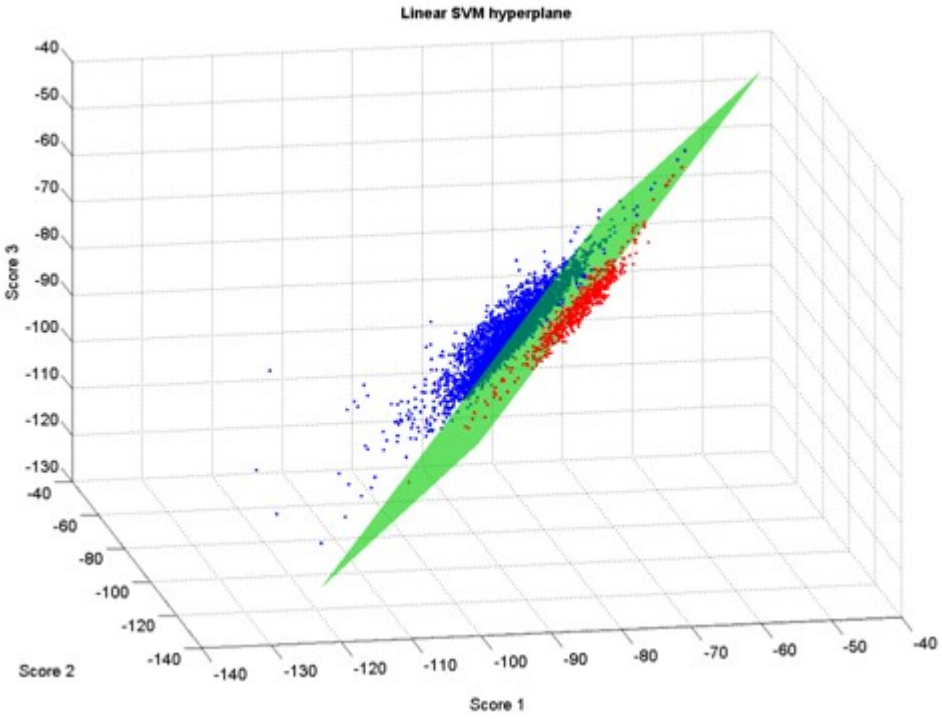
Instituto Federal do Sudeste de Minas Gerais, Campus Barbacena
Curso Superior de Tecnologia em Sistemas para Internet

Semana Acadêmica 2/2016

Introdução à aprendizagem de máquina (*machine learning*) em Python



Prof. Rafael José de Alencar Almeida
<rafael.alencar@ifsudestemg.edu.br>

Linguagem de programação Python

- www.python.org
- *Open-source*;
- Multiplataforma, sendo nativa no Linux e no Mac OS;
- Documentação abrangente e comunidade ativa;
- Biblioteca padrão completa (*batteries included*);
- Dinamicamente e fortemente tipada;
- Blocos delimitados por indentação;
- Paradigmas: estruturado, orientado a objetos e funcional (funções como objetos);
- Baseada em *bytecodes*;
- Implementações:
  - CPython: implementação padrão, em linguagem C;
  - Jython: implementação para a JVM;
  - IronPython: implementação para a plataforma .NET;
  - PyPy: implementação em Python (Rpython). Maior desempenho por usar JIT (*Just-In-Time compilation*).
- Propósito geral: programação *desktop*, *web*, *mobile* (SL4A), redes, *hacking*, *scripting*;
- Sintaxe clara e simples, com pequena curva de aprendizagem.

Hello World

```
# coding: utf-8
print 'Hello World!'
```

Arquivo hello.py

Execução: `python hello.py`

Modo interativo

- REPL (*read–eval–print loop*);
- Acessar: `python`
- Sair:
  - `exit()`
  - `Ctrl + D`
- `type()`
- `help()`
- `dir()`
- `len()`
- Operadores;
- Indexação;
- *Slices*;
- Formatação de *string;*
- Entrada de dados: `raw_input(str)`

Atividade

1. Use o modo interativo para calcular as raízes da equação: **$x^2 + 4x + 3 = 0$**

R: -1 e -3

Paradigmas

- **Estruturado:** por que não?

- **Orientado a Objetos:**

```
class Calculadora:
    def soma(self, x, y):
        return x + y

c = Calculadora()

c.soma(5, 10)
```

Exemplo de orientação a objetos

- **Funcional:**

```
def soma(x, y):
    return x + y

somar = soma

somar(10, 20)
```

Exemplo de função como objeto

```
produto = lambda x, y: x * y
produto(5, 10)


(lambda x, y: x * y)(5, 10)
```

Função anônima com Lambda

Atividade

1. Crie uma função que recebe um número e exibe seu cubo.

Principais estruturas de dados

- Tupla: `()` ou `tuple()`
- Lista: `[]` ou `list()`
- Conjunto: `set()`
- Dicionário (tabela hash, array associativo): `{}` ou `dict()`
- Matrizes como listas de listas:

```
m = [
    [0, 1, 0],
    [1, 1, 1],
    [0, 0, 1],
    [1, 0, 0],
]

>>> m[0][-1]
>>> m[2]
>>> m[-2]
>>> m[3]
```

