Máquinas de Vetores Suporte

Prof. Dr. Geraldo Braz Junior

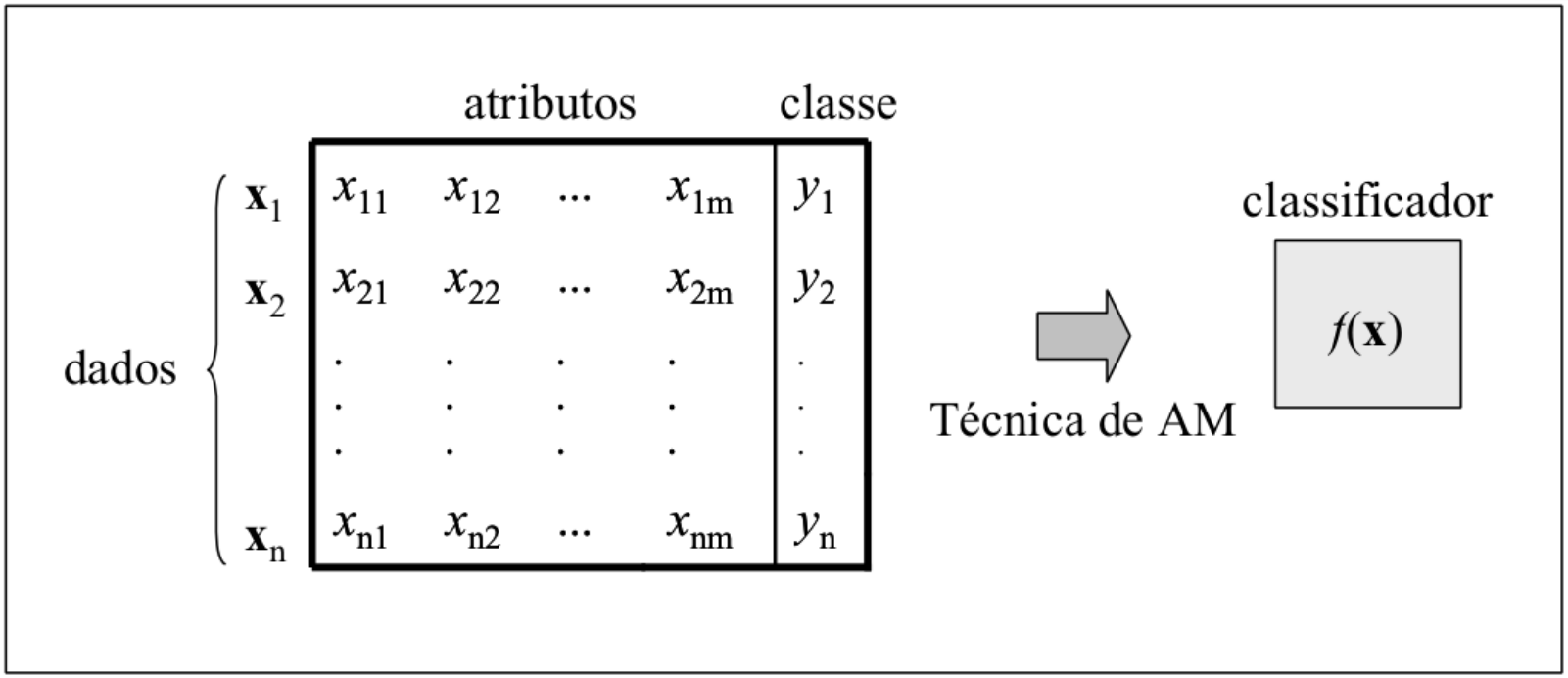
Introdução

- Máquinas de Vetores de Suporte (MVS)
  - Criada por (VAPNIK, 1998) é um método de aprendizagem **supervisionado** usado para estimar uma função que classifique dados de entrada em duas classes (normalmente, mas é multiclasses)
  - O objetivo do treinamento através de MVS é a obtenção de hiperplanos que dividam as amostras de tal maneira que sejam otimizados os limites de generalização.

Introdução

- Os algoritmos de treinamento das MVS possuem forte influência da teoria de otimização e de **aprendizagem estatística**
- Resultados normalmente melhores comparado a redes neurais
  - Bons exemplos de aplicações em processamento de imagens, bioinformática, categorização de textos, análise se solo .....

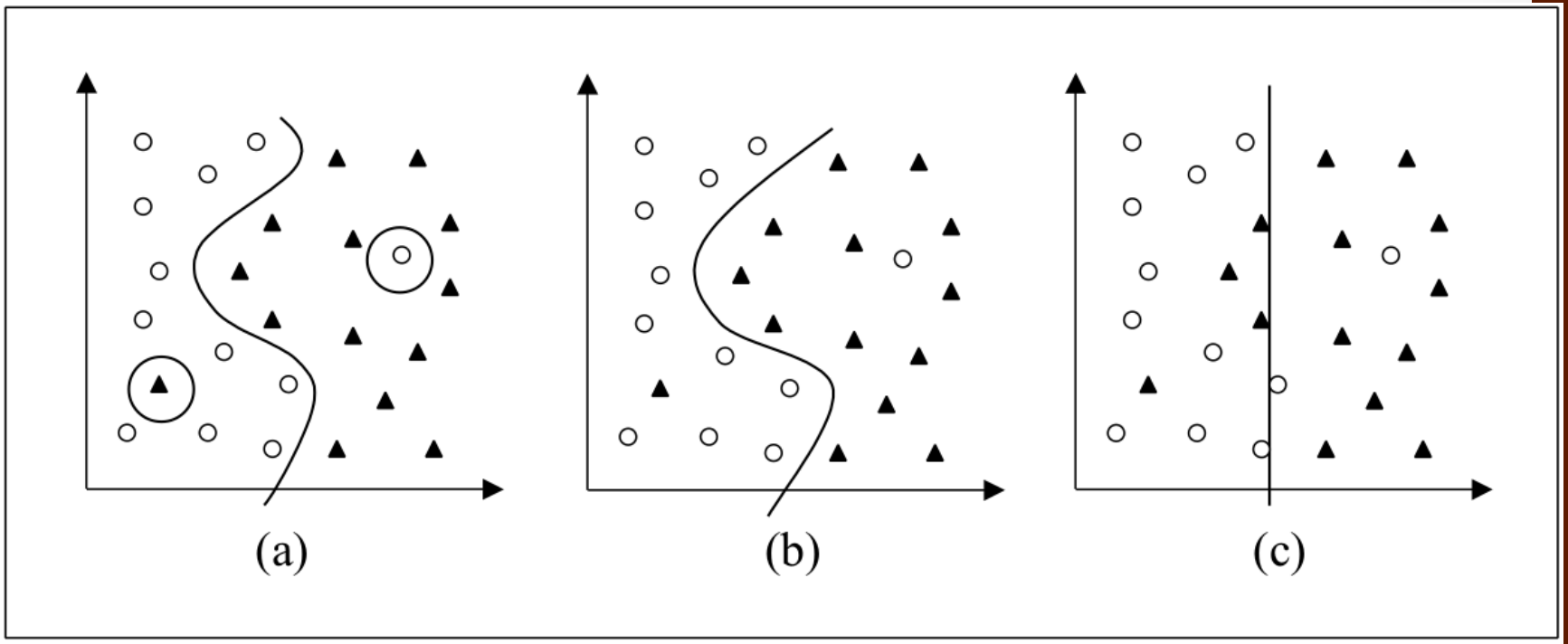
Representação AM



Teoria de Aprendizado Estatístico

- Seja f um classificador e **F** o conjunto de todos os classificadores que um determinado algoritmo de **AM** pode gerar.
- Esse algoritmo, durante o processo de aprendizado, utiliza um conjunto de treinamento **T**, composto de **n pares (xi, yi)**, para gerar um classificador particular $f' \in \mathbf{F}$.

Teoria de Aprendizado Estatístico



Funções representadas pelas curvas de decisão
Conjunto de treinamento binário

Qual classificador *f'* escolher?

- Sendo todos dados do domínio gerados de forma independente e identicamente distribuídos (i.i.d)
  - O risco empírico de um classificador *f* pode ser calculado como:

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} c(f(\mathbf{x}_i), y_i)$$

  - Principio de Minimização de Risco Empírico
  - Nem sempre leva a um bom classificador

Limite do Risco Esperado

- Relaciona o Risco Empírico com o Risco Esperado
  - Garantido com probabilidade 1 – teta onde teta ϵ [0,1]

$$R(f) \leqslant R_{emp}(f) + \sqrt{\frac{h\left(\ln\left(2n/h\right)+1\right) - \ln\left(\theta/4\right)}{n}}$$

  - h denota a dimensão Vapnik-Chervonenkis (VC)
  - n qtd exemplos no treinamento

Dimensão VC Vapnik-Chervonenkis

- A dimensão VC h mede a capacidade do conjunto de funções F.
- Quanto maior o seu valor, mais complexas são as funções de classificação que podem ser induzidas a partir de F.
- Dado um problema de classificação binário, essa dimensão é definida como o número máximo de exemplos que podem ser particionados em duas classes pelas funções contidas em F, para todas as possíveis combinações binárias desses dados.

