Matemática para Aprendizado Automático

▼ Ementa - https://we.tl/t-mCsmZioC9F

1. Técnicas de otimização
2. Aprendizado Linear
   - Regressão Linear,
   - Regressão Logística,
   - Perceptron,
   - Support Vector Machines,
   - Métricas de Classificação,
   - Clusterização via K-means,
   - Engenharia de features: Normalização, PCA Regularização.
3. Aprendizado Não-Linear
   - Regressão Não-Linear,
   - Aprendizado não-linear não-supervisionado,
   - Aproximadores Universais,
   - Metodo dos Kernels, Redes Neurais
   - Árvores de Aprendizagem

▼ Avaliação

- P1: Prova

- P2: 50% seminário e 50% trabalho
- P3: Prova
- Média: $\frac{P_1+P_2+P_3}{3}$

1. Introdução

- 1949: The Organization of Behavior - livro de neurociência que serve de base para o aprendizado automático
- 1990: boosting
- 1997: reconhecimento de voz - LSTM
  - Long Short Term Memory: modelo de RNN que grava informações passadas - dados antigos ainda influenciam novo dado processado

2. Descrição Geral

- Como o computador aprende?
- O que significa aprender?
- Modelo humano: aprende por exemplos
- Modelo computacional: algoritmo que identifica diferentes classes de objetos

2.1. Etapas na Classificação

1. Conjunto de treinamento: selecionar o conjunto de amostras (objetos) que serão usadas na classificação
2. Feature design: seleção de características, variáveis e formatos/dimensões
   a. Vetorização, transformação matricial, tratamentos de dados, limpeza e remoção de colunas
3. Modelo de treinamento
4. Validação do modelo

2.2. Principais Tipos de Aprendizagem

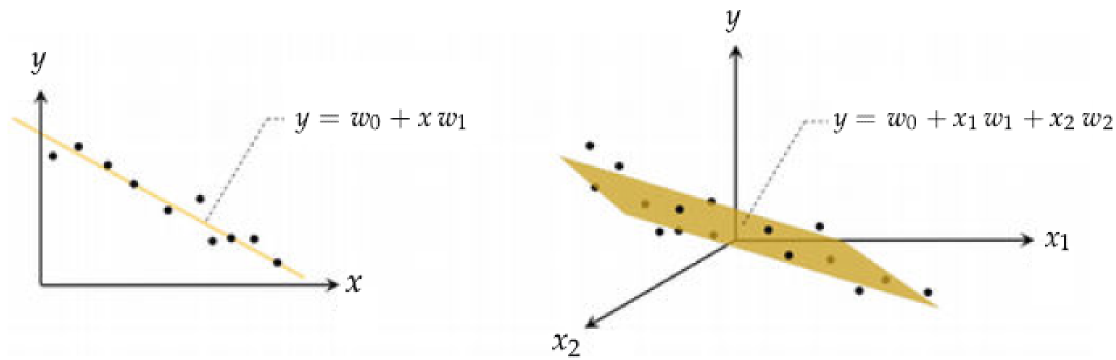
- Supervisionada: o modelo utiliza conjuntos de dados rotulados - “busca resposta para uma pergunta”
- Não Supervisionada: dados não rotulados - “busca resposta sem pergunta”
    - Exemplo - Redução de dimensão por PCA (Análise de Componentes Principais)

