Carlos Henrique Venturi Ronchi

# Estudo matemático do reconhecimento de caracteres

Brasil

2017

Carlos Henrique Venturi Ronchi

# Estudo matemático do reconhecimento de caracteres

Relatório apresentado à disciplina Trabalho de Conclusão de Curso para Bacharelado II como avaliação parcial do semestre no curso de bacharelado em Matemática na Universidade Federal do Paraná.

Universidade Federal do Paraná

Departamento de Matemática

Brasil

2017

# Resumo

Este trabalho reúne conceitos e resultados básicos de Otimização e Aprendizagem de máquinas, além de uma aplicação prática no reconhecimento de caracteres. No primeiro capítulo são discutidos métodos de otimização e seus aspectos teóricos, apresentando o problema dos quadrados mínimos, métodos do gradiente, gradiente conjugado e gradiente estocástico. No segundo capítulo utilizou-se os métodos de otimização para descrever a resolução de problemas na aprendizagem de máquinas. Por último, apresenta-se o reconhecimento de caracteres através da otimização e aprendizagem de máquinas.

**Palavras-chaves**: otimização, aprendizagem de máquinas, reconhecimento, redes neurais.

# Sumário

# 1 Introdução

O poder computacional tem aumentado a cada dia, e assim o uso de métodos computacionais para resolver problemas do dia a dia vem evoluindo constantemente. A aprendizagem de máquinas é uma área que se utiliza desse poder computacional e da matemática para obter soluções para problemas.

Com a otimização, podemos encontrar os melhores parâmetros para modelos baseados em observações empirícas. Tais modelos são a representação matemática de possíveis relações observadas na natureza, por exemplo. Assim, com os métodos de otimização desenvolvidos, como os métodos do gradiente e de Newton, os modelos podem ser bem estimados.

No primeiro capítulo, discutiremos os principais métodos de otimização para a solução de problemas sem restrição. Sendo o primeiro deles o problema de quadrados mínimos e após isso os métodos do gradiente, gradiente conjugado e gradiente estocástico. Também apresentamos uma análise dos métodos, como limitações e convergência.

O Capítulo 2 explica o conceito de aprendizagem de máquina, e como aplicar os métodos descritos no primeiro capítulo para problemas de regressão e classificação. Além do uso dos métodos usuais, há uma breve introdução a redes neurais e máquinas de vetores suporte, duas técnicas amplamente utilizadas nesta área.

Já no último capítulo, descrevemos como é possível reconhecer caracteres em palavras escritas à mão fazendo uso dos métodos descritos no capítulo anterior, utilizando um banco de dados com mais de 150 mil imagens de letras em preto e branco. As técnicas utilizadas e estudos realizados neste trabalho servem como base para o projeto de escaneamento de textos escritos à mão. Todos os códigos presentes neste trabalho encontram-se disponíveis online, no repositório tcc-mat hospedado na plataforma Github.

# 2 Otimização - 1ª parte

A otimização se encontra presente no dia a dia, quando se quer encontrar as melhores condições para lucrar com uma empresa, ou quando se quer minimizar os gastos de uma linha de produção. Para solucionar esses problemas muitos métodos foram criados, gerando uma teoria muito abrangente. Um dos casos é a minimização sem restrição, que consiste em minimizar uma função $f$ sem nenhuma condição imposta no seu domínio, denotada por:

$$\min f(x),$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é chamada de função objetivo (NOCEDAL; WRIGHT, 2006). Esta função $f$ é proveniente da modelagem do problema, como os custos da linha de produção ou o custo de funcionamento de uma empresa. Esse problema de minimização pode ser difícil de resolver, para isso foram criados alguns métodos, como o Método do Gradiente, Método dos Gradientes Conjugados e o Método de Newton.

Após a utilização de um método de otimização, precisamos verificar se a solução obtida é realmente o que se procurava, ou seja, se o problema é solucionado. Para isso, existem as *condições de otimalidade*, que analisam se o conjunto solução é de fato uma resposta para o problema. Uma condição de otimalidade bastante utilizada na prática é se o gradiente da função no ponto é 0, que é escolhida como critério de parada do algoritmo, por exemplo 1. Existem também outras condições de otimalidade, que não são usadas na prática. Caso as condições de otimalidade não sejam satisfeitas, isto pode nos dar uma ideia de como a solução pode ser melhorada. Neste presente capítulo discutiremos sobre alguns métodos da otimização, assim como suas soluções, por exemplo, o ponto crítico da função.

## 2.1 Quadrados mínimos

Nesta seção utilizamos como referência (BURDEN; FAIRES, 2010).

O problema de quadrados mínimos surge da necessidade de aproximar dados eficientemente, determinando a melhor aproximação linear quando o erro envolvido é a soma do quadrado das diferenças entre os valores de $y$ dados e os valores de $y$ na reta. Isto é, para o conjunto de dados $\{(x_i, y_i) \mid \ i = 1, 2, \ldots, m\} \in \mathbb{R}^2$ definimos o erro.

$$E_2(a_0, a_1) = \sum_{i=1}^{m} [y_i - (a_1 x_i + a_0)]^2$$

Para os dados da Tabela 1, encontramos a reta representada na Figura 1.

Tabela 1: Dados.

| i | $x_i$ | $y_i$ |
|---|---|---|
| 1 | 1,0 | 2,0 |
| 2 | 3,5 | 4,1 |
| 3 | 2,1 | 6,2 |
| 4 | 4,5 | 9,2 |
| 5 | 2,9 | 5,0 |

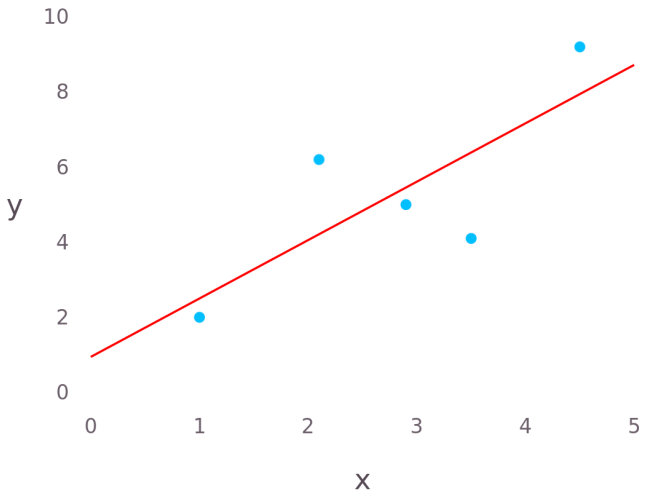


Figura 1: Aproximação dos dados utilizando uma reta, em vermelho neste caso.

O problema de quadrados mínimos não se resume apenas a aproximações lineares. Podemos utilizá-lo para aproximar um conjunto de dados, como acima, com um polinômio

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0,$$

de grau $n < m - 1$. Um exemplo de aproximação é dado pela a Figura 2 utilizando a Tabela 2 de dados com um polinômio de grau 2 para aproximação dos dados.

Tabela 2: Dados.

| i | $x_i$ | $y_i$ |
|---|---|---|
| 1 | 0,00 | 1,000 |
| 2 | 0,25 | 1,2840 |
| 3 | 0,50 | 1,6487 |
| 4 | 0,75 | 2,1170 |
| 5 | 1,00 | 2,7183 |

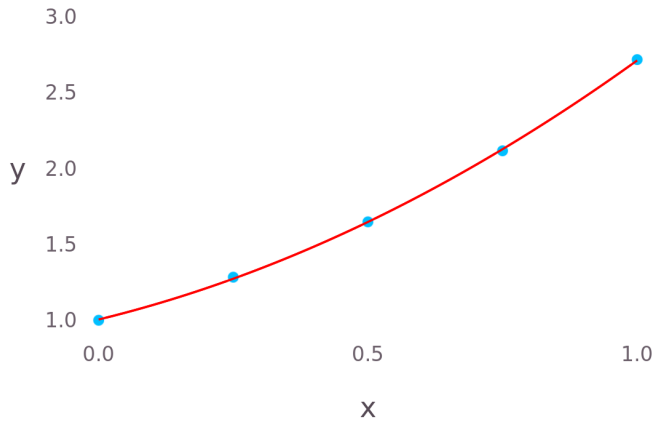


Figura 2: Aproximação dos dados utilizando um polinômio de segundo grau.

### 2.1.1 Determinando o menor erro

O problema de quadrados mínimos pode ser resolvido minimizando o erro total dado um conjunto de dados $\{(x_i, y_i) \mid \ i = 1, 2, \ldots, m\}$, considerando que o erro pode ser dado por

$$E(a_1, ..., a_n) = \sum_{i=1}^{m} [y_i - z(x_i)]^2, \tag{2.1}$$

em que $z(x)$ é a função que tenta melhor aproximar os valores de $y$. Considere $z(x) = \sum_{j=1}^{n} a_j f_j(x)$.

Portanto, o nosso problema consiste em tentar encontrar o mínimo de $E(a_1, ..., a_n)$. Para isso acontecer, vamos derivar em relação a cada parâmetro $a_j$ a Equação (2.1).

$$\frac{\partial E}{\partial a_j} = 0. \tag{2.2}$$

Assim, da Equação (2.2), temos que

$$0 = \frac{\partial}{\partial a_j} \sum_{i=1}^{m} [y_i - \sum_{k=1}^{n} a_j f_k(x_i)]^2 = 2 \sum_{i=1}^{m} [y_i - \sum_{k=1}^{n} a_j f_k(x_i)] f_j(x_i). \tag{2.3}$$

Note que a Equação (2.3) vale para todo $i = 1, ..., m$. Simplificando, obtemos as chamadas *equações normais* dadas abaixo,

$$\sum_{i=1}^{m} (a_1 f_1(x_i) + ... + a_n f_n(x_i)) f_j(x_i) = \sum_{i=1}^{m} y_i f_j(x_i). \tag{2.4}$$

Agora vamos inserir a seguinte notação,

$$\phi_j = [f_j(x_1); f_j(x_2); ...; f_j(x_n)].$$

Note que o lado direito da Equação (2.4) pode ser visto como o produto interno entre o vetor $y$ com $f_j$ aplicada em cada ponto $x_j$, ou seja,

$$\sum_{i=1}^{m} y_i f_j(x_i) = \langle y, \phi_j(x) \rangle. \tag{2.5}$$

Além do mais, observe que para cada $f_j$, temos o seguinte

$$\begin{aligned} \sum_{i=1}^{m} (a_1 f_1(x_i) + ... + a_n f_n(x_i)) f_j(x_i) &= a_1 \sum_{i=1}^{m} f_1(x_i) f_j(x_i) + ... + a_n \sum_{i=1}^{m} f_n(x_i) f_j(x_i) \\ &= a_1 \langle \phi_1, \phi_j \rangle + ... + a_n \langle \phi_n, \phi_j \rangle. \end{aligned} \tag{2.6}$$

Portanto, das Equações (2.5) e (2.6), temos

$$a_1 \langle \phi_1, \phi_j \rangle + ... + a_n \langle \phi_n, \phi_j \rangle = \langle y, f_j(x) \rangle.$$

Para cada $j$, obteremos uma nova equação. Assim encontramos um sistema com $n$ equações que matricialmente pode ser dado como

$$Ma = X^T y, \tag{2.7}$$

onde M pode ser dado como a matriz dos produtos internos entre as funções $f_i$ e X é a matriz, em que cada coluna é o vetor $\phi_j$. A matriz M, também conhecida como matriz de Gram, é dada por

$$M = \begin{bmatrix} \langle \phi_1, \phi_1 \rangle & \langle \phi_2, \phi_1 \rangle & \dots & \langle \phi_n, \phi_1 \rangle \\ \langle \phi_1, \phi_2 \rangle & \langle \phi_2, \phi_2 \rangle & \dots & \langle \phi_n, \phi_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \phi_1, \phi_n \rangle & \langle \phi_2, \phi_n \rangle & \dots & \langle \phi_n, \phi_n \rangle \end{bmatrix},$$

e X é dado por

$$X = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ \vdots & \vdots & & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_n(x_m) \end{bmatrix}.$$

Note que $M$ pode ser escrita como $X^T X$. Logo, M é uma matriz semidefinida positiva, já que

$$w^T M w = w^T X^T X w = (Xw)^T Xw = ||Xw||^2 \geq 0.$$

E o sistema é dado pela Equação (2.8)

$$X^T X a = X^T y. \tag{2.8}$$

Portanto, o problema de quadrados mínimos se resume a determinar quando M é definida positiva, ou seja quando os vetores $\phi_j$ forem linearmente independentes, assim M será invertível e encontramos a solução do nosso problema.

## 2.2 Método do Gradiente

As referências utilizadas nesta seção são encontradas em (SHALEV-SHWARTZ; BEN-DAVID, 2014), (NOCEDAL; WRIGHT, 2006), (RIBEIRO; KARAS, 2014).