Principais conversões

- `int()`
- `float()`
- `bin()`
- `ord()`
- `chr()`
- `str()`
- `abs()`

Controle de fluxo

```
if condicao1:
    codigo 1
elif condicao2:
    codigo2
else:
    codigo3
```

```
while True:
    x = raw_input('Digite 0 para sair: ')

    if x == '0':
        break
```

If e While

```
for iterador in iteravel:
    codigo
else:
    codigo_caso_for_termine_sem_break


for letra in 'String':
    print letra


for  in range(10):
    print n


for cont in xrange(100000):
    print cont
```

For

Arquivo Python

- Extensão:
  - .py
  - .pyc
  - .pyo
- Execução: `python arquivo.py`
- Script Linux:
  - `#!/usr/bin/env python`
  - `chmod +x arquivo.py`
  - `./arquivo.py`
- Usar padrão Unicode:
  - `# coding: utf-8`

- Cometário:

```
# Uma linha

'''
   Múltiplas linhas
'''
```

Funções *built-in* para processamento de dados

- `any()`
- `all()`
- `sum()`
- `min()`
- `max()`
- `reversed()`
- `sorted()`

Processando dados com programação funcional

- **`map`**`(lambda x: x + 1, [1, 2, 3])`
- **`reduce`**`(lambda x, y: x + y, [1, 2, 3])`
- **`filter`**`(lambda x: x > 2, [1, 2, 3])`
- **`zip`**`([1, 2, 3], ['a', 'b', 'c'])`

Ordenação

```
l = [('b', 3), ('a', 1), ('c', 2)]
l.sort()
```

Ordenação avançada

```
'''
   Cria uma função que recebe um item da lista e retorna o elemento
   da tupla que será utilizado como critério de ordenação
'''
def criterio(item):
    return item[1]

l.sort(key=criterio)
```

Processamento de dados com *list comprehensions*

Na matemática:

```
C = {1, 2, 3, 4, 5, 6}
{ x | x ∈ C, x > 2 }
```

```
# Em Python
c = (1, 2, 3, 4, 5, 6)
[x for x in c if x > 2]
```

Exemplos:

```
[v + 1 for v in [1, 2, 3, 4]]

[(n ** 2, n ** 3) for n in range(3)]

[ord(l) for l in 'Palavra']

['<li>{0}</li>'.format(e) for e in ('MG', 'SP', 'RJ')]
```

Atividade

1. Com base na classe Pessoa, definida ao lado, crie uma lista de instâncias da mesma, e uma *list comprehension* que processe a lista de pessoa, retornando o nome apenas das mulheres.

```
class Pessoa:
  def __init__(self, nome='', sexo='M'):
      self.nome = nome
      self.sexo = sexo
```

Manipulação de arquivos

```
f = open('/etc/hostname', 'r')
nome_maquina = f.read()
f.close()

print 'Nome do computador: ', nome_maquina
```

Leitura de arquivo

```
f = open('texto.txt', 'w')
f.write('Oi')
f.close()
```

Escrita em arquivo

```
with open('texto2.txt', 'w') as f:
    f.write('Texto')
```

Gerenciador de contexto

Processamento de strings

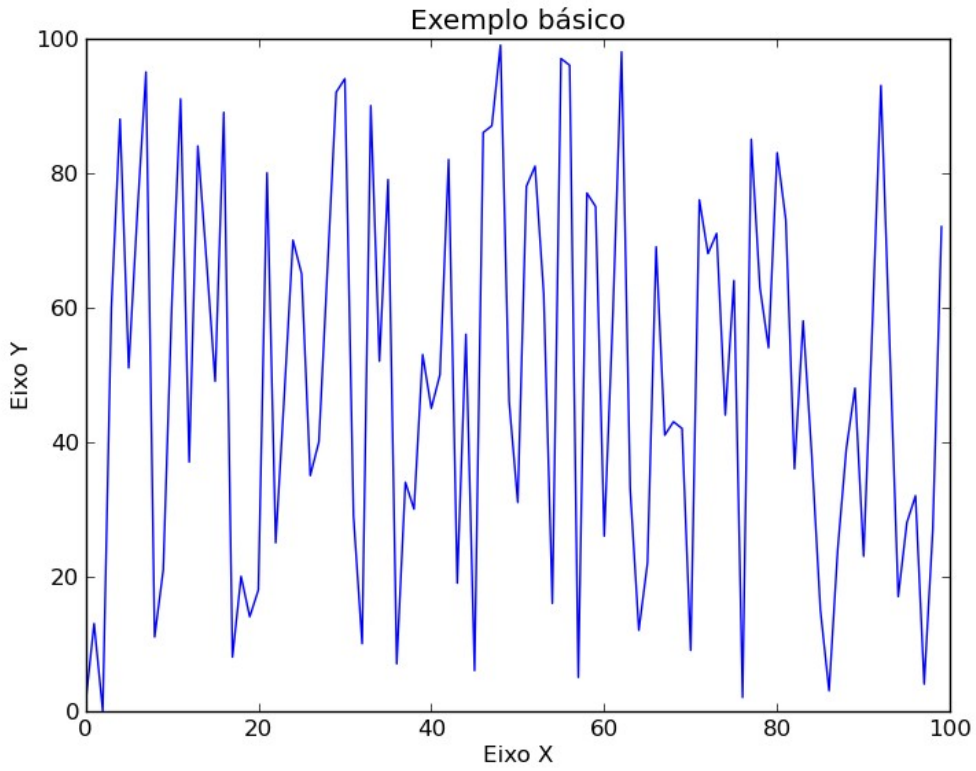
- `join()`
- `split()`
- `strip()`
- `count()`
- `find()`
- `index()`
- `replace()`
- `startswith()`
- `endswith()`