5. Regressão Linear

- Aprendizado supervisionado

5.2. Mínimos Quadrados

- Dados de entrada: P pontos de dimensão n
    - $(x_1, y_1), (x_2, y_2), ...(x_P, y_P), x_p = [x_{1,p}, x_{2,p}, ..., x_{n,p}]$
- Caso Geral:
    - $w_0 + w_1 * x_{1,p} + w_2 * x_{2,p} + ... + w_n * x_{n,p} = y_p$
- Para determinar os parâmetros do hiperplano, selecionamos uma função custo da forma
    - $g(w) = \frac{1}{p} \sum_{p=1}^{P} g_p(w) = \frac{1}{p} \sum_{p=1}^{P} (x_p^t * w - y_p)^2$
- Buscamos w tal que $w * x \approx y$. Logo, nosso objetivo é minimizar g(w)
- Com isso, caímos num sistema de equações lineares
    - $X_{P\times(n+1)} \cdot w_{(n+1)\times 1} = y_{P\times 1}$
- Por fim, temos que
    - $min_w(g(w)) = min_w\|X \cdot w - y\|$
    - $X^t(X * w - y) = 0 \rightarrow w = (X^t * X)^{-1} * X^t * y$
    - OBS.: proj ortogonal



▼ Funções Convexas

- Definição : dizemos que $f : U \subset \mathbb{R}^n \to \mathbb{R}$ é convexa em $I \subset U$ se e somente se, para quaisquer 2 pontos em U, a reta (ou equivalente em $\mathbb{R}^n$) é sempre maior que o valor de f

$$f((1-t)p + t\cdot q) \leq (1-t)f(p) + t\cdot f(q), \forall p, q \in U, \forall t \in [0,1]$$

- Ex.: Verifique que $f(x) = x^2, f : \mathbb{R} \to \mathbb{R}$ é convexa
-

$$\begin{gathered} \mathit{Solução} : [(1-t)p + t\cdot q]^2 \leq (1-t)\cdot p^2 + t\cdot q^2 \\ \therefore p^2\cdot(t^2-t) + q^2\cdot(t^2-t) - 2\cdot p\cdot q\cdot(t^2-t) \\ \therefore (p-q)^2\cdot(t^2-t) \leq 0, \forall p, q \in U, \forall t \in [0,1] \end{gathered}$$

- Teorema: Considere $f : U \subset \mathbb{R}^n \to \mathbb{R}$ de classe C^1 (derivada é contínua) e $p \in U$ um ponto crítico de f. Se f é convexa em U, então p é um ponto de mínimo global de f em U
- Teorema: Considere $f : U \subset \mathbb{R} \to \mathbb{R}$ de classe C^2 (2ª derivada é contínua). f é convexa se, e somente se, $f''(p) \geq 0, \forall p \in U$
  - Para $f : U \subset \mathbb{R}^n \to \mathbb{R}$, f é convexa se, e somente se, a matriz hessiana $\nabla^2 g(w)$ tem sempre autovalores não-negativos.

$$f: \mathbb{R}^2 \to \mathbb{R}$$
$$f(x, y) = x^2 + y^2$$
$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2x, 2y)$$
$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

- Verificar convexidade da função custo $g(w) = \frac{1}{p} \sum_{p=1}^{P} (x_p^t * w - y_p)^2$
    1. Utilizando a propriedade comutativa do produto escalar, simplificamos o quadrado
    2. Obtemos $g(w) = a + b^t * w + w^t * c * w$
    3. Fazer por extenso em $\mathbb{R}^2$ para verificar que a derivação segue o mesmo raciocínio do $\mathbb{R}$
    4. $\nabla g(w) = b^t + 2c \cdot w$
    5. $\nabla^2 g(w) = 2c$
- Because of its convexity, and because the Least Squares cost function is infinitely differentiable, we can apply virtually any local optimization scheme to minimize it properly

▼ Anaconda - 24/08

Regressao - funcao custo.ipynb

▼ 5.3. Mínimo Resíduo Absoluto

- Mínimos quadrados: $g(w) = \frac{1}{p} \sum_{p=1}^{P} (x_p^t * w - y_p)^2$
- Mínimos absolutos: $g(w) = \frac{1}{p} \sum_{p=1}^{P} |x_p^t * w - y_p|$
- Vantagem: reduz efeitos negativos de possíveis ruídos
- Desvantagem: não se aplica métodos de otimização de 2ª ordem (método de Newton) - não é diferenciável

