CLASSIFICATION

Prof. Nielsen Rechia
nielsen.machado@uniritter.edu.br

CLASSIFICATION

Paradigmas	Supervisionado	Não-supervisionado
Tarefas	Classificação	Análise associativa
	Regressão	Agrupamento (*clustering*)
	Outros	Redução de dimensionalidade
		Outros

7 tarefas comuns de aprendizado de máquina: http://vitalflux.com/7-common-machine-learning-tasks-related-methods/

CLASSIFICATION

Dado um conjunto de objetos (instâncias de treino), onde cada objeto possui atributos (features) sendo um deles a classe;

Encontre o modelo mais ajustado para estes dados;

Para novos objetos, o modelo deve classificá-los de forma correta;

Um conjunto de teste deve ser processado para determinar a qualidade do modelo treinado.

CLASSIFICATION

Existem vários algoritmos para a tarefa de classificação:

K-NN

Árvore de Decisão

Naïve Bayes

Redes Bayesianas

Redes Neurais

Regressão Logística

Máquinas de Vetores de Suporte (SVM)

etc...

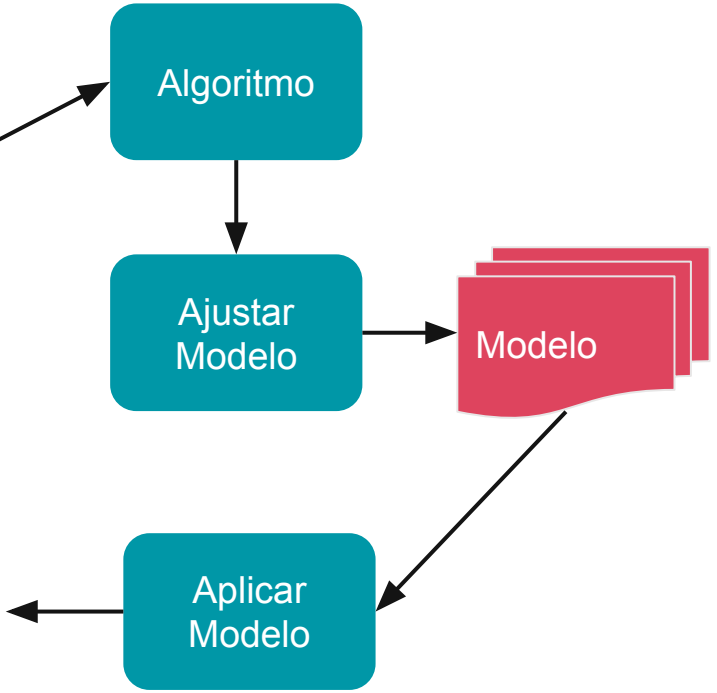
CLASSIFICATION

Treino e teste

Atr1	...	AtrN	Class
. . .			

Treino 2/3

Teste 1/3



CLASSIFICATION

Treino

Teste

CLASSIFICATION

In [205]:
```
import numpy as np
from sklearn.cross_validation import train_test_split
a = np.arange(10).reshape((5, 2))
b = [[0],[1],[1],[0],[0]]
b = np.array([1,0,1,0,0])
print a
print b
```

```
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
[1 0 1 0 0]
```

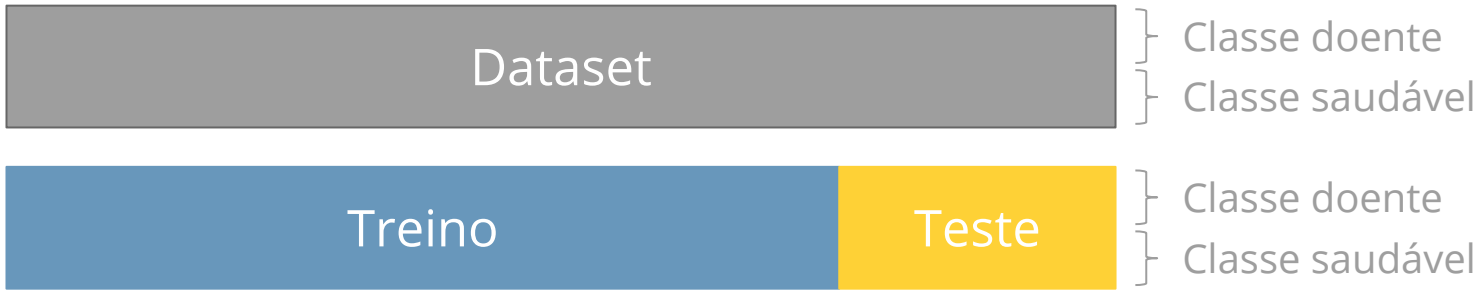
In [206]:
```
x_train, x_test, y_train, y_test = train_test_split(
    a, b, test_size=0.2)

print x_train
print y_train
print x_test
print y_test
```

```
[[6 7]
 [2 3]
 [4 5]
 [0 1]]
[0 0 1 1]
[[8 9]]
[0]
```

