PROCESSAMENTO DE IMAGENS APLICADO A AGROINDUSTRIA

Pedro Luiz de Paula Filho

Aprendizagem de máquina

Recursos

- Material adaptado das aulas de Reconhecimento de Padrões
  - Prof: Luiz Eduardo S. Oliveira, PhD
    - www.lesoliveira.net
    - Programa de Pós Graduação em Informática – UFPR
  - Diego Bertolini
    - UTFPR = Campo Mourão

Aprendizagem de máquina

- Desde que os computadores foram inventados temos nos perguntado: “Eles são capazes de aprender?”
- Se pudéssemos programá-los para aprender para se aperfeiçoar automaticamente com a xperiência o impacto seria surpreendente

Aprendizagem de máquina

- Infelizmente ainda não sabemos como fazer computadores aprender de uma maneira similar a maneira como os humanos aprendem;
- Entretanto, foram desenvolvidos algoritmos que são eficientes em certos tipos de tarefas de aprendizagem e um entendimento teórico de aprendizagem está começando a surgir.

Aprendizagem de máquina

- Aprendizagem é uma propriedade essencialmente humana.
- Aprender significa mudar para fazer melhor (de acordo com um dado critério) quando uma situação similar acontecer.

Objetivo

- **Classificação não supervisionada**
  - Clusterização – Kmeans
- **Classificação supervisionada**
  - Vizinhos mais próximos – Knn
  - Support Vector Machine - SVM
  - Redes Neurais

Recursos

- Kmeans
- Knn
- Percetron
- LIBSVM / Python
- JAVANNS / Analyse

Aprendizagem de Máquina

- Aprendizagem, não é "**memorizar**". Qualquer computador pode "memorizar", a dificuldade é "**generalizar**" um comportamento para uma nova situação.

Formas de Aprendizagem

- **Supervisionada**
  - Fornecemos a “resposta” durante o treinamento
  - É o mais eficiente porque fornece mais informações
- **Por Reforço**
  - Não damos a “resposta“
  - O sistema faz uma hipótese lhe dizemos “bom / ruim”
  - Útil para o controle de robôs
- **Não Supervisionada**
  - Ex: Quais são as características principais dos clientes típicos? (segmentação do mercado)

Fases da Aprendizagem

- **Treinamento (supervisionado)**
  - Apresentamos exemplos ao sistema
  - O sistema "aprende" a partir dos exemplos
  - O sistema modifica gradualmente seus parâmetros ajustáveis para que a saída se aproxime da saída desejada.
- **Utilização/Teste**
  - Novos exemplos jamais visto aparecem
  - Desejamos que o sistema **generalize**!

Aprendizagem Não Supervisionada

- Interesse em capturar uma organização inerente dos dados.
- Assume-se que não se conhece a que classe pertence uma coleção de dados
- Busca encontrar agrupamentos naturais através de medidas de semelhança

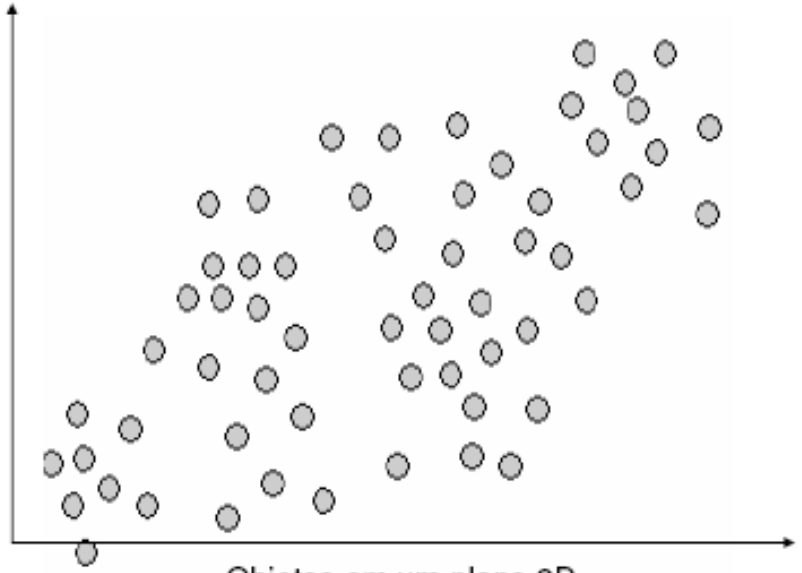
k-Means (*k*-Médias)

- Usado em processamento de sinais, telecomunicações e mineração de dados.
- É a técnica mais simples de aprendizagem não supervisionada.
- Consiste em agrupar (cluster) *k* grupos (classes) através de algum tipo de proximidade espacial (ex. distância euclidiana)

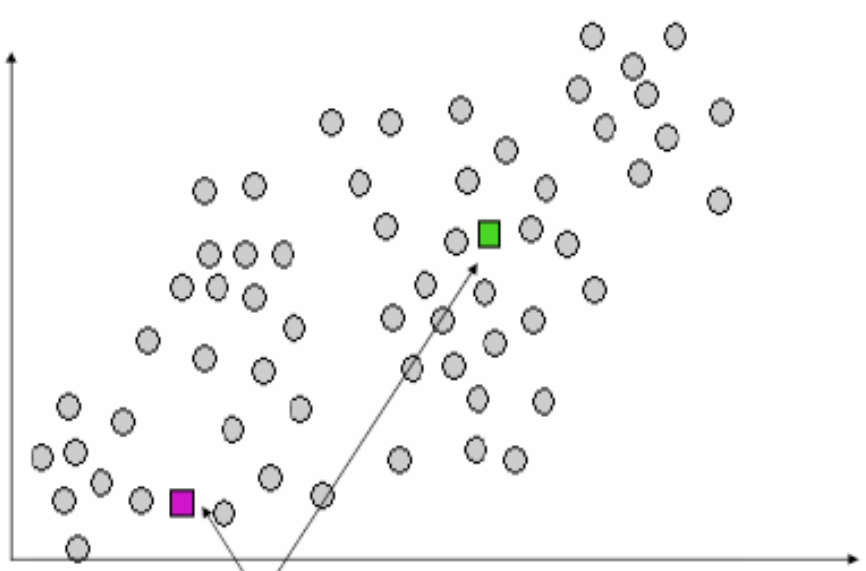
Algoritmo *k*-Means

1. Determinar os centróides
2. Atribuir a cada objeto do grupo o centróide mais próximo.
3. Após atribuir um centróide a cada objeto, recalcular os centróides.
4. Repetir os passos 2 e 3 até que os centróides não sejam modificados.