5.4. Métricas de qualidade da regressão

5.4.1. Usando o modelo de treinamento

- w*: vetor w já otimizado a partir do $model(x_p, w) = y_p$
- 2 modelos de avaliação de qualidade
    1. MSE: $g(w) = \frac{1}{p} \sum_{p=1}^{P} (model(x_p, w^*) - y_p)^2$
    2. MAD: $g(w) = \frac{1}{p} \sum_{p=1}^{P} |model(x_p, w^*) - y_p|$
- É interessante aplicar o MSE na otimização por mínimos absolutos, ou o MAD na otimização por mínimos quadrados, no intuito de avaliar o quão adequada ela é na maneira alternativa de calcular
    - Leia-se: otimizamos por LSM - será que w* tem um MAD pequeno?

5.5. Regressão Ponderada

- 5.5.1. Pontos Duplicados
- Exemplo de uso: conjunto de dados com muita repetição de valores próximos → desbalanceamento do modelo de regressão
- Objetivo: adicionar pesos β_p para cada ponto y_p
- $g(w) = \frac{1}{\beta_1 + \beta_2 + ... + \beta_p} \sum_{p=1}^{P} [\beta_p \cdot (model(x_p, w^*) - y_p)^2]$

5.6. Regressão com Múltiplas Saídas

- Modelo anterior: $f : U \subset \mathbb{R}^n \to \mathbb{R}$
    - Conjunto de pares $(x_p, y_p), x_p = (x_{1,p}, x_{2,p}, ..., x_{n,p}), y_p \in \mathbb{R}$
- Modelo com múltiplas saídas: $f : U \subset \mathbb{R}^n \to \mathbb{R}^m$

$$(x_1, y_1), (x_2, y_2), ..., (x_p, y_p)$$

$$x_p = \begin{pmatrix} x_{1,p} \\ x_{2,p} \\ ... \\ x_{n,p} \end{pmatrix}, y_p^t = \begin{pmatrix} y_{0,p} \\ y_{1,p} \\ y_{2,p} \\ ... \\ x_{m-1,p} \end{pmatrix}, \mathring{x}_p^t = \begin{pmatrix} 1 \\ x_{1,p} \\ x_{2,p} \\ ... \\ x_{n,p} \end{pmatrix}$$

- Assim, o problema da regressão se transforma em m problemas

- w: matriz (N+1)x m

- Função custo: $g(w) = \frac{1}{p} \sum_{p=1}^{P} \sum_{c=0}^{m-1} (\mathring{x}_p^t * w_c - y_{c,p})^2$
- Minimizar cada c

▼ 3. Otimização de 1ª ordem

- Técnicas que utilizam a 1ª derivada da função

▼ Pontos Críticos e Extremos Locais

- Definição : considere $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ de classe C^1 e $p \in Dom(f)$. O vetor gradientede f no pontopé dado por:

$$\nabla f = \left(\frac{\partial f(p)}{\partial x_1}, \frac{\partial f(p)}{\partial x_2}, ..., \frac{\partial f(p)}{\partial x_n} \right)$$

- Definição : considere $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ de classe C^1 e $p \in Dom(f)$. O ponto p é ponto crítico de f se $\nabla f(p) = 0$
- Teorema: considere $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ de classe C^1 e $p \in Dom(f)$. Se p é um extremo local de f e um ponto interior de D, então p é um ponto crítico de f.