Vimos anteriormente que um dos problemas que a otimização trata de resolver são os de minimização irrestrita, ou seja, problemas da forma

$$\begin{aligned} &\min f(x) \\ &\text{sujeito a } x \in \mathbb{R}^n. \end{aligned}$$

Um dos métodos utilizados para minimizar uma função é o método do gradiente, conhecido também como método de Cauchy. Tal método consiste em usar o oposto do gradiente como direção de busca a cada iteração, pois este é o vetor em que $f$ mais decresce. Em cada iteração do método do gradiente, há uma atualização da forma $x^{k+1} = x^k + t_k d^k$, em que o vetor $d^k \in \mathbb{R}^n$ é o oposto do gradiente aplicado no ponto $x^k$.

**Definição 2.2.1.** *Sejam $f : \mathbb{R}^n \to \mathbb{R}$, $\bar{x} \in \mathbb{R}^n$ e uma direção $d \in \mathbb{R}^n - \{0\}$. Dizemos que $d$ é uma direção de descida para $f$, a partir de $\bar{x}$, se existe $\delta > 0$ tal que $f(\bar{x} + td) < f(\bar{x})$, $\forall t \in (0, \delta)$.*

Segue abaixo uma condição suficiente para um vetor ser considerado direção de descida.

**Teorema 2.2.1.** *Se $\nabla f(\bar{x})^T d < 0$, então $d$ é uma direção de descida para $f$, a partir de $\bar{x}$.*

*Demonstração.* Como

$$\nabla f(\bar{x})^T d = \frac{\partial f}{\partial d}(\bar{x}) = \lim_{t \to 0} \frac{f(\bar{x} + td) - f(\bar{x})}{t}, \tag{2.9}$$

pela hipótese e pela equação (2.9), temos que existe $\delta > 0$ tal que para todo $t \in (-\delta, \delta)$ e $t \neq 0$,

$$\frac{f(\bar{x} + td) - f(\bar{x})}{t} < 0.$$

E portanto, $f(\bar{x} + td) < f(\bar{x})$, para $t \in (0, \delta)$. □

Sejam $d = -\nabla f(x)$ e $v \in \mathbb{R}^n$ tal que $\|v\| = \|d\|$, então

$$\frac{\partial f}{\partial d}(x) = \nabla f(x)^T d = -\|\nabla f(x)\|^2 = -\|\nabla f(x)\| \, \|v\| \leq \nabla f(x)^T v = \frac{\partial f}{\partial v}(x).$$

Portanto, temos que o oposto do gradiente é uma direção de descida com o decréscimo mais acentuado. O algoritmo do método do gradiente é dado abaixo.

**Algoritmo 1** Método do Gradiente

1: Dado $x_0 \in \mathbb{R}^n$
2: $k = 0$
3: **enquanto** $\nabla f(x^k) \neq 0$ **faça**
4: Defina $d^k = -\nabla f(x^k)$;
5: Obtenha $t_k > 0$ tal que $f(x^k + t_k d^k) < f(x^k)$;
6: Faça $x^{k+1} = x^k + t_k d^k$;
7: $k = k + 1$;
8: **fim enquanto**
9: **retorne** $x^k$

**Definição 2.2.2.** *Seja $U \subset \mathbb{R}^n$ um conjunto convexo. Uma função $f : U \mapsto \mathbb{R}$ é dita convexa se*

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2), \quad \forall x_1, x_2 \in U, \forall t \in [0,1].$$

**Definição 2.2.3.** *Sejam $U \subset \mathbb{R}^n$, $f : U \mapsto \mathbb{R}$. A função $f$ é dita $L-Lipschitz$, se existe $L \in \mathbb{R}^*$, tal que*

$$|f(x) - f(y)| \leq L \|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Para provar um limitante da diferença entre o valor de $f$ aplicado na sequência gerada pelo método do gradiente e o ponto mínimo, onde f é uma função convexa e Lipschitz, usaremos o seguinte lema.

**Lema 2.2.2.** *Sejam $d^0, ..., d^T$ uma sequência arbitrária de vetores. Qualquer algoritmo com inicialização $x^0 = 0$ e uma atualização da forma $x^{k+1} = x^k - \eta d^k$ satisfaz*

$$\sum_{k=0}^{T} \langle x^k - x^*, d^k \rangle \leq \frac{\|x^*\|^2}{2\eta} + \frac{\eta}{2} \sum_{k=1}^{T} \|d^k\|^2 \tag{2.10}$$

*Em particular, para todo $B, L > 0$, se para todo $k$, $\|d^k\| \leq L$ e $\eta = \sqrt{\frac{B^2}{L^2 T}}$, então para todo $x^*$ com $\|x^*\| \leq B$,*

$$\frac{1}{T} \sum_{k=0}^{T} \langle x^k - x^*, d^k \rangle \leq \frac{BL}{\sqrt{T}}. \tag{2.11}$$

*Demonstração.* Note que

$$\begin{aligned}
\langle x^k - x^*, d^k \rangle &= \frac{1}{\eta} \langle x^k - x^*, \eta d^k \rangle \\
&= \frac{1}{2\eta} \left( - \|x^k - x^* - \eta d^k\|^2 + \|x^k - x^*\|^2 + \eta^2 \|d^k\|^2 \right) \\
&= \frac{1}{2\eta} \left( - \|x^{k+1} - x^*\|^2 + \|x^k - x^*\|^2 \right) + \frac{\eta}{2} \|d^k\|^2,
\end{aligned}$$

onde a última linha segue da atualização. Somando sobre $k$, obtemos

$$\sum_{k=1}^{T} \langle x^k - x^*, d^k \rangle = \sum_{k=1}^{T} \frac{1}{2\eta} \left( - \|x^{k+1} - x^*\|^2 + \|x^k - x^*\|^2 ) \right) + \frac{\eta}{2} \sum_{k=1}^{T} \|d^k\|^2. \tag{2.12}$$

Note que do lado direito da Equação (2.12) temos uma soma telescópica que nos dá

$$-\left\|x^{T+1}-x^*\right\|^2+\left\|x^1-x^*\right\|^2.$$

Logo,

$$\begin{aligned}\sum_{k=1}^{T}\langle x^k-x^*,d^k\rangle &= \sum_{k=1}^{T}\frac{1}{2\eta}\left(-\left\|x^{k+1}-x^*\right\|^2+\left\|x^k-x^*\right\|^2\right)+\frac{\eta}{2}\sum_{k=1}^{T}\left\|d^k\right\|^2\\ &= \frac{1}{2\eta}(-\left\|x^{T+1}-x^*\right\|^2+\left\|x^1-x^*\right\|^2)+\frac{\eta}{2}\sum_{k=0}^{T}\left\|d^k\right\|^2\\ &\le \frac{1}{2\eta}\left\|x^1-x^*\right\|^2+\frac{\eta}{2}\sum_{k=1}^{T}\left\|d^k\right\|^2\\ &= \frac{1}{2\eta}\left\|x^*\right\|^2+\frac{\eta}{2}\sum_{k=0}^{T}\left\|d^k\right\|^2,\end{aligned}$$

onde a última igualdade segue do fato que $x^1 = 0$. Portanto, obtemos a primeira parte do lema. Para a segunda parte do lema, basta limitar $x^*$ por $B$, $d^k$ por $L$ e dividir por $T$ em ambos os lados na Equação (2.10), além de substituir o valor para $\eta$ dado como hipótese. □

Vamos assumir que se $f$ é uma função $L$-Lipschitz, então $\|\nabla f(x^k)\| \le L$. Portanto, satisfazemos as condições do Lema 2.2.2 e obtemos o seguinte resultado.

**Teorema 2.2.3.** *Sejam $f$ uma função convexa e $L$-Lipschitz e $x^* \in \arg\min_{x:\|x\|\le B} f(x)$. Se rodarmos o método do gradiente em $f$ para $T$ passos com $\eta = \sqrt{\frac{B^2}{L^2T}}$, então o vetor de retorno $\bar{x}$, dado pelo método do gradiente, satisfaz*

$$f(\bar{x})-f(x^*)\le\frac{BL}{\sqrt{T}}$$

*Ainda mais, dado $\epsilon > 0$, para obter $f(\bar{x}) - f(x^*) \le \epsilon$, é necessário rodar o algoritmo com um número $T$ de iterações que satisfaz*

$$T\ge\frac{B^2L^2}{\epsilon^2}.$$

Note que o resultado acima vale para um valor de passo fixo, $\eta$. Ou seja, omite-se a quinta linha do algoritmo 1. Para uma escolha do passo que varia a cada iteração, utiliza-se normalmente um método de busca inexata, por exemplo algum baseado no critério de Armijo.

### 2.2.1 Busca inexata - Condição de Armijo

Seja um ponto $\bar{x} \in \mathbb{R}^n$, uma direção de descida $d \in \mathbb{R}^n$ e $\delta \in (0,1)$. A regra de Armijo encontra $t > 0$ tal que

$$f(\bar{x}+td)\le f(\bar{x})+\delta t\nabla f(\bar{x})^T d.$$

O seguinte teorema nos garante a existência de tal $t$.

**Teorema 2.2.4.** *Considere uma função diferenciável $f : \mathbb{R}^n \to \mathbb{R}$, um ponto $\bar{x} \in \mathbb{R}^n$, uma direção de descida $d \in \mathbb{R}^n$ e $\delta \in (0,1)$. Então existe $\beta > 0$ tal que*

$$f(\bar{x} + td) \leq f(\bar{x}) + \delta t \nabla f(\bar{x})^T d,$$

*para todo $t \in [0, \beta)$.*

*Demonstração.* Caso $\nabla f(\bar{x})^T d = 0$, segue da definição de descida a desigualdade. Suponha agora $\nabla f(\bar{x})^T d < 0$. Como $\delta < 1$,

$$\lim_{t \to 0} \frac{f(\bar{x} + td) - f(\bar{x})}{t} = \nabla f(\bar{x})^T d < \delta \nabla f(\bar{x})^T d$$

Portanto, existe $\beta > 0$ tal que

$$\frac{f(\bar{x} + td) - f(\bar{x})}{t} < \delta \nabla f(\bar{x})^T d,$$

para todo $t \in [0, \beta)$. Ou seja

$$f(\bar{x} + td) \leq f(\bar{x}) + \delta t \nabla f(\bar{x})^T d$$

□

Um algoritmo para encontrar um passo satisfazendo Armijo é o seguinte.

**Algoritmo 2** Busca de Armijo

1: Dados $\bar{x} \in \mathbb{R}^n$, $d \in \mathbb{R}^n$, $\gamma, \delta \in (0,1)$
2: $t = 1$
3: **enquanto** $f(\bar{x} + td) > f(\bar{x}) + \delta t \nabla f(\bar{x})^T d$ **faça**
4: $\quad t = \gamma t$
5: **fim enquanto**
6: **retorne** $t$

### 2.2.2 Convergência global do método do gradiente

**Definição 2.2.4.** *Um algoritmo é dito globalmente convergente quando para qualquer sequência $(x^k)$ gerada pelo algoritmo, qualquer ponto de acumulação $x^*$ de $(x^k)$ é um ponto estacionário, ou seja, $\nabla f(x^*) = 0$.*

Em vista da definição acima, dizemos que o método do gradiente é globalmente convergente, como diz o teorema abaixo

**Teorema 2.2.5.** *O Algoritmo 1 com o tamanho do passo calculado pela condição de Armijo, Algoritmo 2, é globalmente convergente.*

A prova deste resultado pode ser encontrada no livro (RIBEIRO; KARAS, 2014, p. 59).

## 2.3 Gradientes conjugados

As referências utilizadas nesta seção são encontradas em (SHALEV-SHWARTZ; BEN-DAVID, 2014), (NOCEDAL; WRIGHT, 2006), (RIBEIRO; KARAS, 2014).

O método dos gradientes conjugados é um método iterativo para a minimização de funções quadráticas convexas, que garante encontrar o minimizador global em um número finito de etapas utilizando direções conjugadas.

