Redes Neurais Artificiais

Conteúdo

1. Base Biológica

- Como o cérebro processa informações? Como ele é organizado? Quais são os mecanismos envolvidos no funcionamento cerebral?
- Estas são apenas algumas das perguntas mais desafiadoras para a ciência.
- O cérebro é especialista em desempenhar funções como reconhecimento de padrões, controle motor, percepção, inferência, intuição, adivinhações, etc. Entretando, o cérebro também é “preconceituoso”, lento, impreciso, realiza generalizações incorretas e, acima de tudo, é geralmente incapaz de explicar suas próprias ações (embora este seja um requisito cultural e não funcional).
- Os neurônios são considerados como as unidades básicas de processamento do cérebro.
- De modo análogo, modelos simplificados dos neurônios biológicos constituem as unidades básicas de processamento das redes neurais artificiais (RNAs).

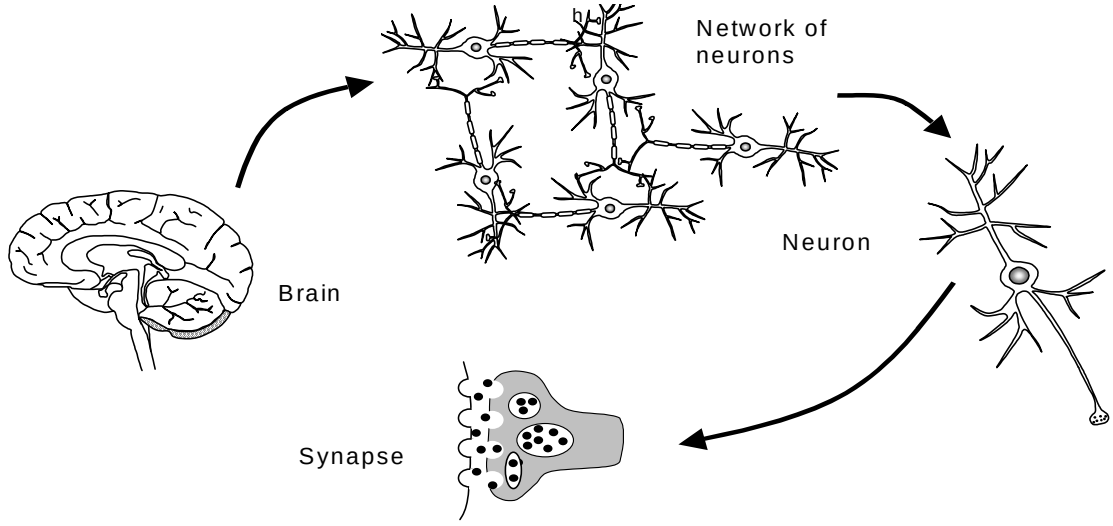
- Os neurônios biológicos estão conectados uns aos outros através de conexões sinápticas. Acredita-se que a capacidade das sinapses serem moduladas é a principal base para todos os processos cognitivos, como percepção, raciocínio e memória.
- Sendo assim, algumas informações essenciais sobre neurônios, sinapses e organização estrutural são importantes para o projeto de RNAs.