Exemplo

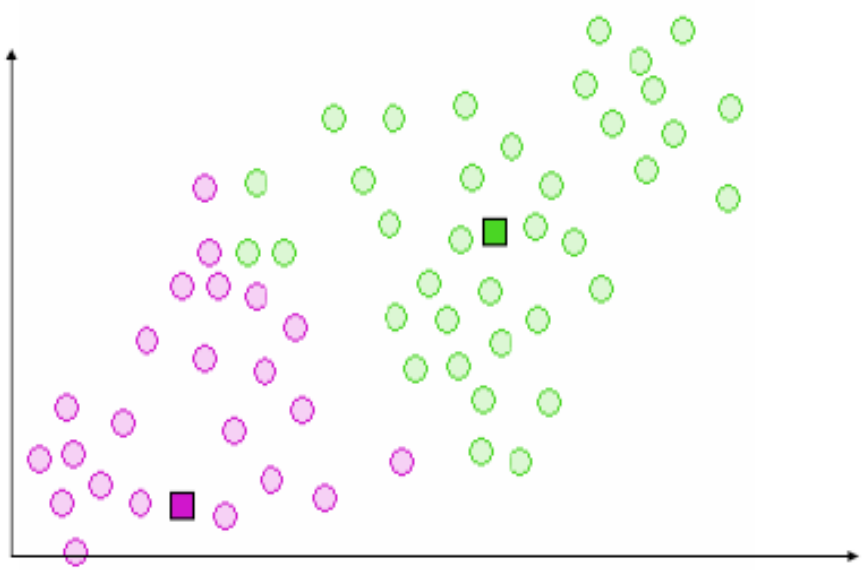


Objetos em um plano 2D

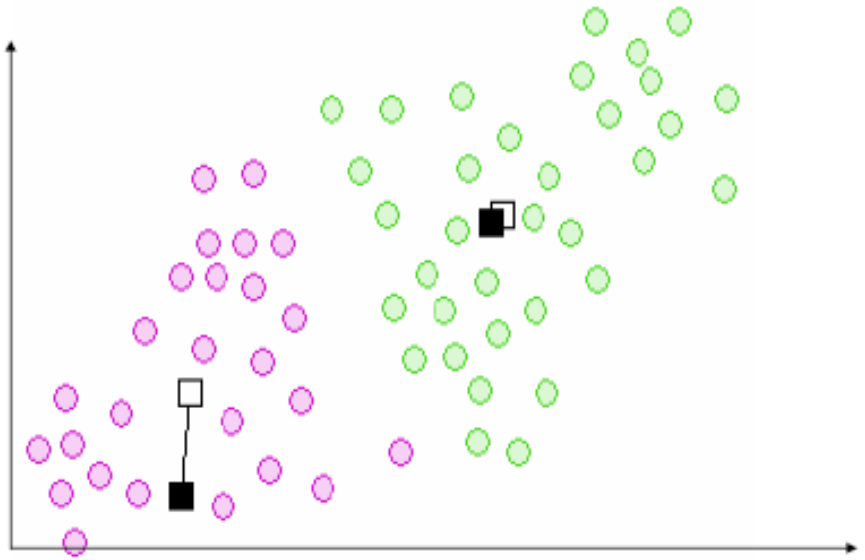


Passo 1:Centróides inseridos aleatoriamente

Exemplo

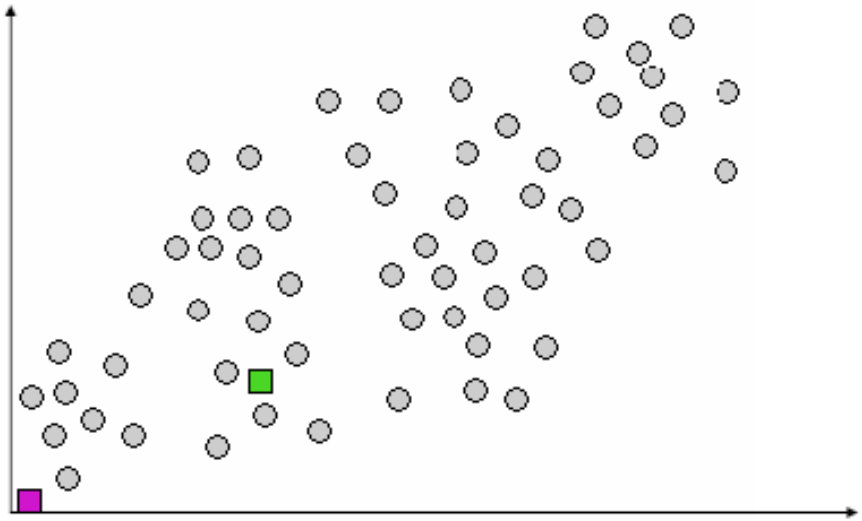


Passo 2: Atribuir a cada objeto o centróide mais próximo

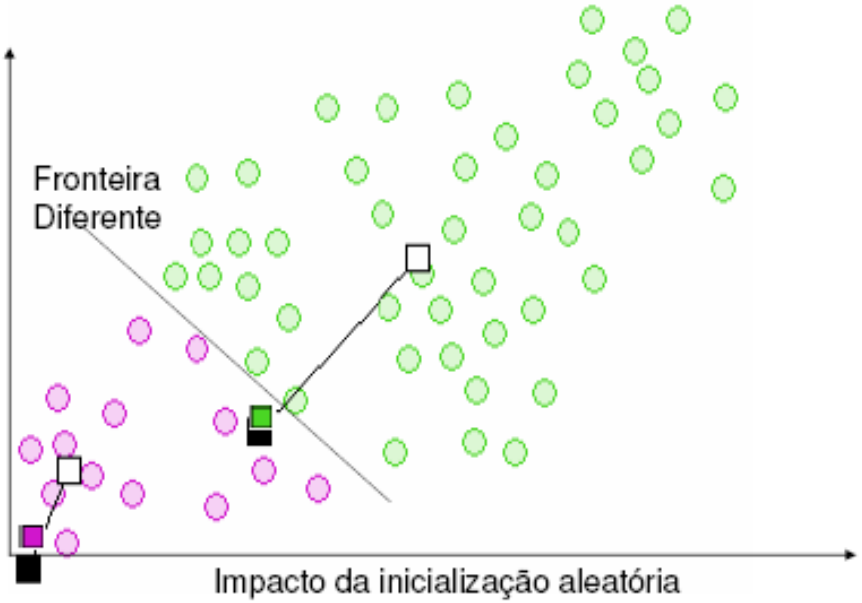


Passo 3: Recalcular os centróides

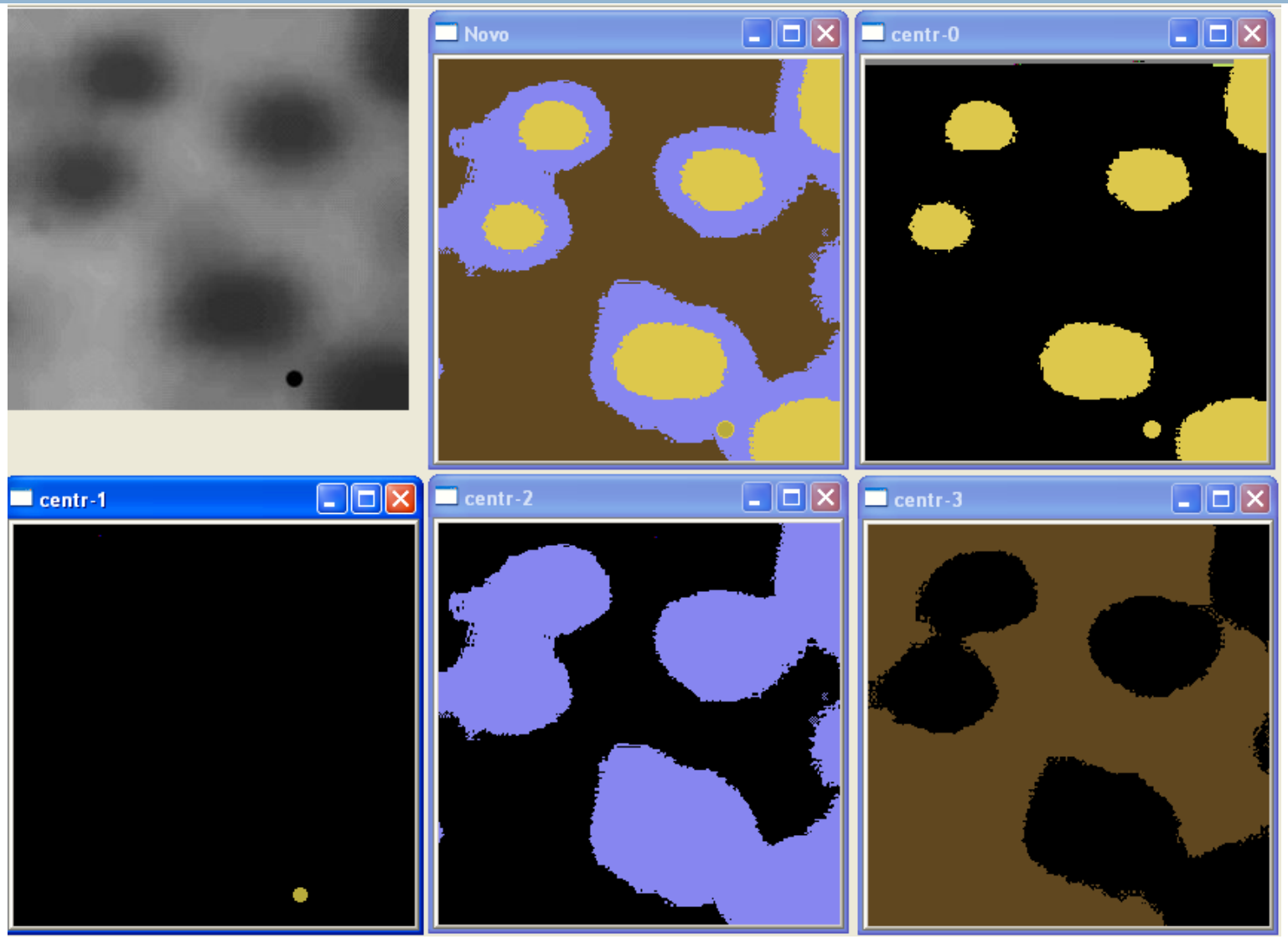
Exemplo – Impacto da Aleatoriedade



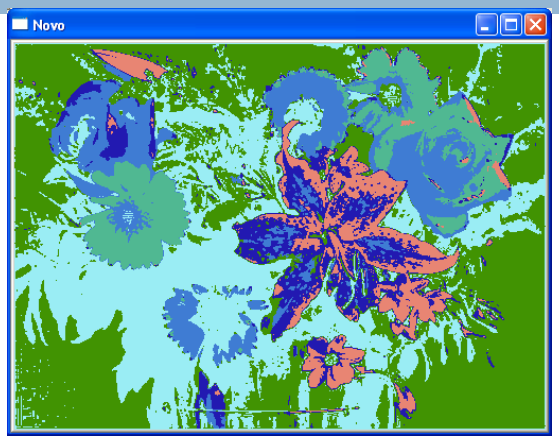
Impacto da inicialização aleatória.

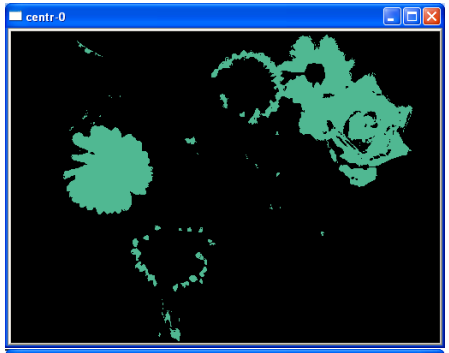


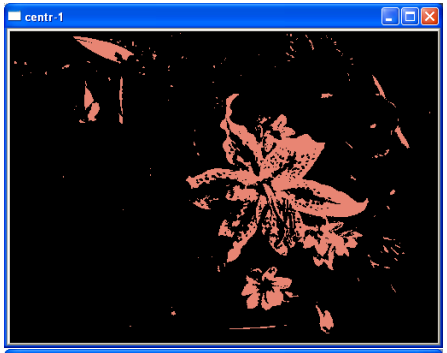
Prática – kmeans – celula.jpg

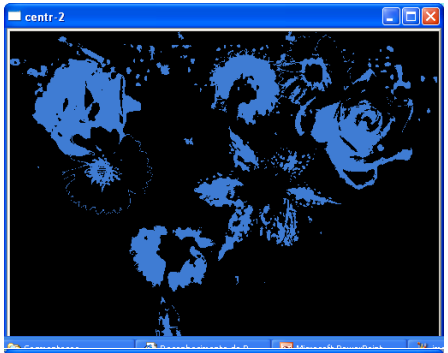


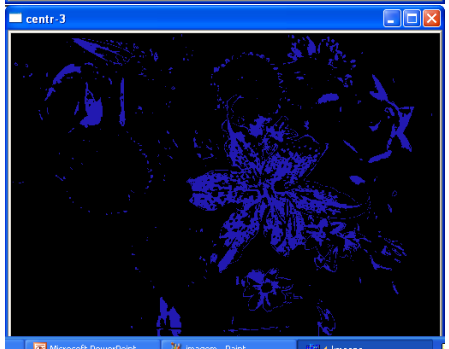
Prática – kmeans – flores.jpg

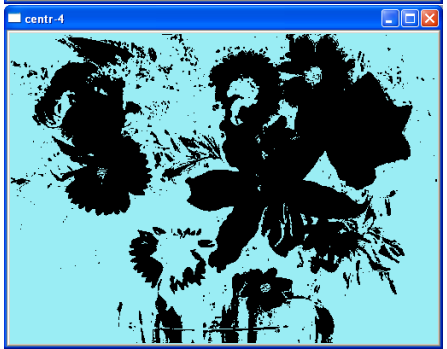












Classificação Supervisionada

- Tem-se um conjunto de dados pré-classificados usados para treinar o sistema
- O classificador é treinado para replicar ou refinar o conhecimento dos padrões
- Podem ser:
  - **Paramétricos**
  - **Não paramétricos**