Visualização de dados

```
sudo apt-get install python-matplotlib
```

```
# coding: utf-8

import matplotlib.pyplot as plt
import random

plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title(u'Exemplo básico')

lista_dados = range(100)
random.shuffle(lista_dados)

plt.plot(lista_dados)
plt.show()
```

Produção de gráfico básico



Aprendizagem de máquina (*machine learning*)

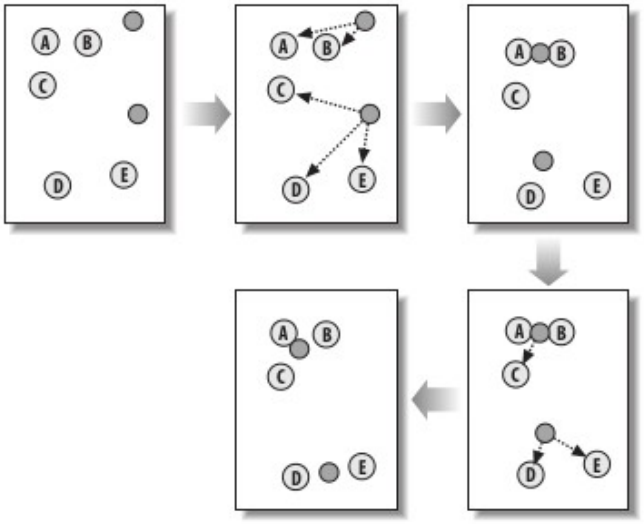
- Sub-área da Inteligência Artificial que consiste no desenvolvimento de algoritmos capazes de "aprender" padrões a partir de conjuntos de dados de exemplos ou observações.

- Aplicações:
  - Detecção de fraudes em cartões de crédito
  - Filtros anti-spam
  - Sugestão de filmes (Netflix), *playlists* (Spotify), amigos/postagens (Facebook)
  - Detecção de terroristas (NSA)
  - Previsão financeira
  - Categorização de resultados contextuais (Google)
  - Marketing
  - Carros autônomos
  - Detecção facial

- Polêmicas:
  - Podemos ser prejudicados por uma previsão? (Planos de saúde)
  - E quando um algoritmo erra a classificação? (Terroristas de panela de pressão)
  - Algoritmos podem nos conhecer melhor do que nós mesmos. (Cupons de gravidez)
  - Até que ponto somos isolados da diversidade de informação? (Bolha dos filtros)

- **Aprendizado supervisionado:**
    - Utilizado para classificação (prever uma categoria) e regressão (prever um valor).
    - O algoritmo é “treinado” com um conjunto de dados inicial, onde cada entrada já possui a resposta esperada para saída.
    - O algoritmo busca padrões e relacionamentos nos dados, de modo a criar um modelo de previsão capaz de relacionar os dados de entrada à saída esperada, generalizando o aprendizado para usos futuros (em entradas de dados diferentes).
    - Exemplo: algoritmos de detecção de rostos são inicialmente treinados com milhares de imagens, já sabendo previamente quais possuem um rosto e quais não possuem. Uma vez assimilados os padrões durante o treinamento, o algoritmo estará apto a receber um imagem que não faz parte da base de treino, e classificar se a mesma possui ou não um rosto.
    - Principais algoritmos: SVM, Redes Neurais, Filtro Bayesiano, Árvores de Decisão.

- **Aprendizado não supervisionado:**
    - O algoritmo não é treinado com um conjunto inicial de dados.
    - Não realiza classificação ou regressão. Organiza os dados e descreve sua estrutura a partir de seus atributos.
    - Exemplo: algoritmo que analisa conteúdos textuais e gera *hashtags* automáticas para os mesmos.
    - Principais algoritmos: Clusterização, descoberta automática de palavras-chave, resumo automático de textos, decomposição de sinais.



Clusterização baseada em centroides (Segaran, 2008).

Kaggle

- Plataforma com desafios de aprendizado de máquina: www.kaggle.com
- Titanic – Machine Learning from Disaster: https://www.kaggle.com/c/titanic

Em 15 de abril de 1912, na sua viagem inaugural, o navio Titanic afundou após colidir com um iceberg, matando 1502 de seus 2224 passageiros. Uma das principais razões para o grande número de mortos foi que não havia barcos salva-vidas para todos.