Contribuição Risco Esperado

- Importância de se controlar a capacidade do conjunto de funções F do qual o classificador é extraído
  - Bom classificador minimiza o Risco Empírico e que possua a uma classe de funções F com baixa dimensão VC h
  - **Minimização de Risco Estrutural**

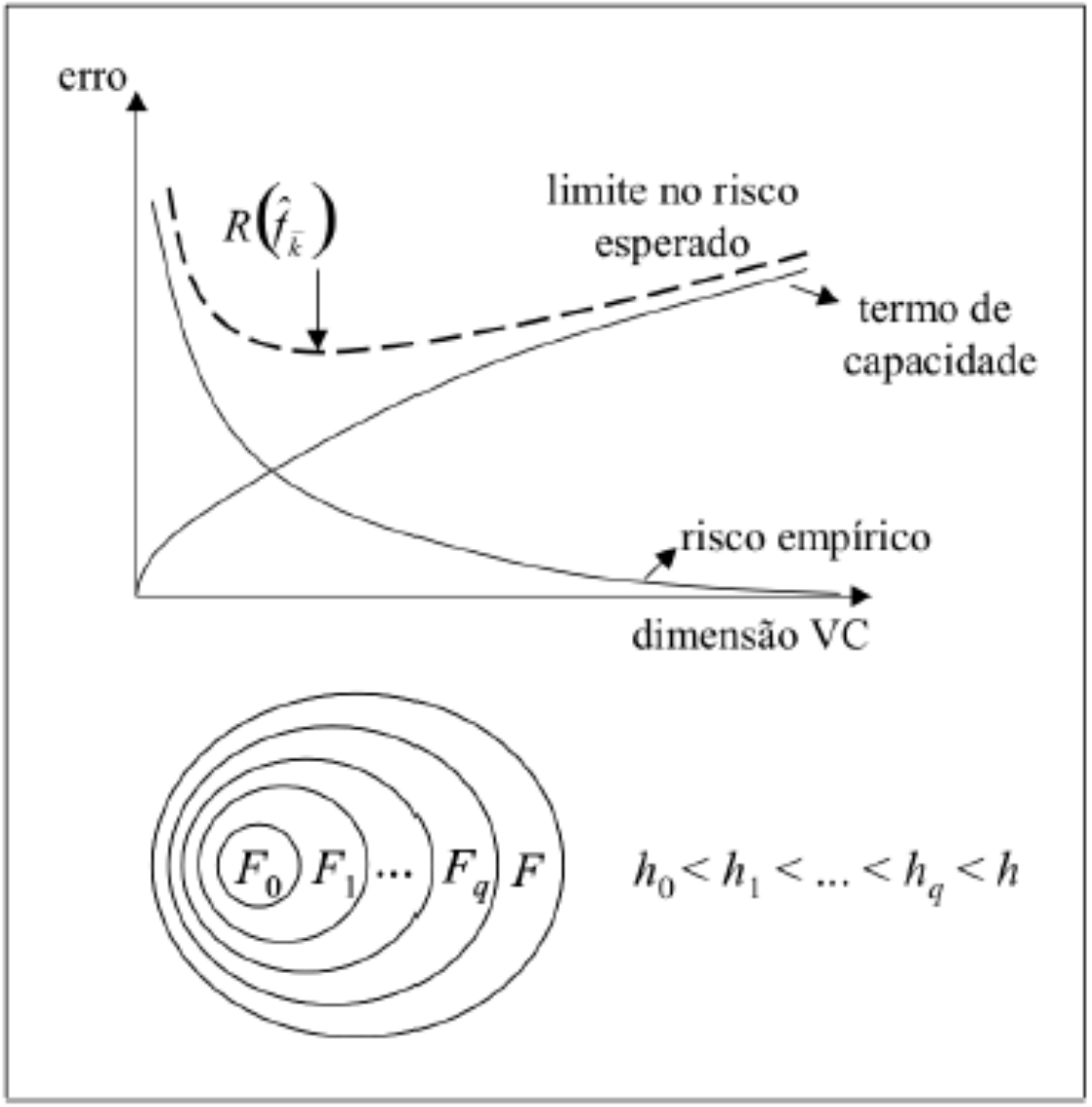
Como escolher

- Divide-se inicialmente F em subconjuntos de funções com dimensão VC crescente

Principio da Minimização do Risco Estrutural

- **Introduzir** uma estrutura (função de classificação) em F
- Minimiza-se então o limite sobre as estruturas introduzidas.
- Quanto **maior a capacidade** menor **o risco empírico** pela complexidade das funções

Principio de Minimização de Risco Estrutural



Na prática

- Útil na definição do procedimento de minimização de risco estrutural
- Mas …
  - Computar a dimensão VC de uma classe de funções geralmente não é uma tarefa trivial.
  - Soma-se a isso o fato de que o valor de h poder ser desconhecido ou infinito

Para funções Lineares

- Relacionamento do risco esperado com a **margem** (distância da fronteira de decisão induzida)

$$\varrho\left(f(\mathbf{x}_i), y_i\right) = y_i f\left(\mathbf{x}_i\right)$$

Erro Marginal

- Assim o erro marginal pode ser calculador por:

$$R_{\rho}(f) = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i f\left(\mathbf{x}_i\right) < \rho\right)$$

- Onde I(q) = 1 se q é verdadeiro e I(q) = 0 se q é falso.
- Um *p* elevado implica uma menor termo de capacidade

Hiperplano Ótimo

- Como conclusão tem-se que:
  - deve-se buscar um hiperplano que tenha margem elevada
  - e cometa poucos erros marginais, minimizando assim o erro sobre os dados de teste e de treinamento, respectivamente.
- Esse hiperplano é denominado ótimo

MVS Lineares (Rígidas)

- Definem fronteiras lineares a partir de dados linearmente separáveis
  - Hiperplano linear é definido por

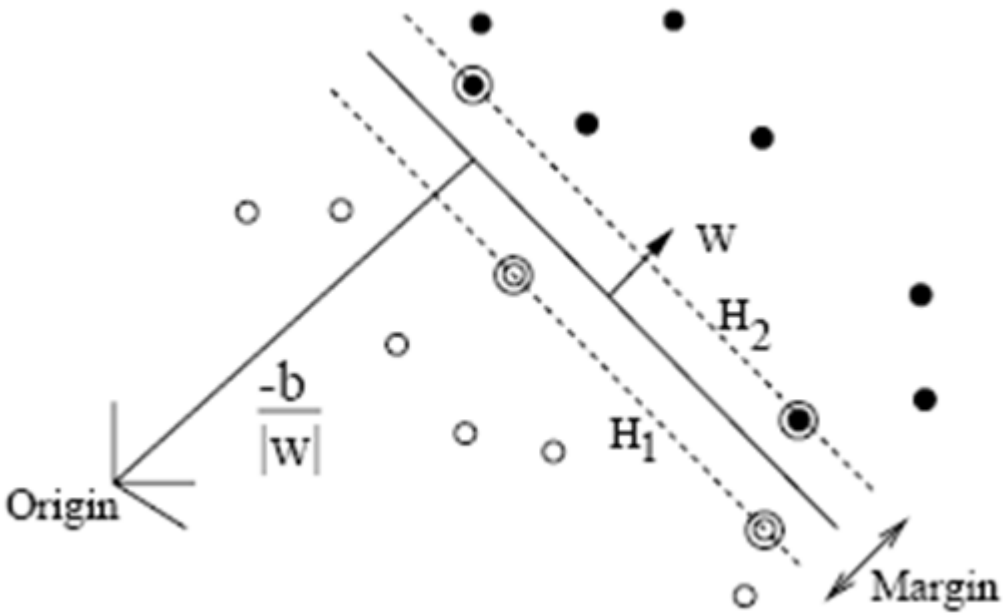
$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0$$

  - Onde em que w. x é o produto escalar entre os **vetores w e x**
  - w $\boldsymbol{\epsilon}$ X é o vetor normal ao hiperplano descrito
  - $b/\ ||w||$ corresponde à distância do hiperplano em relação à origem

MVS Lineares (Rígidas)

- Função g(x) divide o espaço X em duas regiões

$$g(\mathbf{x}) = \operatorname{sgn}(f(\mathbf{x})) = \begin{cases} +1 \text{ se } \mathbf{w}\cdot\mathbf{x}+b>0 \\ -1 \text{ se } \mathbf{w}\cdot\mathbf{x}+b<0 \end{cases}$$



Função Objetivo

- Logo para se obter maximização da margem, deve-se minimizar a norma de w através de:

$$\underset{\mathbf{w},b}{\text{Minimizar}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$y_i\left(\mathbf{w}\cdot\mathbf{x}_i+b\right)-1\geqslant 0,\quad \forall i=1,\ldots,n$$

