



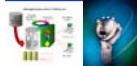
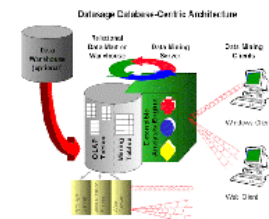
Robert Groth



## Métodos e Algoritmos de Data Mining(parte 3)



Usama Fayyad et al



## Métodos e Algoritmos de Data Mining

- Soluções distância (K-vizinhos mais próximos)
- Naïve-Bayes
- Árvores de decisão
- Regras de associação
- ⇒ Redes neurais
- ⇒ Algoritmos genéticos.
- ⇒ Combinação de múltiplos métodos de predição.
  
- ⇒ Alguns prós e contras das tecnologias mais comuns;  
ferramentas mais relevantes e suas características



# Redes Neuronais



## A Aprendizagem e Técnicas de Machine Learning

A aprendizagem é um aspecto tão importante da natureza que inevitavelmente é fulcral em vários domínios do estudo dos seres vivos e fornece-nos modelos para o estudo do próprio comportamento da aprendizagem.

É interessante notar que duas das técnicas mais importantes no domínio da machine learning, sejam devidas ao estudo dos seres vivos:

- redes neuronais
- algoritmos genéticos.

A primeira centra-se em modelos do cérebro humano, sendo a segunda uma simulação da evolução das espécies.



## Redes Neurais

**Redes Neurais - Programa de computador que implementa detecção de padrões sofisticados e algoritmos de aprendizagem, permitindo construir modelos predictivos a partir de grandes bases de dados históricas.**

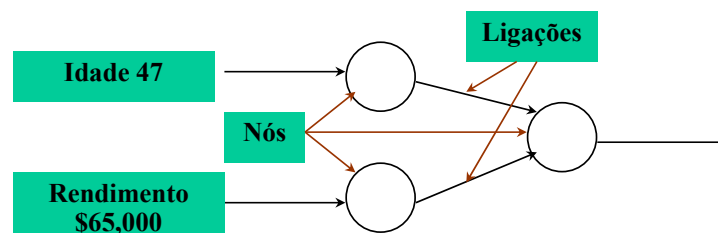
**Derivam o seu nome do paralelismo da sua arquitectura e a arquitectura do cérebro humano:**

- O cérebro humano consiste num grande número de neurónios (cerca de  $10^{11}$ ), ligados através de um largo número de sinapses (1 neurónio pode dispor de milhares delas).
- A rede de neurónios e as suas sinapses constitui um excelente modelo que pode ser implementado artificialmente:
  - através de hardware especial
  - utilizando programas de software

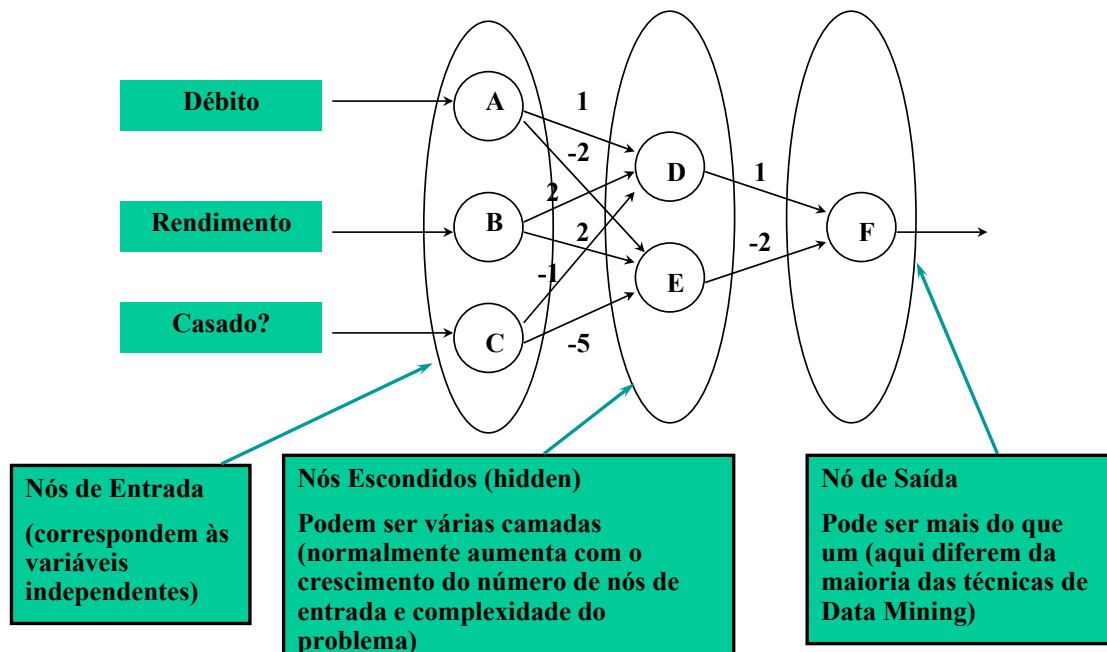


## Aspecto da Rede Neuronal

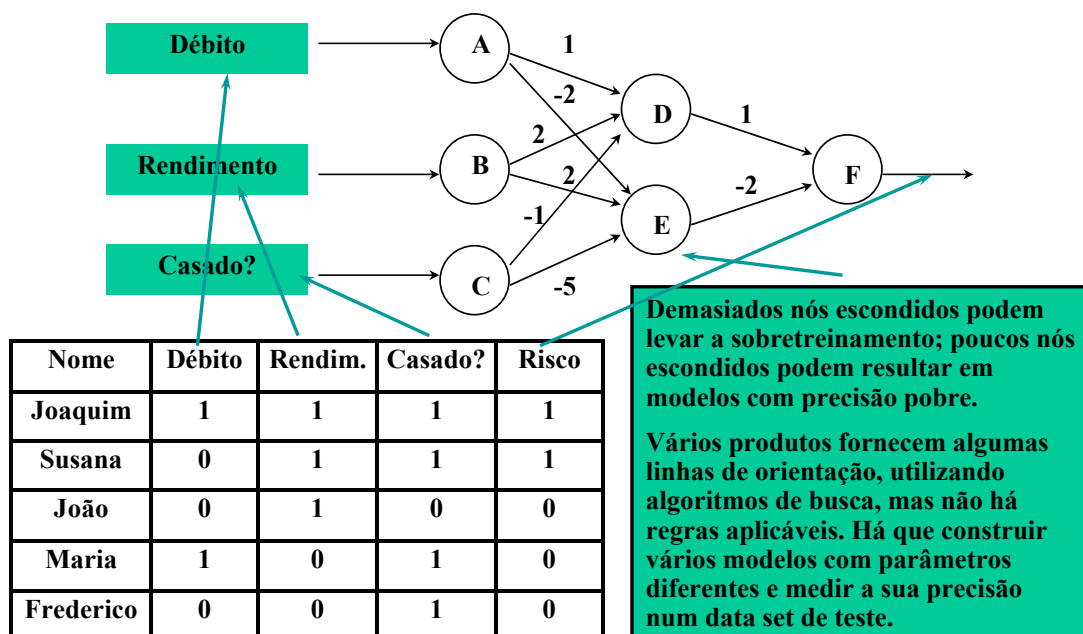
- **Constituída por uma série de nós (que correspondem de perto aos neurónios do cérebro humano):**
  - nós de entrada - recebem os sinais de entrada,
  - nós de saída - fornecem os sinais de saída;
- **Um potencialmente número ilimitado de camadas intermédias que contém nós intermédios;**
- **Ligações - que correspondem às ligações entre neurónios (dendritos e sinapses) no cérebro humano.**



## Outra Rede Neuronal



## Outra Rede Neuronal



## Topologia ou Arquitectura de Rede Neuronal

- O número de nós de entrada, saída e escondidos é muitas vezes designado como topologia da rede ou ainda, arquitectura da rede.
- O arranjo dos nós em camadas é comum, mas não essencial.
- A rede pode ser completamente conectada (em que cada nó numa camada tem ligações para todos os nós da próxima camada), arranjo que, embora típico, não é obrigatório.
- Pesos:
  - cada seta dos dois modelos apresentados mostra números (excepto nas setas de entrada e saída)
  - podem ser positivos ou negativos
  - normalmente são restringidos a valores pequenos
  - são tipicamente números reais com decimais



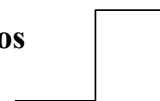
## Como é Calculada a Saída de um Nó

Há dois componentes:

- 1. O peso (visto atrás)
  - permite calcular a saída de um nó como a soma pesada das entradas;
- 2. Uma função de squashing - que converte a soma pesada das entradas (calculadas em 1) num valor de saída, podendo ser a) descontínuas (tipo step) ou b) contínuas (sigmóide).