1.1. O Sistema Nervoso

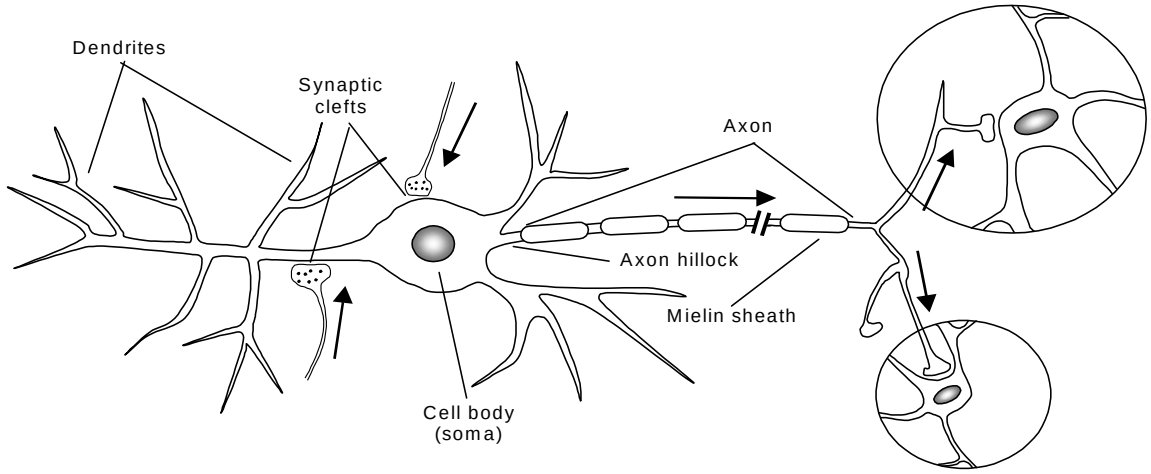
- Todos os organismos multicelulares possuem algum tipo de sistema nervoso, cuja complexidade e organização varia de acordo com o tipo de animal.
- Mesmo os vermes, lesmas e insetos são capazes de adaptar seu comportamento e armazenar informações em seus sistemas nervosos.
- O sistema nervoso é responsável por dotar o organismo, através de entradas sensoriais, de informações sobre o estado do ambiente no qual ele vive e se move.

A informação de entrada é processada, comparada com as experiências passadas, e transformada em ações apropriadas ou absorvidas sob a forma de conhecimento.

- O sistema nervoso pode ser organizado em diferentes níveis: *moléculas*, *sinapses*, *neurônios*, *camadas*, *mapas* e *sistemas*.



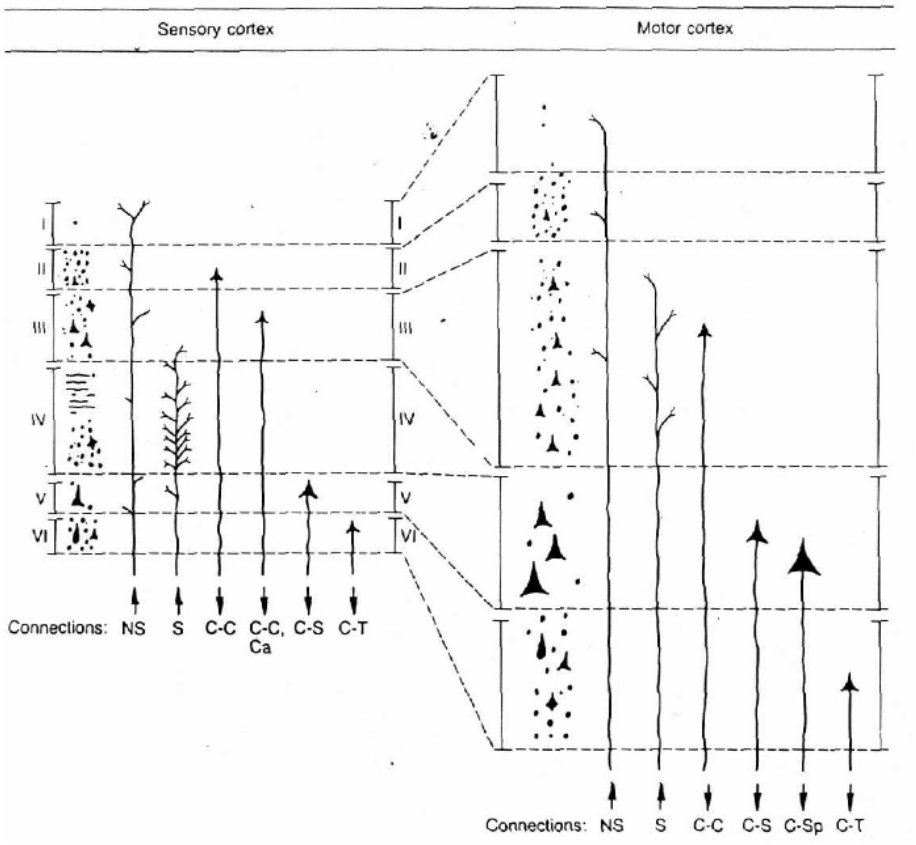
- O processo de transmissão de sinais entre neurônios é central para a capacidade de processamento de informação do cérebro.
- Uma das descobertas mais empolgantes em neurociência foi a de que a *efetividade* da transmissão de sinais pode ser modulada, permitindo o cérebro se adaptar a diferentes situações.
- A *plasticidade sináptica*, ou seja, a capacidade das sinapses sofrerem modificações, é o ingrediente chave para o aprendizado da maioria das RNAs.
- Os neurônios podem receber e enviar sinais a vários outros neurônios.
- Os neurônios que enviam sinais, chamados de neurônios *pré-sinápticos* ou "*enviadores*", fazem contato com os neurônios *receptoress* ou *pós-sinápticos* em regiões especializadas denominadas de *sinapses*.
- A sinapse é portanto, a junção entre o axônio de um neurônio pré-sináptico e o dendrito ou corpo celular de um neurônio pós-sináptico (ver figura).