▼ 3.5. Gradiente Descendente

- Teorema: considere $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ de classe C^1 e $p \in D$ um ponto interior de D . Se $\nabla f(p) \neq 0$ então a direção de maior crescimento de f em p é dada por $\nabla f(p)$ e a direção de menor crescimento é $-\nabla f(p)$
- Método: $w^k = w^{k-1} - \alpha \cdot \nabla f(w^{k-1})$
    - α é o passo iterativo da otimização: incremento utilizado para se aproximar cada vez mais no ponto crítico
- Controle passo:
    1. Passo fixo: α constante
    2. $\alpha = \frac{1}{k}$: k iterações

3. $\alpha = \frac{\alpha}{2}$: quando por exemplo, numa função de minimização $f(w^{k+1}) > f(w^k)$ → refaz a iteração com um passo menor

▼ Seminario

▼ Enunciado

1. Vídeo - Apendice A.4. Advanced Gradient-Based Methods
    - Descrever métodos solicitados, introduzir pré-requisitos e mostrar exemplo
2. Implementar gradiente descendente do A.5 (Adam) e A.6 (RMSProp) e aplicar na regressão linear da Aula 4 (ead)

▼ A.4 - Advanced Gradient-Based Methods

- Momentum-accelerated gradient descent

▼ 6. Classificação Binária

- É um caso de aprendizagem supervisionada com apenas duas saídas, denominadas classes
- Serão definidas novas funções custo associado a este problema, que envolvem elementos como:
    - Regressão Logística
    - Perceptron
    - SVM - Support Vector Machines

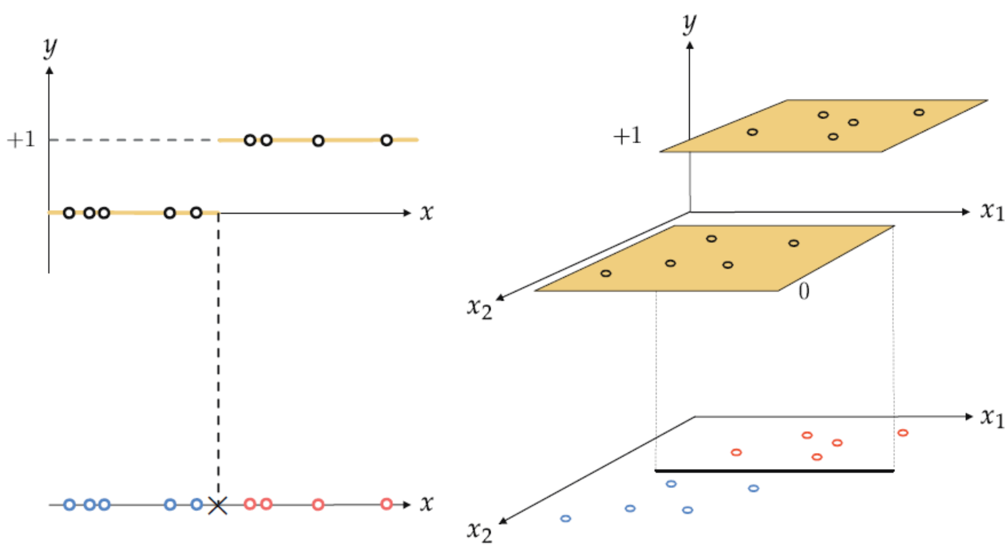
▼ 6.2. Regressão Logística e Entropia Cruzada

- Os dados estão na forma a seguir:
- Cada valor y é um label, e formam conjuntos chamados classes (2-class classification)

$$\{(x_p, y_p)\}, x_p = \begin{bmatrix} x_{p,1} \\ x_{p,2} \\ \dots \\ x_{p,N} \end{bmatrix}, \mathring{x}_p = \begin{bmatrix} 1 \\ x_{p,1} \\ x_{p,2} \\ \dots \\ x_{p,N} \end{bmatrix}, y_p \in \{0, 1\}$$