Em relação aos sobreviventes, apesar de haver o elemento sorte envolvido, algumas pessoas tiveram mais chances de sobreviver do que outras, como mulheres, crianças e os passageiros das classes mais altas.

A partir de um conjunto de dados de treinamento (`train.csv`) com informações sobre 891 passageiros (inclusive se sobreviveram ou não), treine um algoritmo capaz de prever, dadas as informações de um passageiro, se o mesmo sobreviveu ou não.

Teste seu modelo de dados com os passageiros do arquivo `test.csv`, que possui as mesmas informações do arquivo de treinamento, exceto o campo que determina se o passageiro sobreviveu.

Gere um arquivo de saída em CSV, contendo as colunas `PassengerId` e `Survived`, associando o ID de cada linha do arquivo de testes à resposta de seu algoritmo. Envie seus resultados para o desafio no Kaggle, e descubra sua taxa de acerto.

Conjunto de dados (*dataset*)

- https://www.kaggle.com/c/titanic/data
- Baixar os arquivos a seguir:
  - train.csv
  - test.csv

- Estrutura dos dados:

Atributo	Descrição	Exemplo
`PassengerId`	Identificador do passageiro	`1`
`Survived`	Informa se o passageiro sobreviveu (0 = Não; 1 = Sim)	`0`
`Pclass`	Classe do passageiro: (1 = primeira; 2 = segunda; 3 = terceira)	`3`
`Name`	Nome do passageiro, com pronome de tratamento	`Lam, Mr. Len`
`Sex`	Sexo	`female`
`Age`	Idade	`26`
`SibSp`	Número de irmãos/cônjuges a bordo	`0`
`Parch`	Número de pais/filhos a bordo	`1`
`Ticket`	Número do bilhete de passagem	`113803`
`Fare`	Preço da passagem	`53.1`
`Cabin`	Cabine	`C123`
`Embarked`	Porto onde embarcou: (C = Cherbourg; Q = Queenstown; S = Southampton)	`C`

Lendo um arquivo CSV

```
import csv

with open('dados/train.csv') as csv_treino:
    dados = csv.reader(csv_treino)

    # Pula o cabeçalho
    dados.next()

    for linha in dados:
        print linha
```

Escrevendo um arquivo CSV

```
import csv

with open('arquivo.csv', 'w') as f:
    csv_saida = csv.writer(f, delimiter=',')

    cabecalho = [['PassengerId', 'Survived']]

    dados = [
        [1, 1],
        [2, 0],
        [3, 1]
    ]

    csv_saida.writerows(cabecalho + dados)
```

Dados faltantes

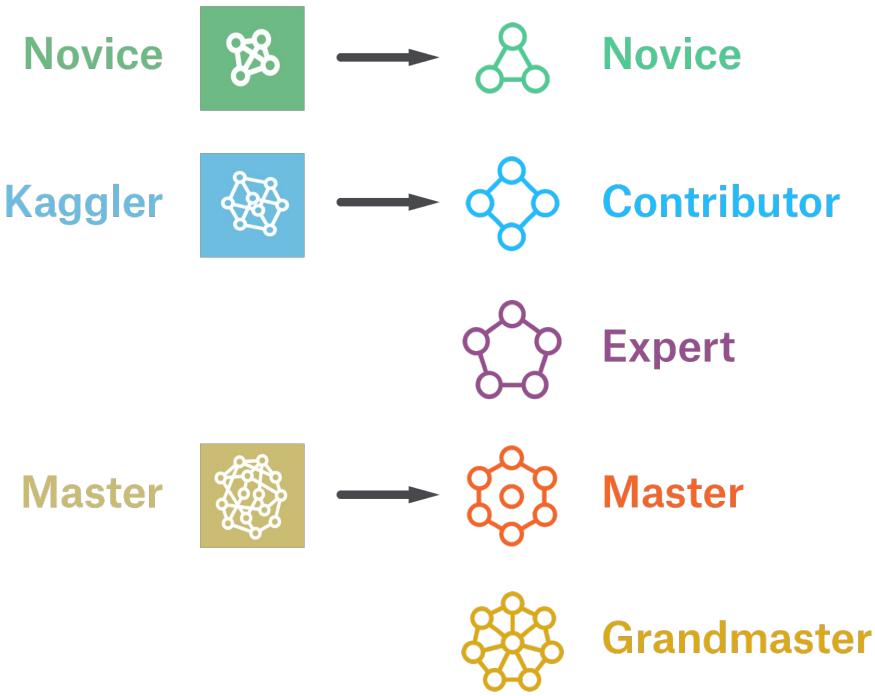
- Alguns campos importantes, como `Age` (idade), estão com dados incompletos. Inicialmente iremos definir as idades faltantes como a média das idades dos demais passageiros.
- Uma alternativa mais sofisticada (e precisa) seria definir as idades a partir do pronome de tratamento do campo `Name`.

```
import csv
import numpy as np

idades = []

with open('dados/train.csv') as csv_treino:
    dados = csv.reader(csv_treino)

    # Pula o cabeçalho
    dados.next()

    for linha in dados:
        idade = linha[5]

        if idade != '':
            idades.append(float(idade))

    print np.mean(idades) # 29.6991176471
```

