MACHINE LEARNING

Algoritmos Regressores

cognitiza

Your Data & AI school

Prof. Altino Dantas

Conceito

- Aprendizado supervisionado, ou seja, existem as características (variáveis independentes) e o alvo (variável dependente);
- A variável alvo é um valor numérico, contínuo ou discreto;
- Necessidade de separação de conjunto de treino e teste;
- Todos os exemplos destes slides usarão o *scikit learn.*

Problemas

- Estimativa de preço de imóveis;
- Previsão de valores de vendas em determinado período;
- Previsão de volume de chuva;
- Número de acidentes em trechos de rodovia;
- Quantidade de leitos de UTI ocupados por pacientes covid considerando o isolamento social;

Alguns algoritmos

- Regressão linear e logística;
- Árvores de decisão;
- Maquina de vetores de suporte (SVMs);
- Redes Neurais Artificiais - Perceptron Multicamadas;
- kNN;
- Muitos outros...

Alguns algoritmos

Veja que várias técnicas de Machine Learning possuem algoritmos tanto para problemas de regressão, quanto para problemas de classificação de padrões. O que difere tais algoritmos é a forma de lidar com os dados e consequentemente realizar predições.

Regressão Linear

Regressão Linear

- Lida com problemas cuja resposta é numérica (tarefas de regressão);
- Como o nome sugere, tenta encontrar uma relação linear entre um valor alvo (variável dependente) e um ou mais preditores (variáveis independentes);
- Regressão simples ou múltipla;

Regressão Linear

- A regressão linear simples relaciona uma variável independente X com uma variável dependente Y, e pode ser dada por:

$$Y \approx \beta_0 + \beta_1 X.$$

- β0 e β1 são constantes desconhecidas, parâmetros ou coeficientes do modelo, que representam a inclinação e a interceptação do modelo linear;

Regressão Linear

- Uma vez utilizados os dados de treinamento do modelo, os valores dos coeficientes podem ser ajustados e tem-se:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

- onde ^y uma predicação para Y considerando a entrada x de X.

$$e_i = y_i - \hat{y}_i$$

- representa o *i*-ésimo erro residual, que é a diferença entre o valor esperado e o valor predito pelo modelo.

Regressão Linear

- A partir dos erros produzidos para cada amostra de X, define-se a Soma dos Erros Residuais (Residual Sum of Squares):

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$

- Esta é uma estratégia para lidar com diferenças negativas e, é equivalente a:

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

- Chamada de Função de perda (Loss Function), deve ser minimizada para que o modelo bons resultados.

Métricas para regressão

- Relative Absolute Error (RAE)
- Mean Squared Error (MSE)
- **Root Mean Squared Error on Prediction (RMSE/RMSEP)**
- Normalized Root Mean Squared Error (Norm RMSEP)
- Relative Root Mean Squared Error (RRMSEP)

- Mean/Median of prediction
- Standard Deviation of prediction
- Range of prediction
- **Coefficient of Determination (R2)**
- Relative Standard Deviation/Coefficient of Variation (RSD)
- Relative Squared Error (RSE)
- **Mean Absolute Error (MAE)**

Gradiente Descendente Estocástico

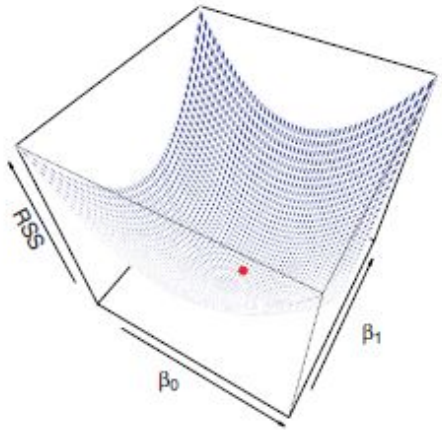
- A acurácia do modelo pode ser mensurada pelo MSE (Mean Squared Error), da seguinte forma:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

- Definindo MSE como a função de perda L, através de uma série de iterações o valor da função é atualizado pelo subtração de sua ***derivada negativa.*** Para isso, usa-se o método conhecido como Gradiente Estocástico que ajuda encontrarmos os mínimos globais (eventualmente, locais) de uma função.