▼ Visualização

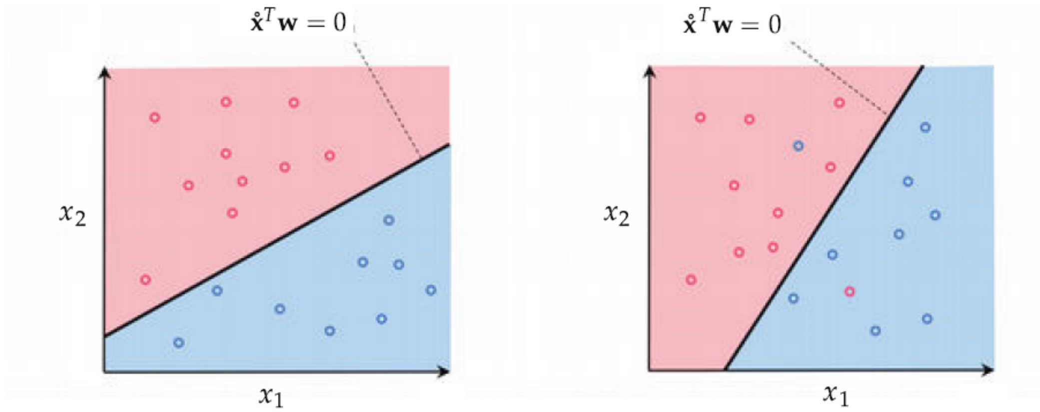
- VIsualização - RLog



one-dimensional input:
decision boundary is a single point

two-dimensional input:
decision boundary is a line

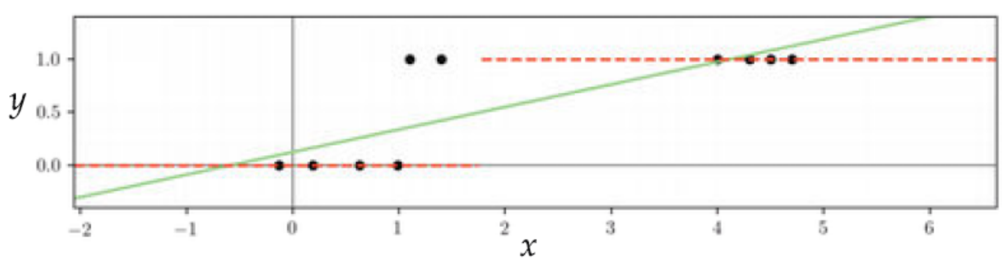
- Visualização - Perceptron



6.2.2. Função degrau

Função degrau: $step(t) = \begin{cases} 0 \quad se \quad t < 0 \\ 1 \quad se \quad t > 0 \end{cases}$

- A regressão linear não é adequada pois não incorpora a função degrau



- No $\mathbb{R}^n$, a função degrau é um hiperplano que corta em 0 e 1

- Função Custo
  - Regressão Linear: $g(w) = \frac{1}{p}\sum_{p=1}^{P}(x_p^t * w - y_p)^2$
  - Regressão Logística: $g(w) = \frac{1}{p}\sum_{p=1}^{P}(step(x_p^t * w) - y_p)^2$
  - Assim, a função custo envolvendo a função degrau tem regiões ou platôs planos, onde o gradiente se anula, impossibilitando aplicar os métodos clássicos de otimização

6.2.3. Função Logística Sigmoide

RLog sigmoide: $\sigma(wx) = \dfrac{1}{1 - e^{-wx}}$

Função degrau: $step(t) = \lim_{w \to \infty} \sigma(wt) = \begin{cases} 0 \ \ se \ \ t < 0 \\ 1 \ \ se \ \ t > 0 \end{cases}$

- Motivação: a minimização por mínimos quadrados da função degrau é inviável dada a horizontalidade dos dados (derivada tende a 0 em quase todo o domínio)
- Substituindo pela sigmoide, é possível minimizar a função custo. A essa minimização, culminando na separação dos dados em 2 classes, é chamada regressão logística.