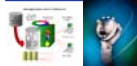
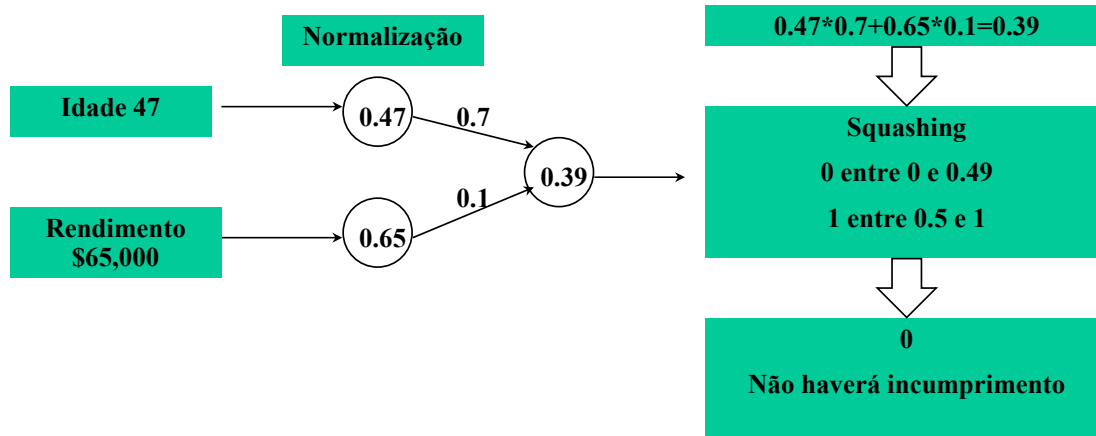
a) a saída será 1 se o somatório de  $w_i x_i > 0$  e -1 nos outros casos

b) a saída será  $1 / (1 + e^{-\text{somatório } w_i x_i})$



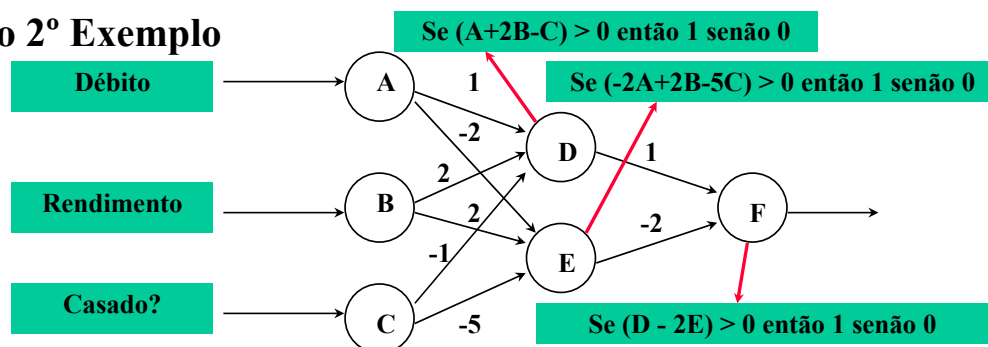
## Como se Faz a Previsão na Rede Neuronal

### Para o 1º Exemplo



## Como se Faz a Previsão na Rede Neuronal

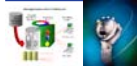
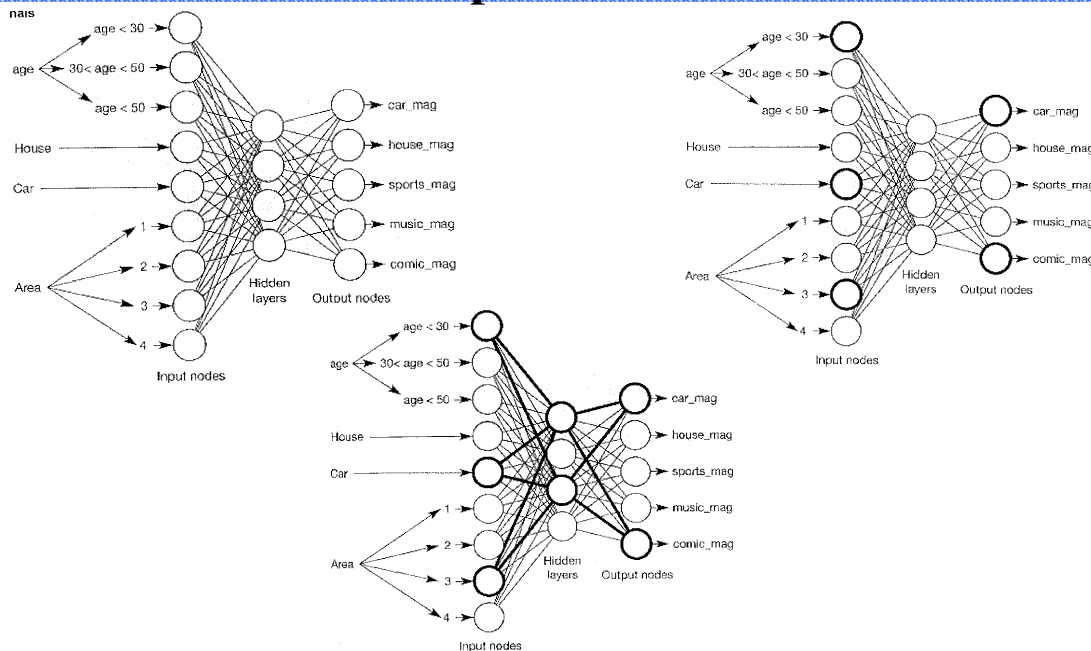
### Para o 2º Exemplo



| Nó        | A      | B       | C       |       | D | E | F |
|-----------|--------|---------|---------|-------|---|---|---|
| Nome      | Débito | Rendim. | Casado? | Risco |   |   |   |
| Joaquim   | 1      | 1       | 1       | 1     | 1 | 0 | 1 |
| Susana    | 0      | 1       | 1       | 1     | 1 | 0 | 1 |
| João      | 0      | 1       | 0       | 0     | 1 | 1 | 0 |
| Maria     | 1      | 0       | 1       | 0     | 0 | 0 | 0 |
| Frederico | 0      | 0       | 1       | 0     | 0 | 0 | 0 |



## Outro Exemplo de Rede Neuronal



## Generalidades Sobre Redes Neurais (1)

- Estão entre os algoritmos mais complicados para classificação e regressão
- O processo de treino é demorado, mas a predição é rápida:
  - pode ser em tempo real (ex. na detecção de utilização fraudulenta de cartão de crédito).
- Utilização bastante alargada:
  - no domínio de análise de dados para negócio:
    - classificação e regressão,
    - clustering;
  - controlo de robots;
  - reconhecimento de escrita e voz.
- Em classificação e regressão
  - a saída de uma rede neuronal é puramente predictiva.





## Generalidades Sobre Redes Neurais (2)

- **Não há componente descritiva no modelo da rede neuronal:**
  - levam a que as escolhas da rede neuronal sejam difíceis de entender
  - desencorajam a sua utilização
  - esta técnica é muitas vezes denominada de “tecnologia de caixa preta”
- **Opera directamente sobre números**
  - quaisquer colunas (independentes ou dependentes) devem ser convertidas para números antes de podermos utilizar uma rede neuronal
- **Ainda que baseado no modelo do cérebro humano, não é mais do que uma função matemática que calcula uma saída baseada num conjunto de valores de entrada**
- **O paradigma de rede torna fácil a decomposição de funções complexas num conjunto de subfunções relacionadas, permitindo aos algoritmos de aprendizagem estimar os parâmetros das subfunções**



## Score para as Redes Neurais

**Automatização** - São muito poderosas na criação de modelos predictivos, requerem significativo pré-processamento dos dados e boa compreensão do alvo de predição. Requer normalmente a normalização dos predictores para valores entre 0 e 1 e a conversão de variáveis categóricas para numéricas. Também requerem a especificação de vários parâmetros que determinam a forma como a rede será construída.

**Clareza** - A desvantagem das redes neuronais é a falta de clareza: que os utilizadores possam ver e compreender os resultados que são apresentados. Esta deficiência é reconhecida e várias implementações comerciais fornecem técnicas poderosas para melhor visualizar os resultados e proporcionar regras compreensíveis ou protótipos para ajudar a compreender os modelos.

**ROI** - Proporcionam modelos predictivos poderosos que, em teoria, são mais gerais do que os de outras técnicas de data mining e estatística. Na prática, os ganhos em precisão são pequenos, dependentes da habilidade do utilizador e muito demorados na sua aprendizagem.





## Treino da Rede Neuronal

Já vimos atrás como é que a rede neuronal faz a predição.

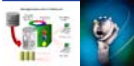
- Há dois componentes:
  - Pesos;
  - Função squash.

Mas:

- como são determinados os pesos?

O modelo da rede neuronal é criado apresentando-lhe muitos exemplos (variáveis de entrada e de saída dos registos do set de treino):

- Comparando a resposta correcta do registo de treino com a resposta obtida da rede neuronal, é possível fazer pequenas alterações ao comportamento da rede neuronal, alterando os pesos da ligações.