Métodos Não Paramétricos

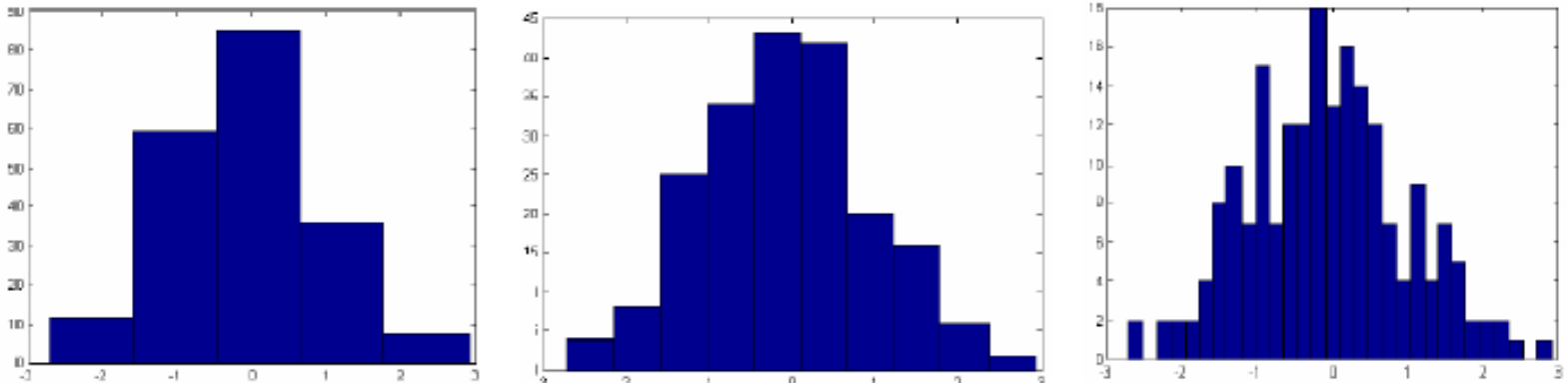
- Usado quando não se sabe como estão distribuídas as classes
- Tem melhores resultados quando se tem um pequeno número de amostras de treino
- Tudo que se tem são dados rotulados
- Busca estimar a distribuição de probabilidade

Métodos Não Paramétricos

- Histogramas
- Janelas de Parzen
- Vizinhos mais próximos – *k*-nn

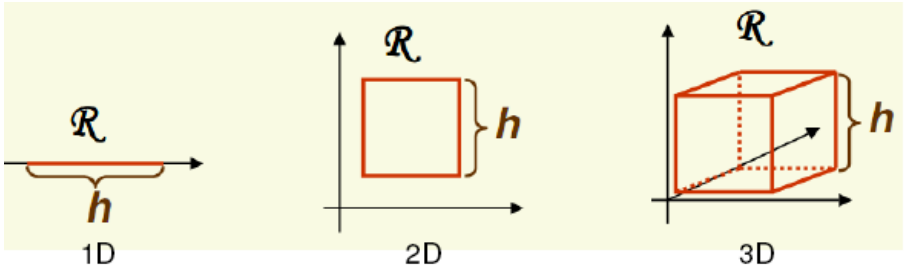
Histograma

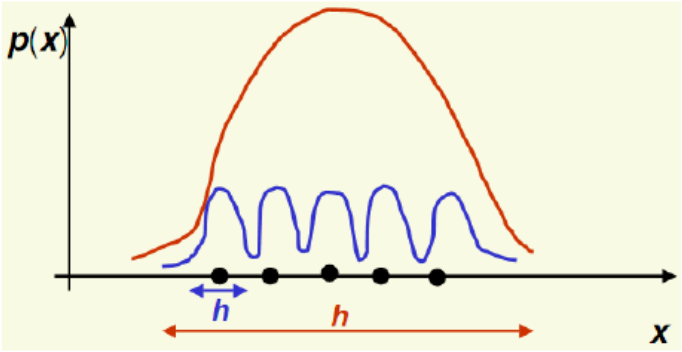
- Método mais antigo e mais simples para estimação de densidade.
- Depende da origem e da largura (h) usada para os intervalos



Janelas de Parzen

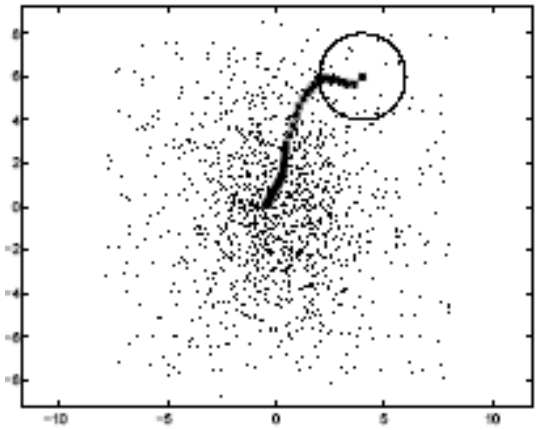
- Nessa abordagem fixa-se o tamanho da região R para estimar a densidade.
- Assumindo que a região R é um hipercubo de tamanho h
- Se o h for grande generaliza demais se for muito pequeno especializa demais

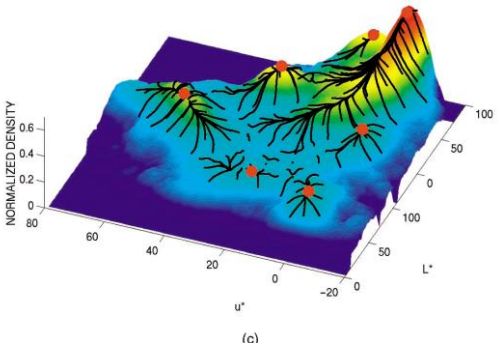




Exemplo – Mean Shift

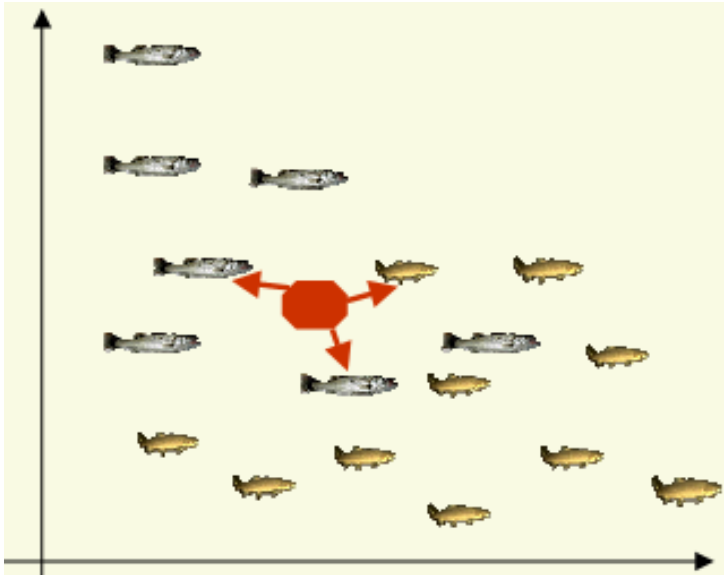
Perseguição do local de densidade máxima





Vizinhos mais Próximos - *k*-nn

- Baseia-se em encontrar os *k* elementos mais próximos a um elemento **x**, não identificado, e através de uma votação identificar a qual classe ele pertence



Se k = 3, teríamos 2 robalos e 1 salmão. Logo, x é um robalo.

Vizinhos mais Próximos - *k*-nn

- Para medir a proximidade dos vizinhos geralmente usa-se a distância euclidiana

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- *k* é geralmente impar para evitar empates

Vizinhos mais Próximos - *k*-nn

A classificação pode mudar de acordo com a escolha de k.

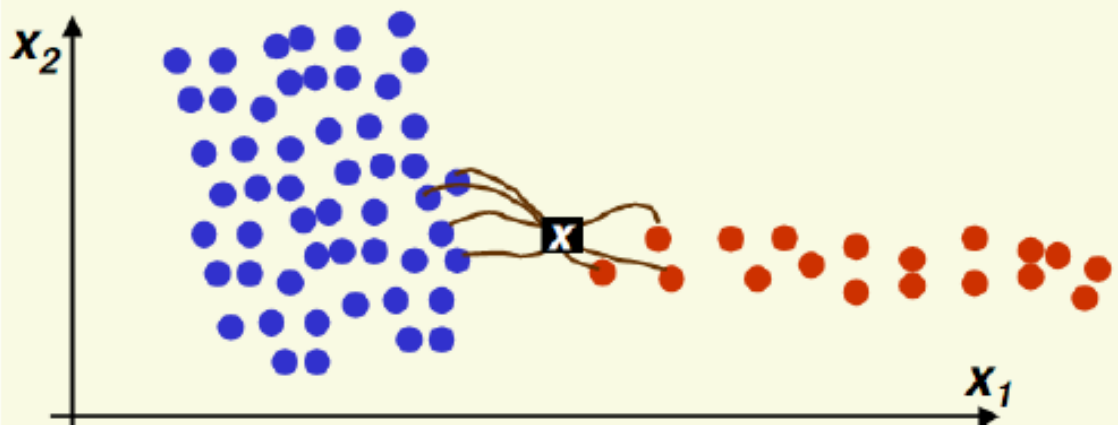
K=1 – Não se pode classificar
K=3 – Vermelho
K=5 – Vermelho
K=7 – Azul

Vizinhos mais Próximos - *k*-nn

- O k-nn é uma regra simples e intuitiva.
- Se tivermos um número ilimitado de exemplos, ele tende a funcionar bem
- O problema é encontrar o k
  - k muito pequeno está sujeito a ruídos
  - k muito grande é muito generalista
  - Segredo - Base de validação

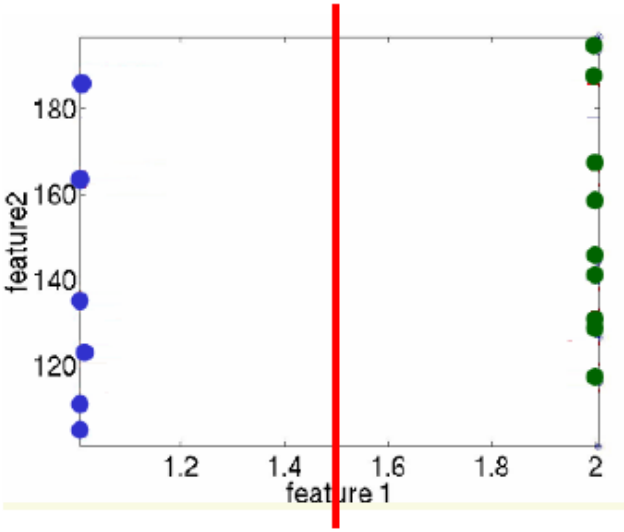
Vizinhos mais Próximos - *k*-nn

O problema é encontrar o k



Normalização

- Quando as características usam escalas diferentes.
  - Característica 1 varia entre 1 e 2
  - Característica 2 varia entre 100 e 20(
- Resolve-se o problema através da normalização, onde a forma mais simples, consiste em dividir cada característica pelo somatório de todas as características



Matriz de Confusão

- Matriz que permite visualizar as principais confusões do sistema.
- Considere um sistema com 3 classes, 100 exemplos por classe.