### 2.3.1 Direções conjugadas

**Definição 2.3.1.** *Dada uma matriz $A \in \mathbb{R}^{n\times n}$ simétrica definida positiva, dizemos que os vetores $d^0, ..., d^k \in \mathbb{R}^n - \{0\}$ são $A$-conjugados quando*

$$(d^i)^T A d^j = 0, \quad \forall i, j = 0, ..., k \ e \ i \neq j.$$

**Lema 2.3.1.** *Seja $A$ uma matriz simétrica definida positiva, e $\{d^0, ..., d^k\}$ vetores $A$ - conjugados. Então $\{d_0, ..., d_k\}$ é linearmente independente.*

*Demonstração.* Sejam $\alpha_0, ..., \alpha_k$ tais que

$$\alpha_0 d^0 + ... + \alpha_k d^k = 0.$$

Multiplicando por $(d^i)^T A$ em ambos os lados, para $i$ qualquer entre 0 e $k$, e usando o fato de que os vetores são $A$-conjugados, temos que

$$\alpha_i (d^i)^T A d^i = 0.$$

Como $A$ é uma matriz definida positiva, temos que

$$\alpha_i = 0, \ \forall i = 0, 1, ..., k. \tag{2.13}$$

□

### 2.3.2 O método dos gradientes conjugados

Seja $f : \mathbb{R}^n \to \mathbb{R}$ da forma $f(x) = c + b^T x + \frac{1}{2} x^T A x$, onde $A$ é uma matriz simétrica definida positiva, $c \in \mathbb{R}$ e $b \in \mathbb{R}^n$. Considere o seguinte problema de otimização como anteriormente visto

$$\begin{aligned} &\min \ f(x) \\ &\text{sujeito a } x \in \mathbb{R}^n. \end{aligned}$$

Para minimizar a função utilizaremos o método dos gradientes conjugados, que consiste em utilizar $n$ direções conjugadas, que vão servir como direção de descida para o método.

Ou seja, considere $x^{k+1} = x^k + t_k d^k$, onde $\{d^0, ..., d^{n-1}\}$ são direções conjugadas. O objetivo é encontrar $t_k$ de modo que se obtenha um mínimo para a função $f$ nessa direção. Caso $f$ possua um ponto crítico, e ele for mínimo local, então o ponto é mínimo global, pois a função f é convexa e diferenciável. Como tem-se $n$ direções conjugadas e elas são *linearmente independentes*, o minimizador global pode ser escrito como combinação linear das direções conjugadas. Para encontrar $t_k$ considere a função $\varphi : \mathbb{R} \to \mathbb{R}$, dada por

$$\varphi(t) = f(x_k + td_k),$$

onde $f$ é a função acima definida. Então, definimos $t_k$ como o minimizador da função $\varphi$. Portanto, note que

$$\begin{aligned} 0 = \varphi'(t_k) &= \nabla f(x^k + t_k d^k)^T d^k \\ &= (A(x^k + t_k d^k) + b)^T d^k \\ &= ((x^k)^T + t_k (d^k)^T) A d^k + b^T d^k. \end{aligned}$$

Como $A$ é uma matriz simétrica definida positiva, temos que $(d^k)^T A d^k > 0$ e portanto

$$t_k = \frac{-\nabla f(x^k)^T d^k}{(d^k)^T A d^k}.$$

**Teorema 2.3.2.** *Seja $f : \mathbb{R}^n \to \mathbb{R}$ e denote por $x^*$ seu minimizador global. Então para qualquer $x^0 \in \mathbb{R}^n$, o método das direções conjugadas converge para o mínimo global em no máximo n passos.*

*Demonstração.* De fato, como $\{d^0, ..., d^{n-1}\}$ são direções conjugadas, logo, pelo Lema 2.3.1, temos que $\{d^0, ..., d^{n-1}\}$ forma uma base para $\mathbb{R}^n$. Seja agora

$$x^* - x^0 = \sum_{i=0}^{n-1} \alpha_i d^i. \tag{2.14}$$

Vamos mostrar que $\alpha_i = t_i$. De fato, multiplicando a equação (2.14) por $(d^k)^T A$ pela esquerda, e usando o fato de que os vetores $d^j$ são $A-conjugados$, temos que $(d^k)^T A d^j = 0$, $\forall j \neq k$ e portanto

$$(d^k)^T A x^* - (d^k)^T A x^0 = \alpha_k (d^k)^T A d^k,$$

logo

$$\alpha_k = \frac{(d^k)^T A x^* - (d^k)^T A x^0}{(d^k)^T A d^k}. \tag{2.15}$$

Além disso, note que

$$x^k = x^0 + t_0 d^0 + t_1 d^1 + ... + t_{k-1} d^{k-1}. \tag{2.16}$$

Logo, de (2.16) e como $d^j$ são vetores $A$-conjugados,

$$(d^k)^T A x^k = (d^k)^T A x^0.$$

E como $\nabla f(x^*) = 0 = Ax^* - b$ por hipótese, temos que

$$\alpha_k = \frac{(d^k)^T Ax^* - (d^k)^T Ax^k}{(d^k)^T Ad^k} = \frac{(d^k)^T b - (d^k)^T Ax^k}{(d^k)^T Ad^k} = -\frac{\nabla f(x^k)^T d^k}{(d^k)^T Ad^k} = t_k.$$

Portanto, segue que $\alpha_k = t_k$ e que $x^* = x^n$. □

Note que para o método acima foi necessário o uso de $n-1$ vetores $A$-conjugados, previamente dados. Na prática é necessário obtê-los de alguma forma. Para tanto, utiliza-se os seguintes passos. Dado um ponto $x^0 \in \mathbb{R}^n$ qualquer, inicialize com $d^0 = \nabla f(x^0)$. Tome $d^{k+1} = -\nabla f(x^{k+1}) + \beta_k d^k$. Como é necessário que os vetores $\{d^0, ..., d^{n-1}\}$ sejam $A$-conjugados, ou seja, $(d^j)^T Ad^i = 0$, $\forall i \neq j$. Então

$$0 = (d^k)^T Ad^{k+1} = -(d^k)^T A\nabla f(x^{k+1}) + \beta_k (d^k)^T Ad^k \implies \beta_k = \frac{(d^k)^T A\nabla f(x^{k+1})}{(d^k)^T Ad^k}. \quad (2.17)$$

Ou seja, o conjunto $\{d^0, ..., d^{n-1}\}$ é um conjunto de vetores $A$-conjugados considerando $\beta_k$ da equação (2.17). O pseudocódigo para o método do gradiente conjugado é dado pelo algoritmo 2.3.2.

**Algoritmo 3** Método do Gradiente Conjugado

1: Dado $x^0 \in \mathbb{R}^n$ e $\delta \in (0, 1)$
2: $k = 0$
3: $d^0 = -\nabla f(x^0)$
4: **enquanto** $\|\nabla f(x^k)\| > \delta$ **faça**
5: Faça $t_k = \dfrac{-\nabla f(x^k)^T d^k}{(d^k)^T Ad^k}$;
6: Faça $x^{k+1} = x^k + t_k d^k$;
7: Faça $\beta_k = \dfrac{(d^k)^T A\nabla f(x^{k+1})}{(d^k)^T Ad^k}$;
8: Faça $d^{k+1} = -\nabla f(x^{k+1}) + \beta_k d^k$
9: $k = k + 1$;
10: **fim enquanto**
11: **retorne** $x^k$

# 3 Conceitos de estatística

Nos próximos capítulos utilizaremos alguns conceitos de estatística. Portanto, este capítulo serve para introduzi-los. A principal referência para este capítulo é (JAMES, 2011).

## 3.1 Definições básicas

Suponhamos que um experimento seja realizado sob certas condições fixas. Seja $\Omega$ o conjunto de resultados possíveis, onde por resultado possível entende-se o resultado elementar e indivisível do experimento. $\Omega$ será chamado de *espaço amostral* do experimento.

**Exemplo 3.1.1.** *Realizar o lançamento de um dado não viciado nos dá*

$$\Omega = \{1, 2, 3, 4, 5, 6\}.$$

**Definição 3.1.1.** *Seja $\Omega$ espaço amostral. Todo subconjunto $A \subset \Omega$ é chamado de evento. $\Omega$ é o resultado certo, $\varnothing$ o evento impossível. Se $\omega \in \Omega$, o evento $\{\omega\}$ é dito elementar (ou simples).*

**Definição 3.1.2.** *Seja $\Omega$ um conjunto não vazio. Uma classe $\mathcal{A}$ de subconjuntos de $\Omega$ é chamada de álgebra se as seguintes condições são satisfeitas.*

1. $\Omega \in \mathcal{A}$
2. $A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A}$
3. $A, B \in \mathcal{A} \Rightarrow A \cup B \in \mathcal{A}$

**Proposição 3.1.1.** *Seja $\mathcal{A}$ uma álgebra de conjuntos de $\Omega$. Então valem as seguintes propriedades.*

1. $\varnothing \in \mathcal{A}$
2. $\forall n, \forall A_1, \ldots, A_n \in \mathcal{A}$, *temos* $\bigcup_{i=1}^{n} A_i \in \mathcal{A}$ *e* $\bigcap_{i=1}^{n} A_i \in \mathcal{A}$

Se substituirmos a terceira propriedade da definição 3.1.2 por

$$3\text{'. Se } A_n \in \mathcal{A} \quad \forall n \geq 1, \text{ então } \bigcup_{i=1}^{\infty} A_n \in \mathcal{A},$$

obtemos a seguinte definição.

**Definição 3.1.3.** *Uma classe $\mathcal{A}$ de subconjuntos de um conjunto não vazio $\Omega$ satisfazendo $1, 2, 3'$ é chamada $\sigma-$álgebra de subconjuntos de $\Omega$.*

Caso $\Omega$ seja um subconjunto dos reais, é conveniente definir a $\sigma-$álgebra de Borel na reta.

**Definição 3.1.4.** *A $\sigma-$álgebra de Borel na reta é a menor $\sigma-$álgebra contendo todos os intervalos.*

**Definição 3.1.5.** *Seja $\Omega$ um conjunto não vazio e $\mathcal{A}$ uma $\sigma-$álgebra associada a $\Omega$. Então $P : \mathcal{A} \to \mathbb{R}$ é chamada de medida de probabilidade em $\mathcal{A}$ se*

1. $P(A) \geq 0, \quad \forall A \in \mathcal{A}$,
2. $P(\Omega) = 1$,
3. *($\sigma-$Aditividade) Se $A_1, A_2, \cdots \in \mathcal{A}$ são disjuntos, então*

$$P\left(\bigcup_{i=1}^{\infty} A_n\right) = \sum_{i=1}^{\infty} P(A_n).$$

Segue algumas propriedades de uma probabilidade $P$ em um $\sigma-$álgebra $\mathcal{A}$. Suponha que todo $A$ abaixo pertença a $\mathcal{A}$.

1. $P(A^c) = 1 - P(A)$,
2. $0 \leq P(A) \leq 1$,
3. $A_1 \subset A_2 \Rightarrow P(A_1) \leq P(A_2)$,
4. $P(\cup_{i=1}^{\infty} A_n) \leq \sum_{i=1}^{\infty} P(A_n)$.

**Definição 3.1.6.** *Um espaço de probabilidade é um trio $(\Omega, \mathcal{A}, P)$, onde*

*a) $\Omega$ é um conjunto não-vazio,*

*b) $\mathcal{A}$ é uma $\sigma-$álgebra de subconjuntos de $\Omega$,*

*c) $P$ é uma probabilidade em $\mathcal{A}$.*

**Definição 3.1.7.** *Seja $\Omega$ um conjunto não vazio, $\mathcal{A}$ uma $\sigma-$álgebra em $\Omega$. A probabilidade condicional de $A$ dado $B$, onde $B \in \mathcal{A}$ e $P(B) > 0$ é dada por*

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

**Teorema 3.1.2.** *Teorema da probabilidade Total (ou Absoluta) Se a sequência (finita ou enumerável) de eventos aleatórios $A_1, A_2, \ldots$ formam um partição de $\Omega$, então*

$$P(B) = \sum_i P(A_i) P(B|A_i)$$

## 3.2 Variáveis Aleatórias

Informalmente, uma variável aleatória é uma função que sai de um espaço amostral e vai para a reta real.