## Calcular os Acertos a Fazer aos Pesos (1)

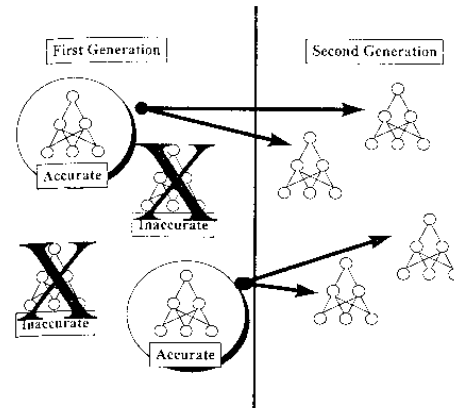
- O cálculo é complicado, porque se trata de um problema de pesquisa significativo, especialmente em redes de arquitectura complexa:
  - Os pesos são valores numéricos contínuos, havendo literalmente um número infinito de combinações de valores de pesos;
  - Por outro lado, nas camada hidden, e dadas as ligações múltiplas entre os nós, calcular a contribuição para o erro devido a cada link torna-se uma tarefa no mínimo complicada.
- São utilizadas várias técnicas, entre as quais:
  - algoritmos genéticos;
  - classe de algoritmos gradient descent.



## Calcular os Acertos a Fazer aos Pesos (2)

### Algoritmos Genéticos:

- trata-se no fundo de permitir uma evolução nos pesos, utilizando os princípios da selecção natural (neste caso sobrevivem e reproduzem-se as redes cujos pesos forneçam um erro mais pequeno); será o algoritmo que estudaremos na próxima secção deste capítulo;
- não foi demonstrada ainda a sua melhor precisão e rapidez de treino.



### Algoritmos Gradient Descent:

- classe de algoritmos, cujo mais corrente é chamado de backpropagation ou backprop, de que trataremos mais adiante, como exemplo de algoritmo de rede neuronal.



## Tipos de Redes Neurais

Há várias formas de redes neurais, mas aqui iremos discutir três delas:

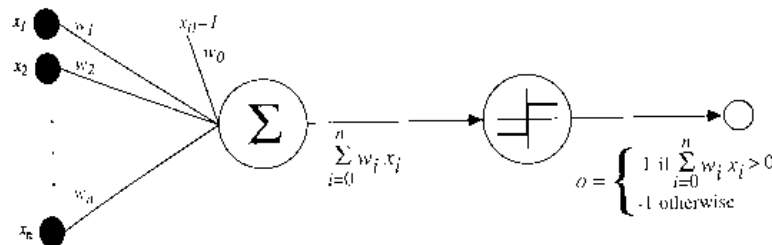
- Perceptrão
- Redes com “back propagation”
- Redes de Kohonen



## Redes Neurais: Perceptrão

### Desenvolvido em 1958 por Frank Rosenblat

- Rede neuronal simples (só uma unidade), que servirá aqui para compreender como são aprendidos os pesos;
- Está apto a aprender categorias simples;
- Pode ser utilizado para tarefas de classificação simples.



**Funções Booleanas Executadas: AND, OR, NAND, NOR**  
**Falha a função XOR**



## Redes Neurais: Perceptrão

### Algoritmo Informal:

1. Já vimos que poderemos iniciar a aprendizagem com pesos aleatórios;
2. Aplicar iterativamente cada exemplo de treino ao perceptrão;
3. Modificar os pesos do perceptrão, sempre que haja erro na classificação de um exemplo;

$$w_i \leftarrow w_i + \Delta w_i$$

em que  $\Delta w_i = \eta(t - o) x_i$

$t$  - saída alvo do exemplo corrente (valor real)

$o$  - saída gerada pelo perceptrão

$\eta$  - constante positiva denominada taxa de aprendizagem (para moderar o grau de alteração dos pesos - normalmente de pequeno valor (0.1), podendo mesmo ser sucessivamente decrementada para ajuste fino final)

4. O processo é repetido, iterando através dos exemplos de treino tantas vezes quantas as necessárias até que o perceptrão classifique todos os exemplos de treino correctamente.



## Redes Neurais: Percepção

### Em resumo:

No percepção, a convergência do processo (ou seja a minimização do erro através da alteração dos pesos), só é assegurada se:

- a taxa de aprendizagem for suficientemente pequena;
- desde que os exemplos de treino estejam separados linearmente (daí a falha em executar a função XOR (não linearmente separável).

Logo em 1969, foi demonstrado que a classe de problemas que poderiam ser resolvidos por uma máquina com arquitectura percepção eram muito limitados (daí a falha em executar a função XOR - não linearmente separável).

Solução: Há que tornar a rede mais complexa (implementar uma arquitectura mais sofisticada):

- introdução de arquitecturas com camadas escondidas (hidden), no que se chamou de redes de backpropagation;
- introduzir não linearidade no limiar.



## Redes Multi-camadas e Algoritmo Backpropagation

Capazes de expressar uma variedade rica de superfícies de decisão não lineares.

| STRUCTURE        | TYPES OF DECISION REGIONS                         | EXCLUSIVE OR PROBLEM | CLASSES WITH MESHEDED REGIONS | MOST GENERAL REGION SHAPES |
|------------------|---------------------------------------------------|----------------------|-------------------------------|----------------------------|
| SINGLE-LAYER<br> | HALF PLANE BOUNDED BY HYPERPLANE                  |                      |                               |                            |
| TWO-LAYER<br>    | CONVEX OPEN OR CLOSED REGIONS                     |                      |                               |                            |
| THREE-LAYER<br>  | ARBITRARY (Complexity Limited By Number of Nodes) |                      |                               |                            |



## Redes Multi-camadas - Unidade Utilizada

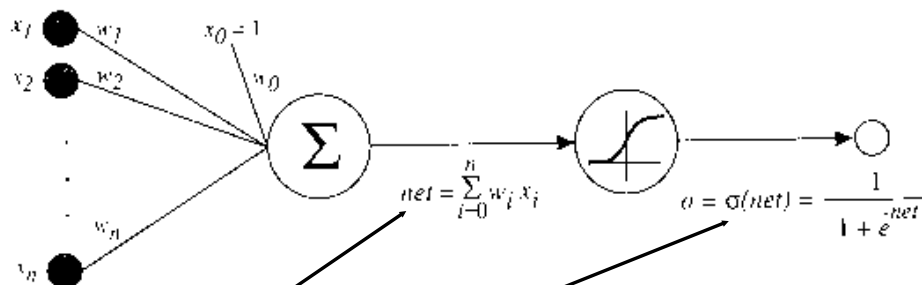
### Alternativas para o tipo de unidade utilizada:

- unidade linear utilizada no perceptrão:
  - mesmo com camadas múltiplas de unidades lineares em cascata, continuamos a produzir somente funções lineares;
- unidade não linear:
  - com unidades não lineares, seremos capazes de construir redes capazes de representar funções altamente não lineares.

**Assim: necessitamos de uma unidade cuja saída será uma função não linear das entradas - uma solução chamada de sigmóide - parecida com o perceptrão, mas utilizando uma função limiar diferenciável.**



## Redes Multi-camadas - Unidade Utilizada



1. calcular a combinação linear das entradas;
2. aplicar o limiar ao resultado anterior, em que:

limiar ( $\sigma$ ) será a função sigmóide ou logística cujo output varia entre 0 e 1, crescendo monotonicamente com as entradas.

além da função sigma mostrada na figura, são utilizadas outras variantes :  
 $1 / (1 + e^{-knet})$ , em que  $k$  é uma constante que determina o step do limiar  
ou ainda a função tanh



## Algoritmos de Backpropagation

**Aprende os pesos duma rede multi-camada,  
com um conjunto fixo de unidades e  
interligações;**

**Problema: com as camadas múltiplas, a função  
erro calculada pode ter vários mínimos locais  
e, assim, não convergir obrigatoriamente  
para o menor erro global;**

**São assim empregues alguns truques para  
minimizar este problema.**



## Algoritmo Backpropagation

**Backpropagation (exemplos\_de\_treino,  $\eta$ ,  $n_{in}$ ,  $n_{out}$ ,  $n_{hidden}$ )**

cada exemplo de treino é um par de forma  $(x, t)$  onde  $x$  é um vector de valores de entrada na rede e  $t$  o vector de valores das saídas

a entrada da unidade  $i$  na unidade  $j$  é chamado de  $x_{ij}$  e o peso da unidade  $i$  para a unidade  $j$  é chamado de  $w_{ji}$

- criar uma rede feed-forward com  $n_{in}$  entradas,  $n_{hidden}$  unidades hidden e  $n_{out}$  unidades de saída
- inicializar todos os pesos da rede para pequenos números aleatórios (ex. entre -0.5 e 0.5)

- Até que a condição de finalização seja satisfeita, Fazer

- Para cada  $(x, t)$  no exemplo de treino, Fazer

- propagar a entrada para a frente através da rede

- 1. aplicar a instância  $x$  à rede e calcular a saída  $o_u$  de cada unidade  $u$  da rede

- propagar o erro para trás, através da rede

- 2. para cada saída  $k$  da rede, calcular o termo do erro  $\Delta_k$

$$\Delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

- 3. para cada unidade  $h_{hidden}$

$$\Delta_h \leftarrow o_h(1 - o_h) \sum w_{kh} \Delta_k$$

- 4. actualizar cada peso da rede  $w_{ji}$

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}, \text{ onde } \Delta w_{ji} = \eta \Delta_{j_{xji}}$$





## Algoritmo Backpropagation

### Explicação:

- começa-se por construir a rede com o número desejado de nós hidden e unidades de saída. inicializado os pesos com valores aleatórios;
- o loop principal repete iterativamente para cada um dos exemplos de treino;
- para cada exemplo de treino que é aplicado à rede:
  - é calculado o erro para esse exemplo
  - calcula-se o gradiente (cada peso é actualizado na proporção da taxa de aprendizagem  $\eta$ , o input  $x_{ji}$  a que o peso é aplicado e o erro na unidade de saída) com respeito ao erro do exemplo
  - actualizam-se todos os pesos da rede

Obs. claro que o erro nas unidades hidden é de mais difícil percepção, dado que o erro só é conhecido para as unidades de saída e não o temos para as unidades hidden. Dessa forma o erro para a unidade  $h$  é calculado somando os termos de erro  $\Delta_k$  para cada unidade de saída influenciada por  $h$ , pesando cada um dos  $\Delta_k$ 's por  $w_{kh}$ , o peso da unidade  $h$  para a unidade de saída  $k$ . Este peso caracteriza o grau de responsabilidade no erro na unidade de saída  $k$ , pela unidade hidden  $h$ .