100% de classificação

	c1	c2	c3
c1	100		
c2		100	
c3			100

Erros de classificação

	c1	c2	c3
c1	90	10	
c2		100	
c3	5		95

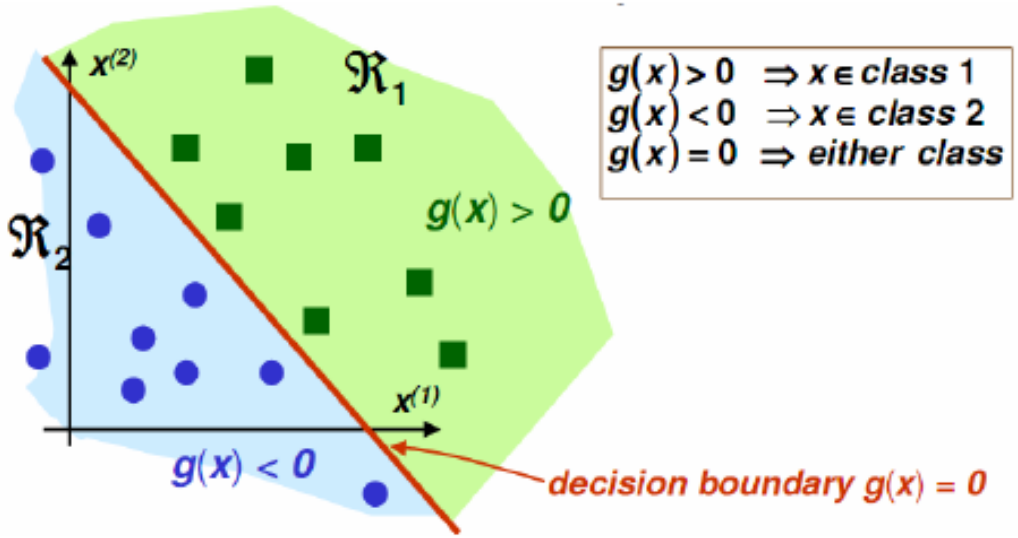
10 exemplos de C1 foram classificados como C2

Exemplo - knn

- Arquivo de Treinamento (training.txt)
  - 1000 digitos classificados de 0 a 9
  - 132 caracteristicas
- Arquivo de Teste (testing.txt)
  - 1000 digitos classificados de 0 a 9
  - 132 caracteristicas
- Matriz de confusão (knn.txt)
  - Resultado com k variando de 1 a 31
  - *k* 15 e 17 melhores resultados

Classificadores Lineares

- Busca encontrar uma função que permita descrever uma fronteira de decisão

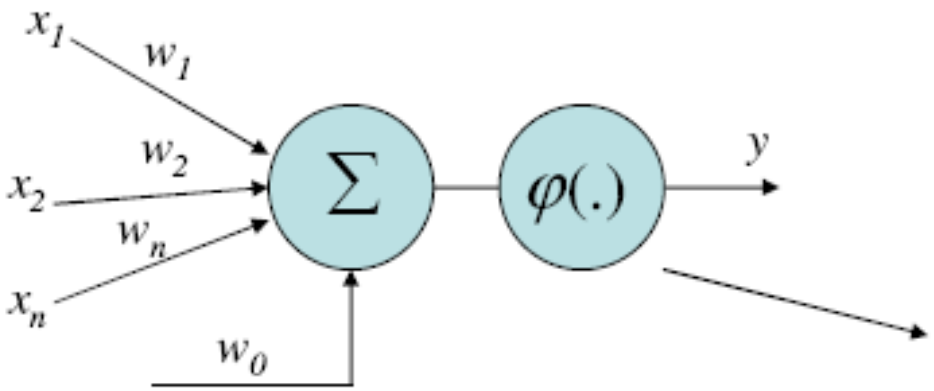


- Exemplos:
  - Perceptron
  - SVM

Perceptron

- Classificador linear bastante simples, mas bastante importante no desenvolvimento das redes neurais
  - É considerado como sendo a primeira e mais primitiva estrutura de rede neural.
  - Concebido na década de 50.
- Tenta encontrar a melhor fronteira linear que separa os dados.

Perceptron



Bias

$$y = \varphi\left(\sum w_i \times x_i + w_0\right)$$

A função de ativação normalmente utilizada no perceptron é a hardlim (threshold)

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Perceptron - Algoritmo

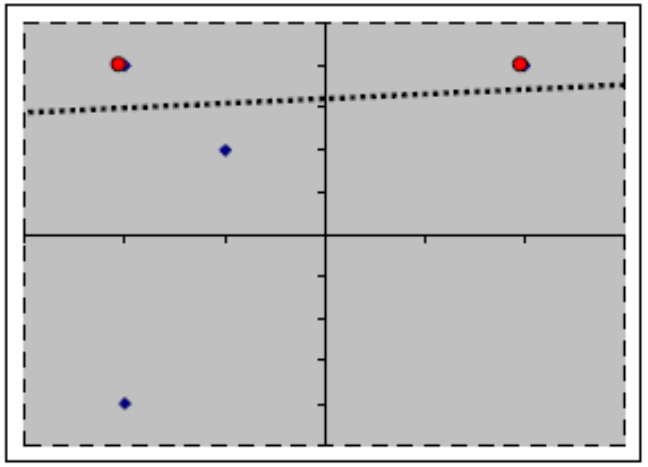
1. Iniciar os pesos e bias com valores pequenos, geralmente no intervalo [0.3-0.8]
2. Aplicar um padrão de entrada com seu respectivo valor desejado de saída (t_i) e verificar a saída y da rede.
3. Calcular o erro da saída $e = t_j - a$
4. Se e=0, volta ao passo 2
5. Se e<>0,
   1. Atualizar pesos $w_i = w_i^{old} + e \times x_i$
   2. Atualizar o bias $b = b^{old} + e$
   3. Voltar ao passo 2

Critério de parada: Todos os padrões classificados corretamente.

Perceptron - Exemplo

- Neste tipo de algoritmo é bom que os dados estejam misturados (shuffle)
- Considere o seguinte conjunto de aprendizagem.

X		t
2	2	0
-2	-2	1
-2	2	0
-1	1	1



Perceptron - Exemplo

- Nesse exemplo, foi inicializado os pesos e bias com 0, ou seja, w =(0,0) e b = 0

$$y = hard\lim\left([0,0]\begin{bmatrix}2\\2\end{bmatrix}+0\right) = hard\lim(0) = 1$$

	X	t
2	2	0
-2	-2	1
-2	2	0
-1	1	1

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Calcula-se o erro

$$e = t_i - y = 0 - 1 = -1$$

Como o erro é diferente de 0, atualizam se os pesos e o bias

$$W = W^{old} + e \times x_i = [0,0] + (-1[2,2]) = [-2,-2]$$

$$b = b^{old} + e = 0 + (-1) = -1$$

Apresentando o primeiro padrão (x_2) a rede:

$$y = hard\lim\left([-2,-2]\begin{bmatrix}-2\\-2\end{bmatrix}+(-1)\right) = hard\lim(7) = 1$$

	X	t
2	2	0
-2	-2	1
-2	2	0
-1	1	1

Calcula-se o erro

$$e = t_i - y = 1 - 1 = 0$$

Como o erro é 0, os pesos e o bias não precisam ser atualizados.

Apresentando o primeiro padrão (x_3) a rede:

$$y = hard\lim\left([-2,-2]\begin{bmatrix}-2\\2\end{bmatrix}+(-1)\right) = hard\lim(-1) = 0$$

	X	t
2	2	0
-2	-2	1
-2	2	0
-1	1	1

Calcula-se o erro

$$e = t_i - y = 0 - 0 = 0$$

Como o erro é 0, os pesos e o bias não precisam ser atualizados.

Apresentando o primeiro padrão (x_4) a rede:

$$y = hard\lim\left([-2,-2]\begin{bmatrix}-1\\1\end{bmatrix}+(-1)\right) = hard\lim(-1) = 0$$

	X	t
2	2	0
-2	-2	1
-2	2	0
-1	1	1

Calcula-se o erro

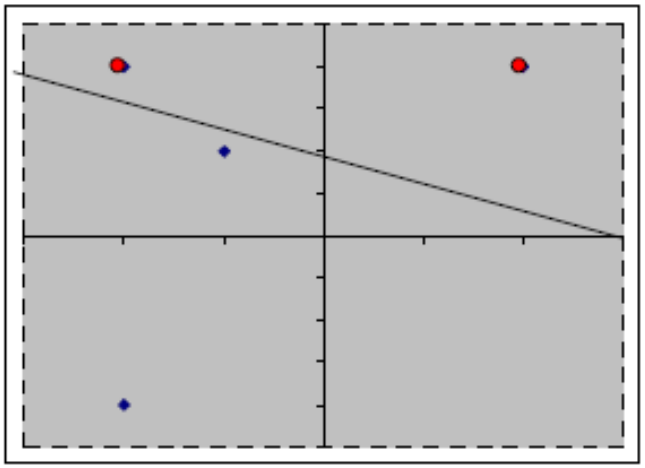
$$e = t_i - y = 1 - 0 = 1$$

$$W = W^{old} + e \times x_i = [-2,-2] + (1[-1,1]) = [-3,-1]$$

$$b = b^{old} + e = -1 + 1 = 0$$

Perceptron - Exemplo

- O processo acaba quando todos os padrões forem classificados corretamente.
- Para esse exemplo, os pesos finais são
  - w=[-1,-3] e b = 2.