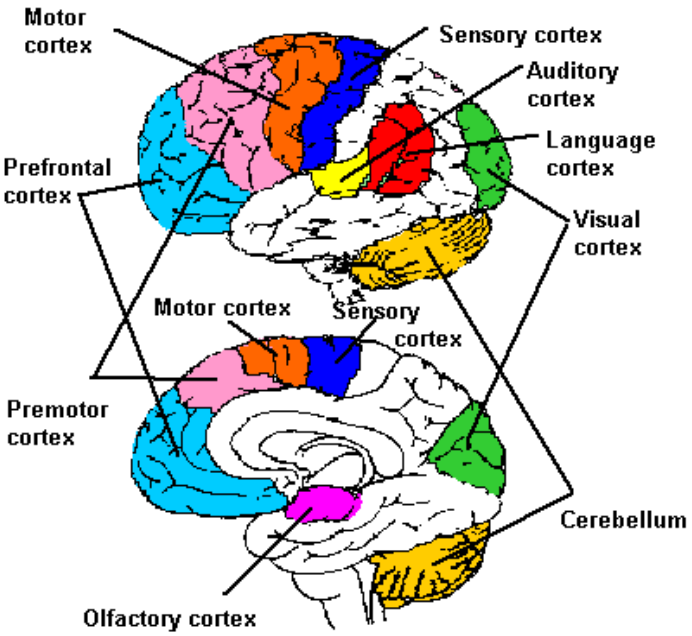
- A capacidade de processamento de informação das sinapses permite que elas alterem o estado de um neurônio pós-sináptico, eventualmente gerando um pulso elétrico, denominado *potencial de ação*, no neurônio pós-sináptico.
- A ativação de um neurônio ocorre apenas quando seu potencial de membrana é maior do que um dado *limiar* (*threshold*).

- Portanto, um neurônio pode ser visto como um dispositivo capaz de receber estímulos (de entrada) de diversos outros neurônios e propagar sua única saída, função dos estímulos recebidos e do estado interno, a vários outros neurônios.
- Os neurônios podem ter conexões *de sentido positivo* (*feedforward*) e/ou *de sentido negativo* (*feedback*) com outros neurônios, ou seja, as conexões podem ter um único sentido ou serem recíprocas.
- Diversos neurônios interconectados geram uma estrutura em rede conhecida como *rede neural*.
- Um agrupamento de neurônios interconectados pode exibir comportamentos complexos e uma capacidade de processamento de informação que não pode ser predita tomando-se cada neurônio individualmente.
- Uma característica marcante das redes neurais é a *representação distribuída* de informação e seu *processamento paralelo*.

- Muitas áreas do cérebro apresentam uma *organização laminar* de neurônios. Lâminas são *camadas de neurônios* em contato com outras camadas.
- Um dos arranjos mais comuns de neurônios é uma estrutura bi-dimensional em camadas organizada através de um arranjo *topográfico* das respostas de saída. O exemplo mais conhecido deste tipo de estrutura é o *córtex* humano.
- O córtex corresponde à superfície externa do cérebro; uma estrutura bi-dimensional com vários dobramentos, fissuras e elevações.
- Diferentes partes do córtex possuem diferentes funções (ver figura).
- Em geral os neurônios do córtex estão organizados em camadas distintas, que são sub-divididas em *camada de entrada*, *camadas intermediárias* ou *escondidas* e *camada de saída*.