## Algoritmo Backpropagation

### Resumo:

- Pretende a uma classe de algoritmos denominados de Gradient-Descent, pois que começam com uma solução (utilizando pesos gerados aleatoriamente) e, através da aplicação de cada exemplo do set de treino, calculam um erro que é utilizado para alterar lentamente os pesos de toda a rede, por forma a que o erro vá sendo minimizado. (esta redução gradual é o descent);
- O ciclo de aplicação de exemplos é repetido para todos os exemplos, sendo efectuadas de cada vez, pequenos ajustamentos aos pesos;
- Depois de todos os exemplos serem processados, voltamos ao 1º (cada passagem pelo conjunto de treino é chamada de época);
- A taxa de aprendizagem pode variar, mas é tipicamente de 5%: muito maior leva a uma aprendizagem relativa aos casos mais recentes (não memorizar os casos anteriores - e possivelmente oscilar); alguns algoritmos permitem um ajustamento dinâmico, começando com um valor alto que vai baixando à medida que o processo de treino progride (processo denominado de têmpera simulada - simulated annealing).





## Algoritmo Backpropagation-Momentum

**Alteração à regra de actualização dos pesos, levando a que cada actualização do peso na  $n$ -ésima iteração dependa parcialmente da actualização ocorrida na iteração  $(n-1)$  por um factor alfa (entre 0 e 1) constante que é denominado de momentum.**

**O efeito deste momentum é aumentar gradualmente o tamanho do step em regiões de procura onde o gradiente não se altera e, dessa forma, aumentar a convergência. Traduzido o efeito em termos de uma bola a rolar. A bola é mantida a rolar na mesma direcção de uma iteração para a outra (ou seja evita-se a convergência para mínimos locais e evitar a paragem em zonas planas).**



## Algoritmo Backpropagation

### Quando parar o treino?

**Podem ser utilizadas diversas regras, sendo as mais comuns:**

- **para após um número especificado de épocas;**
- **parar quando o erro atingir um determinado limiar;**
- **parar quando o erro não conhecer nenhum melhoramento depois de um certo número de épocas;**
- **parar quando a medida de erro num conjunto de controlo se desvia mais do que um determinado valor relativamente ao medido no set de treino; esta regra é particularmente orientada à prevenção do sobre-treinamento;**
- **possibilitar feedback visual, por forma a que o utilizador possa intervir na paragem do processo.**

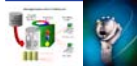
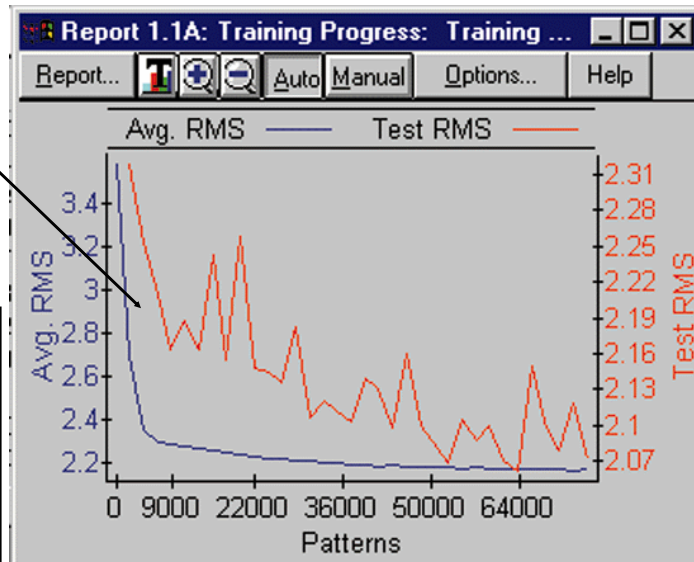


## Algoritmo Backpropagation

### Quando parar o treino?

Vê-se que o erro (quer o erro geral quer de teste) vai diminuindo à medida que prossegue o treino da rede neuronal. O gráfico é actualizado em tempo real; quando o utilizador verificar que o processo está a inverter-se, pode actuar, parando o processo.

Obs. Quando o processo de treino é muito demorado (especialmente se o set de dados for muito grande), muitos produtos permitem parar o processo, testar o modelo e, depois, se necessário, continuar o treino, se melhorias foram consideradas possíveis.



## Redes Neurais: Compreender a Saída

### Já conhecemos a dificuldade das redes neurais, no que concerne a aspectos descritivos:

- No K-NN são disponibilizados registos protótipo que poderão revelar padrões relativos ao porquê de ser feita a predição.
- As árvores de decisão têm características eminentemente descritivas, podendo proporcionar regras que explicam porque a predição foi feita para um dado registo.

### Nas redes neurais:

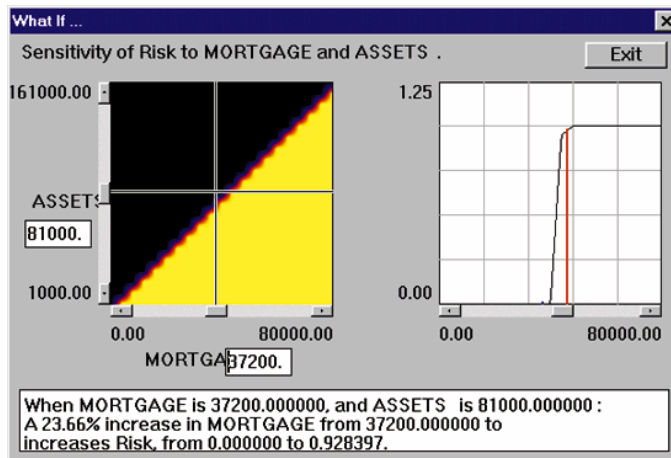
1. podemos criar a rede sem um dado nó de entrada (para o predictor que queremos avaliar) e comparar a precisão com a rede com o predictor.

Claro que esta forma pode ser errónea, pois que mesmo que o predictor retirado seja grandemente influenciador do resultado, a rede pode conseguir na mesma um bom desempenho, se houver outro predictor que esteja fortemente correlacionado com o predictor retirado.



## Redes Neurais: Compreender a Saída

2. **Análise de sensibilidade** - Alterar os valores para uma ou duas variáveis independentes e observar o valor de saída. Da sua alteração poderemos aquilatar da importância do predictor, mas também quão sensível é a pequenas perturbações. Se pequenas alterações tiverem um impacto significativo na saída, o predictor deverá ser considerado significativo e inversamente.



O produto Neural Connection da SPSS possui uma característica "What If" que mostra um gráfico de sensibilidade. Movendo as barras de scroll ao longo do gráfico, o utilizador pode explorar o impacto da alteração em valores de duas variáveis independentes.

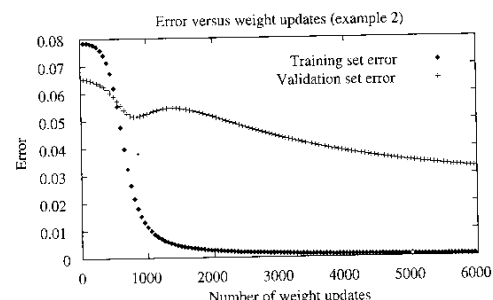
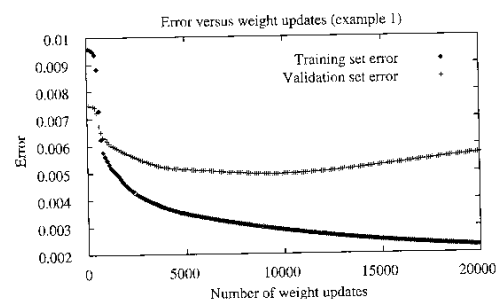


## Redes Neurais: Combater a Sobreadaptação

Uma rede neuronal pode ser treinada eternamente e, mesmo assim, não atingir 100% de precisão no set de treino (contrariamente às árvores de decisão ou K-NN)

Assim o problema da sobreadaptação é aqui menos crítico, mas ainda existe, como se pode verificar nos gráficos ao lado.