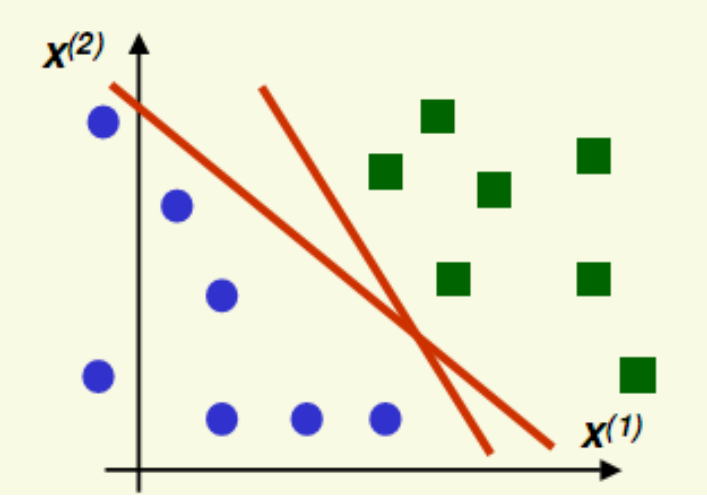


Programa Perceptron

- O programa lê um arquivo de treinamento (trainperceptron.txt) e busca a classificação perfeita do treinamento (ponto de parada)
- Os pesos e bias iniciais são randomicos
- O arquivo (saida perceptron.txt) mostra com quantas eras houve a classificacao perfeita e qual a reta que permite a separação das classes (com 20 inicializações diferentes)

SVM

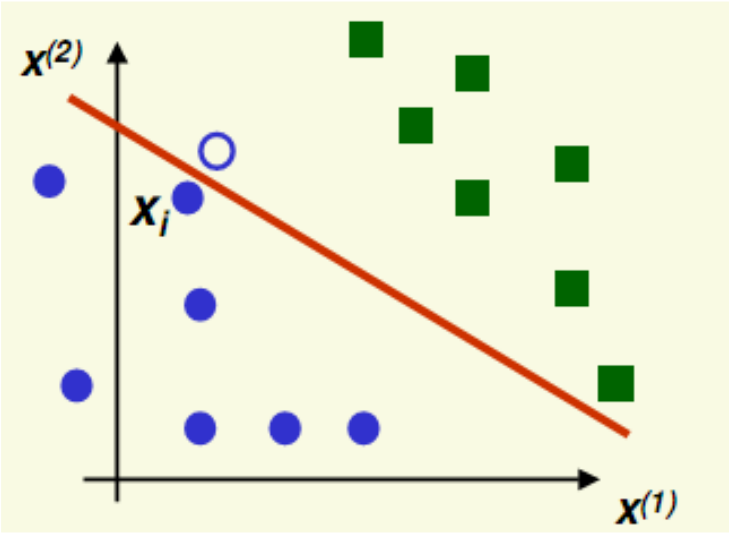
- O perceptron é capaz de construir uma fronteira se os dados forem linearmente separáveis.

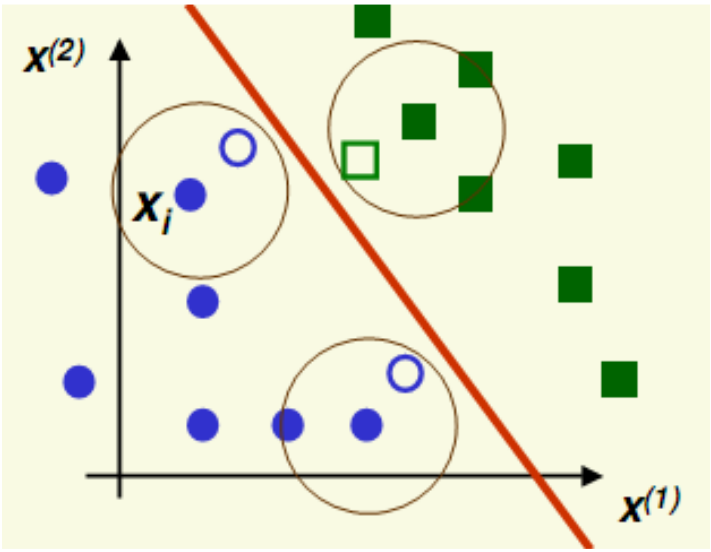


Mas qual a fronteira que deve ser escolhida ?

SVM

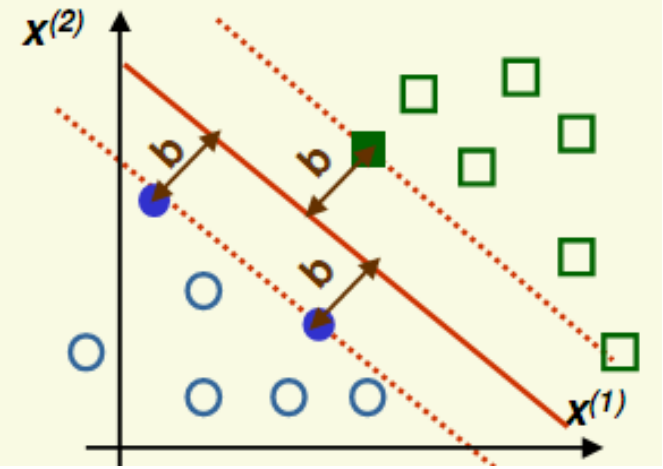
- Conforme a escolha da fronteira pode-se ter valores classificados erroneamente devido a um poder de generalização baixo





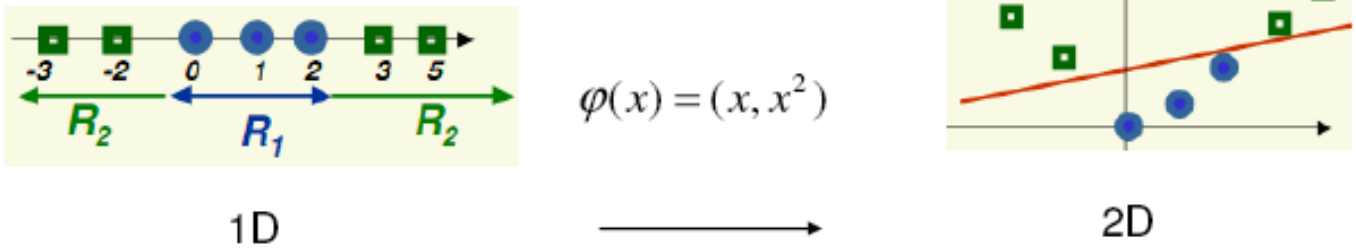
SVM

- A idéia do SVM é maximizar a distância da margem dos dados de treinamento, através dos vetores de suporte
- Os vetores de suporte são os exemplos da base de treinamento mais próximos do hiperplano que separa as classes

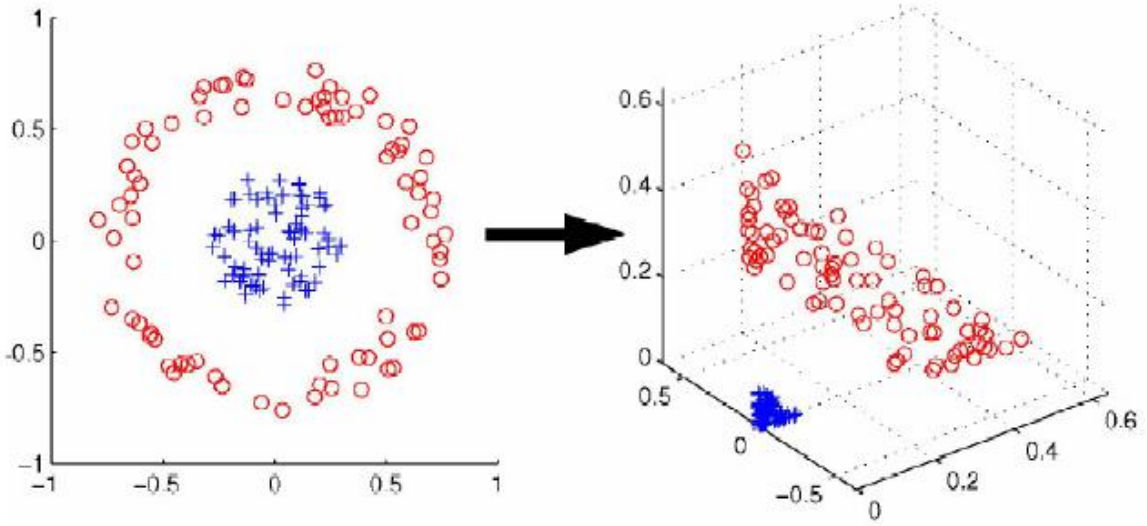


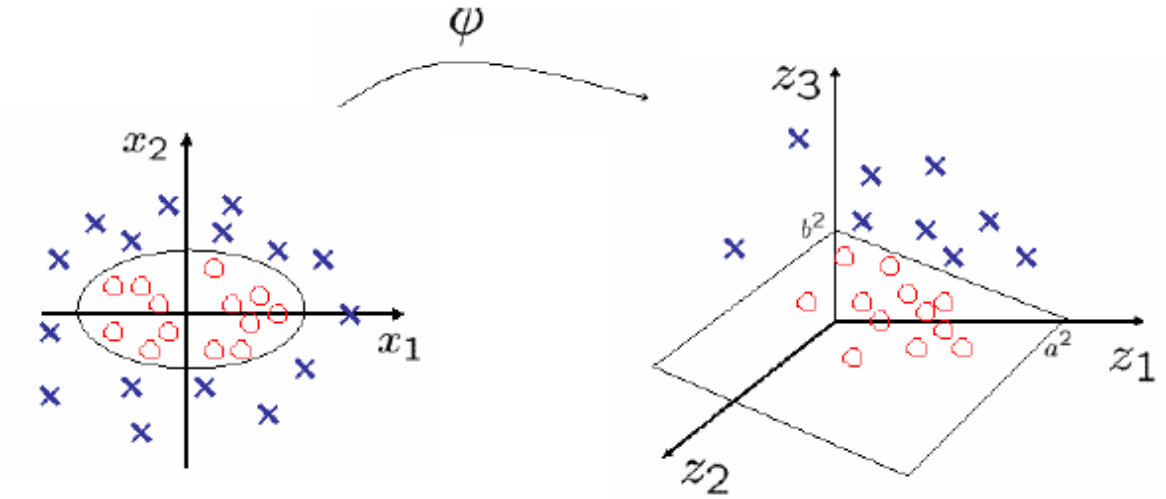
SVM

- Como a grande maioria dos problemas reais não são linearmente separáveis.
- A idéia do SVM é projetar esses dados em um espaço (outra dimensão) onde eles sejam linearmente projetados através de uma função de kernel