**Definição 3.2.1.** *Uma variável aleatória $X$ em um espaço de probabilidade $(\Omega, \mathcal{A}, P)$ é uma função real definida no espaço $\Omega$, $X : \Omega \to \mathbb{R}$ tal que*

$$X^{-1}(I) = \{\omega \in \Omega \mid X(\omega) \in I\} \in \mathcal{A},$$

*para todo intervalo $I \in \mathbb{R}$. Em palavras, $X$ é tal que sua imagem inversa de intervalos $I \in \mathbb{R}$ percentem a $\sigma-$álgebra $\mathcal{A}$.*

**Definição 3.2.2.** *Seja $X$ uma variável aleatória. Se o número de valores possíveis de $X$ for enumerável, denominamos $X$ de variável aleatória discreta.*

**Definição 3.2.3.** *A função de probabilidade de uma variável aleatória discreta é uma função que atribui probabilidade a cada um dos possíveis valores assumidos pela variáveis. Isto é, sendo $X$ uma variável aleatória discreta com valores $x_1, x_2, \ldots,$ temos para $i \geq 1$,*

$$p(x_i) = P(X = x_i) = P(\{\omega \in \Omega \mid X(\omega) = x_i\}),$$

*em que $p(x_i)$ satisfaz*

1. $0 \leq p(x_i) \leq 1, \quad \forall i \geq 1,$
2. $\sum_i p(x_i) = 1$

**Exemplo 3.2.1.** *Considere 2 lançamentos indepentendes de uma moeda equilibrada. Logo, o espaço amostral é dado por $\Omega = \{cc, c\bar{c}, \bar{c}c, \bar{c}\bar{c}\}$, em que $c$ representa cara e $\bar{c}$ representa coroa. Defina $X$ como sendo o número de caras. Portanto, $X$ pode assumir os valores $0, 1$ ou $2$. Assim,*

1. $p(0) = P(X = 0) = 1/4,$
2. $p(1) = P(X = 1) = 1/2,$
3. $p(2) = P(X = 2) = 1/4.$

**Definição 3.2.4.** *Diz-se que $X$ é uma variável aleatória contínua, se existir uma função $f$, denominada função densidade de probabilidade de $X$ em $(\Omega, \mathcal{A}, P)$ que satisfaça*

1. $f(x) \geq 0, \quad \forall x \in \mathbb{R},$
2. $\int_{-\infty}^{\infty} f(x)dx = 1.$

Para obter a probabilidade em um certo intervalo $[a, b]$ fazemos

$$P(a \leq X \leq b) = \int_a^b f(x)dx.$$

**Definição 3.2.5.** *Seja $X$ uma variável aleatória em $(\Omega, \mathcal{A}, P)$. Sua função de distribuição acumulada, ou simplesmente, função de distribuição é $F_X(x) = P(X \leq x) = P(\{\omega \in \Omega \mid X(\omega) \leq x\})$.*

Portanto, se X for uma variável aleatória discreta,

$$F(x) = P(X \leq x) = \sum_{i \in A_x} p(x_i),$$

com $A_x = \{i \mid x_i \leq x\}$. Se X for uma variável aleatória contínua, temos que

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)dt.$$

**Definição 3.2.6.** *Seja $X$ uma variável aleatória em $(\Omega, \mathcal{A}, P)$ e $\mathcal{B}$ a $\sigma-$álgebra de Borel. Então a probabilidade $P_X$, definida em $\mathcal{B}$ por $P_X(B) = P(X \in B) = P(\{\omega \in \Omega \mid X(\omega) \in B\})$ é chamada de distribuição de $X$.*

Note que a distribuição de uma variável aleatória X pode ser determinada por qualquer das seguintes funções.

(1) A função de distribuição $F_X$,

(2) a densidade $f(x)$, se X é contínua,

(3) a função de probabilidade $p(x)$ no caso discreto.

## 3.3 Valor esperado

**Definição 3.3.1.** *Seja $X$ uma variável aleatória discreta com valores possíveis $x_1, x_2, \ldots$. Seja $p(x_i) = P(X = x_i), \forall i \geq q$. Então o valor esperado de $X$ (também chamado esperança matemática de $X$ ou média de $X$) é*

$$\mu_X = \mathbb{E}_X(X) = \sum_{i=1}^{\infty} x_i p(x_i)$$

**Teorema 3.3.1.** *(Lei da expectativa total) Sejam $X, Y$ duas variáveis aleatórias quaisquer em $(\Omega, \mathcal{A}, P)$ espaço de probabilidade. Seja $g$ uma função dos reais, então*

$$\mathbb{E}_X[g(X)] = \mathbb{E}_Y \mathbb{E}_X[g(X) \mid Y],$$

*em que $\mathbb{E}_X[g(X) \mid Y]$ é o valor esperado de $g(X)$ dado $Y$.*

# 4 Otimização - 2ª parte

## 4.1 Gradiente estocástico

Os métodos desenvolvidos anteriormente se basearam no fato de que a função a ser minimizada é diferenciável. Caso isto não aconteça, podemos utilizar o método do gradiente estocástico, que ao invés de utilizar como direção de descida o gradiente em um ponto $w$, usa-se um vetor retirado aleatoriamente de uma distribuição, em que o valor esperado deste vetor seja um elemento do subdiferencial da função no ponto $w$, ou seja, o valor esperado é subgradiente da função em $w$. Nesta seção explica-se o subgradiente e como usá-lo no cálculo estocástico.

Como a seguir estaremos trabalhando apenas com funções diferenciáveis, assumimos sem provar que o subgradiente de uma função diferenciável possui apenas um ponto, o gradiente da função no ponto.

### 4.1.1 Subgradiente

O método do gradiente precisa que a função a ser minimizada seja diferenciável, mas é possível minimizar funções convexas não diferenciáveis, utilizando-se o subgradiente de $f(x)$ em $x^k$, ao invés do gradiente. Para motivar a definição do subgradiente, para uma função convexa $f$ observa-se que, o gradiente em $x$ define a inclinação da reta tangente que se encontra abaixo de $f$, ou seja

$$f(u) \geq f(x) + \langle u - x, \nabla f(x) \rangle, \quad \forall u$$

A existência da reta tangente é uma propriedade importante de funções convexas, e pode ser caracterizada pelo seguinte lema.

**Lema 4.1.1.** *Seja $S$ um conjunto convexo aberto. Uma função $f : S \to \mathbb{R}$ é convexa se, e somente se, para todo $x \in S$, existe um $v$ tal que*

$$f(u) \geq f(x) + \langle u - x, v \rangle, \quad \forall u \in S \tag{4.1}$$

A demonstração se encontra no livro (BORWEIN; LEWIS, 2006), capítulo 3. Isto nos motiva a dar a definição de subgradiente de uma função $f$.

**Definição 4.1.1.** *Um vetor $v$ que satisfaz a equação (4.1) é chamado de* subgradiente *de $f$ em $x$. O conjunto de subgradientes de $f$ em $x$ é chamado de* conjunto diferencial *e é denotado por $\partial f(x)$.*

Abaixo seguem duas proposições que nos dão um método de construção para os subgradientes de uma função convexa.

**Proposição 4.1.2.** *Se $f$ é uma função diferenciável em $x$ então $\partial f(x)$ contém apenas um elemento, o gradiente de $f$ no ponto $x$.*

**Exemplo 4.1.1.** *Usando a proposição 4.1.2, podemos encontrar os subgradientes da função $f(x) = |x|$. Para $x$ maior que $0$ e menor que $0$ tem-se que $\partial f(x) = 1$ e $\partial f(x) = -1$, pois $f$ é diferenciável nestas restrições. O único problema é quando $x = 0$, mas tem-se que $\partial f(0) = [-1, 1]$.*

Uma outra maneira de se encontrar os subgradientes é dado pela proposição 4.1.3

**Proposição 4.1.3.** *Seja $g(w) = \max_{i\in[r]} g_i(w)$ para $r$ funções convexas diferenciáveis $g_1, ..., g_r$. Dado algum $w$, seja $j \in \arg\max_i g_i(w)$. Então $\nabla g_j(w) \in \partial g(w)$.*

**Exemplo 4.1.2.** (Hinge loss) *Uma função importante para a aprendizagem de máquinas é* hinge loss, *$f(x) = \max\{0, 1 - y\langle x, w\rangle\}$ para algum vetor $w$ e escalar $y$. Utilizando a proposição acima, temos que um subgradiente de $f$ em um ponto $x$ é dado por*

$$\begin{cases} -y\boldsymbol{w}, \text{ se } 1 - y\langle x, w\rangle > 0, \\ \quad 0, \text{ se } 1 - y\langle x, w\rangle \leq 0. \end{cases}$$

Outra propriedade interessante de subgradientes é a caracterização de funções $L$-Lipschitz com subgradientes.

**Lema 4.1.4.** *Seja $A$ um conjunto aberto convexo e seja $f : A \to \mathbb{R}$ uma função convexa. Então, $f$ é $L$-Lipschitz sobre $A$ se, e somente se, para todo $x \in A$ e $v \in \partial f(x)$ tem-se que $\|v\| \leq L$.*

*Demonstração.* Seja $v \in \partial f(x)$, $\|v\| \leq L$. Como $v \in \partial f(x)$

$$f(x) - f(u) \leq \langle v, x - u\rangle, \quad \forall u \in A.$$

Utilizando a desigualdade de Cauchy Scharwz em $\langle v, x - u\rangle$ e $\|v\| \leq L$, temos

$$f(x) - f(u) \leq L \|x - u\|.$$

De maneira análoga, obtemos $f(u) - f(x) \leq L\|u - x\|$. Portanto, $f$ é $L$-Lipschitz. Suponha agora o contrário, que $f$ é $L$-Lipschitz. Portanto, seja $x \in A$ e $v \in \partial f(x)$. Como $A$ é aberto, existe $\epsilon > 0$ tal que $u = x + \epsilon v/\|v\| \in A$. Logo,

$$\langle u - x, v\rangle = \epsilon \|v\|,$$
$$\|u - x\| = \epsilon.$$

Como $v$ é um subgradiente, temos que

$$f(u) - f(x) \geq \langle u - x, v \rangle.$$

Do fato que $f$ é L-Lipschitz,

$$f(u) - f(x) \leq L \|u - x\| = L\epsilon.$$

Portanto, $\|v\| \leq L$. □

### 4.1.2 O método do gradiente estocástico

No método do gradiente estocástico, toma-se como direção de descida um vetor aleatório cujo valor esperado em cada iteração vai ser uma direção do gradiente. Em outras palavras, ao invés de se tomar o gradiente do ponto, toma-se $\mathbb{E}[d^k \mid x^k] \in \partial f(x^t)$. Assim como no método do gradiente, há vários métodos para se calcular o tamanho do passo. Mas no algoritmo abaixo considera-se um passo fixo.

**Algoritmo 4** Método do Gradiente Estocástico

1: Dado $x^0 = 0$
2: Parâmetros: $T \in \mathbb{N}$ e $\eta > 0$
3: **para** $k = 1, 2, ..., T$ **faça**
4: Tome um vetor $d^k$ de uma distribuição tal que $\mathbb{E}[d^k \mid x^k] \in \partial f(x^k)$
5: Faça $x^{k+1} = x^k - \eta d^k$;
6: **fim para**
7: **retorne** $\bar{x} = \frac{1}{T} \sum_{k=1}^{T} x^k$

Ou seja, se $f$ for uma função diferenciável, temos que $\mathbb{E}[d^k \mid x^k] = \nabla f(x^k)$ para todo vertor $d^k$ que satisfaz a condição de descida do gradiente estocástico.

Dado o algoritmo acima, é possível encontrar uma limitação para o valor esperado retornado pelo algoritmo do gradiente estocástico. O Teorema 4.1.5 garante tal limitação.

**Teorema 4.1.5.** *Sejam $B, L > 0$. Seja $f$ uma função convexa e $x^* \in \arg\min_{\|x\| \leq B} f(x)$. Assuma que o método do gradiente estocástico rode $T$ iterações com $\eta = \sqrt{\frac{B^2}{L^2 T}}$, e que para todo $k$, $\|d^k\| \leq L$ com probabilidade 1. Então,*

$$\mathbb{E}[f(\bar{x})] - f(x^*) \leq \frac{BL}{\sqrt{T}}.$$

*Demonstração.* Como $f$ é uma função convexa, podemos aplicar a desigualdade de Jensen para $f(\bar{x})$ e tomar o valor esperado em $f(x^k) - f(x^*)$, de forma a obter

$$\mathbb{E}_{d_{1:T}}[f(\bar{x}) - f(x^*)] \leq \mathbb{E}_{d_{1:T}}\left[\frac{1}{T} \sum_{k=1}^{T} (f(x^k) - f(x^*))\right],$$

onde $d_{1:T}$ é uma sequência de vetores $d^1, ..., d^T$. Como o Lema 2.2.2 se aplica para qualquer sequência de vetores, se aplica para $d^1, ..., d^T$. Portanto, tomando o valor esperado da limitação na Equação (2.11), do lema anterior, temos que

$$\mathbb{E}_{d_{1:T}}\left[\frac{1}{T}\sum_{k=0}^{T}\langle x^k - x^*, d^k\rangle\right] \leq \frac{BL}{\sqrt{T}}$$

Apenas resta mostrar que

$$\mathbb{E}_{d_{1:T}}\left[\frac{1}{T}\sum_{k=1}^{T}(f(x^k) - f(x^*))\right] \leq \mathbb{E}_{d_{1:T}}\left[\frac{1}{T}\sum_{k=1}^{T}\langle x^k - x^*, d^k\rangle\right]. \tag{4.2}$$