Gerando um arquivo de saída para avaliação no Kaggle

```
import csv
import random

saida = []

with open('dados/test.csv') as csv_teste:
    dados = csv.reader(csv_teste)
    dados.next()

    for linha in dados:
        # Define aleatoriamente se o passageiro sobreviveu
        sobreviveu = random.choice([0, 1])
        saida.append([linha[0], sobreviveu])

with open('saida.csv', 'w') as f:
    csv_saida = csv.writer(f, delimiter=',')

    cabecalho = [['PassengerId', 'Survived']]
    csv_saida.writerows(cabecalho + saida)
```

- Submissão da resposta: https://www.kaggle.com/c/titanic/submissions/attach
- Taxa de acerto (irá variar): **0.49321** (onde 1.0 equivale a 100% de acerto)
- Repare que um algoritmo realmente capaz de prever os sobreviventes deverá possuir uma taxa de acerto significativamente maior do que 50%.



Aprendizado de máquina com SVM

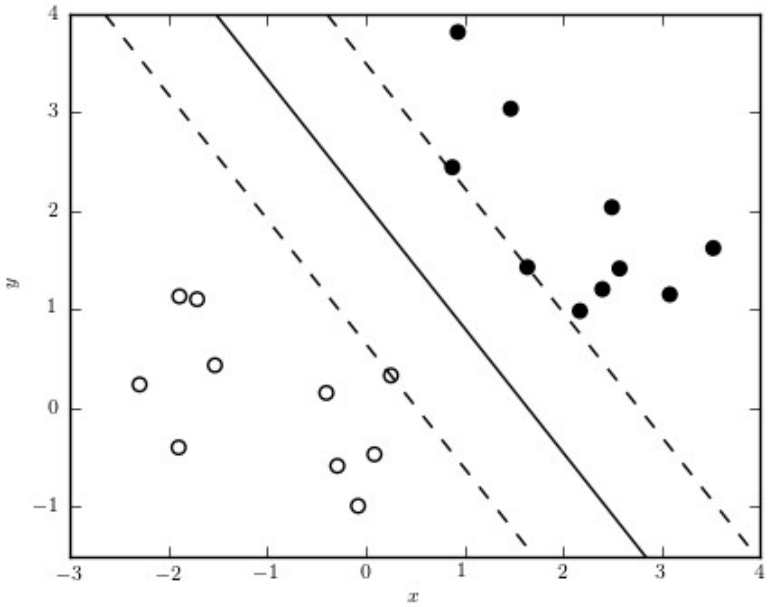
- Intuitivamente podemos perceber que uma série de informações dos dados de treinamento podem indicar se o passageiro sobreviveu ou não, como idade, sexo, classe,número da cabine, etc.

- Entretanto, apenas olhando os dados, não podemos determinar até que ponto cada variável ou combinação de variáveis influencia na chance de sobrevivência dos passageiros.

- Para buscar padrões e relacionamentos profundos nos dados de forma automatizada, treinando um modelo de classificação, utilizaremos o algoritmo de aprendizado de máquina supervisionado SVM (*Support Vector Machine* – Máquina de Vetores de Suporte).

- Uma SVM treinada recebe entradas numéricas (*features*), a partir das quais tenta prever em quais categorias ou classes elas se encaixam. No nosso desafio, as entradas para um passageiro poderia ser o vetor `[3, 25, 2]`, representando numericamente os valores `[classe, idade, sexo]`. A saída seria `0` (morreu) ou `1` (sobreviveu).