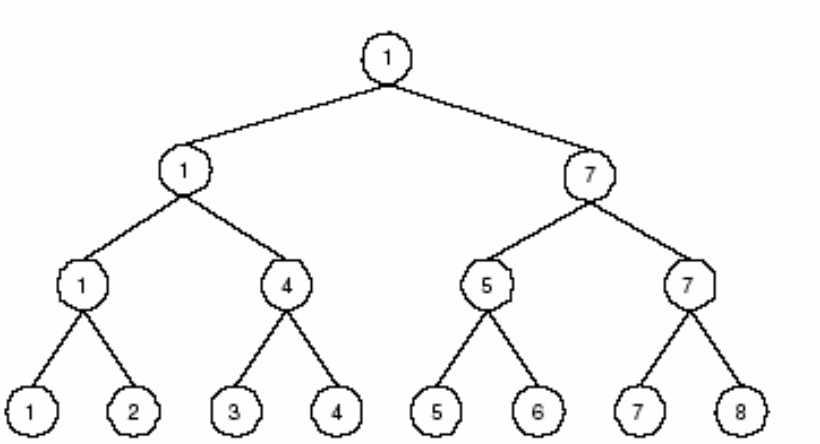
SVM


1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
0.6
0.4
0.2
0
0.5
0
-0.5
0
0.2
0.4
0.6


φ
x_2
x_1
z_3
b^2
a^2
z_1
z_2

SVM

- O SVM é um classificador binário, ou seja, separa apenas duas classes
- Porém a grande maioria dos problemas reais possuem mais que duas classes, ele usa um conceito de Pairwise (um-contra-todos)



SVM

- O SVM tem se mostrado muito eficiente, onde tem tido resultados melhores do que a grande maioria dos classificadores em diversas aplicações.

SVM - Ferramentas

- Existem várias ferramentas para usar essa técnica
  - http://www.smartlab.dibe.unige.it/Files/sw/Applet%20SVM/svmapplet.html
  - www.csie.ntu.edu.tw/~cjlin/libsvm/
- Através de um programa em python chamado easy.py define os melhores parâmetros para sua base de treinamento

SVM - Prática

- Baixe o libsvm e instale (descompacte) (www.csie.ntu.edu.tw/~cjlin/cgi-bin/libsvm.cgi?+http://www.csie.ntu.edu.tw/~cjlin/libsvm+zip)
  - Manual - www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
- Instale/baixe o gnuplot (sourceforge.net/project/showfiles.php?group_id=2055)
- Instale/baixe o python 2.7 (https://www.python.org/downloads/)

SVM - Prática

- Copie os arquivos abaixo no diretório “tools”
  - treinamento/teste/validação
    - 50% - treinamento e validação (70% treinamento e 30% validação)
    - 50% - teste
  - easyDiego.py
    - Ajustar linha 25 – diretório gnuplot
  - grid.py
    - Ajustar linha 27 – diretório gnuplot
  - Testa2.bat

Arquivos Gerados

- .scale – Dados normalizados.
- .range – Valores mínimos e máximos dos atributos.
- .scale.out – Resultados parciais obtidos com diferentes parâmetros durante o treinamento.
- .scale.png – Gráfico com a variação dos resultados com diferentes parâmetros durante o treinamento.
- .predict – Resultado da classificação dos exemplos de teste.
- .model – Modelo do classificador treinado.

SVM-SCALE


```
48    cmd = "%s -s %s %s > %s" % (svmscale_exe, range_file, train_pathname, scaled_file)
```

- Normaliza o arquivo de treinamento (train_pathname) gerando o arquivo “range” que guarda os maiores e menores valores de cada atributo e por fim gera o arquivo “scale” com os dados de treinamento normalizados de -1 a 1

```
Usage: svm-scale [options] data_filename
options:
-l lower : x scaling lower limit (default -1)
-u upper : x scaling upper limit (default +1)
-y y_lower y_upper : y scaling limits (default: no y scaling)
-s save_filename : save scaling parameters to save_filename
-r restore_filename : restore scaling parameters from restore_filename
```

Grid.Py


```
52    cmd = "%s -svmtrain %s -gnuplot %s %s" % (grid_py, svmtrain_exe, gnuplot_exe, scaled_file)
```

- Encontra os melhores valores de C e gamma
  - O C refere-se ao quão branda é a margem do SVM
  - O gamma é um parâmetro usado nos Kernels da SVM

Kernel	Formula	Parameters	R name
Linear	$\mathbf{u}^T\mathbf{v}$	none	
Polynomial	$\gamma(\mathbf{u}^T\mathbf{v} + c_0)^d$	γ, d, c_0	gamma=γ coef0=c_0 degree=d
Gaussian Radial basis fct.	$exp[-\gamma\|\mathbf{u} - \mathbf{v}\|^2]$	γ	gamma=γ
Sigmoid	$\tanh[\ \gamma[\mathbf{u}^T\mathbf{v} + c_0]]$	γ, c_0	gamma=γ coef0=c_0

SVM-TRAIN


```
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC              (multi-class classification)
        1 -- nu-SVC             (multi-class classification)
        2 -- one-class SVM
        3 -- epsilon-SVR        (regression)
        4 -- nu-SVR             (regression)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
        4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates : whether to train a SVC or SVR model for probability e
stimates, 0 or 1 (default 0)
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-q : quiet mode (no outputs)
```

SVM-TRAIN


```
65    cmd = "%s -c %s -g %s -b 1 %s %s" % (svmtrain_exe,c,g,scaled_file,model_file)
```

- Faz o treinamento do SVM definindo os parâmetros c e gamma encontrados no grid.py trazendo as estimativas de probabilidade (-b 1) baseado no arquivo “scale” e gera o modelo

SVM-PREDICT


```
75      cmd = "%s -b 1 %s %s %s" % (svmpredict_exe, scaled_test_file, model_file, predict_test_file)
```

- Faz a predição retornando a probabilidade estimada para cada classe, baseando-se no modelo previamente definido

```
Usage: svm-predict [options] test_file model_file output_file
options:
-b probability_estimates: whether to predict probability estimates, 0 or 1 (defa
ult 0); for one-class SVM only 0 is supported
-q : quiet mode (no outputs)
```

Testa2.bat

- python easyDiego.py treinamentoLbpFelibSVM.txt validacaoLbpFelibSVM.txt
- ..\windows\svm-scale -r treinamentoLbpFelibSVM.txt.range testeLbpFelibSVM.txt > testeLbpFelibSVM.txt.scale
- ..\windows\svm-predict -b 1 testeLbpFelibSVM.txt.scale treinamentoLbpFelibSVM.txt.model testeLbpFelibSVMLbpFelibSVM.txt.predict > resultadoLbpFelibSVM

.predict

Classes possíveis

```
labels 0 1 2 3 4 5 6 7 8 9
5 0.212199 0.0143314 0.00741456 0.00714644 0.0915952 0.613157 0.0436371 0.00374805 0.00409676 0.00267477
0 0.680876 0.00303281 0.00352728 0.00199059 0.0692076 0.218749 0.018619 0.00140381 0.00176898 0.000825302
0 0.934503 0.00104625 0.00561463 0.000547298 0.0279716 0.0227777 0.00401205 0.00104242 0.000668837 0.00181567
```

Classe mais votada

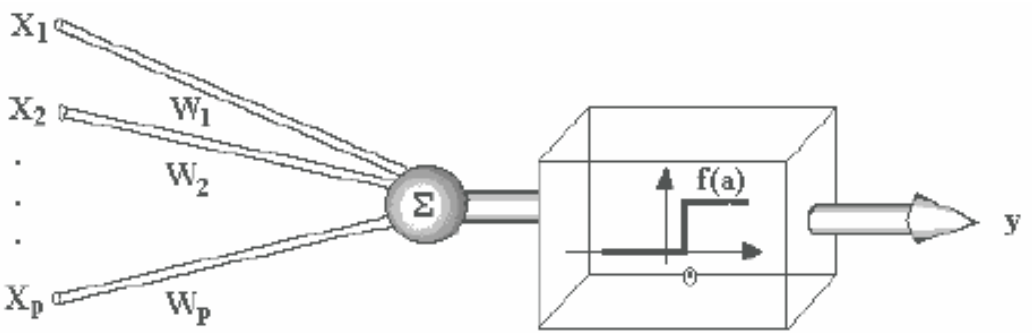
"percentual de retorno" de cada classe

Redes Neurais Artificiais (RNA)

- Fornecem um método geral e prático para a aprendizagem de funções de valor real e de valor discreto a partir de exemplos.
- A aprendizagem de RNA é robusta a erros e ruídos nos dados de treinamento.
- Modelo inspirado na aprendizagem de sistemas biológicos (redes complexas de neurônios interconectados.)

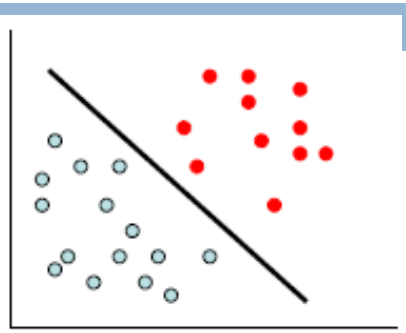
Neurônio Artificial

- Sinais são apresentados à entrada.
- Cada sinal é multiplicado por um peso.
- Soma ponderada produz um nível de ativação.
- Se esse nível excede um limite (threshold) a unidade produz uma saída.