CLASSIFICATION

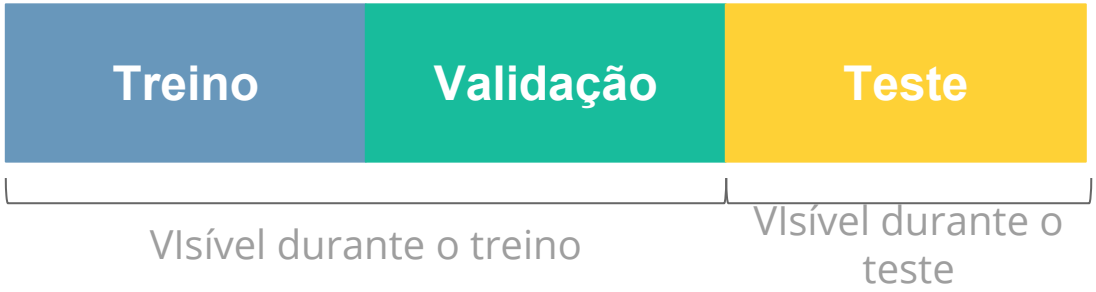
Devemos conservar a distribuição das classes presentes para cada um dos subconjuntos (treino e teste)



CLASSIFICATION

Para alguns modelos iterativos, normalmente é utilizado um conjunto adicional de validação.

EX: Algoritmos evolutivos, Redes Neurais

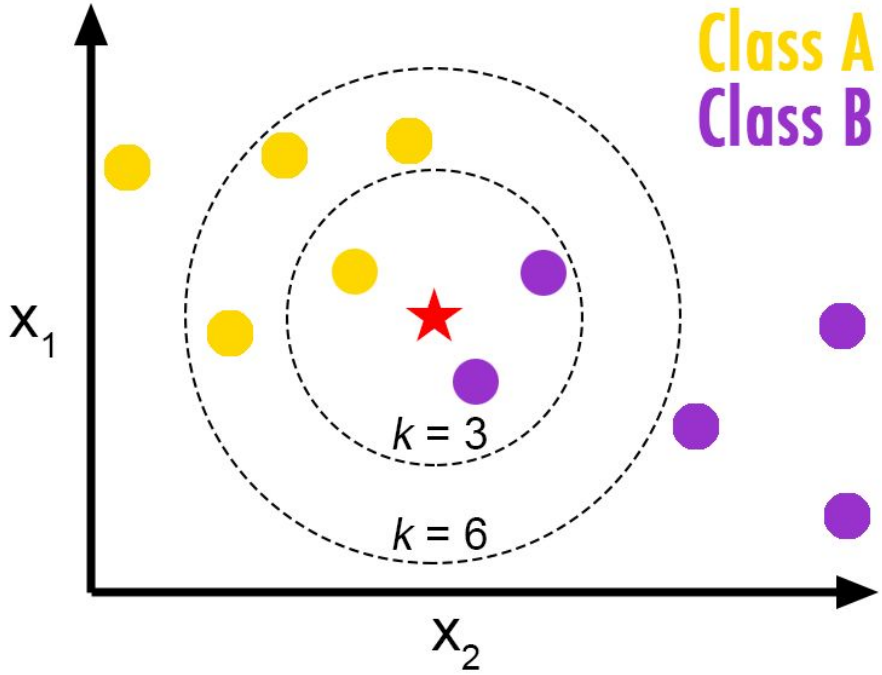


Lembre de SEMPRE estratificar os dados!

Faça o balanço entre classes nos conjuntos de treino e teste

Evita generalização pobre

KNN



kNN

3 necessidades:

- Precisamos de uma base de treinamento
- Medida de (dis)similaridade
- Valor de K (número de vizinhos)

Para classificar:

- Calcular a (dis)similaridade para todas as instâncias de treino
- Obter as K instâncias mais similares (k mais próximos)
- Classificar a instância nova na classe da maioria dos k vizinhos mais próximos

kNN

Pseudocódigo:

Para i=1 até m:

Encontre distâncias do objeto corrente para todos os demais

Ordene as distâncias de forma decrescente

Pegue os k primeiros itens da lista

Encontre a classe majoritária

Retorne.