MVS com Margem Suave

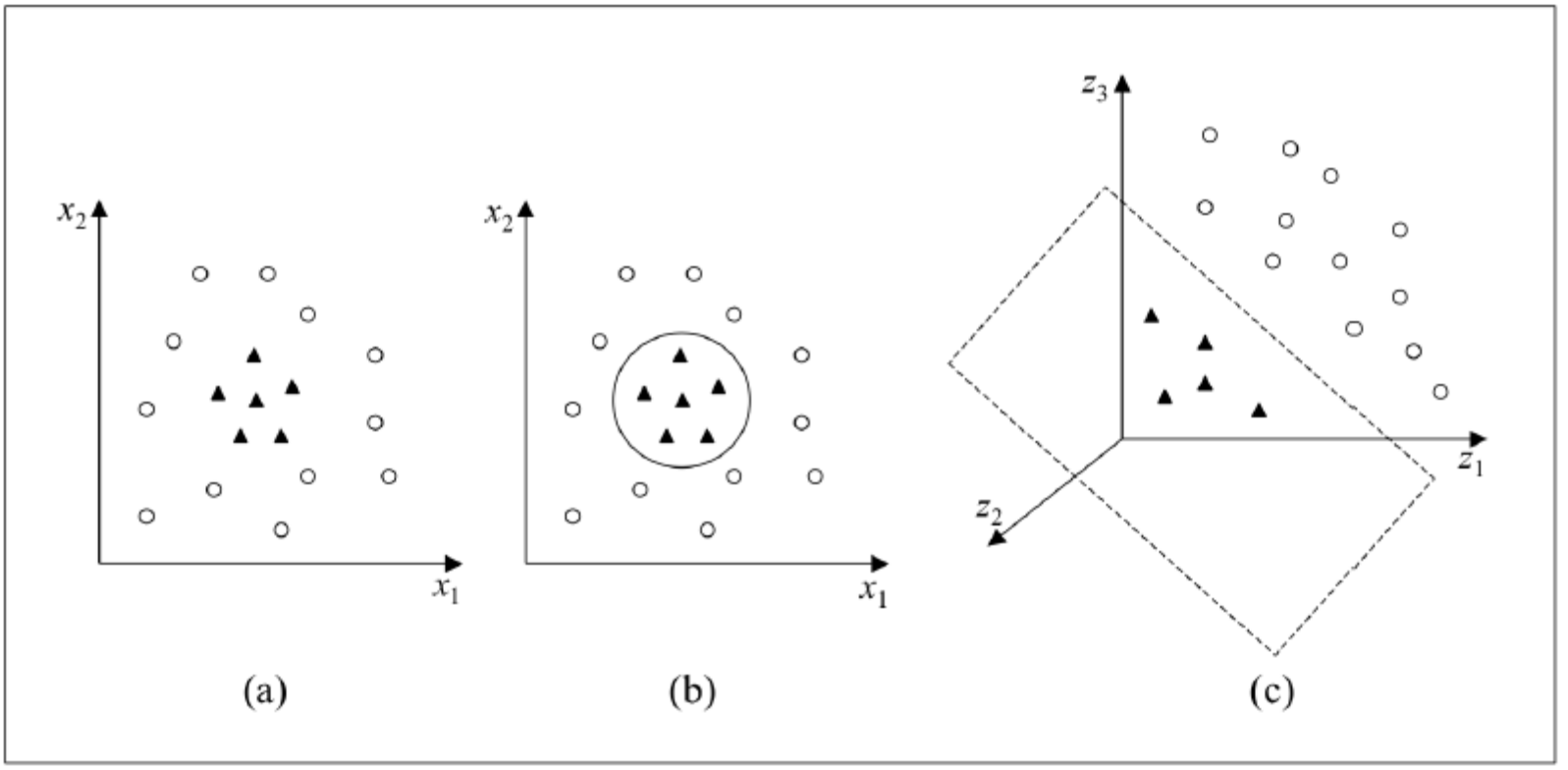
- Presença de ruídos e *outliers*
- Sujeito a restrição:

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \quad \forall i = 1, \ldots, n$$

- Nova função objetivo:

$$\underset{\mathbf{w}, b, \boldsymbol{\xi}}{\text{Minimizar}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^{n} \xi_i \right)$$

MVS não Linear



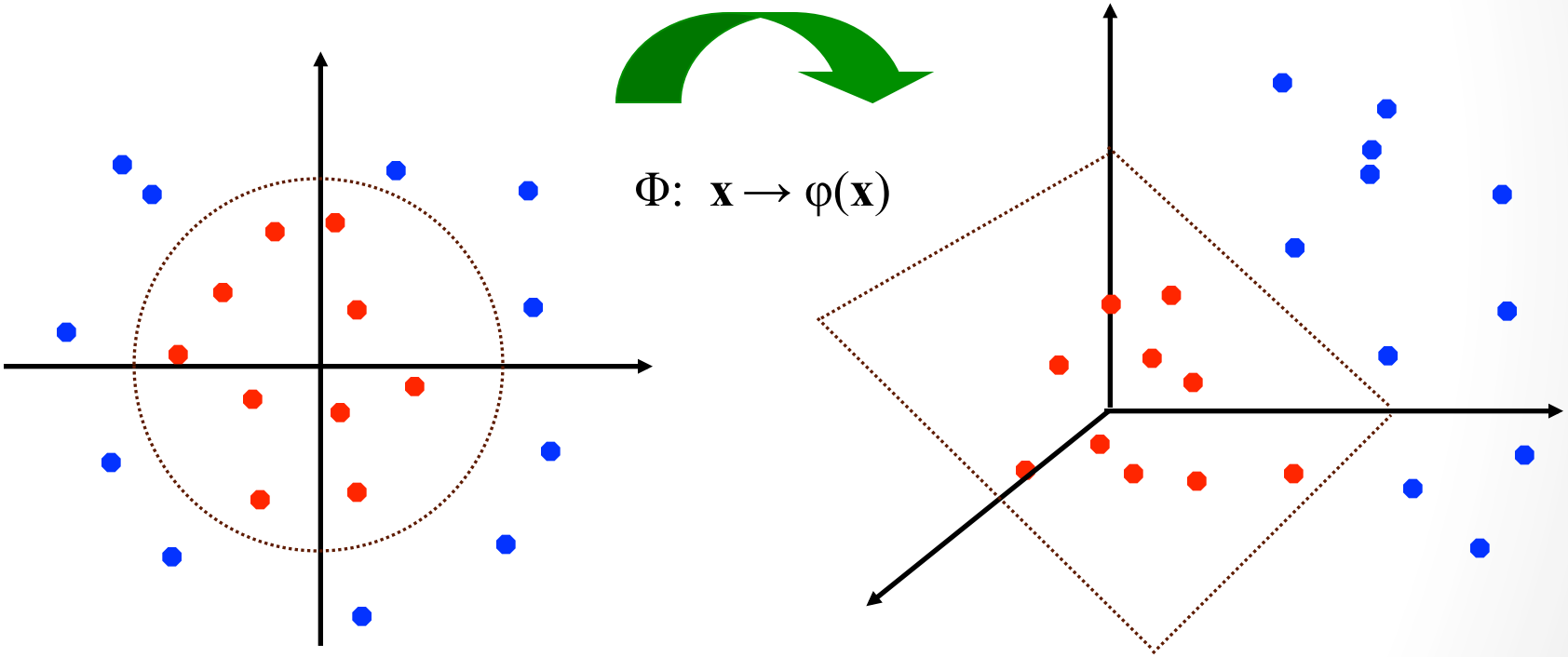
MVS não Linear

- Lidam com este problema mapeando o espaço de treinamento para um novo espaço de maior dimensão demonado *feature space*
  - Através de uma função Φ
  - Mapeamento da figura anterior realizado por:

$$\mathbf{\Phi}(\mathbf{x}) = \Phi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) + b = w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 + b = 0$$

Função Φ



Funções Kernel Comum

Tipo de Kernel	**Função $K(\mathbf{x}_i, \mathbf{x}_j)$**	**Parâmetros**
Polinomial	$(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^d$	δ, κ e d
Gaussiano	$\exp(-\sigma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$	σ
Sigmoidal	$\tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)$	δ e κ

Usando SVM no opencv

Pacote SVM em ML

- Funções básicas
  - train: treina SVM com parâmetros informados
  - trainAuto: descobre parâmetros para SVM e treina
  - predict: realiza o teste

Vide exemplos

Usando SVM

com libsvm

LIBSVM

- Biblioteca básica com implementação do SVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
  - Java
  - C/C++
  - Depende de python para rodar alguns scripts

- Implementações
  - Classificação (C-SVC,nu-SVC)
  - Regressão (episolon-SVR, nu-SVR)
  - Estimação de distribuição (one-class SVM)
  - Utilitários para facilitar o processo de classificação

Como usar? Classificação

- Para estratégia dividindo **treinamento** e teste (ou validação)
- Colocar a base no formato .libsvm

```
1  1:0.1   2:2.3   3:1.0
-1 1:0.2   2:1.3   3:0.5
1  1:0.3   2:3.3   3:1.2
…
```

- Utilitários básicos (são aplicados na ordem abaixo)
  1. svm-scale
  2. subset.py
  3. grid.py
  4. svm-train
  5. svm-predict

1) svm-scale

- Serve para normalizar as variáveis. Melhora a convergência do método
- Faixas definidas pelos parâmetros –l (lower) e –u (upper)
  - vide outras opções de salvar…

USO:

svm-scale **heart_scale** > heart_scale.scale

Normaliza entre a faixa -1 a 1, por padrão.