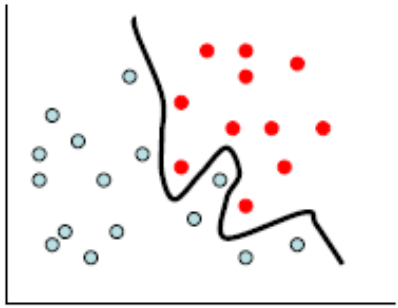


[PERCEPTRON]

Neurônio Artificial



- O perceptron só resolve problemas linea separáveis, porém, os problemas reais, n das vezes são mais complexos.
- Se for criada uma nova dimensão os problemas tornam-se linearmente separáveis (SVM), em RNA, isso é feito usando camadas escondidas



Multi-Layer Perceptron (MLP)

- Refere-se a várias camadas de perceptrons
- A adição de uma camada extra (escondida) entre as camadas de entrada e saida pode ser vista como uma característica a mais que aumenta a dimensionalidade do vetor
- Dado uma quantidade suficiente de neurônios na camada escondida, é possível resolver “qualquer” tipo de problema

Redes Multicamadas

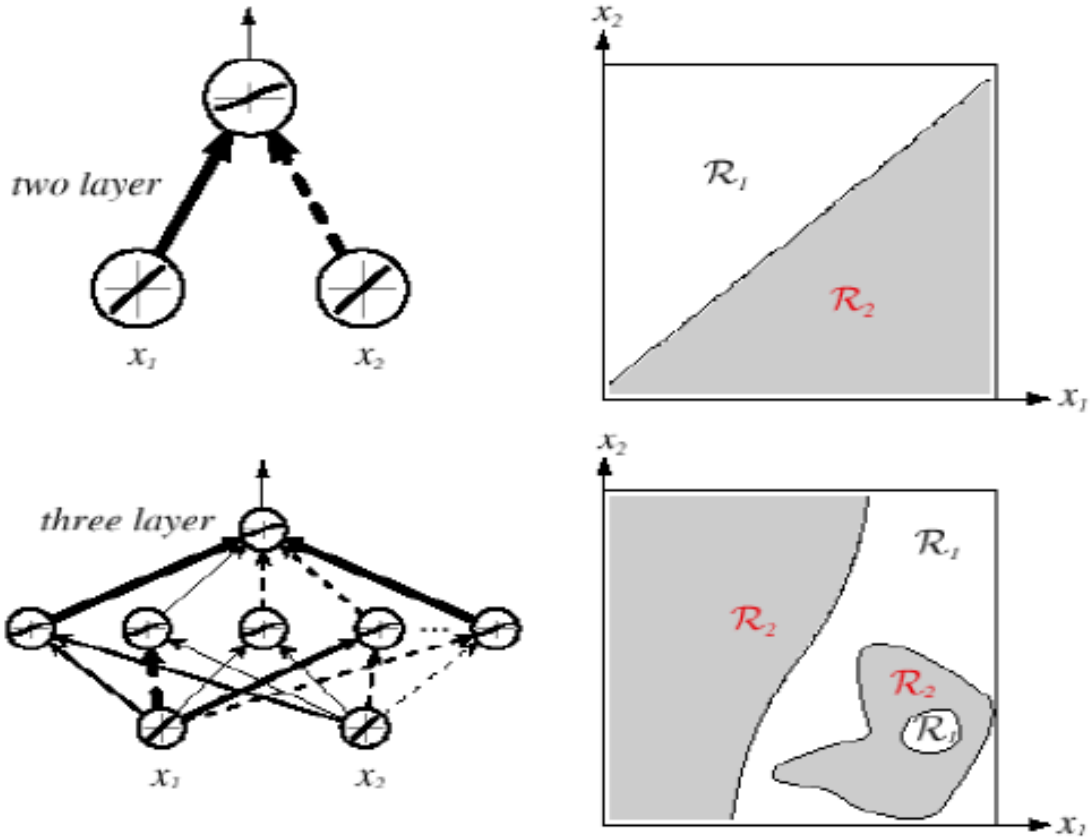


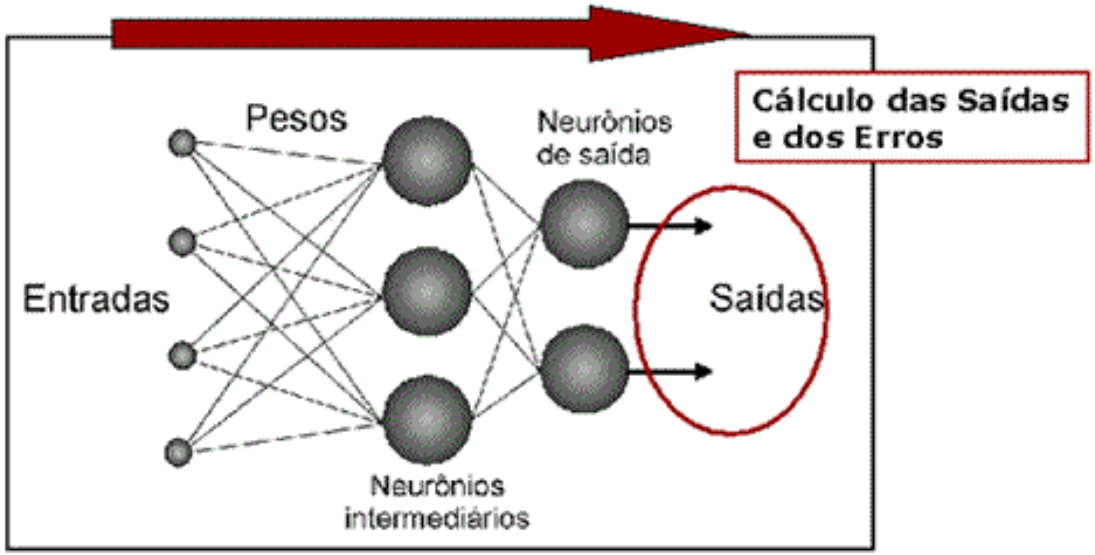
FIGURE 6.3. Whereas a two-layer network classifier can only implement a linear decision boundary, given an adequate number of hidden units, three-, four- and higher-layer networks can implement arbitrary decision boundaries. The decision regions need not be convex or simply connected. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Algoritmo Backpropagation

- É o algoritmo para treinamento de Redes Multi-Camadas mais difundido
- Pode ser dividido em duas etapas
  - Propagação
  - Retropropagação

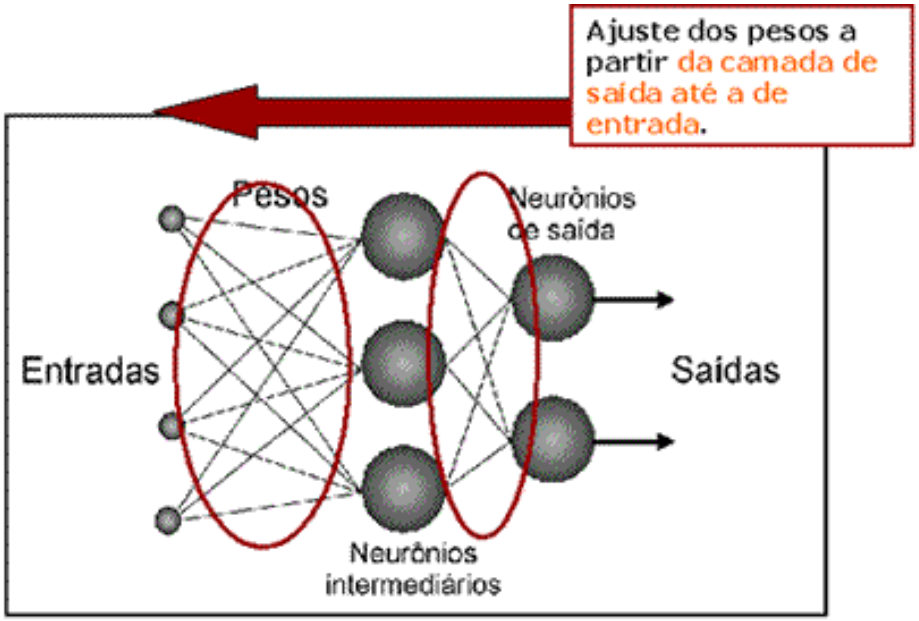
Propagação

- Depois de apresentado o padrão de entrada, a resposta de uma unidade é propagada como entrada para as unidades na camada seguinte, até a camada de saída, onde é obtida a resposta da rede e o erro é calculado



Retropropagação

- Desde a camada de saída até a camada de entrada, são feitas alterações nos pesos sinápticos

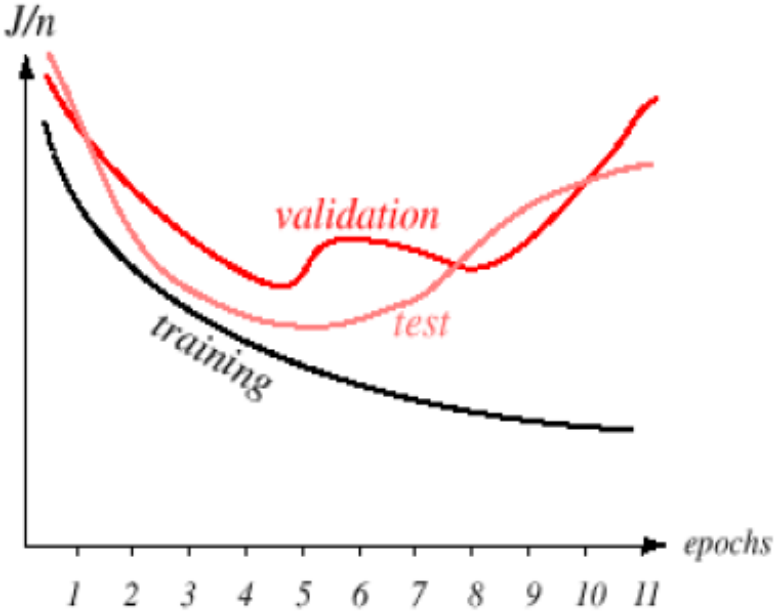


Algoritmo Backpropagation

- Durante o treinamento apresenta um conjunto de características e o valor da resposta.
- A saída é comparada à resposta e é calculado o erro global da rede, que influenciará na correção dos pesos no passo de retropropagação.
- Apesar de não haver garantias que a rede forneça uma solução ótima para o problema, este processo é muito utilizado por apresentar uma boa solução para o treinamento de MLP