Como podemos ver, o erro no set de treino diminui monotonicamente. Para o set de validação, o erro diminui a princípio e depois, pode crescer devido a sobreadaptação aos exemplos de treino. Note-se, no segundo caso, que há que ter cuidado, pois que, apesar de se verificar um aumento de erro cerca das 1000-1500 épocas, depois voltou a baixar, ou seja não parar logo que o erro cresce.



## Redes Neurais: Combater a Sobreadaptação

### Actuar de duas formas:

- Parar o treino (já abordado atrás);
- Controlar a arquitectura da rede (complexidade) adequando-a à complexidade do problema:
  - Podem ser utilizadas um conjunto de regras mais ou menos empíricas (dependentes muitas vezes do produto utilizado).
  - Alguns exemplos:
    - o nº de links deverá ser 5x, 2x ou igual ao número de exemplos
    - o nº de nós de entrada deverá ser 10 a 40x menos do que o número de exemplos;
    - o nº de nós hidden deverá ser de 2/3 do nº de nós de entrada e saída
    - quanto ao número de camadas hidden é, muitas vezes, recomendado ser apenas 1 e, no máximo, 2.



## Redes Neurais: Preparação dos Dados

As entradas de uma rede neuronal esperam normalmente valores entre 0 e 1 ou -1 e 1.

Para uma variável independente ser mapeada para esse âmbito de valores, teremos de empregar escalas.

Para variáveis contínuas:

escala - dividir o menor e maior valor da variável independente por um número, por forma a que a diferença entre os dois seja 1. O número mais pequeno será 0 e o maior, o 1.

escala não uniforme - por exemplo para rendimentos - uma dada diferença entre valores pequenos é mais significativa que a mesma diferença para valores grandes. Para ajudar a rede neuronal a fazer esta distinção, poderemos empregar uma escala não uniforme, p. ex. utilizando logaritmos dos valores da variável.

binning - dividir os valores do predictor em intervalos e assignar um nó de entrada a cada intervalo que indica quando o valor cai ou não em cada um dos intervalos (0.0 ou 1.0)



## Redes Neurais: Preparação dos Dados

### **Para variáveis contínuas (continuação):**

**codificação termómetro** - quando dados numéricos são transformados em intervalos fixos, sendo colocados a 1.0 todos os intervalos abaixo do valor; análogo à subida do mercúrio no termómetro.

**diferença temporal embebida** - proporcionar as diferenças de valores ao longo do tempo, além dos valores reais. Isto representa a taxa de alteração do valor. Também poderemos incluir a taxa da mudança (ou seja a aceleração ou estabilidade).



## Redes Neurais: Preparação dos Dados

### **Para variáveis Categóricas:**

**Não há ordem a estabelecer, mas ela é implícita ao mapearmos obrigatoriamente para valores.** Assim:

**codificação numérica** - a cada valor da variável à assignado aleatoriamente um número (1-louro, 2-castanho, ...) e esses valores são convertidos para intervalo 0-1. Claro que o problema é a ordenação implícita dos valores categóricos (inexistente na realidade) que pode polarizar a rede, influenciando a sua aprendizagem.

**codificação 1-de-N** - Neste caso, cada valor categórico diferente tem o seu nó de entrada (havendo p. ex. tantos nós de entrada, quantas as cores do cabelo). Cada nó recebe o valor 0 se o valor para o predictor não estiver presente e 1, se o estiver.

**codificação Binária** - A cada predictor é assignado um número aleatório e esse número é convertido em representação binária (ex. louro 1=001, castanho 2=010...) Um nó de entrada é assignado a cada posição da string binária.



## Redes Neuronais: Kohonen

Tuevo Kohonen demonstrou em 1981 uma versão completamente nova das redes neuronais: conhecidas como redes Kohonen ou Mapas Auto-Organizáveis de Kohonen.

Podem ser vistas como contrapartida dos mapas que existem em várias áreas do cérebro, como mapas visuais, mapas de detecção de bordas, etc.

Actualmente são utilizadas principalmente para aprendizagem não supervisionada e clustering.

As redes de Kohonen são redes neuronais feed-forward geralmente sem camadas hidden. A rede contém apenas a camada de entrada e a de saída, mas os nós na camada de saída competem entre si para mostrar a maior activação a um dado registo, uma estratégia denominada de “vencedor leva tudo”.



## Redes Kohonen

### Algoritmo:

1. Os nós de saída formam um grelha bidimensional com nenhuma camada hidden.
2. Ligar completamente os nós de entrada aos de saída;
3. Ligar os nós de saída por forma a que os nós próximos se fortaleçam e os nós distantes se enfraqueçam;
4. Começar com pesos aleatórios nos nós;
5. Treinar para determinar que nó de saída responde mais fortemente ao caso corrente apresentado à rede;
6. Alterar o peso por forma a que esse nó como maior resposta passe de futuro a responder mais fortemente;
7. Normalizar os pesos dos links, de forma a que a sua soma seja um determinado valor constante, ou seja, o aumentar o peso de um link, corresponderá a baixar o de um outro;
8. Continuar o treino até que alguma forma global de organização seja gerada na grelha de saída bi-dimensional (onde haverá claros nós vencedores para cada entrada e, geralmente, nós na vizinhança também activados).

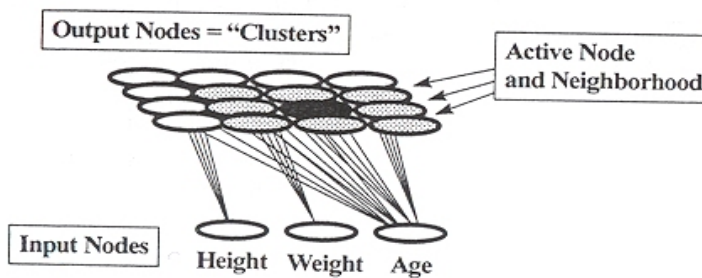




## Redes Neurais: Kohonen

O treino destas redes permite que a organização construída seja útil para segmentação e clustering de uma base de dados.

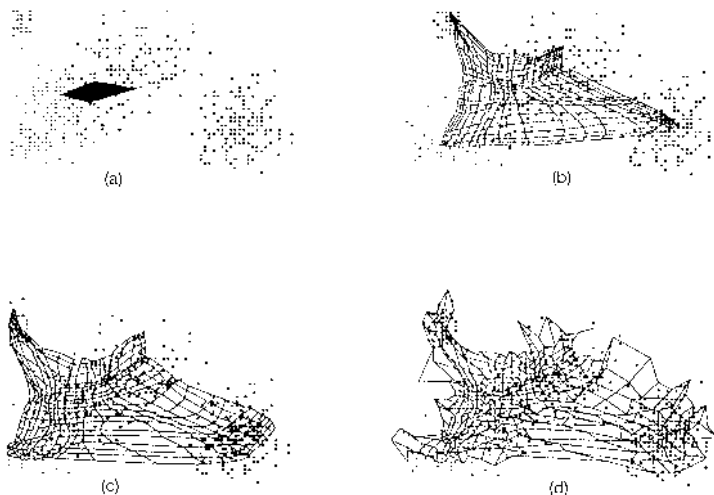
- cada nó de saída representa um cluster e os clusters mais próximos serão saídas próximas na camada de saída;
- cada registo da base de dados cairá em um e um só desses clusters (o mais activo), mas os outros clusters nos quais pode ser também inserido serão mostrados e apresentar-se-ão como próximos candidatos mais adequados.



Arranjo dos nós de saída na rede Kohonen numa grelha bi-dimensional que simulam o reforço positivo de neurónios do cérebro.



## Redes Neurais: Kohonen



Visualização do processo de treino da rede Kohonen. A rede comporta-se como uma superfície elástica que é estendida pelo espaço de amostragem.

A 1ª. figura mostra o estado inicial dos vectores;

As figuras seguintes mostram estados subsequentes que permitem visualizar o mapa que se organiza gradualmente em direcção a uma melhor cobertura do espaço de amostragem.





## Redes Neurais: Avaliação

**Podem ser aplicadas em grande número de problemas de data mining:**

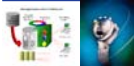
- classificação
- regressão
- clustering

**Mas:**

- são mais complicadas do que as outras técnicas:
- há muitos algoritmos de redes neurais
- incluem muitos parâmetros com que os utilizadores têm de lidar
- natureza não descritiva dos modelos.

**Contudo:**

- não há limitações em relação ao tipo de relacionamentos que podem modelar (ex. relações trigonométricas ou logarítmicas, que podem revelar-se um problemas para as outras técnicas)



## Redes Neurais: Avaliação

**De qualquer modo:**

- em muitas aplicações reais de negócio, as aproximações conseguidas em árvores de decisão ou Naïve-Bayes são mais do que suficientemente boas;
- mas onde a precisão é importante e em relacionamentos complexos, uma rede neuronal pode ser o caminho a seguir.