*https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/heart

2) subset.py

- Fica na pasta tools do aplicativo
- Utilizado para gerar bases de treinamento e bases de teste
  - deve utilizar o parâmetro –s 1 para garantir aleatóriedade

USO

tools/subset.py -s 1 heart_scale.scale 135 train test

Gera os arquivos train e test.

Train é usado para treinamento (grid e svm-train)

Test é usado para avaliar a metodologia (svm-predict)

3) grid.py

- Fica na pasta tools
- Serve para estimar os parâmetros do SVM (C e gama) quando for utilizado o núcleo radial

USO

tools/grid.py train

```
[local] 13 -5 75.5556 (best c=8.0, g=0.0078125, rate=85.9259)
[local] 13 -15 85.9259 (best c=8.0, g=0.0078125, rate=85.9259)
[local] 13 3 60.7407 (best c=8.0, g=0.0078125, rate=85.9259)
[local] 13 -9 79.2593 (best c=8.0, g=0.0078125, rate=85.9259)
[local] 13 -3 77.037 (best c=8.0, g=0.0078125, rate=85.9259)
8.0 0.0078125 85.9259
```

C Gama Acc

4) svm-train

- Realiza o treinamento SVM criando um modelo
- Por padrão usa:
  - -s svm-type = C-SVC (para classificação)
  - -t kernel-type = radial
- Deve ser informado o C e gama obtidos na etapa anterior
  - -c XXX
  - -g YYY
- Pode aplicar –v para validação cruzada informando o número de folds

4) svm-train

USO

svm-train -c 8 -g 0.0078125 train modelo

```
*
optimization finished, #iter = 88
nu = 0.436144
obj = -407.886185, rho = 0.191294
nSV = 64, nBSV = 54
Total nSV = 64
```

Representa quantos vetores foram utilizados para o modelo de treinamento.
Quanto menor, melhor (menor dimensão VC)

O arquivo modelo gerado representa o treinamento do SVM

5) svm-predict

- A partir do arquivo de modelo, usa o arquivo de test para avaliar o método

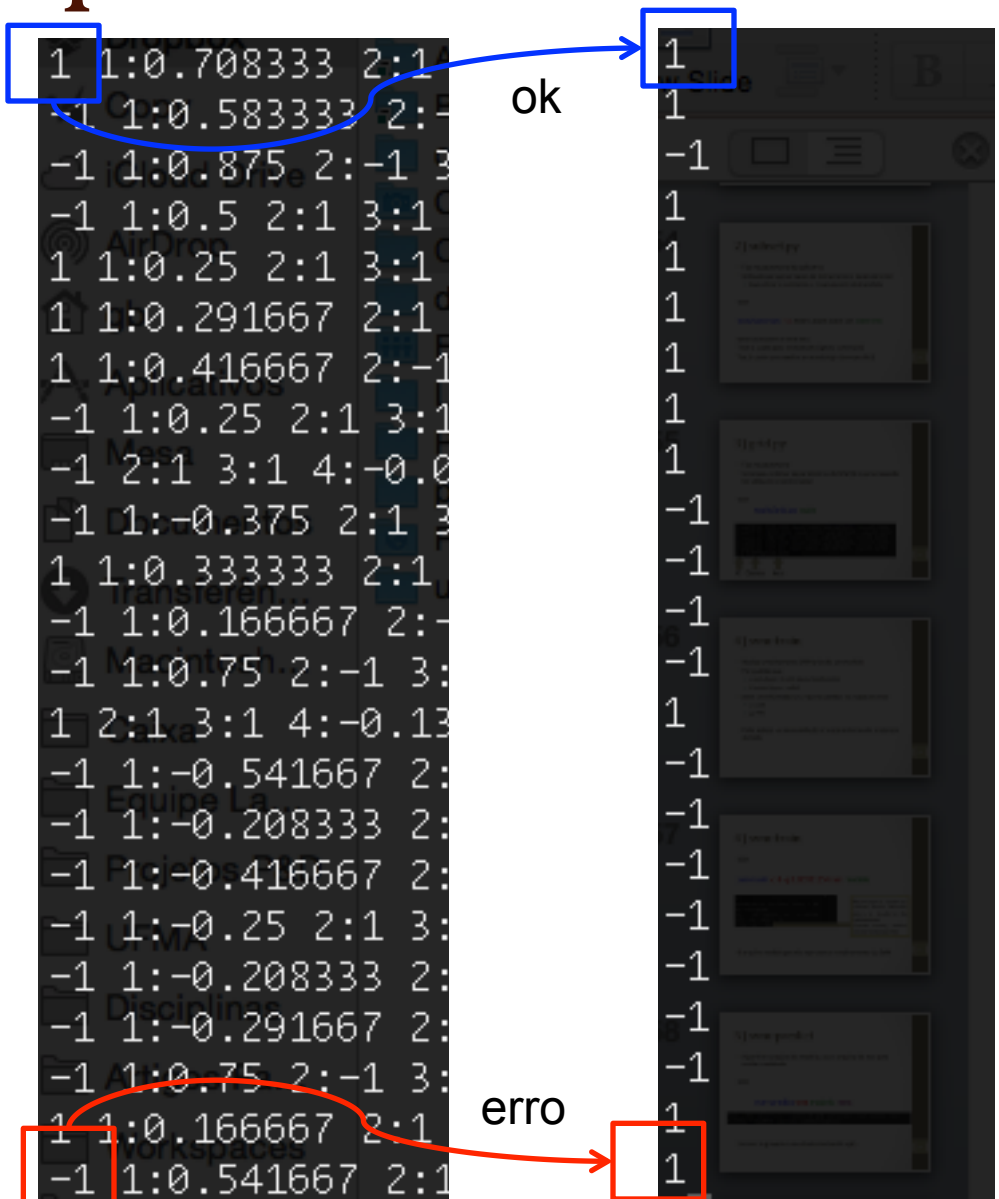
USO

svm-predict test modelo rtest

```
MacBook-Pro-de-Geraldo:libsvm-3.20 gb$ svm-predict test modelo rtest
Accuracy = 82.963% (112/135) (classification)
```

Em rtest é gravado o resultado da classificação

Arquivo rtest



- Usado para avaliar os resultados
- Compara cada resposta esperada, com a respost gerada