A *lei da expectativa total* diz que para quaisquer duas variáveis aleatórias $a,b$ e uma função $g$, temos que $\mathbb{E}_a[g(a)] = \mathbb{E}_b\mathbb{E}_a[g(a)|b]$. Colocando $a = d_{1:k}$ e $b = d_{1:k-1}$

$$\begin{aligned}\mathbb{E}_{d_{1:T}}[\langle x^k - x^*, d^k\rangle] &= \mathbb{E}_{d_{1:k}}[\langle x^k - x^*, d^k\rangle] \\ &= \mathbb{E}_{d_{1:k-1}}\mathbb{E}_{d_{1:k}}[\langle x^k - x^*, d^k\rangle|d_{1:k-1}]\end{aligned}$$

Uma vez que sabemos $d^1, ..., d^{k-1}$, $x^k$ não é mais uma variável aleatória, portanto

$$\mathbb{E}_{d_{1:k-1}}\mathbb{E}_{d_{1:k}}[\langle x^k - x^*, d^k\rangle|d_{1:k-1}] = \mathbb{E}_{d_{1:k-1}}[\langle x^k - x^*, \mathbb{E}_{d^k}[d^k|d_{1:k-1}]\rangle]$$

Como $x^k$ depende apenas de $d_{1:k-1}$, pois $x^0 = 0$, e o método do gradiente requer que $\mathbb{E}_{d^k}[d^k|x^k] \in \partial f(x^k)$, obtemos que

$$\mathbb{E}_{d^k}[d^k|d_{1:k-1}] \in \partial f(x^k).$$

Portanto, pela definição de subgradiente,

$$\mathbb{E}_{d_{1:k-1}}[\langle x^k - x^*, \mathbb{E}_{d^k}[d^k|d_{1:k-1}]\rangle] \geq \mathbb{E}_{d^{k-1}}[f(x^k) - f(x^*)]$$

O que nos dá

$$\mathbb{E}_{d_{1:T}}[\langle x^k - x^*, d^k\rangle] \geq \mathbb{E}_{d_{1:k-1}}[f(x^k) - f(x^*)] = \mathbb{E}_{d_{1:T}}[f(x^k) - f(x^*)]$$

A última igualdade segue do fato que se $x^k$ depende apenas de $d_{1:k-1}$, então $x^k$ depende de $d_{1:T}$. Somando ambos os lados em relação a $k$ e dividindo por $T$ encontramos que vale a Equação (4.2). □

# 5 Aprendizagem de máquinas

Neste capítulo foram usadas as referências (FAN, 2014), (MITCHELL, 1997), (NG, 2012), (ZHANG, 2004),(SHALEV-SHWARTZ; BEN-DAVID, 2014).

## 5.1 Aprendizado supervisionado

A aprendizagem de máquinas, mais em específico o aprendizado supervisionado, é uma subárea da inteligência artificial que desenvolve algoritmos para que o computador possa aprender. Um exemplo é como um algoritmo que prevê se uma pessoa possui um tumor ou não dado um conjunto de informações, onde este conjunto de informações é rotulado. De modo geral, um dos problemas dessa área é como prever qual será a função e também qual é o erro estimado. Seja $(x^i, y^i)$ conjunto de $m$ exemplos, em que $x^i$ são os atributos e $y^i$ são a classe. O objetivo é, dado um conjunto de exemplos, desenvolver um algoritmo de modo que seja possível prever os atributos com apenas os dados já existentes do conjunto de exemplos. Por exemplo, se $x_i, y_i \in \mathbb{R}$ e supormos um modelo linear,

$$h(x) = \alpha_1 + \alpha_2 x, \tag{5.1}$$

ou $x^i \in \mathbb{R}^2$, $y^i \in \mathbb{R}$, e usamos um modelo quadrático

$$h(x) = \alpha_1 + \alpha_2 x_1 + \alpha_3 x_2 + \alpha_4 x_1 x_2 + \alpha_5 x_1^2 + \alpha_6 x_2^2. \tag{5.2}$$

Em ambos os casos, a dependência de $\alpha$ é linear, mas isso não é necessário. Podemos ter um modelo mais complexo, como

$$h(x) = \alpha_1 e^{\alpha_2 x_2} + \alpha_3 x_1 + \alpha_4.$$

Em todo caso, queremos encontrar os parâmetros $\alpha_1, \ldots, \alpha_n$ tal que $h(x^i)$ e $y^i$ sejam os mais próximos possível para o caso de regressão e para classificação de modo que o sinal de $h(x^i)$ seja compatível com o sinal de $y^i$. Uma maneira de quantificar isso é definir a função

$$f(\alpha) = \sum_{i=1}^{m} [h(x^i) - y^i]^2,$$

e procurar o minimizador de $f$, que corresponde ao problema de mínimos quadrados já descrito.

Mas note que a formulação acima é apenas um jeito para descrever o problema. Nesta seção vamos descrever outros métodos de modo que $h(x^i)$ e $y^i$ sejam os mais próximos possível.

### 5.1.1 Requisitos para o aprendizado

Quando se utiliza algum algoritmo para ajustar um modelo, são utilizados os seguintes tópicos.

1. **Um domínio:** Um conjunto $\mathbb{X}$. Este é o conjunto dos dados que queremos classificar. Por exemplo, se tivermos informações sobre o número de quartos e banheiros de uma casa e um valor para a casa, podemos definir o conjunto $\mathbb{X}$ como o conjunto que nos diz a quantidade de quartos e banheiros por casa.

2. **Conjunto dos rótulos:** Um conjunto em que cada elemento está relacionado com um valor do domínio. No caso anterior da casa, temos que o conjunto dos rótulos é formado pelo preço das casas. Denota-se tal conjunto por $\mathbb{Y}$.

3. **Dados de treinamento:** Seja $S \coloneqq \{(x_1, y_1), \ldots, (x_m, y_m)\}$ um conjunto finito de pares em $\mathbb{X} \times \mathbb{Y}$. Tais pares são chamados normalmente de exemplos de treinamento. O conjunto $S$ pode ser também chamado de conjunto de treinamento.

4. **Um modelo:** Como anteriormente descrito, é necessário uma função $h : \mathbb{X} \to \mathbb{Y}$. O modelo dessa forma é utilizado para problemas de classificação, onde o conjunto $\mathbb{Y}$ é discreto.

5. **Um simples modelo de generalização dos dados**: Assume-se uma função $f : \mathbb{X} \to \mathbb{Y}$ tal que ela classifica corretamente os dados, ou seja, $f(x_i) = y_i$. Assume-se também uma que os dados são gerados por uma distribuição de probabilidade $\mathbb{D}$. Não se conhece a expressão de tal função.

6. **Uma medida de sucesso:** Definimos o erro de um modelo como a probabilidade de ele não prever corretamente um rótulo em um ponto aleatoriamente gerado pela distribuição como acima mencionada. Definimos o *erro do classificador* $h$ por

   $$L_{\mathbb{D},f}(h) \coloneqq \mathbb{P}_{x\sim\mathbb{D}}[h(x) \neq f(x)].$$

   Onde $\mathbb{P}$ é uma função probabilidade. L acima definida pode ser chamada de *generalização do erro*, *risco*, ou *risco verdadeiro de* $h$.

7. O observador não consegue saber a distribuição e a função $f$ acima descrita *a priori*. Abaixo será descrito como proceder com o treinamento.

### 5.1.2 Minimização do risco empírico

Como dito previamente, não sabemos a distribuição de probabilidade de um conjunto, muito menos a função $f$ que classifica corretamente todos os dados. Para tanto, considera-se o *erro de treinamento* dado abaixo

$$L_S(h) \coloneqq \frac{|\{i \in \{1, \ldots, m\} \mid \ h(x_i) \neq y_i\}|}{m}$$

Onde $h$ é um modelo, $(x^{(i)}, y^{(i)})$ os dados e $m$ é a quantidade de dados. $L_S$ também pode ser chamada de *erro empírico* ou *risco empírico*.

Para problemas de regressão podemos considerar o *erro verdadeiro* dado a seguir

$$L_{\mathbb{D}}(h) := \mathop{\mathbb{E}}_{(x,y)\sim\mathbb{D}}[(h(x) - y)^2].$$

Pode-se generalizar a ideia da medida do sucesso de um modelo como dado abaixo. Denote por $\mathbb{H}$ o conjunto de todos os modelos. Dado $\mathbb{H}$ e algum domínio $Z$, seja $\ell$ qualquer função de $\mathbb{H} \times Z$ para os reais não negativos, $\ell : \mathbb{H} \times Z \to \mathbb{R}_+$. Tal função $\ell$ é chamada de função de perda.

Agora podemos definir a *função de risco* como a expectativa da função de perda de um classificador, $h \in \mathbb{H}$, com respeito a uma distribuição de probabilidade $\mathbb{D}$ sobre $Z$,

$$L_{\mathbb{D}}(h) := \mathop{\mathbb{E}}_{z\sim\mathbb{D}}[\ell(h, z)].$$

De maneira similar, para o conjunto $S = \{z_1, \ldots, z_m\} \in Z^m$ antes mencionado, temos o *risco empírico*,

$$L_S(h) := \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i).$$

Alguns exemplos de funções perda são dadas abaixo

1. **Perda quadrada:** Usada para problemas de regressão, é dada por

   $$\ell(h, (x, y)) = (h(x) - y)^2$$

   onde $(x, y) \in \mathbb{X} \times \mathbb{Y}$.

2. **Perda 0-1:** Sejam $(x, y) \in \mathbb{X} \times \mathbb{Y}$. A função perda 0-1 é dada por

   $$\ell_{0-1}(h, (x, y)) := \begin{cases} 0, \text{ se } h(x) = y \\ 1, \text{ se } h(x) \neq y \end{cases}$$

   utilizada em problemas de classificação.

Em situações práticas do aprendizado supervisionado, procura-se minimizar o risco empírico. Geralmente escolhe-se um modelo $h$ que depende de alguns parâmetros. Exemplos da função $h$ são dados nas Equações (5.1) e (5.2). Note que o primeiro exemplo é linear, sendo o segundo quadrático. Neste trabalho utilizaremos um modelo linear $h_\theta(x) = \theta^T x$ por facilitar as contas. Portanto, temos o seguinte problema de otimização, onde $z_i \in \mathbb{R}^{n+1}$

$$\min_{\theta\in\mathbb{R}^{n+1}} L_S(\theta) = \min_{h_\theta} L_S(h_\theta)$$

Como estamos tomando $h_\theta = \theta^T x$, podemos escrever $\ell(h_\theta, (x, y)) = \ell(\theta, (x, y))$, indicando que a função perda depende de $\theta$.

Durante o aprendizado, podem ocorrer dois fenômenos, o *underfitting* e o *overfitting*. No *underfitting* os seus dados não se ajustam muito bem ao modelo, levando a um valor alto para função risco. Uma maneira para solucionar este problema é adicionar mais parâmetros, tornar o modelo mais complexo.

Já no caso do *overfitting*, o modelo é muito bom para os dados de treinamento, levando a um valor baixo da função risco. Porém quando testado em dados ainda não vistos, apresenta-se falho. Um dos motivos para que isto aconteça é que o modelo é muito complexo, e ele esta se ajustando muito bem aos dados de treinamento. Uma maneira de resolver este problema é adicionar um termo de regularização no problema de minimização da função risco em função de $\theta$, assim tendo o seguinte problema.

$$\min_{\theta \in \mathbb{R}^{n+1}} \frac{1}{m} \sum_{i=1}^{m} \ell(\theta, z_i) + \frac{\lambda}{2} \|\theta\|_2^2$$

Nas próximas seções vamos descrever como obter tal parâmetro $\theta$, tanto para problemas de regressão como de classificação.

## 5.2 Os métodos do gradiente, gradiente conjugado e gradiente estocástico

### 5.2.1 Regressão

Dado um conjunto de exemplos de treinamento $S = \{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, queremos determinar o valor de

$$L_S(\theta) := \frac{1}{m} \sum_{i=1}^{m} \ell(\theta, (x^{(i)}, y^{(i)}))$$

Como se trata de um problema de regressão linear, iremos utilizar a seguinte função perda $\ell$

$$\ell(\theta, (x^{(i)}, y^{(i)})) = (h(x^{(i)}) - y^{(i)})^2.$$

Tal função é a mesma deduzida no problema de quadrados mínimos, visto na seção 2.1. Vamos considerar o modelo $h$ como

$$h_\theta(x) = \theta^T x,$$

onde $x_0 = 1$. Na aprendizagem de máquinas, é comum adicionar uma coordenada a mais em cada um dos dados $x^{(i)}$, dando a reta mais um grau de liberdade, ou seja, $\theta \in \mathbb{R}^{n+1}$. O termo $\theta_0$ geralmente é chamado de viés, ou interceptação da reta.