**Mas a única forma de avaliar se uma rede neuronal será a abordagem indicada é através da experiência: comparar a sua precisão com outras técnicas** (árvores de decisão ou Naïve-Bayes)

- se a precisão da rede neuronal não for significativamente maior, isso aumentará a confiança no modelo de árvore de decisão ou Naïve-Bayes.
- se a precisão do modelo da rede neuronal for significativamente melhor, isso quer dizer que há mais trabalho a fazer:
  - examinar os casos onde o modelo fica em desacordo, podendo encontrar-se, porventura, formas de melhorar a a árvore de decisão, p. ex. através do pré-cálculo de alguns ramos;
  - ou pode ser que a melhor solução seja mesmo o modelo da rede neuronal.



# Algoritmos Genéticos

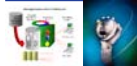
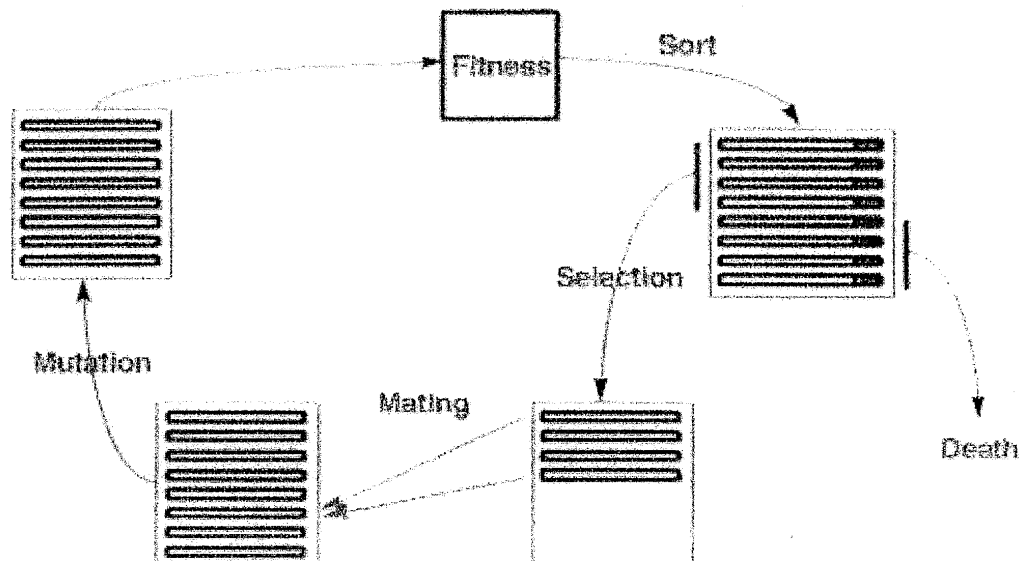


## Algoritmos Genéticos: Generalidades

- Baseados na evolução biológica e no DNA;
- As soluções para o problema são codificadas na forma de palavras;
- A população de soluções (genomas) é avaliada em termos da função de adaptação (fitness);
- Os genomas com maior adaptação são seleccionados para sobreviver; os restantes são mortos;
- Os sobreviventes seleccionam parceiros, reproduzem-se, sofrem mutações e geram descendentes (novas soluções);
- O processo é repetitivo.



## Algoritmos Genéticos: Generalidades



## Evolução das Espécies e Data Mining

Em 1859, Charles Darwin publicou o livro “Origem das Espécies” detalhando como organismos complexos podem ser criados e melhorados através dum processo de tentativas aleatórias, reprodução sexual e selecção. Na chamada teoria da evolução, a noção central é a “selecção natural”: cada espécie tem uma sobreprodução de indivíduos em luta acirrada pela vida e os indivíduos mais bem adaptados ao ambiente, sobrevivem.

**Mas como surge a evolução das espécies ligada ao Data Mining?**

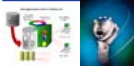
**A questão fundamental é sempre a mesma: encontrar modos de resolução de problemas!**



## **Evolução das Espécies e Data Mining**

**Na Natureza** a questão é abordada sob a forma de duas vertentes fundamentais:

- sobrevivência do mais adaptado como técnica de resolução de problemas;
- DNA para codificar os problemas (utilizando 4 blocos C,G,A,T, na linguagem da genética).  
(para o caso do genoma humano, teremos uma cadeia de cerca de 3 bilhões de letras que encheria milhares de bíblias)
  - cada indivíduo herda algumas características do pai e outras da mãe;
  - diferenças individuais entre pessoas (cor de cabelo, olhos e também a predisposição para doenças, são causadas por diferenças na codificação genética).



## **Evolução das Espécies e Data Mining**

**Em Data Mining** a resolução de problemas com a aplicação de mecanismos evolutivos (computação evolucionária) - utiliza algoritmos genéticos para criar uma versão da evolução biológica em computadores:

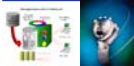
- Os algoritmos operam sobre programas em computadores, sujeitos a mutações, reprodução sexual e sobrevivência dos mais aptos.
- Com o tempo, estes programas melhoram o seu desempenho na resolução de um determinado problema - atingindo eventualmente um elevado grau de competência:
  - A sobrevivência do mais apto é aqui traduzida na função de adaptação (fitness function)
  - a codificação das soluções é efectuada numa cadeia de um alfabeto limitado (do C,G, C, T -suficiente para a concepção de criatura vivas, para binário, p. ex. - suficiente para o registos de outros sistemas complexos).



## Porquê um tal Mecanismo de Optimização?

### Tem Desvantagens:

- **larga sobreprodução de indivíduos**  
mas, uma ou duas espécies a mais ou a menos não tem consequências na natureza
- **envolve enorme quantidade de energia e tempo**  
e também sofrimento
- **todo o processo parece ter pouco propósito, pois que o melhoramento das espécies depende do factor sorte**  
como a evolução tem a ver com a mutação e boa combinação destes, a probabilidade de que uma determinada mutação / combinação leve a algo significativo será tão grande quanto a de um macaco escrever um belo poema, simplesmente batendo nas teclas com um processador de texto.



## Porquê um tal Mecanismo de Optimização?

### Tem Vantagens:

- **solidez e paralelismo inerente:**
  - a natureza não aposta tudo num só cavalo, mas leva a cabo grande número de experiências simultâneas;
  - implica decerto um esforço extra, mas
  - o processo não fica preso num local sub-ótimo,
  - ou fica retido em soluções genéricas.

Em resumo, se houver algo a ser encontrado, sê-lo-á, em regra.



## Popularidade dos Algoritmos Genéticos

1. É conhecido que a evolução é um método robusto e com sucesso para adaptação em sistemas biológicos;
2. Os algoritmos genéticos podem pesquisar espaços de hipóteses contendo partes inter-actuantes complexas, onde o impacto de cada parte nas hipóteses genéricas de elegância possam ser difíceis de modelar;
3. Os algoritmos genéticos são facilmente paralelizáveis e podem tirar vantagens dos custos sucessivamente menores de hardware cada vez mais poderoso.

Em geral, um organismo é alocado a cada processador ou processador virtual, numa máquina paralela. A adaptação é avaliada localmente e, todos os organismos que não atingem a adaptação requerida podem ser apagados e substituídos por material genéticos de indivíduos mais aptos. A mutação e cruzamento podem ser acomodados por computação local eficiente, uma vez os cromossomos para cruzamento tenham sido movidos para um processador, a cargo da comunicação inter-processos.



## Utilização de A.G. em Negócio

Há três áreas principais:

**Optimização** - dado um problema de negócio com determinadas variáveis e uma definição bem formada de lucro, um algoritmo genético pode ser utilizado para determinar automaticamente os valores óptimos para as variáveis que otimizarão o lucro,

**Predição** - Os algoritmos genéticos têm sido utilizados como operadores metanível, utilizados para ajudar na optimização de outros algoritmos de data mining.

têm sido utilizados para otimizar os pesos numa rede neuronal ou encontrar regras de associação óptimas, em análise de cesto de compras.

**Simulação** - Algumas vezes, um problema específico de negócio não está bem definido, em termos do que é o lucro ou de avaliar se uma dada solução é melhor do que outra.

O analista de negócio tem um grande número de entidades (clientes ou competidores) que gostaria de simular através de regras simples de interações no tempo.





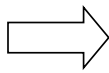
## Utilização de A.G.

No seu âmbito, os algoritmos genéticos são técnicas de optimização:

- são sistemas que abordam problemas muito complexos e de difícil solução.

Chegam a uma boa solução, não carecendo de grandes detalhes de compreensão sobre o problema, excepto como avaliar a boa solução;

- podem ser vistos como um tipo de estratégia de meta-aprendizagem;
- podem ser aplicados a um conjunto de problemas incrivelmente diversificado - trata-se de uma abordagem de largo espectro.



Não podem competir com algoritmos concebidos especificamente para a resolução de problemas particulares.