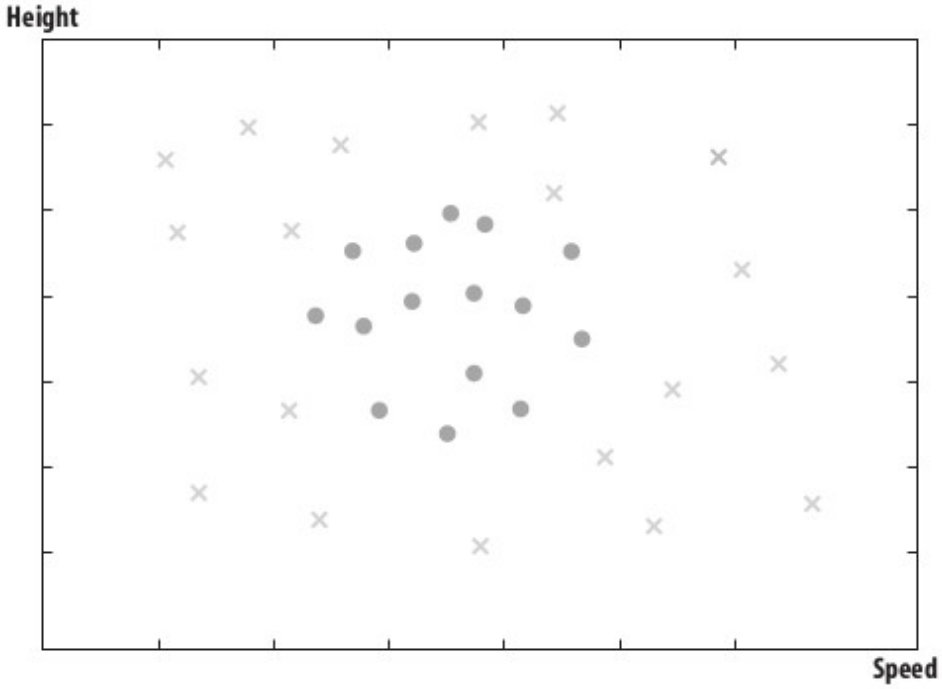
- SVM é um algoritmo de **classificação linear**. Dessa forma, ele supõe que as categorias em que os dados podem ser divididos (no nosso problema, sobrevivente ou morto) podem ser divididas por uma linha reta, chamada hiperplano de margem máxima.



- A linha divisória é calculada de modo dividir o melhor possível as categorias durante o treinamento. Uma vez definida, a SVM classifica uma nova entrada apenas definindo em qual lado da linha a mesma se posiciona.

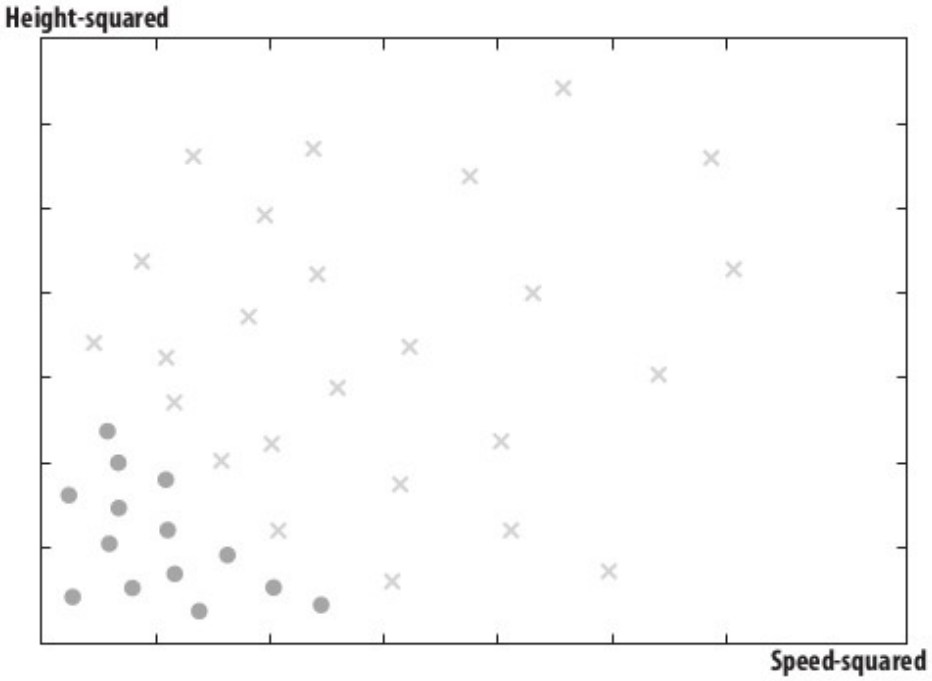
- Os pontos nas margens (linhas pontilhadas) são utilizados para calcular a linha divisora, e são chamados de vetores de suporte.

- Entretanto, um problema de classificação nem sempre é linearmente separável – ou seja, não é possível calcular a linha divisora de categorias:



Dados que não podem ser separados linearmente (Segaran, 2008).

- Para resolver esse problema, as máquinas de vetores de suporte utilizam uma técnica de transformação polinomial chamada truque do núcleo (*kernel trick*), onde os dados são transformados para um espaço dimensional diferente, até que seja possível traçar uma linha separadora (este processo ocorre durante a etapa de treinamento). No exemplo, os dados foram apenas elevados ao quadrado:



Transformação dos dados de modo a poderem ser separados linearmente (Segaran, 2008).