Vamos definir agora a função custo, $J(\theta)$ como

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\ell(\theta,(x^{(i)},y^{(i)})) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2.$$

Com a função definida, é possível utilizar os métodos descritos no capítulo anterior. Note que a derivada parcial de $J$ é dada por

$$\begin{aligned}\frac{\partial J}{\partial\theta_j}(\theta) &= \frac{1}{2m}\frac{\partial}{\partial\theta_j}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\right]\\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\frac{\partial h}{\partial\theta_j}(h_\theta(x^{(i)}))\\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}.\end{aligned} \tag{5.3}$$

Em que $x_j^{(i)}$ é a $j$-ésima coordenada do $i$-ésimo exemplo. Logo, o gradiente de $J$ é

$$\nabla J(\theta) = \frac{1}{m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_0^{(i)},\ldots,\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_n^{(i)}\right)$$

onde $x$ é um vetor. Do mesmo modo que colocamos a função perda em função de $\theta$, podemos fazer isso para a função risco $L$. Portanto, utilizando os métodos do capítulo anterior, podemos determinar $\theta$. Note que para minimizar

$$L_{\mathbb{D}}(\theta) \coloneqq \mathop{\mathbb{E}}_{z\sim\mathbb{D}}[\ell(\theta,z)] \tag{5.4}$$

utilizando os métodos do gradiente e gradiente conjugado, minimizamos a função

$$L_S(\theta) \coloneqq \frac{1}{m}\sum_{i=1}^{m}\ell(\theta,z_i).$$

Já para o método do gradiente estocástico, podemos minimizar (5.4) diretamente. Como não sabemos $\mathbb{D}$, não conseguimos calcular $\nabla L_{\mathbb{D}}(\theta)$ e minimizar com o método do gradiente. Mas com o método do gradiente estocástico nós precisamos apenas achar uma estimativa não tendenciosa do gradiente de $L_{\mathbb{D}}(\theta)$, ou seja, um vetor aleatório cuja esperança condicional é $\nabla L_{\mathbb{D}}(\theta)$.

Para uma função perda diferenciável, temos que $L_{\mathbb{D}}$ é diferenciável. Vamos construir o vetor $d^k$ definido no capítulo anterior da seguinte maneira: tome um exemplo $z\sim\mathbb{D}$, que significa $z=(x,y)$, onde $x$ é um dado de treinamento e $y$ o seu rótulo. Defina $d^k$ como o gradiente da função perda $\ell(\theta,(x,y))$ em relação a $\theta$ aplicado no ponto $\theta^k$. Portanto, pela linearidade do gradiente

$$\mathbb{E}[d^k \mid \theta^k] = \mathop{\mathbb{E}}_{z\sim\mathbb{D}}[\nabla\ell(\theta^k,(x,y))] = \nabla\mathop{\mathbb{E}}_{z\sim\mathbb{D}}[\ell(\theta^k,(x,y))] = \nabla L_{\mathbb{D}}(\theta^k)$$

Assim, o gradiente da função perda $\ell(\theta,(x,y))$ em $(\theta^k,(x,y))$ é um bom estimador sem tendência do gradiente da função risco $L_{\mathbb{D}}(\theta^k)$ e é facilmente construido tirando um novo dado $z\sim\mathbb{D}$ a cada iteração $k$.

O mesmo argumento vale para funções de perda não diferenciáveis. De fato, basta tomar $d^k$ como um subgradiente da função $\ell(\theta,(x,y))$ em $\theta^k$. Então para todo $u$, temos que

$$\ell(u,(x,y)) - \ell(\theta^k,z) \geq \langle u-\theta^k, d^k\rangle$$

Tomando a esperança de ambos os lados com respeito a $z \sim \mathbb{D}$ e condicionado ao vetor $x^k$, temos que

$$\begin{aligned} L_{\mathbb{D}}(u) - L_{\mathbb{D}}(\theta^k) &= \mathbb{E}\left[\ell(u,(x,y)) - \ell(\theta^k,(x,y))|\theta^k\right] \\ &\geq \mathbb{E}\left[\langle y-\theta^k, d^k\rangle \mid \ \theta^k\right] \\ &= \langle u-\theta^k, \mathbb{E}\left[d^k \mid \ \theta^k\right]\rangle. \end{aligned}$$

Segue então que $\mathbb{E}[d^k|\theta^k]$ é um subgradiente de $L_{\mathbb{D}}(\theta)$ em $\theta^k$. Portanto, o método do gradiente estocástico para minimizar a função risco pode ser resumido no algoritmo 5.

**Algoritmo 5** Método do Gradiente Estocástico para minimizar $L_{\mathbb{D}}(w)$

1: Dado $x^0 = 0$
2: Parâmetros: $T \in \mathbb{N}$ e $\eta > 0$
3: **para** $k = 1,2,...,T$ **faça**
4: Tome $z \sim \mathbb{D}$;
5: Tome $d^k \in \partial\ell(x^k,z)$;
6: Faça $x^{k+1} = x^k - \eta d^k$;
7: **fim para**
8: **retorne** $\bar{x} = \frac{1}{T}\sum_{k=1}^{T} x^k$

Vamos criar um conjunto de dados para utilizarmos os métodos. A implementação é feita utilizando a linguagem *Julia*. Gerou-se um conjunto com 10000 exemplos. Para medir o erro, utilizou-se o erro quadrático médio. Usamos 9000 dados para o treinamento e 1000 para teste.

Na tabela 3 encontra-se uma comparação entre os métodos para o problema de regressão linear. O tempo foi calculado realizando 10000 iterações. O gradiente estocástico

Tabela 3: Comparação entre os métodos apresentados.

| **Método** | **Erro (em %)** | **Tempo (em segundos)** |
|---|---|---|
| Gradiente | $6 \cdot 10e-5$ | 0,01 |
| Gradiente conjugado | $9 \cdot 10e-5$ | 5,68 |
| Gradiente estocástico | $6 \cdot 10e-5$ | 0,47 |

e método do gradiente conjugado são executados de forma mais rápida, já que o primeiro leva em consideração apenas um exemplo por iteração, enquanto o outro realiza menos iterações. Utilizou-se um passo fixo para os métodos do gradiente e gradiente estocástico de $\eta = 0,01$.

### 5.2.2 Classificação

Neste momento utilizaremos o banco de dados *Iris*. Íris é um tipo de flor, e que será classificada de acordo com 4 características. Além disso, o banco de dados possui três classes diferentes sendo que cada uma representa uma espécie de Íris, ver Figura 3.

Figura 3: Íris

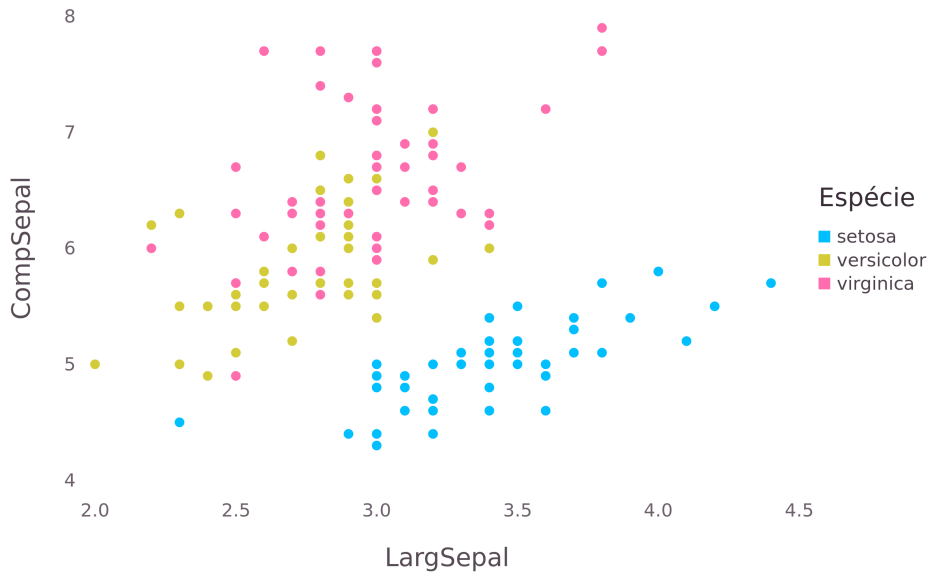


Figura 4: Distribuição dos exemplos do banco de dados Íris em relação à largura e ao comprimento da sépala. Neste caso os exemplos possuem 4 atributos diferentes e se divide em 3 classes, que são *setosa*, *versicolor* e *virginica*.

Matematicamente falando, gostaríamos de encontrar um hiperplano de forma que para dado $x$, um atributo, se $\theta^T x \geq 0$, então $x$ pertence a classe 1, caso contrário $x$ pertence a classe 0. Para facilitar as contas e garantir propriedades importantes da função custo, vamos utilizar a função sigmoide (5.5)

$$g(z) = \frac{1}{1 + e^{-z}}. \tag{5.5}$$

Note que para z tendendo ao infinito, a função sigmóide se aproxima de 1. Já para z tendendo a menos infinito, a função sigmóide se aproxima de 0, como nos mostra a Figura

5. Tal função é uma escolha conveniente para o nosso modelo. Considere $m_\theta(x) = g(\theta^T x)$. Se $m_\theta(x) \geq 1/2$ vamos dizer que $x$ pertence à classe positiva, 1, caso contrário à classe negativa, 0.

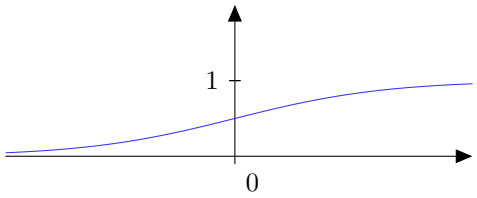


Figura 5: Gráfico da função sigmoide.

Portanto, gostaríamos de ter uma função custo que se adeque ao modelo. Seja $S = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ um conjunto de pares ordenados onde cada $x^{(i)}$ é um exemplo e $y^{(i)}$ sua classe. Simplesmente não podemos utilizar a função custo anterior apenas substituindo $h_\theta(x) = \theta^T x$ pela função $m_\theta(x)$, pois assim perdemos a convexidade da função custo. Para tanto considere os seguintes casos.

$$\ell(\theta, (x, y)) = \begin{cases} -log(m_\theta(x)), \text{ se } y = 1 \\ -log(1 - m_\theta(x)), \text{ se } y = 0 \end{cases} \tag{5.6}$$

Observe que existem duas possibilidades para a função $\ell$ acima definida, representadas na Figura 6. Considerando a função custo acima dada, podemos determinar o

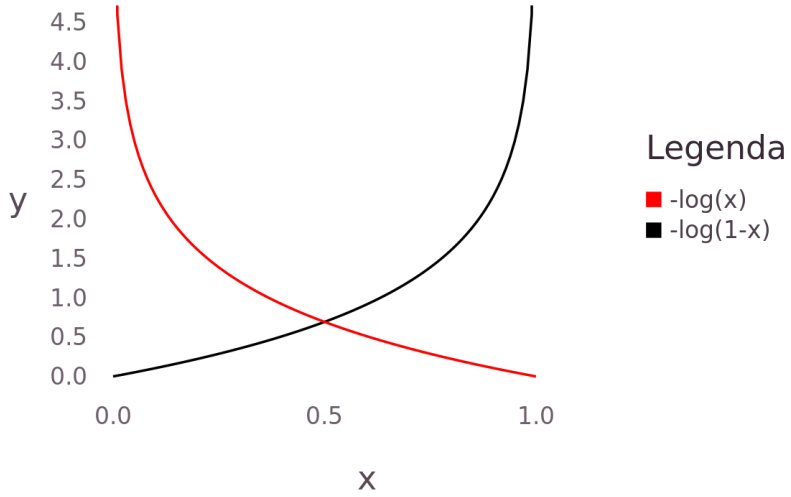


Figura 6: Gráfico das funções $-log(x)$ em vermelho e $-log(1-x)$ em preto.

parâmetro $\theta$ minimizando a função custo

$$J(\theta) = \sum_{i=1}^{m} \ell(\theta, (x^{(i)}, y^{(i)})). \tag{5.7}$$

A função acima é convexa e diferenciável, portanto qualquer ponto crítico é um minimizador global da função. Para facilitar as contas do gradiente da função, considere

$$\ell(\theta,(x,y)) = -ylog(m_\theta(x)) - (1-y)log(1-m_\theta(x)). \tag{5.8}$$

Observe que para $y = 1$ recuperamos $-log(m_\theta(x))$, e para $y = 0$ temos $log(1 - m_\theta(x))$. Portanto, juntando as Equações (5.7) e (5.8), temos a seguinte função de custo

$$J(\theta) = \sum_{i=1}^{m} -y_i log(m_\theta(x_i)) - (1-y_i)log(1-m_\theta(x_i)).$$

Agora podemos calcular o gradiente da função custo acima definida para enfim aplicarmos os métodos já desenvolvidos. Note que

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}e^{-z} \\ &= \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

Para apenas um dado $(x, y)$, temos que

$$\begin{aligned} \frac{\partial J}{\partial \theta_j}(\theta) &= -\left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial g}{\partial \theta_j}(\theta^T x) \\ &= -\left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial \theta_j}\theta^T x \\ &= -\left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x))x^j \\ &= -\left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x^j \\ &= (m_\theta(x) - y)x_j, \end{aligned}$$

onde $x_j$ é a $j$-ésima coordenada de $x$. Portanto, para o conjunto $S$ de dados, temos que

$$\frac{\partial J}{\partial \theta_j}(\theta) = \sum_{i=1}^{m}(m_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}.$$

Portanto, utilizando os métodos desenvolvidos anteriormente, podemos classificar os tipos de Íris. Na Tabela 4 temos uma comparação entre os 3 métodos apresentados anteriormente para o problema de classificação. Para todos os métodos utilizou-se passo constante de $0,03$.