▼ 6.2.5. Entropia Cruzada

- O erro de um ponto $x_p \to g_p(w) = (\sigma(\mathring{x}_p^T w) - y_p)^2$ não é o melhor jeito de calcular o erro no caso de 2-class, onde o output y é sempre 0 ou 1
- Em vez disso, utiliza-se o log do erro: $x_p \to g_p(w) = \begin{cases} -log(\sigma(\mathring{x}_p^T w)) & se \ y_p = 1 \\ -log(1 - \sigma(\mathring{x}_p^T w)) & se \ y_p = 0 \end{cases}$
  - Sempre ≥ 0
  - Penaliza mais que o squared error
- Assim, podemos escrever o custo de Entropia Cruzada total como:
  - Sempre convexo, permitindo maior gama de otimizações

$$g(w) = -\frac{1}{P} \sum_{p=1}^{P} y_p \cdot \log\left(\sigma(\mathring{x}_p^T w)\right) + (1 - y_p) \log(1 - \sigma(\mathring{x}_p^T w))$$

$$\therefore \nabla g(w) = -\frac{1}{P} \sum_{p=1}^{P} (y_p - \sigma(\mathring{x}_p^T w)) \cdot \mathring{x}_p$$

▼ 6.3 Regressão Logística e Custo Softmax

- Limitação do custo de entropia cruzada: assime que o output é somente 0 ou 1

- Custo Softmax: custo similar ao de enropia cruzada, mas para quaisquer valores de output y
- Assuma que $y_p \in \{-1, 1\}$
  - Adapta-se a função sigmoide para a tangente hiperbólica: $\tanh(x) = 2 \cdot \sigma(x) - 1 = \frac{2}{1+e^{-x}} - 1$
  - Utilizamos o artifício do log do erro, e chegamos no custo Softmax, função convexa

$$g(w) = \frac{1}{P} \sum_{p=1}^{P} \log(1 + e^{-y_p \mathring{x}_p^T w})$$

$$\therefore \nabla g(w) = -\frac{1}{P} \sum_{p=1}^{P} \frac{e^{-y_p \mathring{x}_p^T w}}{1 + e^{-y_p \mathring{x}_p^T w}} y_p \mathring{x}_p$$

- Ruído (noise): pontos x_p mal classificados pelo modelo

6.4. Perceptron

- Até aqui, tratamos a regressão logística como uma forma particular de regressão não-linear
- O Perceptron é uma abordagem linear da RLog, pois busca determinar diretamente o linear decision boundary, ou seja, a reta, plano ou hiperplano que divide as classes
- Embora o resultado seja o mesmo, é uma perspectiva importante e nos permite entender geometricamente o conceito de regularização

6.4.1 Função Custo do Perceptron

- Exemplo $\mathbb{R}^3$: $f(x_1, x_2) = y$
  - Visto de cima, o linear decision boundary é uma reta $\mathring{x}^T w$, dividindo em 2 subespaços
  - Logo, os pesos w ideais dividem o espaço tal que

$$\begin{cases} \mathring{x}^T w > 0 \ \ se \ \ y_p = +1 \\ \mathring{x}^T w < 0 \ \ se \ \ y_p = -1 \end{cases}$$

- Assim, temos que o custo é $g(w) = \frac{1}{P} \sum_{p=1}^{P} \max(0, -y_p \mathring{x}_p^T w)$
  - Rectified Linear Unit cost: ReLU
  - Sempre ≥ 0
  - Sempre convexa
  - 1x diferenciável (somente zero e 1-ordem)

6.5. Suport Vector Machines (SVMs)

- Sofmax function $soft(s_0, s_1, ..., s_n) = log(e^{s0} + e^{s1} + ... + e^{sn})$
- Margin-Perceptron: uso do Perceptron para maximizar a margem da reta limite
- A forma regularizada do custo do margin-perceptron é o SVM

$$g(b, w) = \sum_{p=1}^{P} \max(0, 1 - y_p(b + \mathring{x}_p^T w)) + \lambda \|w\|_2^2$$