Máquina de vetores de suporte em Python

- Instalar o módulo **scikit-learn**, que implementa algoritmos de aprendizagem de máquina em Python.
- Suas dependências são: NumPy (cálculo numérico), SciPy (módulos científicos) e matplotlib (plotagem de dados)
- No Linux: `sudo apt-get install python-sklearn`

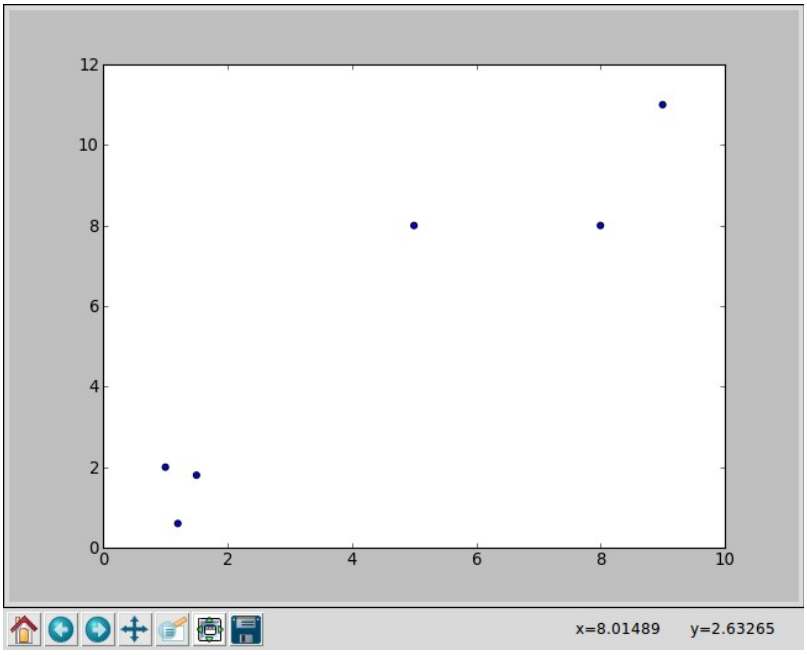
Exemplo básico

- Dividir entradas numéricas com duas *features* cada (x e y) em categorias (0 e 1).

```
x = [1, 5, 1.5, 8, 1.2, 9]
y = [2, 8, 1.8, 8, 0.6, 11]
```

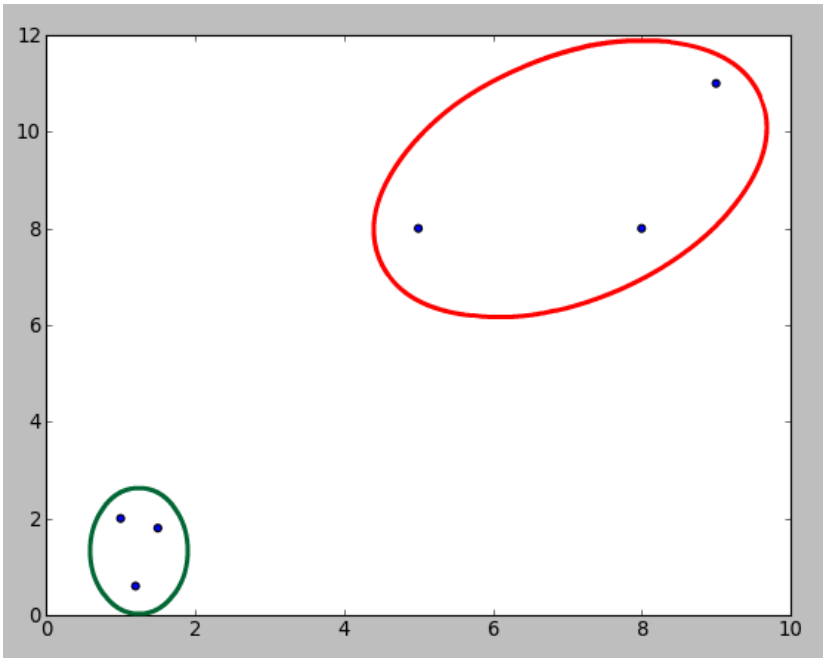
- Visualizando graficamente as entradas:

```
import matplotlib.pyplot as plt

x = [1, 5, 1.5, 8, 1.2, 9]
y = [2, 8, 1.8, 8, 0.6, 11]

plt.scatter(x,y)
plt.show()
```



- Definindo as categorias para cada entrada:

```
c = [0, 1, 0, 1, 0, 1]
```



- Treinando e testando a SVM:

```
import numpy as np
from sklearn import svm
import matplotlib.pyplot as plt

X = np.array(
    [
        [1, 2],
        [5, 8],
        [1.5, 1.8],
        [8, 8],
        [1.2, 0.6],
        [9, 11]
    ]
)

c = [0, 1, 0, 1, 0, 1]

# Cria a SVM
maquina = svm.SVC(kernel='linear', C=1.0)

# Treina a SVM
maquina.fit(X, c)

# Classifica uma nova entrada
print maquina.predict([[7, 11]])
```