Gradiente Descendente Estocástico

- A posição da bola vermelha indica o valor da função de perda considerando os coeficientes βs, assim pode-se formalizar o GDE como um procedimento de atualização:



$$L(m) = \text{MSE Error}$$

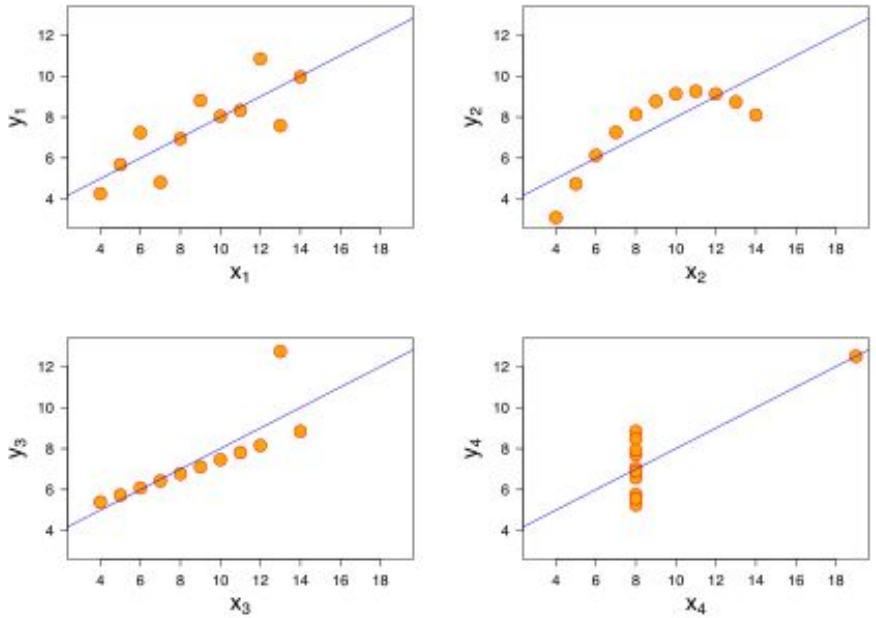
$$m_0 = 1$$

$$m_i = m_{i-1} - \alpha \frac{\partial L}{\partial m_{i-1}}$$

Tome *m* como os coeficientes β.

Verifique o efeito do gradiente descente aqui: https://altinodantas.github.io/singleArtificialNeuron/gradient.html

Exemplos de cenários com modelo de regressão linear



- Dependente da ligação linear entre as variáveis independentes e da variável dependente;
- Sensível a existência de outliers;
- A distribuição normal dos erros pode gerar instabilidade no processo de ajuste;

Exemplo de código em Python

```
'''
importação dos recursos necessários
'''
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
import numpy as np

'''
carregando os dados disponíveis na biblioteca scikitlearn
'''
data = load_boston()

```

Exemplo de código em Python

```
'''
Separando conjuntos de treino e teste
'''

x_train, x_test, y_train_labels, y_test_labels = train_test_split(data.data,
                                                                  data.target,
                                                                  test_size=0.33,
                                                                  random_state=42)
```

Exemplo de código em Python

```
'''
Criando e treinando o modelo
'''
gnb = LinearRegression()
model = gnb.fit(x_train, y_train_labels)
```

```
'''
Fazendo predições
'''

preds = gnb.predict(x_test)
print(preds[0])
```

Exemplo de código em Python

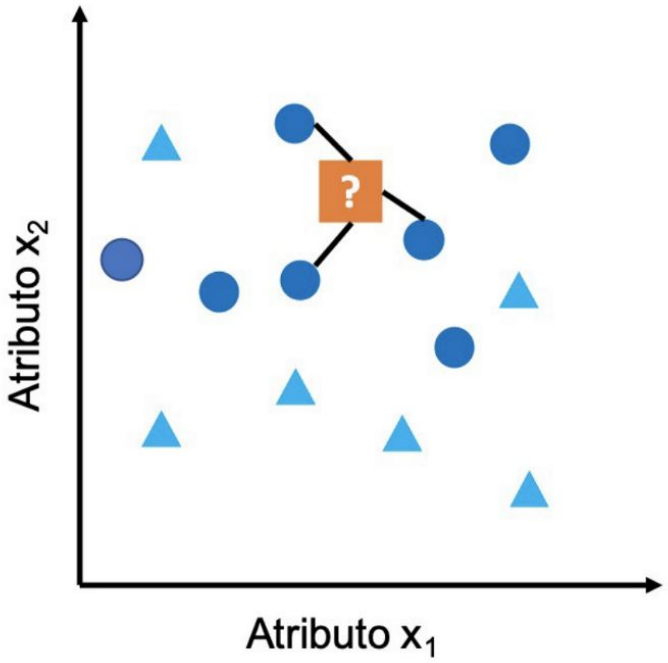
```
'''
Avaliando o modelo
'''

rmse = (np.sqrt(mean_squared_error(y_test_labels, preds)))
print("Valor de MSE:", rmse)

r2 = r2_score(y_test_labels, preds)
print("Valor de R2:", r2)
```