Note que o método do gradiente estocástico é um método mais rápido por iteração que os outros dois, pois no método do gradiente conjugado foram realizados apenas 10 iterações, enquanto que para o método do gradiente estocástico realizou 1050 iterações.

Tabela 4: Comparação entre os métodos para o problema de classificação.

| **Método** | **Precisão (em %)** | **Tempo (em segundos)** |
|---|---|---|
| Gradiente | 96 | 0,12 |
| Gradiente conjugado | 94 | 0,04 |
| Gradiente estocástico | 96 | 0,07 |

## 5.3 Redes neurais

Uma rede neural artificial é um modelo de computação inspirado nas redes neurais do nosso cérebro. Modelos simplificados do cérebro humano consistem em um grande número de componentes (neurônios) que estão conectados entre si através de uma rede complexa de comunicação, em que o cérebro consegue executar tarefas complexas. A rede neural artificial é uma construção computacional formal que é modelada tendo como inspiração o cérebro humano.

Pode-se descrever uma rede neural como um grafo direto cujos nós (E) correspondem a neurônios e as arestas (V) correspondem as relações entre eles. Em uma rede neural, cada neurônio recebe uma soma de valores provenientes dos neurônios na camada anterior a este, como na Figura 7. Este processo acontece através de uma função peso, que associa pesos entre cada conexão entre os neurônios, ou seja, cada seta na Figura 7 correspondem a um valor, ou peso.

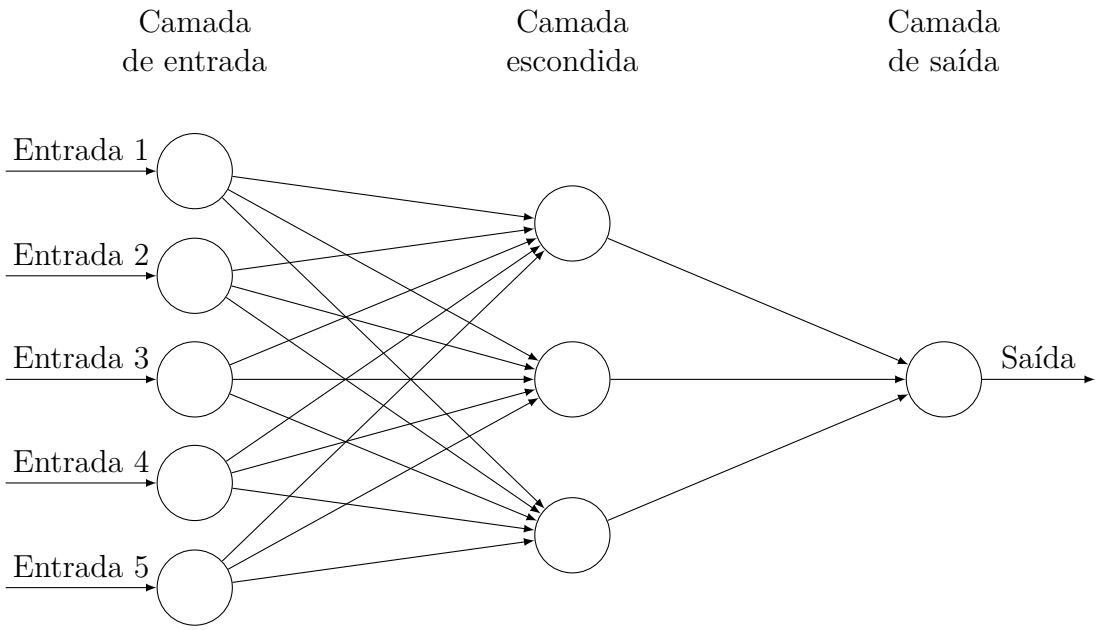


Figura 7: Esquema de uma rede neural artificial.

Um dos motivos para se usar redes neurais é que podemos aproximar qualquer

função $f$ de um conjunto compacto do $\mathbb{R}^n$ para a reta real utilizando uma rede neural com uma camada de entrada, uma camada escondida e uma camada de saída.

**Teorema 5.3.1.** *(CYBENKO, 1989) (Teorema da aproximação universal) Seja $g(.)$ uma função não constante, limitada, monotonicamente crescente e contínua. Seja $U \in \mathbb{R}^m$ um conjunto compacto qualquer. O espaço das funções contínuas em $V$ é denotado por $C(V)$. Então, dada qualquer função em $f \in C(V)$ e $\epsilon > 0$, existe um inteiro $N$, constantes reais $v_i, b_i \in \mathbb{R}$ e vetores reais $w_i \in \mathbb{R}^m$, com $i = 1, \ldots, N$ tais que podemos definir*

$$F(x) = \sum_{i=1}^{N} v_i g(w_i^T x + b_i)$$

*como uma aproximação da função $f$, onde $f$ é independente de $g$, ou seja,*

$$|F(x) - f(x)| < \epsilon,$$

*para todo $x \in V$. Em outras palavras, funções da forma $F(x)$ são densas em $C(V)$.*

Seja $E$ o conjunto que representa os neurônios. Como $E$ é finito, podemos pensar na função peso como um vetor $w \in \mathbb{R}^{|E|}$. Suponha que a rede tem $n$ neurônios na camada de entrada e $k$ neurônios na camada de saída. Seja $h_w : \mathbb{R}^n \to \mathbb{R}^k$ a função calculada pela rede neural se a função peso é definida por $w$, em que $w : E \mapsto \mathbb{R}$.

Por exemplo, se tomarmos a função sigmóide $g$ e um conjunto de dados $(x, y)$, em que $x \in \mathbb{R}^{n-1}$ e $y \in \mathbb{R}^k$, onde $n-1$ é o número de coordenadas de $x$ e $k$ é o número de classes possíveis para o dado $x$. Então, para uma rede neural com apenas uma camada escondida, a função $h_w$ assume a forma

$$h_w(x) = g(W_2 * g(W_1 * x')),$$

onde $x'$ é o vetor x acrescido de uma coordenada $x_0 = 1$, $W_1$ é a matriz de pesos entre os nós da primeira e segunda camada, uma matriz $n \times n$. Já $W_2$ é a matriz de pesos entre os nós da segunda e a terceira camadas. Na próxima subseção vamos mostrar como calcular estas matrizes de pesos através do *backpropagation*.

Denote por $\ell(w, (x, y))$ a função perda. Vamos tomar $\ell$ como anteriormente, o erro quadrático, $\frac{1}{2}\|h_w(x) - y\|^2$, sendo $h_w$ a função que depende do peso $w$. Dada uma distribuição $\mathbb{D}$ sobre o domínio dos dados, $\mathbb{R}^n \times \mathbb{R}^k$, seja $L_{\mathbb{D}}(w)$ o risco da rede neural, ou seja

$$L_{\mathbb{D}}(w) = \mathop{\mathbb{E}}_{(x,y)\sim\mathbb{D}} \left[\ell(w, (x, y))\right].$$

Vamos utilizar o método do gradiente estocástico para a minimização da função risco. Abaixo segue o pseudocódigo do método do gradiente estocástico para o caso de

uma rede neural. Note que no código abaixo estamos usando a função sigmóide $g$, a função de ativação, que realiza as operações entre um neurônio de uma camada para um neurônio de outra camada. Portanto, perdemos as propriedades de convexidade, para tanto se faz necessário algumas mudanças.

**Algoritmo 6** Método do Gradiente Estocástico em Redes neurais

**Parâmetros**: Número de iterações $T$, sequência do tamanho de passos $\{\eta_i\}_{i=1}^{T}$
**Entrada:** Grafo $(V, E)$, função de ativação $g : \mathbb{R} \to \mathbb{R}$
**Inicie:** Tome $w_1 \in \mathbb{R}^{|E|}$ de uma distribuição tal que $w_1$ está próxima de zero.
**para** $t = 1, 2, ..., T$ **faça**
    Tome $(x, y) \sim \mathbb{D}$
    Calcule gradiente $v_i = backpropagation(x, y, w, (V, E), g)$
    Faça $w_i = w_i - \eta_i(v_i + \lambda w_i)$;
**fim para**
**retorne** $\bar{w}$, o melhor $w_i$ testado em um conjunto de validação.

Note que o gradiente é calculado através de um método chamado de *backpropagation*.

**Algoritmo 7** *Backpropagation* para redes neurais

**Entrada:** exemplo $(x, y)$, vetor peso $w$, grafo $(V, E)$, função ativação $g$.
**Inicie:** denote as camados do grafo $V_0, \ldots, V_T$, onde $V_t = \{v_{t,1}, \ldots v_{t,k}\}$. Defina $W_{t.i.j}$ como o peso de $(v_{t,j}, v_{t+1,i})$.
**forward:**
coloque $\mathbf{q}_0 = x$
**para** $t = 1, \ldots, T$ **faça**
    **para** $i = 1, \ldots, k_t$ **faça**
        $a_{t,i} = \sum_{j=1}^{k_t - 1} W_{t-1,i,j} q_{t-1,j}$
        $q_{t,i} = g(a_{t,i})$
    **fim para**
**fim para**
**fim forward**
**backward:**
coloque $\delta_T = \mathbf{q}_0 - y$
**para** $t = T - 1, \ldots, 1$ **faça**
    **para** $i = 1, \ldots, k_t$ **faça**
        $\delta_{t,i} = \sum_{j=1}^{k_t + 1} W_{t,j,i} \delta_{t+1,j} g'(a_{t+1,j})$
    **fim para**
**fim para**
**fim backward**
**Retorne:** para cada $(v_{t-1,j}, v_{t,i}) \in E$.
    coloque a derivada parcial igual a $\delta_{t,i} g'(a_{t,i}) q_{t-1.j}$

### 5.3.1 Como o *Backpropagation* calcula o gradiente

Explicamos agora como o algoritmo de *Backpropagation* calcula o gradiente da função perda no exemplo $(x, y)$ com respeito a $w$. Podemos decompor $V$ em função das suas camadas, ou seja, $V = \cup_{t=0}^{T} V_t$, onde $V_t$ é o conjunto das arestas da $t$-ésima camada. Para todo $t$, escrevemos $V_t = \{v_{t,1}, \dots v_{t,k_t}\}$, onde $k_t = |V_t|$. Além disso, para todo $t$, denotamos $W_t \in \mathbb{R}^{k_{t+1},k_t}$ a matriz dos pesos entre as camadas $V_t$ e $V_{t+1}$. Colocamos $W_{t,i,j}$ como o peso entre os neurônios ($v_{t,j}$ e $v_{t+1,i}$). Após isso, discutimos sobre como calcular as derivadas parciais com respeito as arestas de $V_{t-1}$ para $V_t$, ou seja, com respeito aos elementos de $W_{t-1}$. Como todos os outros pesos da rede neural estão fixados, os elementos de $V_{t-1}$ não dependem de $W_{t-1}$. Denote então por $\mathbf{q}_{t-1}$ o vetor dos elementos de $V_{t-1}$. Represente por $\ell_t : \mathbb{R}^{k_t} \to \mathbb{R}$ a função perda da subrede neural das camadas $V_t, \dots, V_T$ como uma função das saídas dos neurônios de $V_t$. A entrada dos neurônios em $V_t$ pode ser denotada por $a_t = W_{t-1}\mathbf{q}_{t-1}$, e a saída como $\mathbf{q}_t = g(a_t)$, onde $g$ é a função de ativação, ou função sigmóide e ela é aplicada em cada ponto do vetor $a_t$.