## Utilização de A.G. (cont.)

Mas se forem verificadas as condições:

- o valor das várias soluções propostas puder ser bem definido,
- o problema ser complexo e não poder ser directamente solucionado,
- o problema ser relativamente novo, não bem compreendido e não terem sido encontradas outras técnicas de optimização a ser utilizadas na sua solução,
- o problema envolver um grande número de variáveis que trabalham em conjunto na produção de um efeito em larga escala:



O Algoritmo Genético será a abordagem a eleger



## Score dos Algoritmos Genéticos

**Embora muitas empresas forneçam produtos que utilizam algoritmos genéticos, estes produtos estão ainda na sua infância.**

**Automatização** - São relativamente automáticos, uma vez definida a função de adaptação e a forma como o problema será codificado no material genético simulado no computador. Estes dois processos podem, no entanto, revelar-se bastante difíceis. Apesar disso, mesmo esquemas de codificação não óptimos podem resultar em soluções úteis.

**Clareza** - Quando os algoritmos genéticos são utilizados na optimização de outras técnicas de data mining, não poderão trazer qualquer contribuição adicional à clareza característica da técnica. Proporcionam um bom sentido de clareza no procedimento de optimização e a codificação simbólica das soluções encontradas são muitas vezes facilmente legíveis.

**ROI** - dado que a solução do problema pode ser genérica, muitos dos factores, relacionados com o ROI geral do processo de negócio, podem ser levados em conta, simultaneamente.



## Estrutura Geral dos A.G.

- O algoritmo opera actualizando iterativamente uma associação de hipóteses chamada população;
- Em cada iteração, todos os membros da população são avaliados de acordo com a função de aptidão (fitness);
- É gerada uma nova população pela selecção probabilística dos indivíduos mais aptos da população corrente;
  - alguns desses indivíduos são movidos intactos para a próxima população
  - outros são utilizados para criar uma nova prole de indivíduos, aplicando-lhe operadores genéticos: cruzamentos e mutações;
- Parar quando uma família de soluções com sucesso tiverem sido produzidas.



## Protótipo de Algoritmo Genético (1)

### GA (fitness, limiar\_fitness, p, r, m)

fitness - função que atribui um nível de avaliação, dada uma hipótese

limiar\_fitness - um limiar que especifica o critério de finalização

p - número de hipóteses a ser incluídas na população

r - fracção da população a ser substituída em cruzamentos, em cada passo

m - taxa de mutação

- Inicializar população: p - gerar p hipóteses aleatoriamente

- Calcular para cada h em p, fitness (h)

- Enquanto [Max fitness(h)] < limiar\_fitness,

Fazer criar uma nova geração  $P_s$

#### 1. selecção

probabilisticamente seleccionar (1-r)p membros de p, para adicionar a  $P_s$ . A

probabilidade  $P_r(h_i)$  das hipóteses a seleccionar  $h_i$  de p é dada por:

$$P_r(h_i) = \text{Fitness}(h_i) / \sum_{j=1}^p \text{Fitness}(h_j)$$

a probabilidade de uma hipótese ser seleccionada é proporcional à sua própria aptidão

$\text{Fitness}(h_i)$  e inversamente proporcional à aptidão das outras hipóteses competidoras, na população corrente Somatório de  $j=1$  a p de  $\text{Fitness}(h_j)$



## Protótipo de Algoritmo Genético (2)

### 2. cruzamento

probabilisticamente seleccionar (r.p)/2 pares de hipóteses de p, de acordo com

$P_r(h_i)$ , dada em 1. Para cada par ( $h_1$   $h_2$ ) produzir dois sucessores, aplicando

o operador cruzamento.

**Cruzamento:** toma como pais duas hipóteses da geração corrente de acordo com a mesma probabilidade e cria uma descendência de duas hipóteses recombinação porções de ambos os pais.

### 3. mutação

escolher m por cento dos membros de  $P_s$  com probabilidade uniforme. Para cada um, inverter um bit seleccionado aleatoriamente, na sua representação.

**Neste ponto temos uma população constituída por membros idênticos à geração pai e outros gerados por cruzamentos. Aqui, uma fracção m desses membros é seleccionada aleatoriamente e são executadas sobre esses mutações aleatórias para alterar esses membros.**

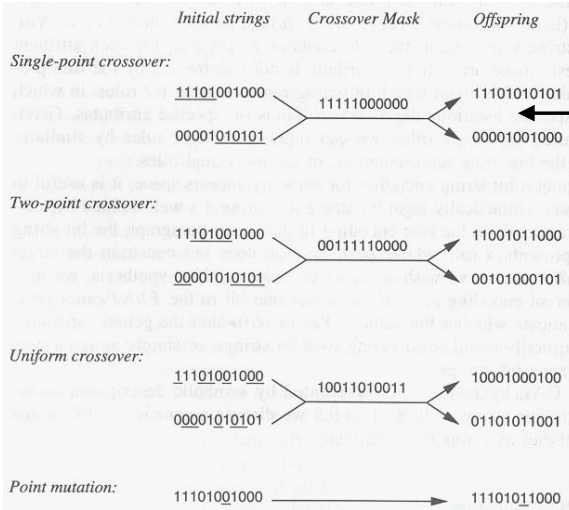
### 4. Actualizar $P \leftarrow P_s$

### 5. Calcular para cada h em P, calcular Fitness (h)

- Retornar a hipótese P que tem a maior aptidão.



# Operadores Genéticos



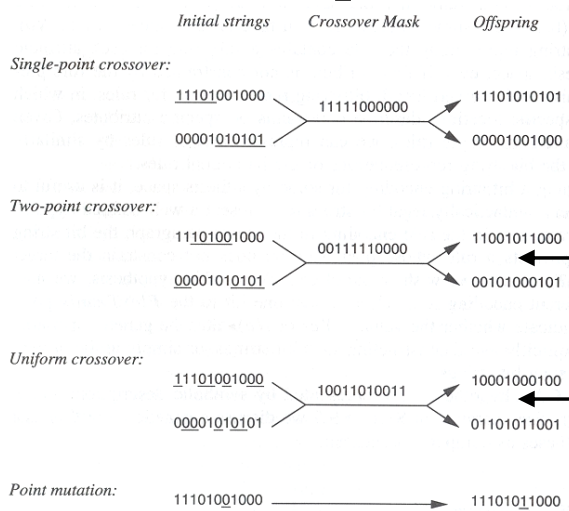
**Operador ponto simples:** começa com n uns contíguos, seguidos de zeros suficientes para completar a string

**Operadores comuns em algoritmos genéticos**  
Estes operadores formam hipóteses descendentes, representadas por strings de bits. O operador cruzamento cria dois descendentes a partir de dois pais, utilizando a máscara de cruzamento para determinar que pais contribuem com que bits.  
Mutação cria um único descendente de um único pai, através da alteração do valor de um bit, seleccionado aleatoriamente.

**Obs.** Podem ser utilizados outros operadores adicionais, especialmente operadores especializados para a representação particular das hipóteses usadas no sistema.



# Operadores Genéticos



**Operador dois pontos -** começa com n<sub>1</sub> zeros, seguidos de n<sub>2</sub> uns contíguos, seguidos de zeros necessários para completar a string.  
Cria descendentes substituindo segmentos intermédios de um pai no meio do segundo.

**Operador uniforme:** Combina bits amostrados uniformemente dos dois pais. Ou seja, a máscara é gerada como uma cadeia aleatória, sendo cada bit escolhido aleatoriamente e independentemente dos outros.

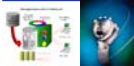
**Operador mutação:** Escolhe um único bit numa posição aleatória e altera o seu valor.



## Função Adaptação (fitness)

**A função de adaptação define o critério de rank de hipóteses potenciais, para as seleccionar probabilisticamente, com vista à sua inclusão na população da próxima geração.**

- Se a tarefa é aprender regras de classificação, então a função de adaptação terá tipicamente um componente que classificará a precisão da regra num conjunto, através de exemplos de treino.
- Outros critérios podem ser incluídos também, tal como a complexidade ou generalidade da regra.



## Seleccção

### **Tipos de Seleccção:**

- Seleccção de adaptação proporcional ou seleccção roleta (algoritmo atrás apresentado) - a probabilidade de uma hipótese ser seleccionada é dada pela razão da sua adaptação relativamente à adaptação dos outros membros da população corrente.
- Seleccção competição - duas hipóteses são escolhidas primeiro de forma aleatória na população corrente. Com uma probabilidade pré-definida  $p$ , a mais adaptada dessas duas é então seleccionada e com probabilidade  $(1-p)$  a hipótese menos adaptada é seleccionada. Este tipo de seleccção leva à existência de uma população mais diversa do que a anterior.
- Seleccção Rank - as hipóteses na população corrente é ordenada inicialmente. A probabilidade de que uma dada hipótese seja seleccionada é proporcional à sua posição no rank dessa lista ordenada, em vez da sua adaptação.