kNN (k-Nearest Neighbours)

k-vizinhos mais próximos

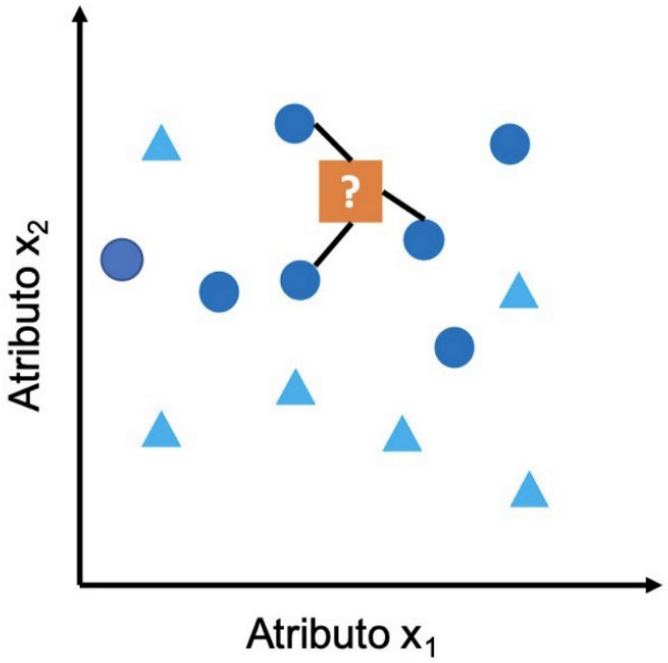
Definição

- Algoritmo de fácil entendimento e implementação;
- Considera-se que os exemplos vizinhos são similares ao exemplo cuja informação se deseja inferir;
- Para tanto, o algoritmo considera que cada amostra é um ponto no espaço de dimensões dos atributos;



Definição

- O conjunto de dados de treinamento é armazenado e, quando um novo exemplo chega, ele é comparado a todos os exemplos armazenados para identificar os k vizinhos mais próximos;
- k é um parâmetro do algoritmo, que estabelece os k elementos;
- Para um problema de regressão, o valor atribuído à nova observação é a média aritmética dos k vizinhos.



kNN - k-Nearest Neighbours | Funcionamento

1. Definição da métrica de distância utilizada e valor de *k*.
2. Cálculo da distância do novo exemplo a cada um dos exemplos existentes no conjunto inicial de entrada.
3. Identificação dos *k* exemplos do conjunto de referência que apresentaram menor distância em relação ao novo exemplo (mais similares).
4. Apuração da classe mais frequente entre os *k* exemplos identificados no passo anterior, usando votação majoritária (para problemas de classificação) ou estimação do valor Y como a média aritmética dos *k*-vizinhos mais próximos.

Exemplo de código

```
1 from sklearn.neighbors import KNeighborsRegressor
2 from sklearn.metrics import r2_score
3 import pandas as pd
4 dados_treino = pd.read_csv("sample_data/california_housing_train.csv")
5 dados_teste = pd.read_csv("sample_data/california_housing_test.csv")
```

```
1 X_train = dados_treino[['longitude','latitude','housing_median_age',
2                         'total_rooms','total_bedrooms','population',
3                         'households','median_income']]
4 y_train = dados_treino.pop('median_house_value')
5
6 X_test = dados_teste[['longitude','latitude','housing_median_age',
7                         'total_rooms','total_bedrooms','population',
8                         'households','median_income']]
9 y_test = dados_teste.pop('median_house_value')
```

```
1 neigh = KNeighborsRegressor(n_neighbors=50)
2 neigh.fit(X_train, y_train)
3
4 preds = neigh.predict(X_test)
5 print("R2 score ", r2_score(y_test,preds))
```

```
R2 score  0.29814558187368423
```

Outros algoritmos

- Alguns algoritmos vistos em suas versões para classificação também possuem implementações para regressão:
  - **sklearn.svm**.SVR
  - **sklearn.neural_network**.MLPRegressor
  - **sklearn.tree**.DecisionTreeRegressor

Obrigado.