Quando se deseja obter Probabilidades

- Use o parâmetro –b 1 no svm-train para obter as probabilidades por classe

USO

svm-train –b 1 -c 8 -g 0.0078125 train modelo

Para o heart_scale

Perceba que foram usados os mesmos parâmetros do grid

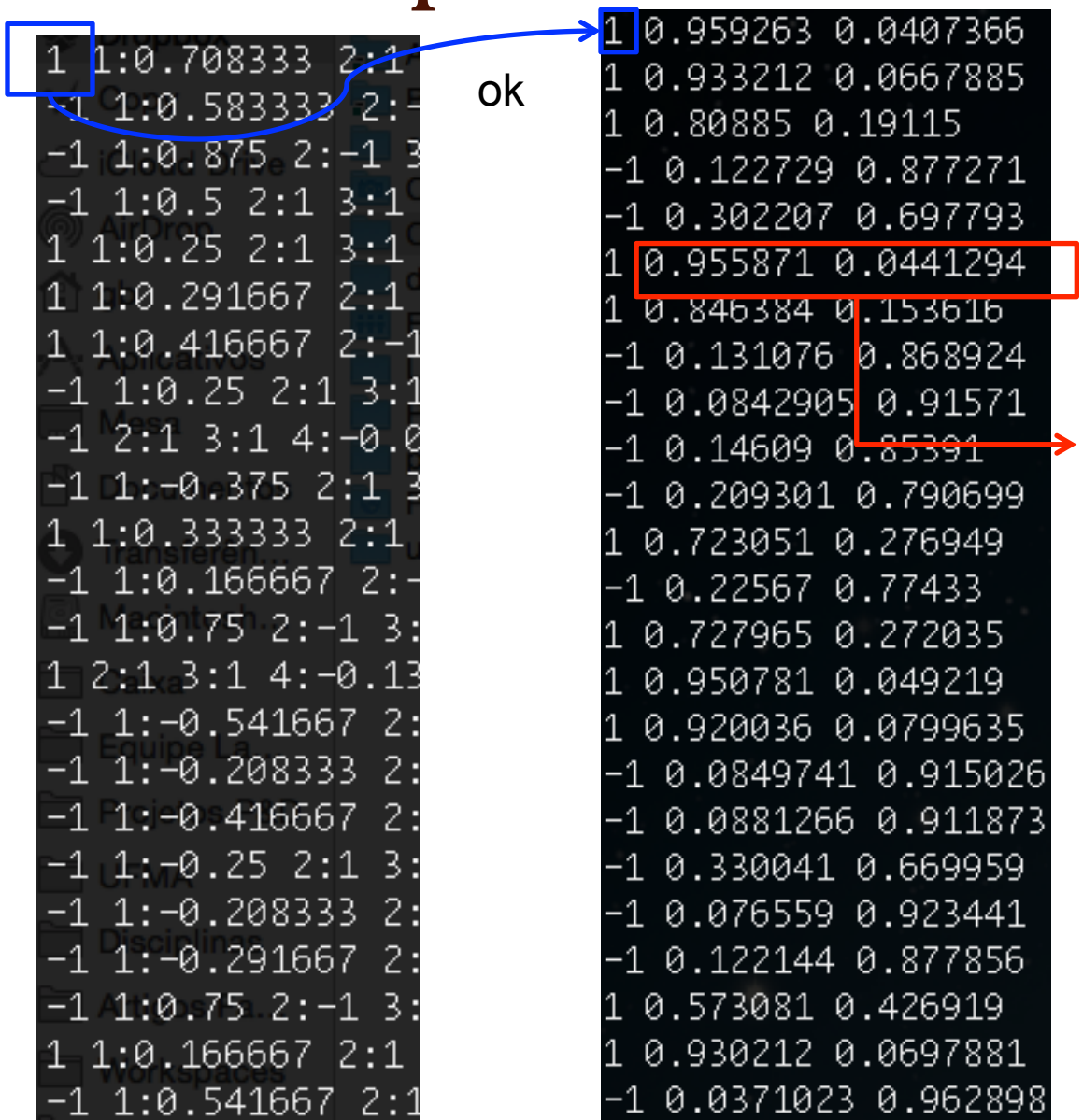
Quando se deseja obter Probabilidades

- e incluir o mesmo parâmetro no svm-predict

USO

svm-predict –b 1 test modelo rtest

Saída com probabilidades



- Usado para avaliar os resultados
- Aqui cada linha também contém o grau de pertinência entre classes

Conjuntos desbalanceados?

- Desbalanceamento: existem mais representantes da classe 1 do que da classe 0
- SVM não necessita de ajustes para conjuntos levemente desbalanceados
- Para conjuntos fortemenente desbalanceados é necessário ajustar a quantidade de erro que pode ser suportada
  - Ou para conjuntos onde existe uma necessidade maior de acerto de um conjunto em relação ao outro
- **Pense: Porque?**

Weighted SVM

- Uma das maneiras é usando o parâmetro –wi VALOR
  - onde **i** é o número da classe e **VALOR** a quantidade de peso

- **Na prática diz ao classificador quanto irá custar a mais errar (VALOR) um indivíduo da classe I**

- O que muda?

  - tools/grid.py -w1 5 train
  - svm-train -c 8 –g 0.5 -w1 5 train modelo

```
MacBook-Pro-de-Geraldo:datasets gb$ svm-predict test modelo rtest
Accuracy = 73.3333% (99/135) (classification)
```

Como usar? Regressão

- Para estratégia dividindo **treinamento** e teste (ou validação)
- Colocar a base no formato .libsvm

```
1  1:0.1   2:2.3   3:1.0
-1 1:0.2   2:1.3   3:0.5
1  1:0.3   2:3.3   3:1.2
…
```

- Utilitários básicos (são aplicados na ordem abaixo)
  1. svm-scale
  2. subset.py
  3. **gridregression.py**
  4. svm-train
  5. svm-predict

1) svm-scale

- Serve para normalizar as variáveis. Melhora a convergência do método
- Faixas definidas pelos parâmetros –l (lower) e –u (upper)
  - vide outras opções de salvar…

USO:

svm-scale **body_fat** > body_fat.scale

Normaliza entre a faixa -1 a 1, por padrão.

*https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/bodyfat

2) subset.py

- Fica na pasta tools do aplicativo
- Utilizado para gerar bases de treinamento e bases de teste
  - deve utilizar o parâmetro –s 1 para garantir aleatóriedade

USO

tools/subset.py -s 1 body_fat.scale 125 train test

Gera os arquivos train e test.

Train é usado para treinamento (**gridregression** e svm-train)

Test é usado para avaliar a metodologia (svm-predict)

3) gridregression.py

- Deve ser baixado como add-on do libsvm
- Serve para estimar os parâmetros do SVM (C e gama) quando for utilizado o núcleo radial e do epsilon para regressão

USO

tools/gridregression.py train

```
[local] -1 -5 -2 0.000373225 (best c=4.0, g=0.00390625, p=0.00390625, mse=2.25268e-05)
[local] -1 -5 -7 4.28891e-05 (best c=4.0, g=0.00390625, p=0.00390625, mse=2.25268e-05)
[local] -1 -5 -3 0.000373225 (best c=4.0, g=0.00390625, p=0.00390625, mse=2.25268e-05)
[local] -1 -5 -5 0.000245227 (best c=4.0, g=0.00390625, p=0.00390625, mse=2.25268e-05)
[local] -1 -5 -1 0.000373225 (best c=4.0, g=0.00390625, p=0.00390625, mse=2.25268e-05)
[local] -1 -5 -8 2.49217e-05 (best c=4.0, g=0.00390625, p=0.00390625, mse=2.25268e-05)
4.0 0.00390625 0.00390625 2.25268e-05
```

C Gama p mean square error

4) svm-train

- Realiza o treinamento SVM criando um modelo
- Por padrão usa:
  - -s svm-type = epsilon-SVR (para regressão)
  - -t kernel-type = radial
- Deve ser informado o C e gama obtidos na etapa anterior
  - -c XXX
  - -g YYY
  - -p YYY
- Pode aplicar –v para validação cruzada informando o número de folds

4) svm-train

USO

svm-train –s 3 -c 4.0 -g 0.00390625 –p 0.00390625 train modelo

```
*
optimization finished, #iter = 77
nu = 0.057647
obj = -0.330552, rho = -1.078779
nSV = 17, nBSV = 3
```

Representa quantos vetores foram utilizados para o modelo de treinamento.
Quanto menor, melhor (menor dimensão VC)