## Mutação

**A mutação opera alterações no material genético, nos organismos simulados, de forma aleatória.**

**A sobrevivência do mais apto torna certo que as coisas aprendidas acerca do espaço que o algoritmo genético está a pesquisar são passadas, tirando vantagem disso, nas gerações sucessivas.**

**Mas:**

- **este processo tem apenas alguma exploração activa, não levando a uma solução muito boa: é análogo a dar pequenos passos à volta da solução corrente, para ver se alguma pequena perturbação é melhor.**



## Mutação

**Temos várias formas de executar a mutação, além da mutação de bit aleatório, já vista atrás.**

- **mutação aleatória de gene** - neste caso os valores permitidos dos genes são conhecidos e o gene inteiro é mudado para outro valor válido.
- **mutação creep** - os genes são modificados com alguma pequena perturbação, em vez de alterações verdadeiramente aleatórias. Ex. um gene com valor 6, pode ser modificado para 5 ou 7; esta mutação permite alterações que seriam difíceis como operador mutação aleatório de gene.
- **mutação heurística** - em alguns casos, são conhecidas regras ou heurística que podem ser aplicadas a cromossomas que podem melhorar dramaticamente a adaptação do organismo.





## Algoritmos Genéticos: Exemplo

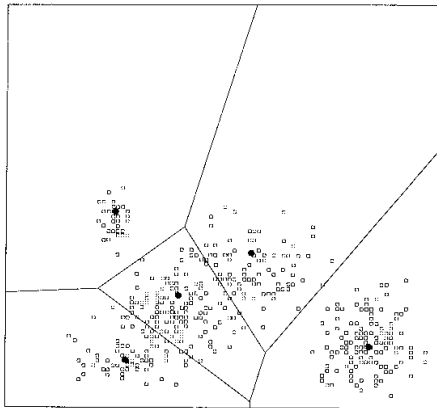
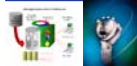


Diagrama Voronoi para a base de dados de revistas: a divisão em áreas diferentes que resultam da definição de fronteiras - linhas equidistantes de cada dois pontos-guia, que descrevem áreas de cluster para o espaço de pesquisa.

**Problema:** descobrir clusters para a base de dados de revistas utilizando algoritmos genéticos.

**1º passo:** encontrar uma codificação elegante para o problema em termos de uma cadeia de símbolos, num alfabeto limitado.

- uma forma de lidar com este problema será descrever os clusters como uma série de pontos-guia, no espaço em investigação.



## Algoritmos Genéticos: Exemplo

A codificação do problema consistirá numa string com as coordenadas dos nossos 5 pontos-guia:

- cada indivíduo consistirá em 10 números reais, descrevendo as coordenadas dos 5 pontos-guia.
- no estágio inicial, esses pontos serão seleccionados aleatoriamente; iniciaremos com vinte indivíduos que descrevem 20 soluções aleatórias para o problema de clustering.

**2º passo:** escolha de uma boa função de adaptação.

- neste caso é muito fácil: vamos utilizar a distância média dos pontos no espaço de busca ao ponto guia mais próximo. O conjunto de pontos-guia que constitua o melhor clustering será aquele em que a distancia média de todos os pontos ao ponto-guia mais próximo for mínima.

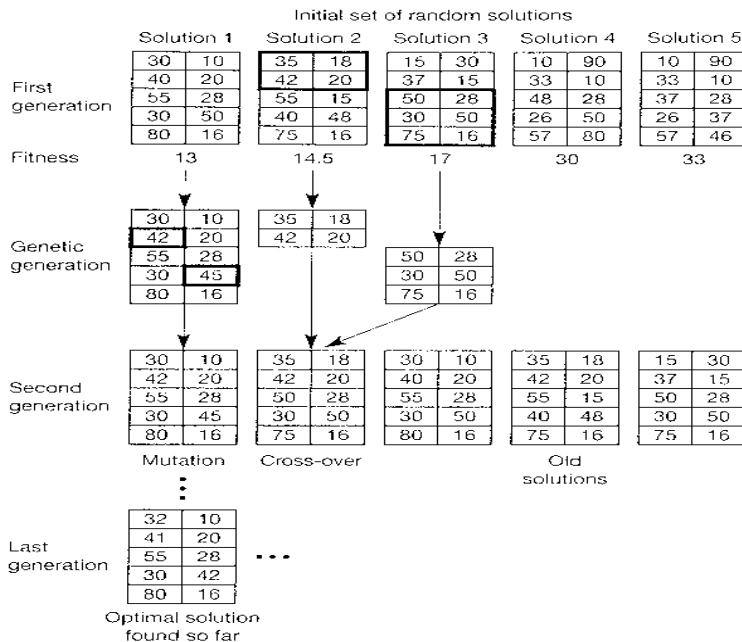
**3º passo:** seleccionar o tipo de operador genético de cruzamento e mutação.

**Cruzamento:** pegamos em três pontos de um indivíduo e em dois de outro, trocamos e, assim, criamos dois descendentes.

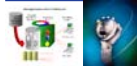
**Mutação:** mudaremos alguns pontos de um indivíduo para uma sua vizinhança, aleatoriamente.



## Algoritmos Genéticos: Exemplo



**Evolução do algoritmo genético: população que representa um clustering estável, obtida após um certo número de gerações, relativa ao problema do cluster de revistas.**



## Algoritmos Genéticos: Desafios

### Convergência Prematura:

A alocação exponencial de tentativas envolvendo o organismo mais adaptado (denominado Teorema de Schema) é, simultaneamente, uma bênção e uma maldição!

- Embora permita ao sistema chegar rapidamente a uma solução bem sucedida, leva, muitas vezes, a uma população a convergir, carecendo de diversidade e, desta forma, à não exploração efectiva de novas áreas ou ideias.

### Muitos Parâmetros:

Na maioria dos sistemas há que atribuir valores a um grande número de parâmetros, antes de iniciar a evolução. Esses parâmetros incluem coisas como: tamanho de população, nº de gerações, algoritmos de selecção e de cruzamento.

Muitas vezes, a escolha desses parâmetros é feita por tentativas e, se não forem correctos, podem ter um impacto negativo no desempenho do sistema.



## Convergência Prematura: Solução

### Técnica chamada Partilha

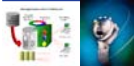
Foi mostrado que o desempenho óptimo do algoritmo genético é atingido, utilizando a maior população possível:

- necessitaríamos de um computador perfeitamente paralelo e capacidade de processamento ilimitada
- mas, ainda assim, subsiste o problema da convergência prematura - podemos ficar com uma população de soluções sub-óptimas - milhões de cópias do mesmo material genético. Equivalente a termos uma população reduzida, mas a grandes custos.

Esta selecção natural, gera uma força que tende a puxar a população em direcção ao domínio de muitas cópias de um organismo único e relativamente bem adaptado.

- Uma vez esta “convergência total” atingida, o cruzamento não poderá gerar variações nesta população;
- Resta a mutação, mas é uma fonte de variação cega e lenta.

**Em resumo:** a procura da melhor solução pára, efectivamente.



## Convergência Prematura: Solução

**Vejamos o que sucede na natureza:**

- se um único tipo de organismo constituir uma grande sub-população:
  - haverá um grande consumo dos recursos presentes no ambiente;
  - estes recursos tornam-se escassos;
  - passa-se à competição pelos recursos escassos entre os indivíduos:
    - todos passam a sofrer carências;
    - o crescimento da população abrandar;
    - estabiliza à volta da “capacidade sustentável” - o número de organismos de um tipo particular fica em equilíbrio com a disponibilidade de recursos.
- Este efeito de auto-controlo é simulado através do chamado **efeito de partilha:**
  - supõe a existência de recursos abstractos pelos quais todos os organismos que exploram um determinado nicho devem competir ou partilhar.



## Efeito de Partilha

**Se houver muitos organismos num nicho particular, a diminuição de recursos leva à diminuição da adaptação de todos os indivíduos nesse nicho.**

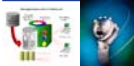
- assume-se que dois organismos que estejam adjacentes no espaço de pesquisa, consumirão mais dos mesmos recursos, do que dois organismos que estejam longe.

**Em termos do algoritmo:**

- o efeito consegue-se multiplicando a adaptação base do organismo por um factor que diminui a adaptação, dependendo de quantos organismos tenham uma composição genética semelhante.

**Na prática:**

- a técnica de partilha proporciona melhores soluções, mais depressa, evitando a convergência prematura da população, enquanto mantém a pesquisa útil e alocação, por tentativas, do mais adaptado.
- necessita de mais computação na fase da avaliação da adaptação, mas é mais do que justificada.



## Muitos Parâmetros: Automatização

**Evolução Metanível**

- Codificar muitos desses parâmetros, no próprio material genético.
- O seu nome advém do facto de que os processos que controlam a evolução, estão também a evoluir.

**Uma desvantagem é imediatamente visível: o tamanho do espaço de pesquisa, regista um aumento dramático.**

**Vantagens:** apesar do atrás dito, a codificação da taxa de mutação nos cromossomas, em sistemas evolucionários, proporcionam desempenho equivalente aos sistemas evolucionários standard, apesar do aumento de espaço de pesquisa.

**Esses sistemas provaram ser mais robustos, relativamente à introdução de alterações na dificuldade dos problemas.**

Realmente, em biologia, encontramos paralelo. Descobriu-se que um terço das mutações num dado cromossoma ocorrem apenas num local. Muito longe da ideia de que a radiação ambiente e outros factores produziram mutações consistentes em todos os genes.