- Vantagem: lida melhor com ruídos

6.8. Métricas de classificação de qualidade

6.8.1. Predições de Modelo

- $model(x, b^*, w^*) = b + w_1^* \cdot x_1 + w_2^* \cdot x_2 + ... + w_n^* \cdot x_n$

6.8.2. Escore de Confiança

- Podemos definir a noção de confiança pela distância à fronteira de decisão: $d = \dfrac{b^* + x_p^t \cdot w^*}{||w^*||_2}$
  - Podemos aplicar a sigmoide: $\sigma(d) = \dfrac{1}{1 + e^{-d}}, \sigma(d) \in (0, 1)$
  - OBS.: O valor na fronteira é próximo de 0.5 - incerto, enquanto pontos distantes da fronteira (0 ou 1) têm maior confiança

▼ 6.8.3. Medidas de Acurácia

- Função identidade

$$I(\hat{y}_p, y_p) = \begin{cases} 0 \;\; se \;\; \hat{y}_p = y_p \\ 1 \;\; se \;\; \hat{y}_p \neq y_p \end{cases}$$

- Avaliada sobre o conjunto de treinamento
- Número de elementos classificados incorretamente: $\sum_{p=1}^{P} I(\hat{y}_p, y_p)$

$$A = 1 - \frac{1}{P}\sum_{p=1}^{P} I(\hat{y}_p, y_p)$$

▼ 6.8.4. Acurácia Balanceada

- $\Omega_{+1} =$ conjunto dos pontos classificados com label +1
- $\Omega_{-1} =$ conjunto dos pontos classificados com label -1
- $|\Omega_{-1}| =$ número de elementos do conjunto

$$A_{+1} = 1 - \frac{1}{|\Omega_{+1}|}\sum_{p \in \Omega_{+1}}^{P} I(\hat{y}_p, y_p)$$
$$A_{-1} = 1 - \frac{1}{|\Omega_{-1}|}\sum_{p \in \Omega_{-1}}^{P} I(\hat{y}_p, y_p)$$
$$A_{balanceada} = \frac{A_{+1} + A_{-1}}{2}$$

▼ 6.8.5. Matriz de Confusão

- Matriz 2x2 positivo e falso

TRUE POSITIVE	FALSE POSITIVE
FALSE NEGATIVE	TRUE NEGATIVE

$$A_{+1} = 1 - \frac{1}{|\Omega_{+1}|} \sum_{p \in \Omega_{+1}}^{P} I(\hat{y}_p, y_p)$$
$$A_{-1} = 1 - \frac{1}{|\Omega_{-1}|} \sum_{p \in \Omega_{-1}}^{P} I(\hat{y}_p, y_p)$$
$$A_{balanceada} = \frac{A_{+1} + A_{-1}}{2}$$

▼ 8. Aprendizado Não-Supervisionado Linear

- Objetivo: aprendizado automático de regras e padrões de um conjunto de input; não há output
- 2 tipos: redução dimensional e clusterização (clustering)
- Muitas vezes utilizado como preprocessamento para modelos supervisionados em grandes datasets

▼ 8.2. Revisão Álgebra Linear

- Dataset de um modelo não-supervisionado: $\{x_p\}_{p=1}^{P}$
- Cada ponto x_p pode ser visualizado como ponto ou como vetor a partir da origem; ambas são importantes para o aprendizado NS
- Conjunto gerador: conjunto de vetores a partir dos quais podemos gerar todos os pontos x_p do conjunto
  - $\sum_{k=1}^{K} c_k w_{p,k} = x_p \Rightarrow C \cdot w_p = x_p \qquad p = 1, ..., P$
  - Matriz C: vetores geradores como colunas
  - Vetor w_p: pesos da combinação linear $[w_{p,1}, w_{p,2}, ..., w_{p,N}]$
- Logo, o objetivo torna-se encontrar w_p, e podemos utilizar os métodos já estudados, como Mínimos Quadrados ou Equação Normal

$$g(w_1, w_2, ..., w_p) = \frac{1}{P} \sum_{p=1}^{P} \|C \cdot w_p - x_p\|_2^2$$
$$C^T C w_p = C^T x_p$$

