FIAP GRADUAÇÃO

FIAP

ENGENHARIA DE COMPUTAÇÃO

Inteligência Artificial e Computacional

PROF. ANTONIO SELVATICI

Versão 4 – Agosto/2018

SHORT BIO

É engenheiro eletrônico formado pelo Instituto Tecnológico de Aeronáutica (ITA), com mestrado e doutorado pela Escola Politécnica (USP), e passagem pela Georgia Institute of Technology em Atlanta (EUA). Desde 2002, atua na indústria em projetos nas áreas de robótica, visão computacional e internet das coisas, aliando teoria e prática no desenvolvimento de soluções baseadas em Machine Learning, processamento paralelo e modelos probabilísticos. Desenvolveu projetos para Avibrás, IPT, CESP e Systax.

PROF. ANTONIO SELVATICI
profantonio.selvatici@fiap.com.br

2. MACHINE LEARNING

Uma implementação simples para o kNN

- A implementação básica da classificação por **k-vizinhos mais próximos** (ou **kNN**) é bastante simples.
- Dados que temos os dados de treinamento armazenados em um DataFrame, para uma nova amostra de teste **x**, devemos:
  1. Encontrar o array de distâncias das amostras de treinamento com relação a **x**;
  2. Ordenar o array de distâncias encontrado para determinar os vizinhos mais próximos;
  3. Encontrar a classe que mais aparece na vizinhança determinada
- Para o passo 1, vamos aproveitar as facilidades de cálculo matricial fornecidas pelos módulos `pandas` e `numpy` do Python, representando tanto a amostra de teste **x** quanto as amostras rotuladas por matrizes ou data frames
- A matriz M das amostras de treinamento é dada pela concatenação dos vetores de atributo como colunas dessa matriz:
  - No caso de usarmos os atributos "petal lenght" e "petal width" armazenados no dataframe `df`:
  - `M = df[['petal_length', 'petal_width']];`

Calculando as distâncias entre dois vetores de mesmas dimensões

- Para calcular a distância euclidiana entre dois vetores n-dimensionais $\mathbf{a} = (a_1, a_2, a_3, \ldots, a_n)$ e $\mathbf{b} = (b_1, b_2, b_3, \ldots, b_n)$, fazemos a raiz quadrada da soma das diferenças:

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$$

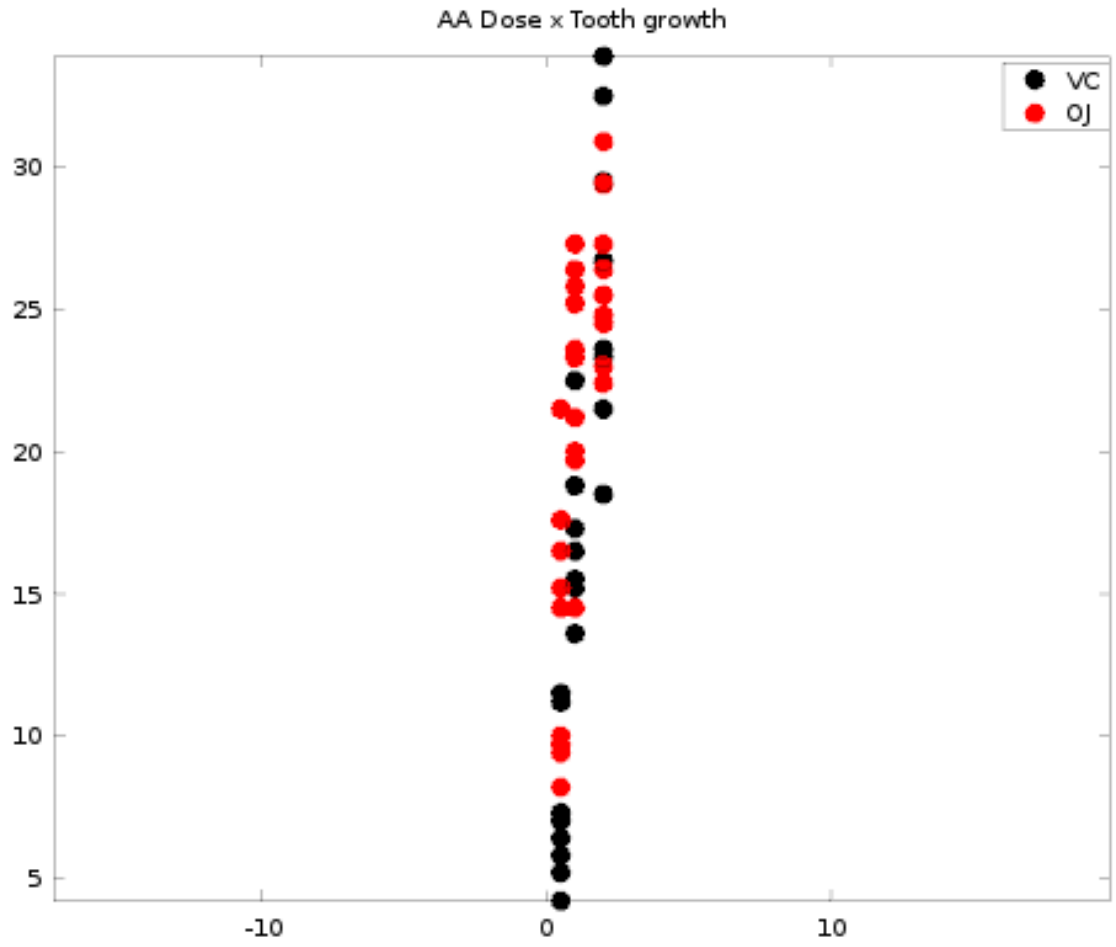
Implementação do kNN no Python

- Cada linha da matriz M de corresponde a uma amostra de treinamento.
- Dada uma amostra de teste **x**, para calcularmos as distâncias temos que primeiro calcular as diferenças entre as coordenadas (colunas) de cada linha de M para as coordenadas de **x**, fazendo:
  - Matriz de diferenças: `D = M - x`
  - Embora o número de colunas de `M` e `x` seja o mesmo, M contém várias entradas (linhas) enquanto x só tem uma. As operações entre essas matrizes se baseiam no princípio do broadcasting, pelo qual a dimensão unitária de um array é repetida várias vezes para casar com a mesma dimensão do outro array que está sendo processado.
- Cálculo do vetor de distâncias ao quadrado, onde a soma é feita ao longo da segunda dimensão (eixo das colunas) de `D`: `dists2 = sum(D ** 2).sum(axis=1)`
- Ordenação do vetor de distâncias quadráticas e recuperação dos índices:
  - `idx = dists2.sort()`
  - Agora as distâncias ordenadas podem ser recuperadas através de: `dists2_ord = dists2(idx)`.
- Contando os `k` vizinhos pertencentes a cada classe
  - `n_setosa = np.count_nonzero(df['species'][idx[:k]]=='Iris-setosa')`
  - `n_versicolor = np.count_nonzero(df['species'][idx[:k]]=='Iris-versicolor')`
  - `n_virginica = np.count_nonzero(df['species'][idx[:k]]=='Iris-virginica')`
- Vamos implementar esse programa no PyCharm, incluindo as linhas que estão faltando

Visualizando diferenças nas escalas dos atributos

- Seja o dataset "toothgrowth" disponível no site: https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/ToothGrowth.html
- Ele guarda as observações de um estudo do crescimento dos incisivos de 60 porquinhos-da-índia que são separado em três grupos que recebem dosagens diárias de vitamina C diferentes. Em cada grupo, metade dos porquinhos recebem na forma de suco de laranja (OJ) e metade recebe na forma de vitamina C sintética (VC)
- Código para a visualização dos dados:
  - `url = "https://vincentarelbundock.github.io/Rdatasets/csv/datasets/ToothGrowth.csv"`
  - `df = pd.read_csv(url)`
  - `df = df['dose', 'len', 'supp'] #Remove a primeira coluna`
  - `sns.scatterplot('dose', 'len', hue='supp', data=df)`
  - `plt.axis('equal')`
- Observe a dimensão "dose" indica um valor na unidade miligramas, e a dimensão "len" está em uma unidade de comprimento
- Para calcular as distâncias de um ponto a outro neste espaço de atributos, temos que somar as diferenças de dosagens (mg) e as diferenças de comprimento (décimos de mm), o que não faz sentido fisicamente falando

Exemplo de espaço de atributos com escalas iguais nos dois eixos



Normalização dos dados

- Observe que as tratar as distâncias em todas as direções (coordenadas) de forma igualitária não é razoável, uma vez que pequenas diferenças em determinados atributos são importantes para definir a classe da amostra, porém diferenças maiores aplicadas a outros atributos parecem não fazer tanta diferença
- Dessa forma, melhores resultados serão alcançados se fizermos um ajuste de escala nos dados, de forma equalizar a importância dos atributos para efeito de classificação. A esse ajuste vamos chamar de **normalização** dos dados.
- Uma normalização simples, mas efetiva, é aquela que torna cada vetor de atributos um conjunto de dados de média zero e desvio padrão unitário
  - Para tanto, basta deduzir os valores de cada vetor de atributos de sua média (**`mean`**) e dividir pelo desvio padrão (standard deviation, ou **`std`**)
- No Python, podemos operar sobre todos os campos de dados do DataFrame:
  - `data = df[['dose', 'len']]`
  - `data_norm = (data - data.mean())/data.std()`
  - `data_norm["supp"] = df["supp"] # Acrescentando o campo das classes`
- Ao executar o teste, temos que normalizar os atributos com os mesmos dados obtidos na fase de treinamento, ou seja, para a amostra de teste x, sua versão normalizada é:
  - `x_norm = (x - data.mean())/data.std()`