O arquivo modelo gerado representa o treinamento do SVM

5) svm-predict

- A partir do arquivo de modelo, usa o arquivo de test para avaliar o método

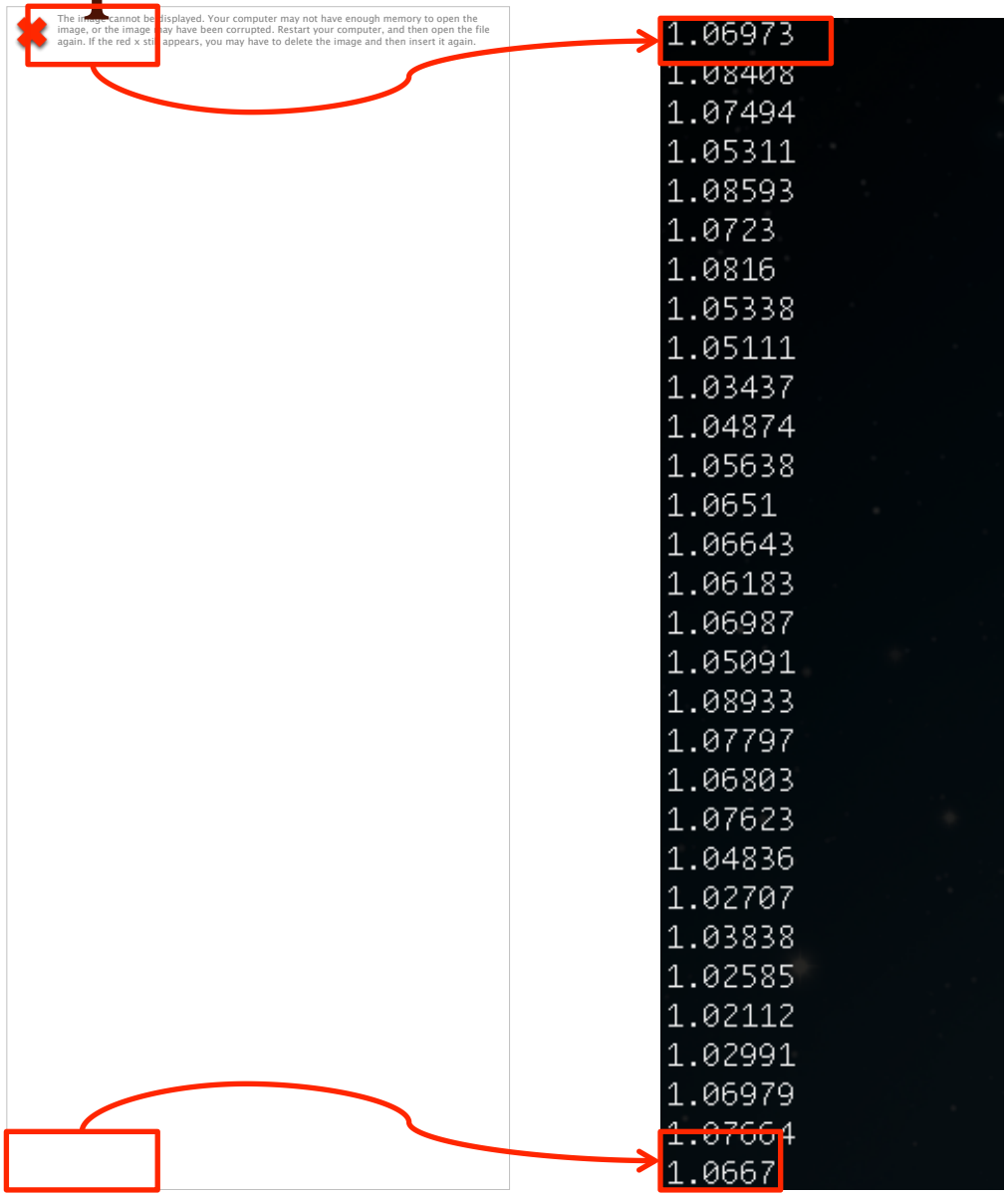
USO

svm-predict test modelo rtest

```
Mean squared error = 6.48333e-06 (regression)
Squared correlation coefficient = 0.984702 (regression)
```

Em rtest é gravado o resultado da classificação

Arquivo rtest

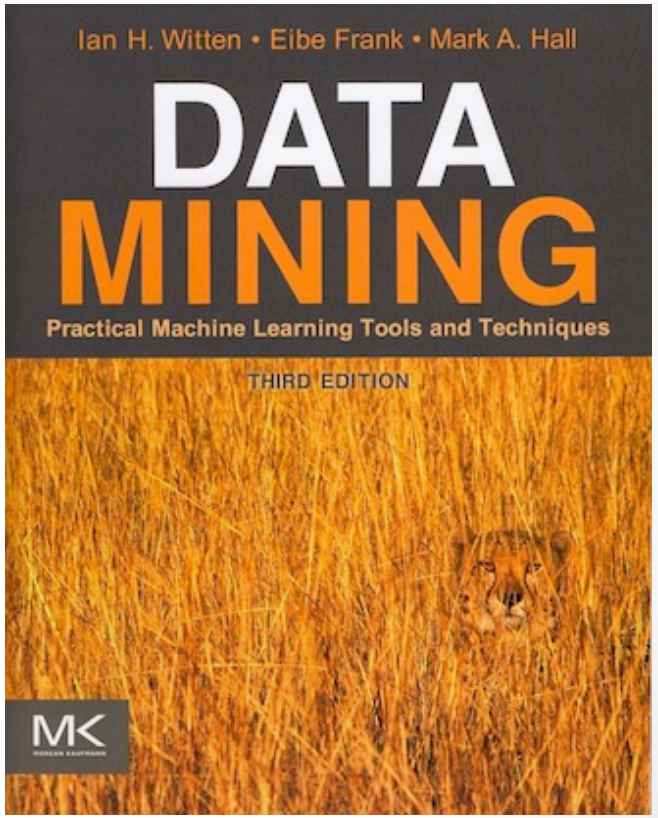


- Usado para avaliar os resultados
- Compara cada resposta esperada, com a respost gerada

Usando o Weka a nosso favor

O que é

- Software livre, voltado para a mineração dados
- Mantenedora principal: University of Waikato (http://www.cs.waikato.ac.nz/ml/weka/)



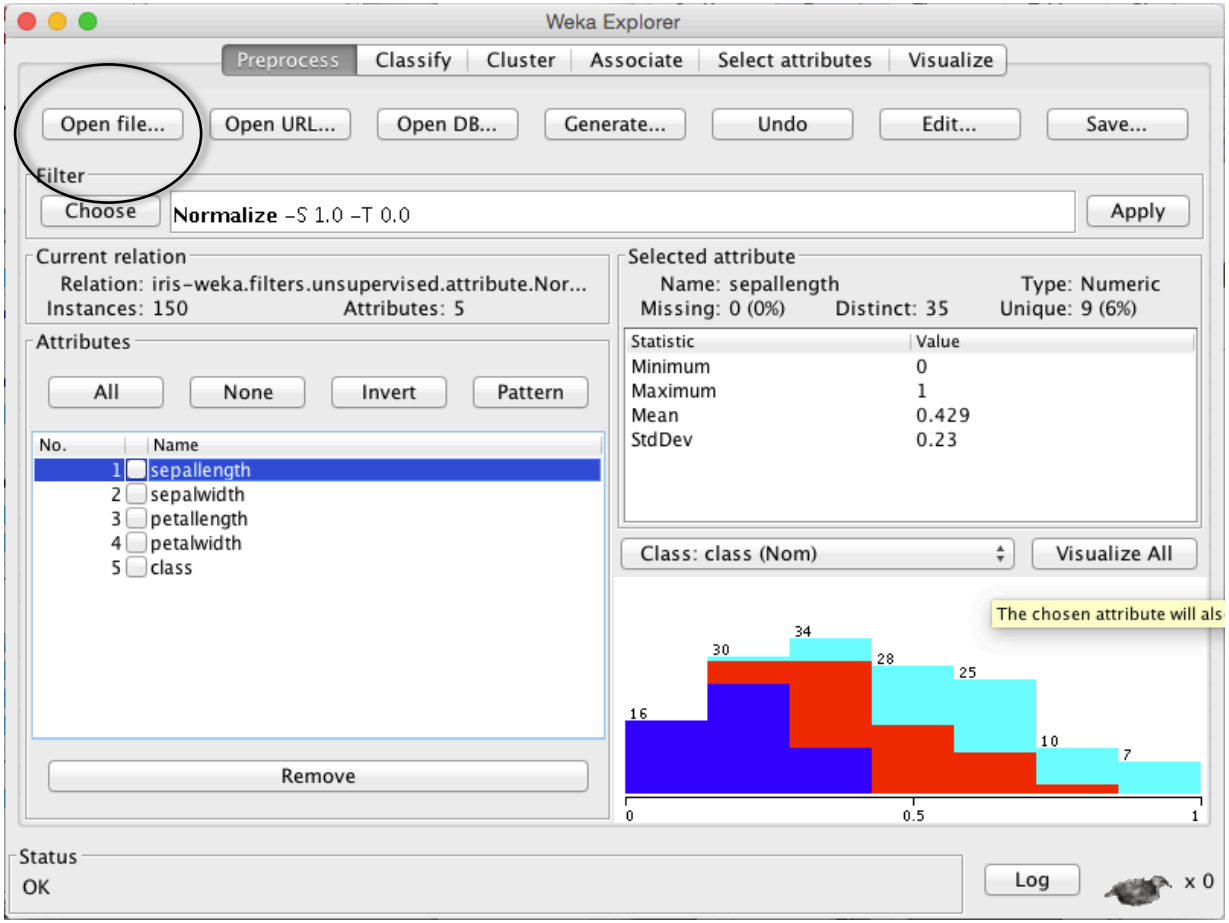
Weka Explorer – Usando SVM

- Inicialmente deve ser linkar o libsvm com o weka
- Maneira #1: incluir no classpath

```
java -classpath $CLASSPATH:weka.jar:libsvm.jar
weka.gui.GUIChooser
```