kNN

Exemplo simples

```
data = load_iris()
data = pd.DataFrame(np.hstack((data['data'], data['target'].reshape(data['target'].shape[0], 1))),
                    columns=data['feature_names'] + ['class']
)

X = data[data.columns[:-1]]
Y = data[data.columns[-1]]
# print Y

t1 = np.array([6.1,2.1,4.7,1.3])

k = 3
dist_x = []

for idx, linha in X.iterrows():
    dist_x.append((idx, euclidean_distances(linha.reshape(1,-1), t1.reshape(1,-1))))

dist_sorted = sorted(dist_x, key=lambda d: d[1])[:k]

xx = [Y.iloc[idx] for idx, dist in dist_sorted]

# print xx

print sorted([(classe, xx.count(classe))
        for classe in Y.unique()], key=lambda a: a[1], reverse=True)[0][0]
```

Exemplo com treino e teste:

In [2]:

```
import numpy as np
import pandas as pd
from sklearn.cross_validation import train_test_split
from sklearn.metrics.pairwise import euclidean_distances

def getNeighbors(trainset, test, k):
    dist_test = []
    for id_train, train in trainset.iterrows():
        t = np.array(train[0:4])
        dist_test.append((id_train, euclidean_distances(t.reshape(1, -1), test[0:4].reshape(1, -1))))
    dist_sorted = sorted(dist_test, key=lambda d: d[1])[:k]
    xx = [trainset.ix[id_dist]['c'] for id_dist, dist in dist_sorted]
    return sorted([(classe, xx.count(classe))
                   for classe in trainset['c'].unique()], key=lambda a: a[1], reverse=True)[0][0]

def getAccuracy(testset, predictions):
    correct = 0
    for id_test, test in testset.iterrows():
        if test[-1] == predictions.ix[id_test][0]:
            correct += 1
    return (correct / float(len(testset))) * 100.0

def main():
    iris = pd.read_csv('iris.data', nrows=None, header=None, index_col=None)
    iris.reset_index()
    iris.columns = ['sl', 'sw', 'pl', 'pw', 'c']
    trainset, testset, = train_test_split(iris, test_size=0.2)
    predictions = []
    k = 7
    for id_test, test in testset.iterrows():
        predicted = getNeighbors(trainset, test, k)
        predictions.append(predicted)
        # print('> predicted=' + str(predicted) + ', actual=' + str(test[-1]))
    accuracy = getAccuracy(testset, pd.DataFrame(predictions, index=testset.index.values))
    print accuracy
main()
```

kNN

Sensível:

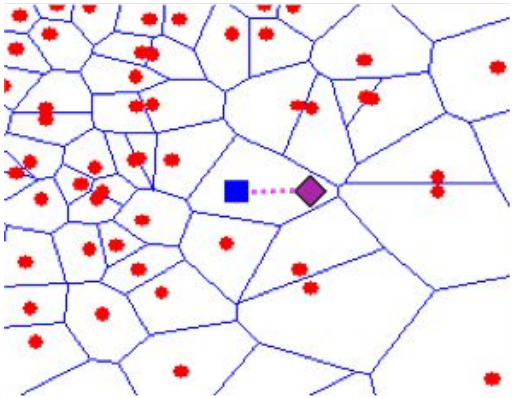
- A escolha de K
- A escolha da medida de (dis)similaridade
- **A ruído e outliers:**
  - Principalmente para K pequeno
  - Robusto com grande valor para K

Poder de classificação elevado:

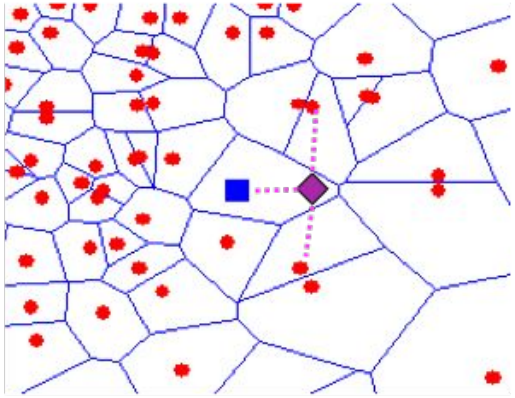
- Depende do nível do problema
- Depende da quantidade de K

Incrivelmente simples de implementar

kNN



k=1



k=3

Sensível a atributos irrelevantes

- Distorcem as distâncias
- Dimensionalidade alta (seleção de atributos, análise exploratória dos dados)

Exercício prático

Para o dataset iris:

Realizar uma classificação com k-NN;

Varie o tamanho dos dados para teste;

Varie k de 1 a 5;

Anote os resultados de acurácia

Compare os resultados

Exercício para entregar

Para o dataset zoo.csv:

Analisar o conjunto de dados, quais são os atributos e onde está a classe;

Realizar uma classificação com k-NN;

Varie o tamanho dos dados para teste;

Varie k de 1 a 5;

Anote os resultados de acurácia;

Qual melhor acurácia? Qual k do melhor modelo segunda acurácia?