Média e desvio padrão

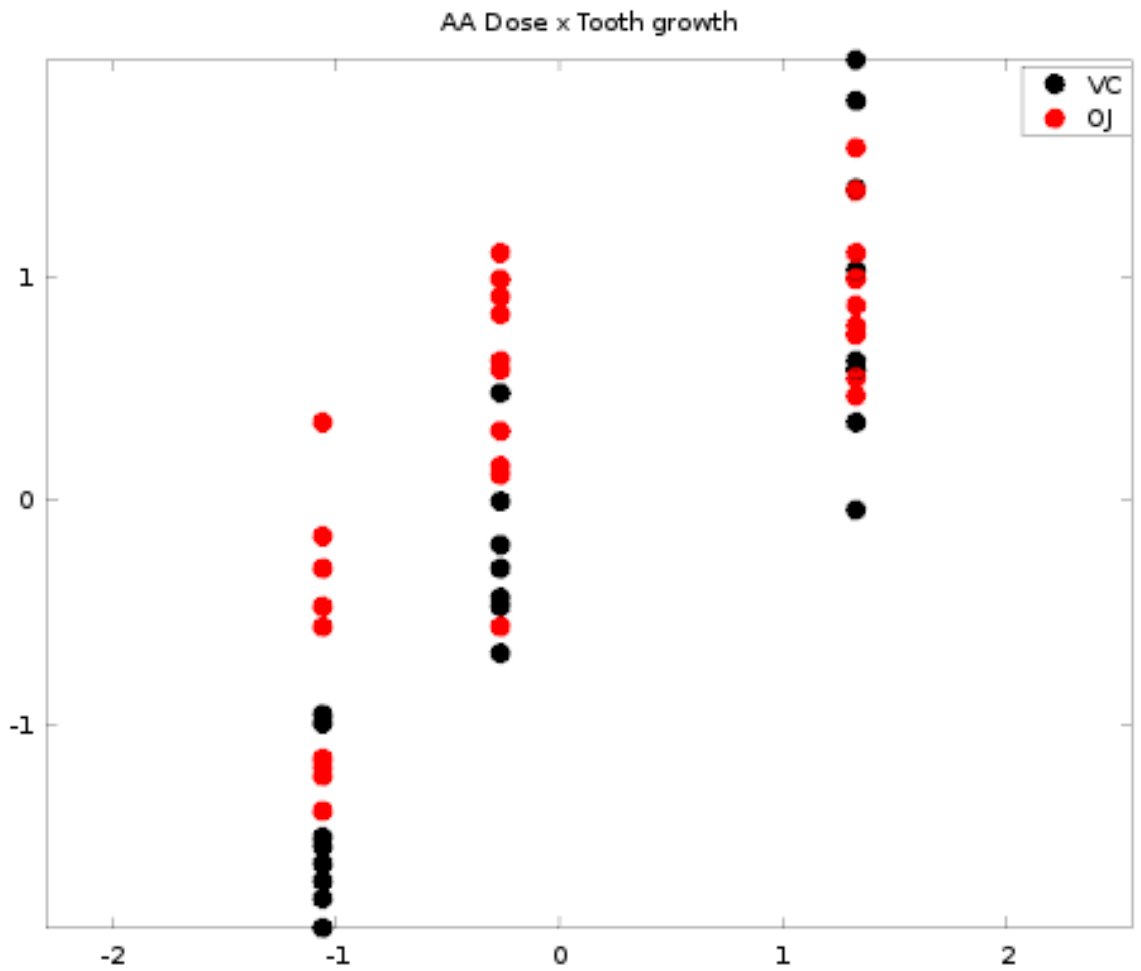
- Dado um vetor de dados $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n)$, podemos calcular:
  - Sua média, dada por

$$\bar{x} = \sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

  - Seu desvio padrão, dado por

$$\sigma_x = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2 - n\bar{x}^2}{n-1}}$$

Exemplo de espaço de atributos após a normalização, com escalas iguais nos dois eixos



Exercício

Qual é o resultado da classificação por kNN para o vetor de teste v=(*x*=4,0; *l*=20,8), considerando k=1 e k=3, respectivamente, realizando a normalização pelo desvio padrão?

Tipo de peixe	Brilho (x)	Comprimento (l)	Brilho normalizado	Comprimento normalizado	Distância	Ordem
Salmão	3,5	21				
Badejo	3,5	21,5				
Badejo	3,5	20,2				
Badejo	3,5	20				
Salmão	4,5	21				
Salmão	5	21				

Condensando o conhecimento

- O que é Machine Learning?
- Quais as três modalidades de aprendizado segundo o livro texto?
- O que é reconhecimento de padrões?
- Quais são os três passos do reconhecimento de padrões?
- O que são os atributos?
- O que é um classificador?
- Que tipos de classificadores nós vimos nesta aula?
- O que significa normalização de dados? Por que é importante?

REFERÊNCIAS

- Stuart Russel & Peter Norvig. Inteligência Artificial – tradução da 2ª ed. Editora Campus, 2004, capítulo 18
- Duda, Hart & Stork. Pattern Classification, 2nd. Ed., 2000, capítulo 4

FIAP