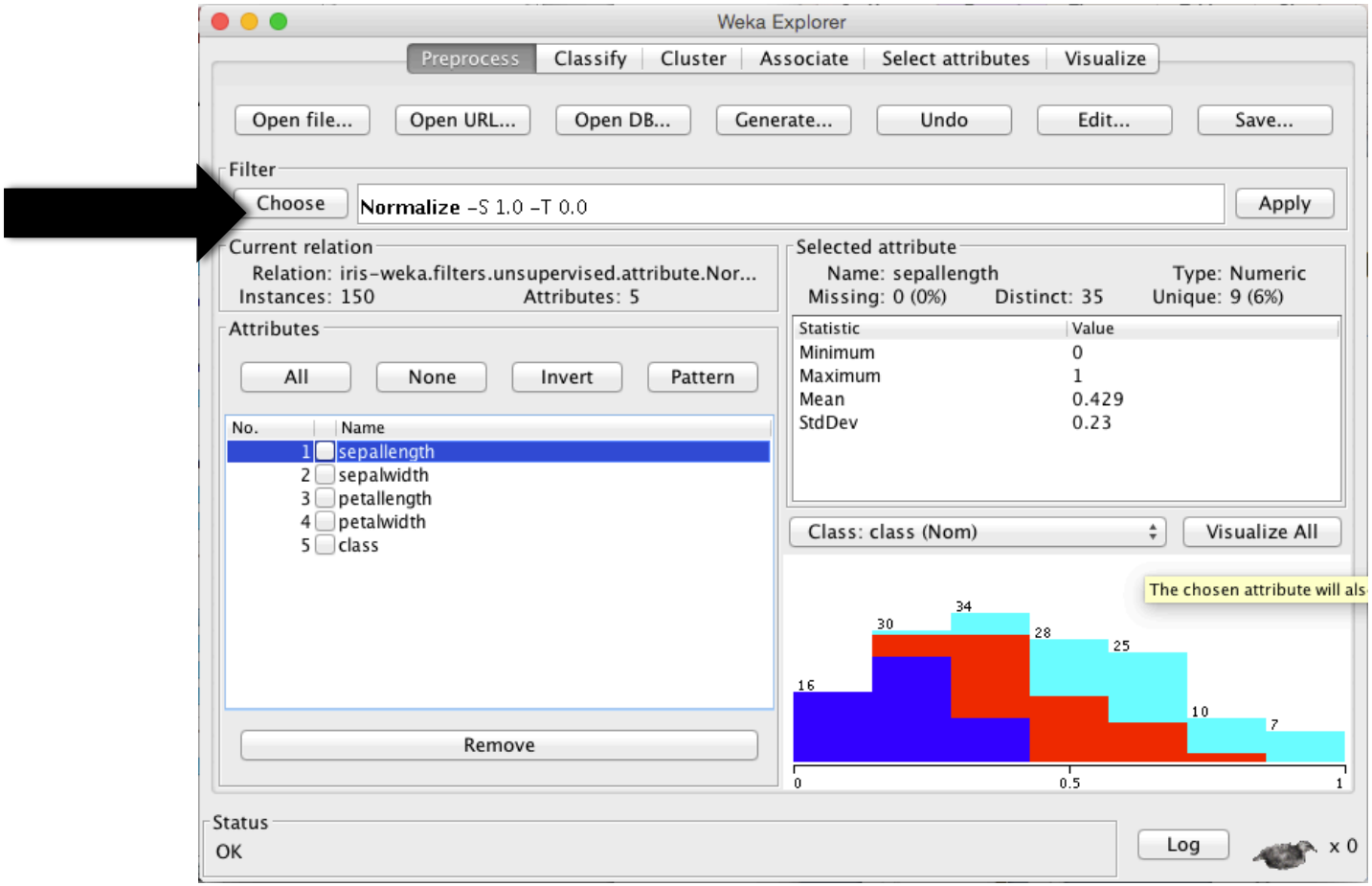
- Maneira #2: maior parte das vezes já está incluida no pacote completo do weka

Weka Explorer

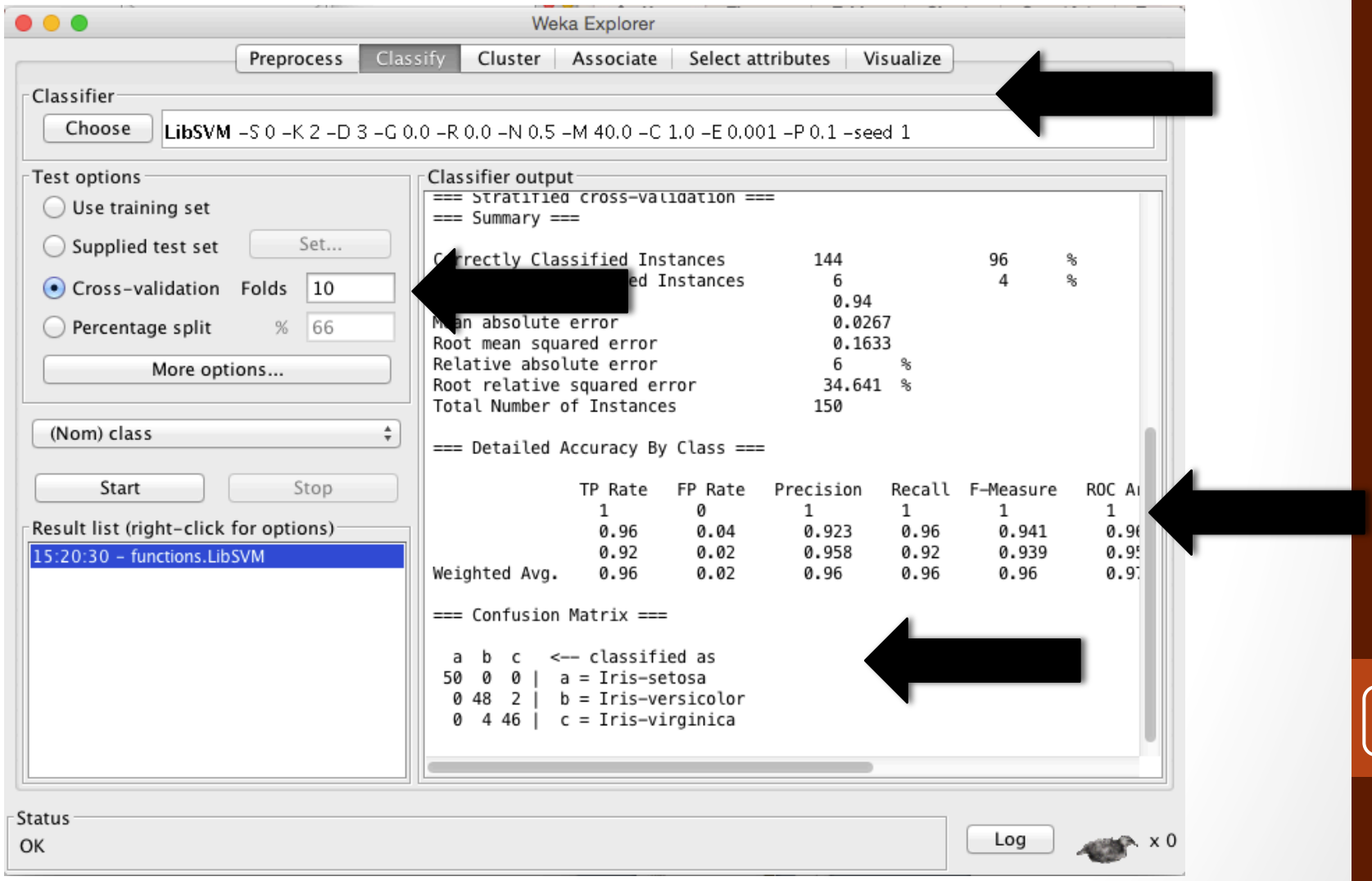


Filter

- Filter -> unsupervised -> attribute -> Normalize

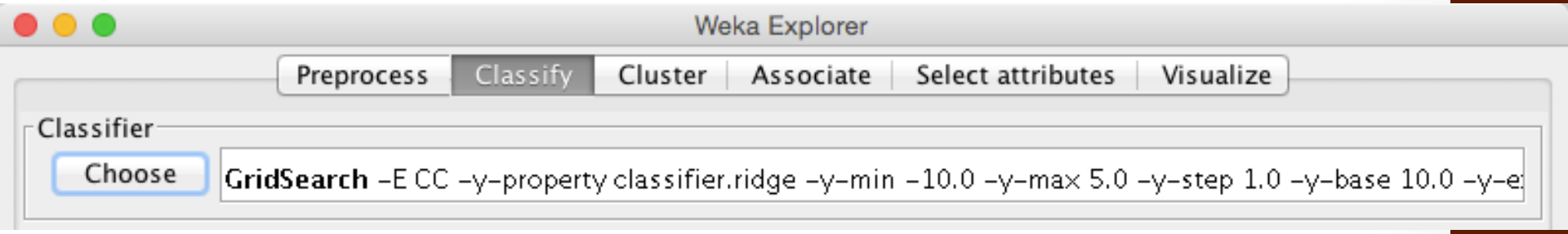


Classify


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -seed 1
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 144 96 %
Incorrectly Classified Instances 6 4 %
0.94
Mean absolute error 0.0267
Root mean squared error 0.1633
Relative absolute error 6 %
Root relative squared error 34.641 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Ar
1 0 1 1 1 1
0.96 0.04 0.923 0.96 0.941 0.96
0.92 0.02 0.958 0.92 0.939 0.95
Weighted Avg. 0.96 0.02 0.96 0.96 0.96 0.97
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 48 2 | b = Iris-versicolor
0 4 46 | c = Iris-virginica
(Nom) class
Start
Stop
Result list (right-click for options)
15:20:30 - functions.LibSVM
Status
OK
Log
x 0

Ajuste de Parâmetros

- Meta -> GridSearch



Ajuste de Parâmetros

- Meta -> GridSearch
  - classifier: escolha libsvm
  - evaluation: accuracy (veja se é o melhor para você)
  - filer: escolha Allfilter (não vamos fazer nenhuma filtragem)
- Em:
  - Xexpression = pow(BASE,I)
  - Xmax = 15
  - Xmin = -5
  - Xproperty = classifier.cost
  - Xstep = 1
  - Xbase = 2

- ….