- Encoding de um vetor: o vetor otimizado w_p^* é chamado de encoding de x_p para a matriz geradora C
- Decoding de um vetor: a multiplicação matricial Cw_p^*

8.2.3. Conjunto Gerador Ortonormal

- Conjunto cujos vetores têm norma 1 e são ortogonais entre si: $CC^T = I_{N \times N}$
- Bastante útil, pois permite a resolução do vetor de pesos imediatamente pela Equação Normal. Portanto, o processo de encoding e decoding se torna barato, obtido a partir de uma multiplicação matriz-vetor:
  - $C^T C w_p = C^T x_p \rightarrow w_p = C^T x_p$
- Autoencoder: expressa o encoding seguido de decoding de um vetor x_p, de modo a retornar para ele mesmo. Isso é possível em um conjunto ortonormal pois $C^T = C^{-1}$
  - $CC^T x_p = x_p$

- Em exemplos reais, não há um conjunto gerador do espaço $\mathbb{R}^N$. Assim, a otimização obtém um encoding w_p tal que Cw_p é a projeção de x_p no subespaço gerado por C.

8.3. Autoencoder e PCA

- Principal Component Analysis (PCA): principal método não-supervisionado
  - Além de obter w, obtém também o melhor conjunto gerador C
  - Como?
- Adaptando a função custo (não convexa)

$$g(w_1, w_2, ..., w_p, C) = \frac{1}{P} \sum_{p=1}^{P} \|C \cdot w_p - x_p\|_2^2$$

- Todavia, se C for ortonormal, podemos aplicar também a fórmula do Autoencoder, de modo a depender somente de C:

$$g(C) = \frac{1}{P}\sum_{p=1}^{P} \|CC^T x_p - x_p\|_2^2$$

- Matriz de covariância: dados os vetores $\{x_p\}_{p=1}^{P}$, e X a matriz formada por eles (em coluna), a matriz de covariância entre os pontos é $\frac{1}{P}XX^T$
- Podemos decompor a matriz a partir de seus autovalores e autovetores, onde V é a matriz de autovetores (em coluna) e D é a matriz diagonal dos autovalores
    - Os autovetores ortonormais são os componentes principais

$$\frac{1}{P}XX^T = VDV^T$$

▼ 8.5. K-Means Clustering

- Diferente do PCA: não visa reduzir dimensões, mas reduzir ou agrupar os pontos de input, para melhor compreensão do dataset
- Dado P pontos $\{x_1, x_2, ..., x_P\}$, podemos agrupá-los em K clusters, sendo K ≤ P
- Notação
    - P pontos $\{x_1, x_2, ..., x_P\}$
    - K clusters de centroides $\{c_1, c_2, ..., c_K\}$
    - Cada cluster possui um set de pontos: $S_k = \{p \mid \text{if } x_p \text{ belongs to the kth cluster}\}$
- Cada centroide k é a média dos pontos de S_k
    - $c_k = \frac{1}{|S_k|}\sum_{p \in S_k} x_p$
- Assignment: nome dado ao cluster c_k ao qual x_p pertence, ou seja, o que possui menor distância
    - $a_p = \underset{k=1,2,...,K}{\arg\min} \|x_p - c_k\|_2$

▼ Determinando o número de clusters: k-means

- Elbow Method: para determinar o melhor k, iniciamos calculando a distância dos pontos ao cnetroide, da forma
  - Montar o gr[afico de sse x k
  - Buscar k onde ocorre o máximo

$$SSE = \sum_{k=1}^{K} \sum_{k=1}^{K}$$

- M[etodo da silhueta? medida de similaridade dos pontos com seus clusters

▼ 10. Non-linear Feature Engineering

Listas