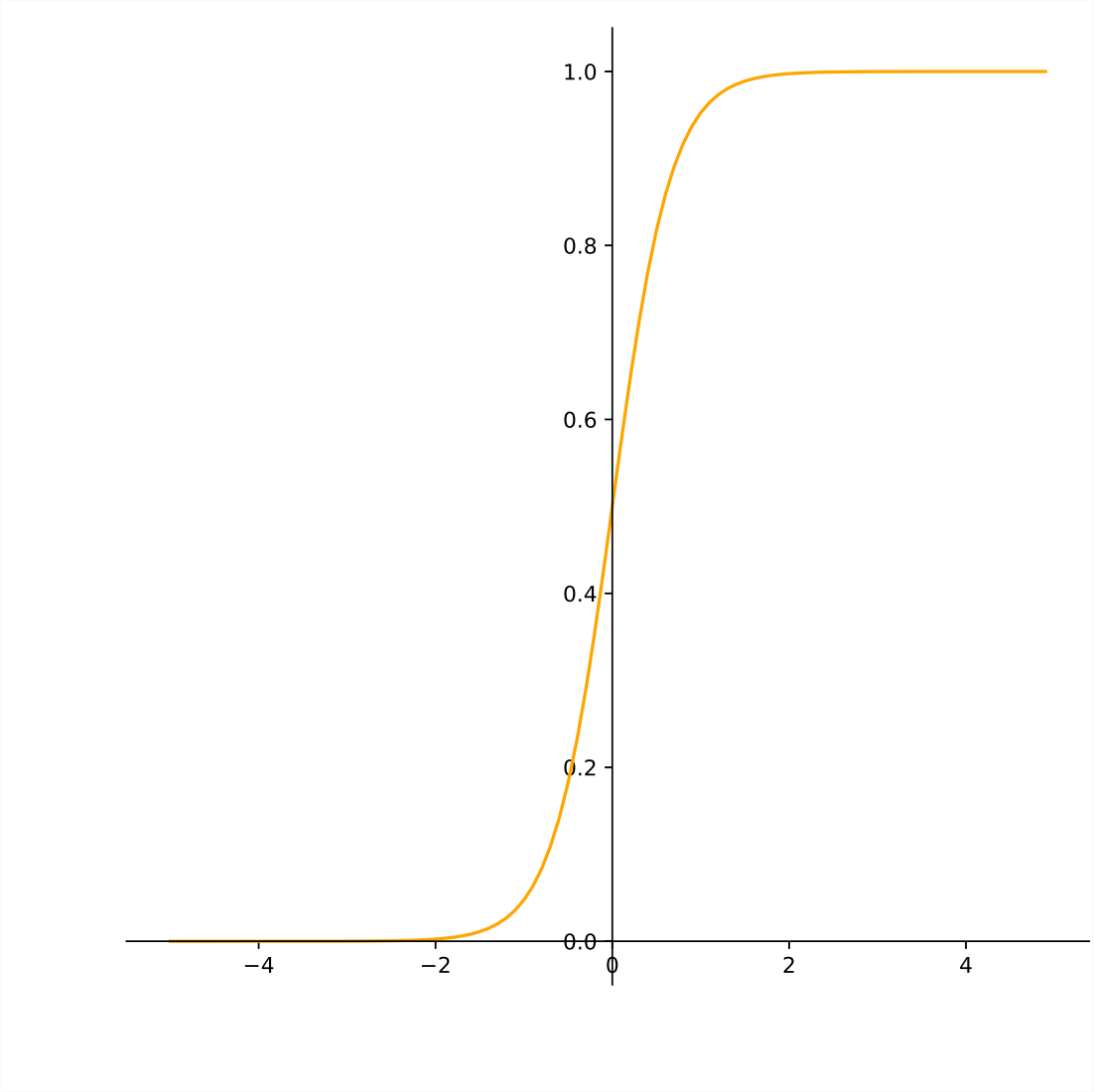
E a derivada $\hat{y} \cdot (1 - \hat{y})$:



UFABC

Regressão Logística

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



UFABC

Regressão Logística

A função sigmoid representa a probabilidade de $\mathbf{x}$ pertencer a classe $y = 1$.

Como consequência, $1 - f(\mathbf{x})$ irá representar a probabilidade de $\mathbf{x}$ pertencer a classe $y = 0$.

Regressão Logística

A função de erro deve ser definida como:

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

UFABC

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 $-\log(1 - \hat{y}) \to 0$.

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

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

E o erro médio:

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

A derivada parcial em função de w_j fica:

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

E o gradiente:

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

Regressão Logística

O algoritmo se torna idêntico ao de Regressão Linear com apenas uma modificação:

```
def logistic(x):
  return 1.0/(1. + np.exp(x))

def gradDesc(X, y, alpha, max_it):
    (n,d) = X.shape
    w     = np.random.normal(size=(1,d))
    for it in range(max_it):
        yhat  = logistic(X @ w.T)
        erro  = y - yhat
        nabla = np.mean(erro*x, axis=0)
        w    += alpha * nabla
    return w
```

UFABC

Regressão Logística - Scikit-Learn

```
from  sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(X,y)
yi = clf.predict(xi)
```

UFABC

Regressão Logística

Todas as adaptações do algoritmo de Gradiente Descendente para Regressão Linear, também funcionam para regressão logística:

- Regularização.
- Batch, Stochastic.
- etc.

Multi Classes

Para adaptar esse algoritmo para o caso de múltiplas classes, podemos adotar duas estratégias:

- One-vs-All
- One-vs-One

One-vs-All

Dadas c classes, construímos c classificadores com a regressão logística.

Cada classificador i vai tentar separar a classe i das demais.

Para uma nova amostra, classificamos ela de acordo com o classificador que devolver a maior probabilidade.

One-vs-One

Nesse caso construímos $c \cdot (c-1)/2$ classificadores, cada um responsável por classificar cada par de classe.

Da mesma forma, podemos predizer utilizando a maior resposta dos classificadores.

Próxima Aula

Na próxima aula aprenderemos sobre:

- Áprendizado Probabilístico
- Naive Bayes

UFABC