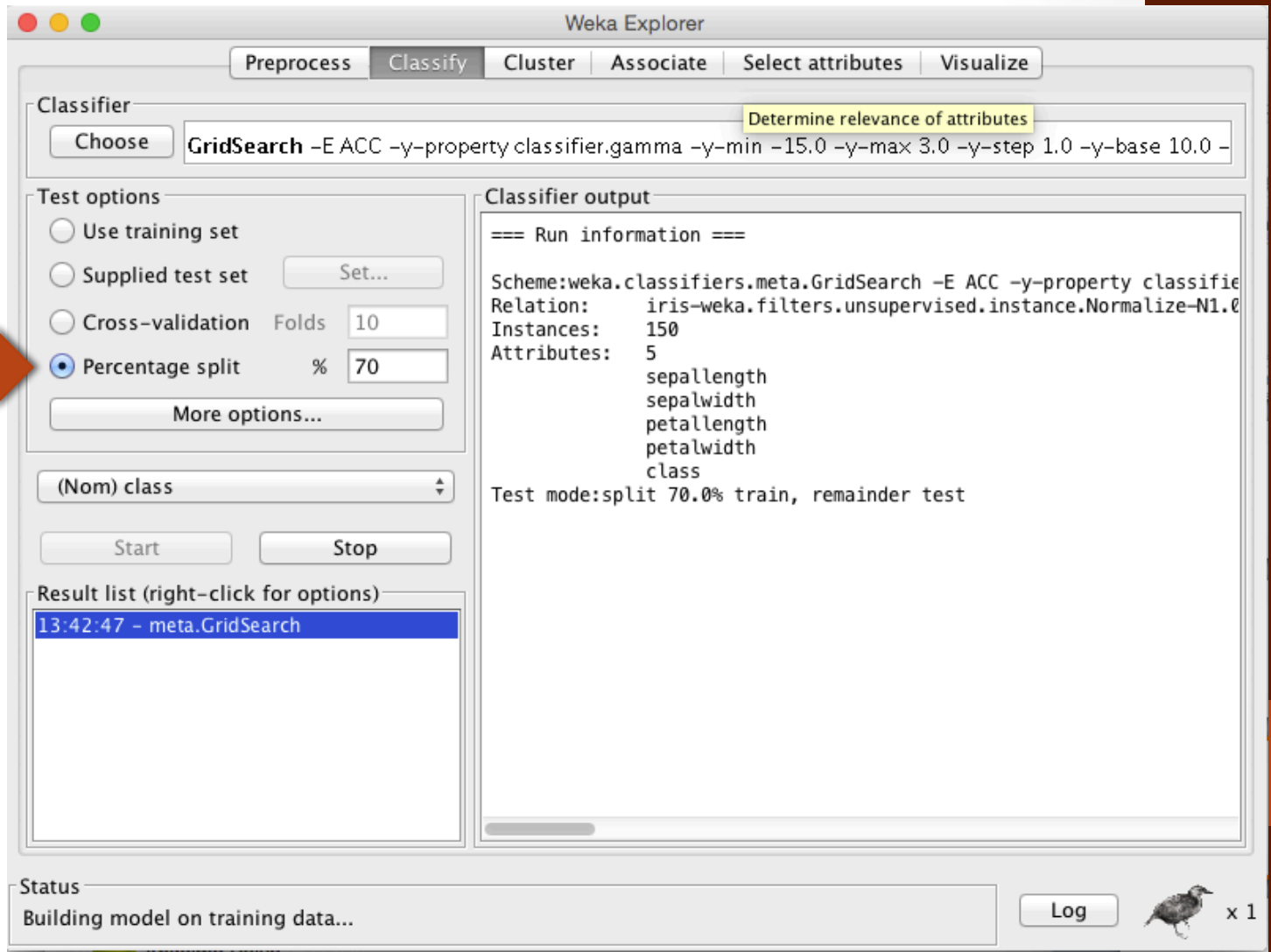
Ajuste de Parâmetros

- Em:
  - Yexpression = pow(BASE,I)
  - Ymax = 3.0
  - Ymin = -15
  - Yproperty = classifier. gamma
  - Ystep = 1
  - Ybase = 2

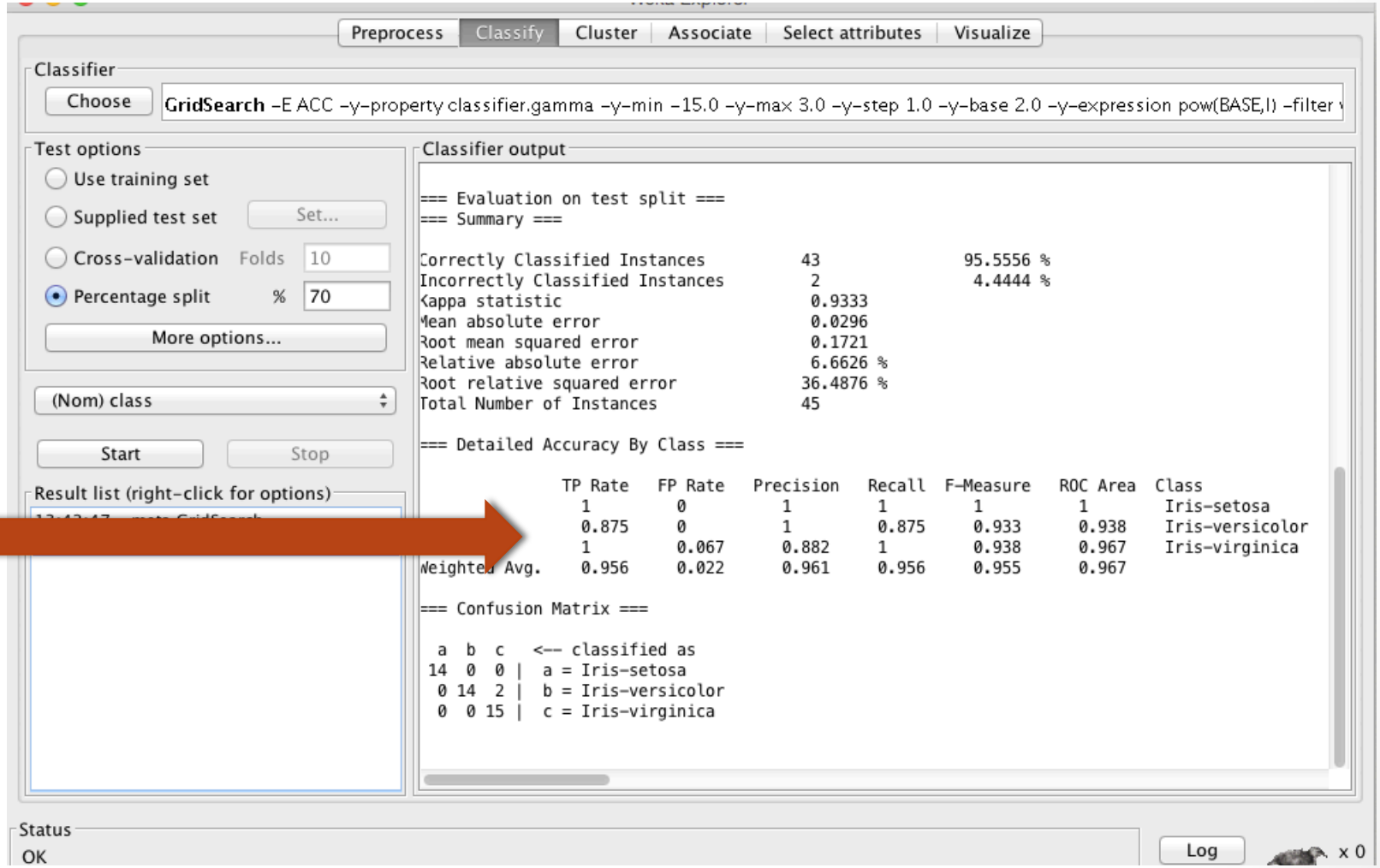
- E SALVE!

Ajuste de Parâmetros

Ajuste o percentual!



Novo Resultado



Parâmetros

```
ibutes:   5
          sepallength
          sepalwidth
          petallength
          petalwidth
          class
 mode:split 70.0% train, remainder test

Classifier model (full training set) ===

.classifiers.meta.GridSearch:
er: weka.filters.AllFilter
sifier: weka.classifiers.functions.LibSVM -S 0 -K 2 -D 3 -G 0.125 -R 0.0

operty: classifier.cost
operty: classifier.gamma

uation: Accuracy
dinates: [15.0, -3.0]
es: 15.0 (X coordinate), 0.125 (Y coordinate)
```

Detalhe

- Para fazer sucessivos treinos e testes no Weka, com bases aleatórias:
  - Gere por fora bases aleatórias para ele
  - ou
  - Utilize o filtro Randomize, sempre mudando o **seed!!!**

Referências

- Ana Carolina Lorena e André C. P. L. F. de Carvalho. Uma Introdução às Support Vector Machines
- C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998. http://citeseer.nj.nec.com/burges98tutorial.html
- LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/