Portanto, a perda como função de $W_{t-1}$ pode ser escrita como

$$f_t(W_{t-1}) = \ell_t(q_t) = \ell_t(g(a_t)) = \ell_t(g(W_{t-1}q_{t-1})).$$

Considere agora o vetor $w_{t-1} \in \mathbb{R}^{k_{t-1}k_t}$ concatenando as linhas da matriz $W_{t-1}$. Portanto, podemos escrever $W_{t-1}\mathbf{q}_{t-1}$ como

$$Q_{t-1}w_{t-1},$$

em que $Q_{t-1} = diag(\mathbf{q}_{t-1}^T)$ é uma matriz $k_t \times (k_{t-1}k_t)$. Logo

$$f_t(w_{t-1}) = \ell_t(g(Q_{t-1}w_{t-1})).$$

Portanto, derivando $\ell$ com respeito a $w_{t-1}$ temos que

$$H_{w_{t-1}}(f_t) = H_{g(Q_{t-1}w_{t-1})}(\ell_t)H_{w_{t-1}}(g(Q_{t-1}w_{t-1})). \tag{5.9}$$

Observe que $H_{w_{t-1}}(g(Q_{t-1}w_{t-1})) = diag(g'(Q_{t-1}w_{t-1}))Q_{t-1}$, pois o jacobiano de $g$ em um ponto $x \in \mathbb{R}^n$ é $diag(g'(x))$, já que $g$ é aplicada em cada ponto de g, ou seja,

$$H_x(g) = \begin{bmatrix} g'(x_1) & 0 & \dots & 0 \\ 0 & g'(x_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & g'(x_n) \end{bmatrix}.$$

E como o Jacobiano da função $p(w) = Q_{t-1}w$ é $H_w(p) = Q_{t-1}$, segue a equação (5.9). Sendo $a_t = Q_{t-1}w_{t-1}$, e $q_t = g(a_t)$. Então,

$$H_{w_{t-1}}(f_t) = H_{q_t}(\ell_t)diag(g'(a_t))Q_{t-1}.$$

Denote $\delta_t = H_{q_t}(f_t)$. Portanto

$$H_{w_{t-1}}(f_t) = \left(\delta_{t1} g'(a_{t,1}) q_{t-1}^T, \ldots, \delta_{tk_t} g'(a_{tk}) q_{t-1}^T\right). \tag{5.10}$$

Falta apenas calcular $\delta_t$ para todo $t$, pois este é o gradiente de $f_t$ no ponto $q_t$. Vamos calcular isto recursivamente. Note que para última camada, temos que $\ell_T(u) = \ell(u, y)$. Previamente, assumimos que $\ell(u, y) = \frac{1}{2}\|u - y\|^2$, temos que $\delta_T = H_{q_T} = (q_T - y)$. Observe que

$$\ell_t(u) = \ell_{t+1}(g(W_t u)).$$

Portanto, pela regra da Cadeia, temos que

$$H_u(\ell_t) = H_{g(W_t u)}(\ell_{t+1}) diag(g'(W_t u)) W_t.$$

Em particular

$$\begin{aligned}\delta_t = H_{q_t}(\ell_t) &= H_{q_{t+1}}(\ell_{t+1}) diag(g'(W_t u)) W_t \\ &= \delta_{t+1} diag(g'(a_t)) W_t.\end{aligned}$$

Resumindo, primeiro calculamos os vetores $\{a_t, q_t\}$ desde a primeira camada até a última. Então, utilizando o *Backpropagation*, calculamos cada $\delta_t$, $t = 1, \ldots, T$, obtendo assim as derivadas parciais desejadas com a Equação (5.10). Assim, mostramos que o Algoritmo 7 realmente calcula o gradiente (SHALEV-SHWARTZ; BEN-DAVID, 2014).

### 5.3.2 Classificação - Íris

Uma aplicação para as redes neurais é a classificação da íris em suas espécies. Utilizamos uma rede neural artificial com 3 camadas, sendo uma de entrada, uma escondida e a de saída. Na primeira camada há 4 neurônios, já que há apenas 4 parâmetros como descrito na seção anterior. Com 5 neurônios é constituída a segunda camada, e na última há 3 neurônios, que representam cada classe, ou seja, cada espécie da flor. Utilizamos a função $\ell(w, (x, y)) = \frac{1}{2}\|(\| h_w(x) - y)^2$, onde $h_w(x)$ é a função dada pela rede neural. Como no conjunto de dados há 150 exemplos, utilizamos 120 como dados de treinamento, 30 para avaliação do modelo. Utilizou-se um tamanho de passo constante no valor 1, termo de regularização para evitar o overfitting no valor $10^{-4}$ e 5 épocas, ou seja, utilizamos o *backpropagation* 5 vezes. Assim, obtemos uma precisão de 97%.

# 6 Reconhecimento de caracteres

## 6.1 Descrição do projeto

Após os estudos de otimização e aprendizagem de máquinas, aplicamos as técnicas adquiridas no reconhecimento de caracteres. O objetivo é detectar as letras em uma folha, e após a detecção fazer a classificação do seu conteúdo.

Para fazê-lo, primeiro converte-se a imagem em uma matriz de píxeis, de modo que um algoritmo detecta a posição das letras e as retorna no formato de uma lista.

Então, com a lista, recupera-se uma matriz para cada caracter e após isso tal matriz é redimensionada para o tamanho de $32 \times 32$. Assim é possível utilizar um modelo para prever o caracter. Então, na saída do programa temos os caracteres identificados.

Para o treinamento do modelo, que será uma rede neural convolucional, descrita abaixo, utilizou-se um banco de dados, de imagens preto e branca, disponibilizado por (THOMA, 2017). O banco de dados disponibiliza 369 classes, ou seja, 369 símbolos diferentes presentes no LaTeX. Na Figura 8 se encontram 100 classes do banco de dados.

Figura 8: 100 classes do banco de dados HASYv2.

Portanto, este projeto visa oferecer o início do desenvolvimento de uma ferramenta que consegue ler manuscritos, notas de aula, anotações e converter para o LaTeX. Este projeto é inspirado na ferramenta *InftyReader* (FUJIMOTO et al., 2017) feita por pesquisadores japoneses.

## 6.2 Testes

O conjunto de dados possui 168233 exemplos classificados em 369 classes. Separamos o conjunto de dados em duas partes, uma para o treinamento e outra para o teste. No conjunto de treinamento há 151241 dados e no de testes há 16992 dados.

Para a criação do modelo, utilizou-se o API do *Google*, TensorFlow (ABADI et al., 2015), que originalmente foi desenvolvido em Python, mas utilizou-se a versão na linguagem Julia. O modelo consiste de uma rede neural convolucional, com duas camadas convolucionais e uma camada densamente conectada. A primeira camada convolucional vai aplicar uma convolução que calcula 32 atributos em um caminho 5x5. Então, aplicamos um retificador linear, ReLU, que é dado pela função $f(x) = \max(0, x)$ e utilizamos o *pooling*, que diminui a nossa imagem para 16x16. Já a segunda camada possui a mesma arquitetura, com a diferença de que calcula-se 64 atributos ao invés de 32. Ou seja, a imagem foi diminuida para 8x8. Na terceira camada, densamente conectada, a imagem está no tamanho 8x8. Adicionamos uma camada de 1024 neurônios completamente conectada. Nós redimensionamos o tensor da última camada de pooling em um conjunto de vetores. Utilizou-se uma técnica chamada *dropout*, que é uma técnica de regularização para diminuir o *overfitting* durante o treinamento. E por último, adiciona-se uma camada linear totalmente conectada que nos dá o resultado final. O método de otimização utilizado para o cálculo dos melhores parâmetros é o método de Adam, um método para a otimização estocástica (KINGMA; BA, 2014).

Após montarmos a arquitetura da nossa rede neural convolucional, treinamos o modelo e obtendo uma precisão de 82% no conjunto de testes.

Com o modelo treinado, podemos aplicá-lo no nosso problema inicial, reconhecer os caracteres em uma palavra ou frase em uma linha. Para tanto, aplicamos a função que reconhece a posição de cada letra na Figura 9.

O modelo conseguiu reconhecer os caracteres como visto na Tabela 5.

Tabela 5: Letras reconhecidas da Figura 9 pelo modelo.

| **Letra Original** | $\partial$ | U | F | P | R |
|---|---|---|---|---|---|
| **Letra reconhecida** | $\partial$ | $\cup$ | F | $\nabla$ | $\mathcal{R}$ |

Utilizamos o modelo também para classificar os caracteres da Figura 10. Na Tabela

Figura 9: Imagem utilizada para teste prático do modelo.

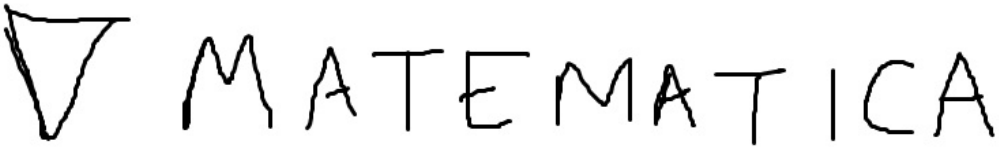

Figura 10: Imagem para utilizar no teste do programa.

6 encontram-se os resultados obtidos pelo modelo para a Figura 10.

Observe que os caracteres que não foram classificados corretamente, mas foram classificados como símbolos parecidos. Algumas razões para que isso tenha acontecido são a semelhança entre classes e dados insuficientes para a respectiva classe que foi classificada erroneamente.

Para melhorar a classificação dos caracteres precisa-se de uma coleta de dados maior e um pré-processamento de imagens mais eficiente, como o redimensionamento das imagens.

Tabela 6: Letras reconhecidas da Figura 10 pelo modelo.

| **Letra Original** | $\nabla$ | M | A | T | E | M | A | T | I | C | A |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Letra reconhecida** | $\nabla$ | M | $\Delta$ | $\top$ | $\in$ | M | $\lambda$ | $\top$ | & | $\subset$ | A |

# Conclusão

O estudo da matemática e otimização nos permite resolver muitos problemas do nosso cotidiano. O presente trabalho teve como objetivo desenvolver uma base teórica matemática para disponibilizar ferramentas que solucionem tais problemas e a aplicação de de redes neurais, por exemplo.

O reconhecimento de caracteres é o início de um projeto que tem como objetivo a leitura de textos matemáticos escritos à mão. O presente trabalho disponibiliza um modelo para o reconhecimento de um caracter individual, assim como uma ferramenta para segmentação de letras conexas em uma única palavra. Os próximos passos do projeto são o desenvolvimento de ferramentas mais sofisticadas, como ferramentas para a segmentação, e a criação de mais modelos para identificação de caracateres para melhorar a previsão.

# Referências


ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Disponível em: <http://tensorflow.org/>. Citado na página 40.

BORWEIN, J. M.; LEWIS, A. S. *Convex analysis and nonlinear optimization: Theory and examples*. 2. ed. 233 Spring Street New York, NY 10013-1578 USA: Springer, 2006. (CMS Books in Mathematics). ISBN 9780387295701,0387295704. Citado na página 21.

BURDEN, R. L.; FAIRES, J. D. *Numerical Analysis*. 9. ed. 20 Channel Center Street, Boston, MA 02210, USA: CENGAGE Learning, 2010. Citado na página 6.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, v. 2, p. 303–314, 1989. Citado na página 35.

FAN, M. *How and Why to Use Stochastic Gradient Descent?* 2014. http://anson.ucdavis.edu/~minjay/SGD.pdf. Citado na página 25.

FUJIMOTO, M. et al. *InftyProject*. 2017. http://www.inftyproject.org/en/index.html. Citado na página 40.

JAMES, B. R. *Probabilidade: Um Curso em Nível Intermediário*. 3. ed. Estr. Dona Castorina, 110 - Jardim Botânico, Rio de Janeiro - RJ, 22460-320: IMPA, 2011. Citado na página 17.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. 2014. Citado na página 40.

MITCHELL, T. M. *Machine Learning*. 1. ed. 2 Penn Plz # 12, New York, NY 10121, EUA: The McGraw-Hill Companies, 1997. Citado na página 25.

NG, A. *CS229 Lecture Notes*. 2012. http://cs229.stanford.edu/notes/cs229-notes1.pdf. Citado na página 25.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2. ed. 233 Spring Street New York, NY 10013-1578 USA: Springer, 2006. Citado 3 vezes nas páginas 6, 10 e 14.

RIBEIRO, A. A.; KARAS, E. W. *Otimização Contínua, Aspectos Teóricos e Computacionais*. 1. ed. 20 Channel Center Street, Boston, MA 02210, USA: CENGAGE Learning, 2014. Citado 3 vezes nas páginas 10, 13 e 14.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understading Machine Learning: From Theory to Algorithms*. 1. ed. Cambridge University Press, University Printing House, Shaftesbury Road, Cambridge, CB2 8BS, United Kingdom: Cambridge University Press, 2014. Citado 4 vezes nas páginas 10, 14, 25 e 38.

THOMA, M. *The HASYv2 dataset*. 2017. Citado na página 39.

ZHANG, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Proceedings of the 21st International Conference on Machine Learning*, 2004. Citado na página 25.