Generalização x Sobreajuste

- A linha inferior mostra o decréscimo do erro sobre os exemplos de treinamento em função do número de iterações de treinamento (Erro de Aprendizagem)
- A linha superior mostra o erro medido sobre exemplos de validação (Precisão da Generalização)



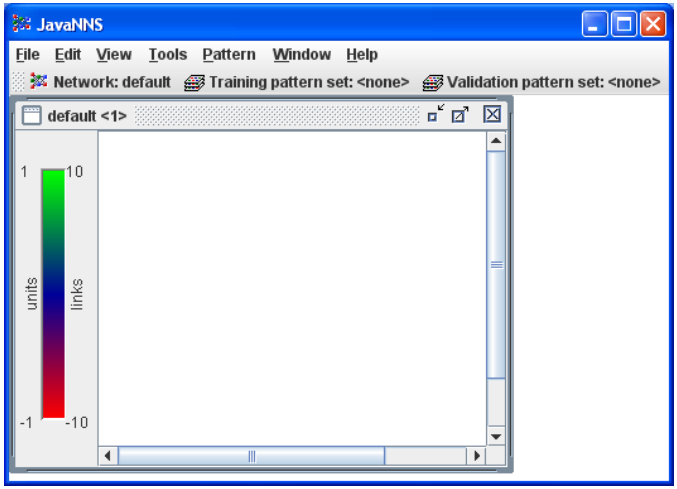
Prática

- Na pasta javanns tem-se:
  - JavaNNS-win – Simulador de RNA
  - tr300.pat – base de treinamento
  - vl300.pat – base de validação
  - Analyze – Interpretador de resultados JavaNNS
  - Snns – gera os resultados na linguagem C

Treinamento

- Na base tem-se 300 elementos com 132 características (vetor de entrada) de letras manuscritas, gerando 26 possíveis classes (vetor de saída)
- A arquitetura da rede depende do vetor de caracteristica e do número de classes.
- Para se definir a quantidade de elementos na camada oculta usa-se “tentativa e erro”
  - Para iniciar (entrada + saida)/2 → (132+26)/2 = 79 neurônios na camada oculta.

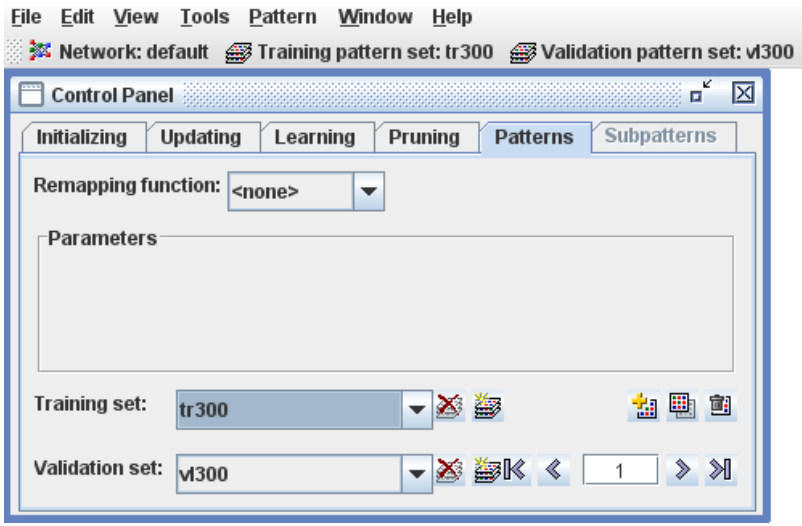
Construindo a rede

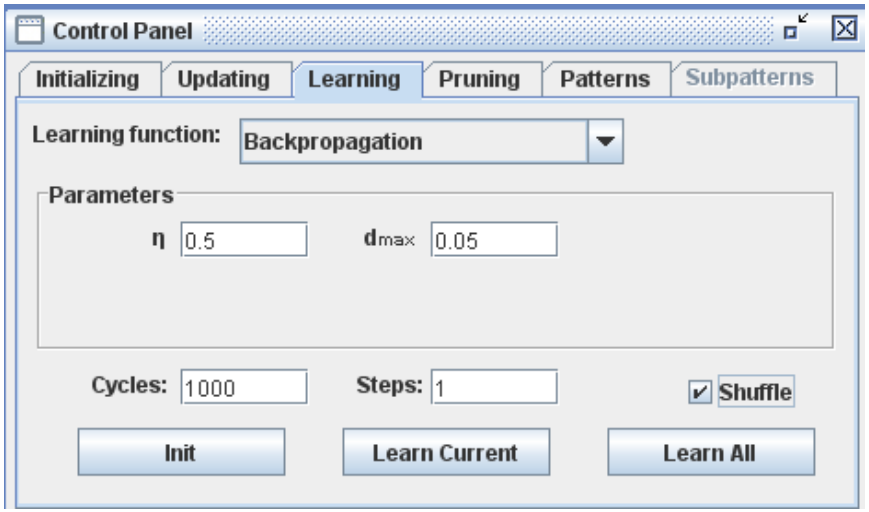


- Abra o JavaNNS
- File->Open
  - Marque tr300.pat e vl300.pat
- Tools->Create->Layers
  - Entrada (Height = 132 e Unit Type = Input)
  - Oculta (Height = 79 e Unit Type = Hidden)
  - Saída (Height = 26 e Unit Type = Output)
- Tools->Create->Connections
  - Connect feed-forward

Configurando a Rede

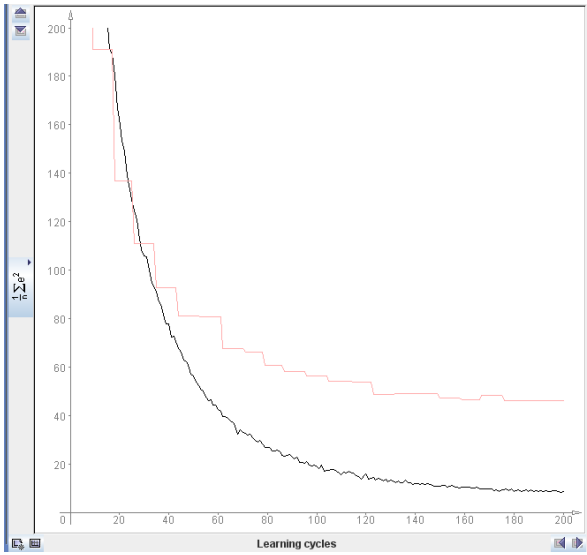
- Tools->Control Panel->Patterns
  - Training set – tr300
  - Validation set – vl300
- Na paleta Learning



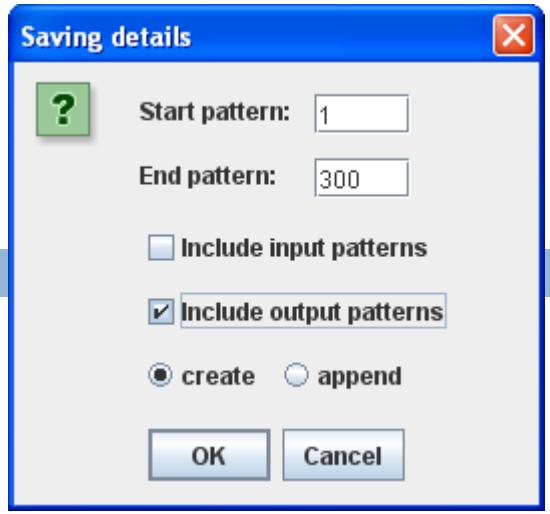


Treinando a rede

- Antes de iniciar o treinamento minimize a tela da rede (o processo é mais rápido)
- Vá em View->Error Graph
  - Isso permitirá acompanharmos a evolução do treinamento e validação
  - Escolha o erro médio quadrático $\frac{1}{n}\sum e^2$
- Clique no botão "Init"
  - Irá randomizar os pesos iniciais
- Clique no botão "Learn All"



Testando a rede



- Salve a rede
  - File->Save as
- Para testar a rede precisa-se gerar um arquivo de resultado
  - File->Save data
  - Inclua os padrões de saída
- Agora para testar os reultados usa-se o programa "Analyze"
  - Analyze –scvm –e WTA –i arq.res

Resultado da rede


```
STATISTICS ( 300 patterns )
wrong   :  1.00 %  ( 3 pattern(s) )
right   : 99.00 %  ( 297 pattern(s) )
unknown :  0.00 %  ( 0 pattern(s) )
error   : 5.221146


1ST ORDER STATISTICS FOR CLASS NO. : 0
wrong   :  0.00 %  ( 0 pattern(s) )
right   : 100.00 %  ( 12 pattern(s) )
unknown :  0.00 %  ( 0 pattern(s) )
```

```
1ST ORDER STATISTICS FOR CLASS NO. : 6
wrong   :  0.00 %  ( 0 pattern(s) )
right   : 100.00 %  ( 12 pattern(s) )
unknown :  0.00 %  ( 0 pattern(s) )

1ST ORDER STATISTICS FOR CLASS NO. : 7
wrong   : 16.67 %  ( 2 pattern(s) )
right   : 83.33 %  ( 10 pattern(s) )
unknown :  0.00 %  ( 0 pattern(s) )


1ST ORDER STATISTICS FOR CLASS NO. : 15
wrong   :  9.09 %  ( 1 pattern(s) )
right   : 90.91 %  ( 10 pattern(s) )
unknown :  0.00 %  ( 0 pattern(s) )
```

CONFUSION MATRIX (rows: teaching input, colums: classification)

class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	unknown
0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
noclass	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Outros Exemplos

www.bostondynamics.com

www.cs.cmu.edu/afs/cs/project/alv/www/index.html

0:30
3:00
5:30

https://www.youtube.com/user/BostonDynamics

Exercícios

- Usando a base de letras, e as características extraídas na semana passada, apresente um relatório com as matrizes de confusão e os resultados usando ao menos dois classificadores