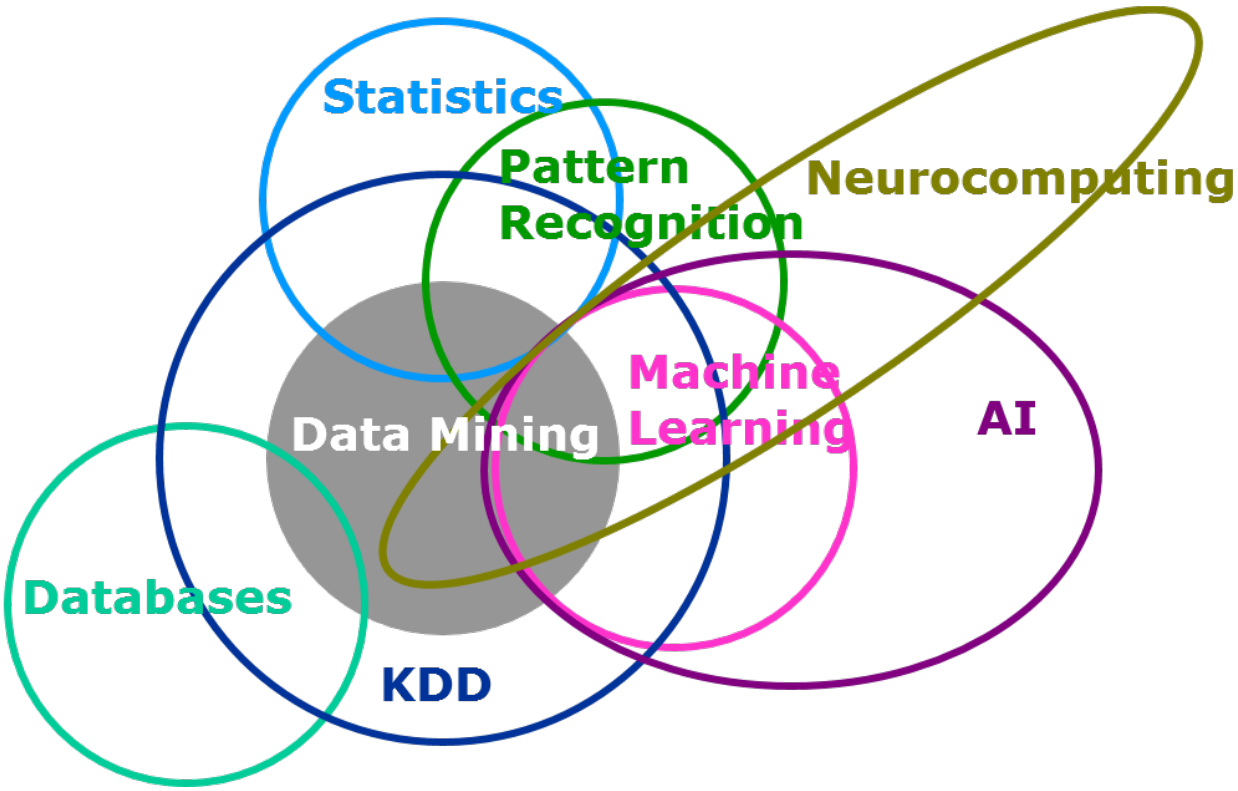
Prevendo os sobreviventes do Titanic com SVM

```
import csv
import numpy as np
from sklearn import svm

caracteristicas = []
categorias = []

with open('dados/train.csv') as csv_treino:
    dados = csv.reader(csv_treino)
    dados.next()

    for l in dados:
        classe = int(l[2])

        # Define o valor 1 para masculino e 2 para feminino
        sexo = 1 if l[4] == 'male' else 2

        if(l[5]):
            idade = float(l[5])
        else:
            idade = 30 # Média aproximada das idades

        caracteristicas.append([classe, sexo, idade])
        categorias.append(int(l[1]))

# Converte os dados de treinamento para arrays do numpy
np_caracteristicas = np.array(caracteristicas)
np_categorias = np.array(categorias)

# Treina a SVM
maquina = svm.SVC(kernel='linear', C=1.0)
maquina.fit(np_caracteristicas, np_categorias)

# Categoriza os resultados para o arquivo de teste
saida = []
with open('dados/test.csv') as csv_teste:
    dados = csv.reader(csv_teste)
    dados.next()

    for l in dados:
        classe = int(l[1])
        sexo = 1 if l[3] == 'male' else 2
        if(l[4]):
            idade = float(l[4])
        else:
            idade = 30
        saida.append([l[0], int(maquina.predict([[classe, sexo, idade]]))])

# Gera CSV de saída
with open('saida.csv', 'w') as f:
    csv_saida = csv.writer(f, delimiter=',')
    dados = [['PassengerId', 'Survived']] + saida
    csv_saida.writerows(dados)
```

Atividade final

Melhore o algoritmo de modo a obter uma taxa de acerto superior a **0.76555**. Experimente alterar o cálculo das idades não definidas e utilizar outras *features* no treinamento, como cabine, ponto de embarque, ticket e relações familiares.

Continua...



Referências

KINSLEY, Harrison. **Linear SVC Machine learning SVM example with Python**. PythonProgramming.net. 2016.

LERNER, Reuven M. **Geek Guide: Machine Learning with Python**. Linux Journal. 2016.

PYTHON.ORG. **Python 2.7.12 documentation**. Python Software Foundation. 2016.

SEGARAN, Toby. **Programando a inteligência coletiva**. Alta Books. 2008.