- A camada de entrada recebe os sinais sensoriais ou de entrada, a camada de saída envia sinais para outras partes do cérebro e as camadas intermediárias recebem (enviam) sinais de (para) outras camadas do córtex. Isso significa que as camadas intermediárias nem recebem entradas diretamente e nem produzem uma saída do tipo motora, por exemplo.
- Um princípio organizacional importante em vários sistemas sensoriais e motores é o *mapa topográfico*.
  - Por exemplo, neurônios em áreas visuais do córtex estão arranjados topograficamente, no sentido de que neurônios adjacentes possuem campos de recepção visual adjacentes e, coletivamente, eles constituem um mapa da retina.
  - Obs: Como neurônios vizinhos ocupam-se de representações similares, mapas topográficos constituem uma forma parcimoniosa de organização do cérebro (há economia de conexões, por exemplo).



- A organização estrutural em redes, mapas topográficos e camadas são todos casos especiais de um princípio mais geral: a exploração das propriedades estruturais e espaço-temporais para o processamento e armazenagem de informação.

1.2. Base Biológica e Física da Aprendizagem e Memória

- O sistema nervoso está continuamente sofrendo modificações e atualizações. Virtualmente todas as suas funções, incluindo percepção, controle motor, regulação térmica e raciocínio, são modificadas por estímulos.
- Observações comportamentais permitiram verificar graus de plasticidade do sistema nervoso: existem mudanças rápidas e fáceis, mudanças lentas e profundas, e mudanças mais permanentes (porém ainda modificáveis).
- Em geral, a aprendizagem global é resultado de alterações locais nos neurônios.
- Existem diversas formas de modificações possíveis em um neurônio:
  - Dendritos podem nascer, assim como também podem ser removidos
  - Alguns dendritos podem se esticar ou ser encolhidos permitindo ou eliminando, respectivamente, a conexão com outras células
  - Novas sinapses podem ser criadas ou sofrerem alterações
  - Sinapses também podem ser removidas
  - Todo neurônio pode morrer e também se regenerar.

- Toda esta vasta gama de adaptação estrutural pode ser convenientemente condensada simplesmente referindo-se às sinapses, pois estas modificações envolvem a modificação sináptica de forma direta ou indireta.
- Sendo assim, a *aprendizagem via modulação sináptica* é o mecanismo mais importante para as redes neurais, sejam elas biológicas ou artificiais.
- A modulação sináptica poderá depender de mecanismos de adaptação de neurônios individuais e de redes neurais como um todo.

2. Redes Neurais Artificiais

- Uma RNA pode ser definida como sendo uma estrutura de processamento (rede), passível de implementação em dispositivos eletrônicos, composta por um número de unidades interconectadas (neurônios artificiais), sendo que cada unidade apresenta um comportamento específico de entrada/saída (computação local), determinado pela sua função de transferência, pelas interconexões com outras

unidades, dentro de um raio de vizinhança, e possivelmente pelas entradas externas.

- Uma rede neural artificial é um circuito composto por uma grande quantidade de unidades simples de processamento inspiradas no sistema neural (Nigrin, 1993).
- Uma RNA é um sistema massivamente paralelo e distribuído, composto por unidades de processamento simples que possuem uma capacidade natural de armazenar e utilizar conhecimento. (Haykin, 1999)
- As RNAs apresentam diversas características em comum com o sistema nervoso:
  - O processamento básico de informação ocorre em diversas unidades simples denominadas de *neurônios artificiais* ou simplesmente *neurônios* (ou *nós*);
  - Os neurônios estão interconectados gerando redes de neurônios, ou redes neurais;
  - A informação (sinais) é transmitida entre neurônios através de conexões ou sinapses;

- A eficiência de uma sinapse, representada por um *peso* associado, corresponde à informação armazenada pelo neurônio e, portanto, pela rede neural; e
- O conhecimento é adquirido do ambiente através de um processo de *aprendizagem* que é, basicamente, responsável por adaptar os pesos das conexões aos estímulos recebidos do ambiente.

- Uma característica importante das RNAs é o local onde o conhecimento está armazenado. Nos casos mais simples, este conhecimento é armazenado nos pesos das conexões entre neurônios.
- Esta característica tem grandes implicações para a capacidade de processamento e aprendizagem da rede.
- A representação de conhecimento é feita de forma que o conhecimento necessariamente influencie a forma de processamento da rede, ou seja, o seu comportamento de entrada-saída.

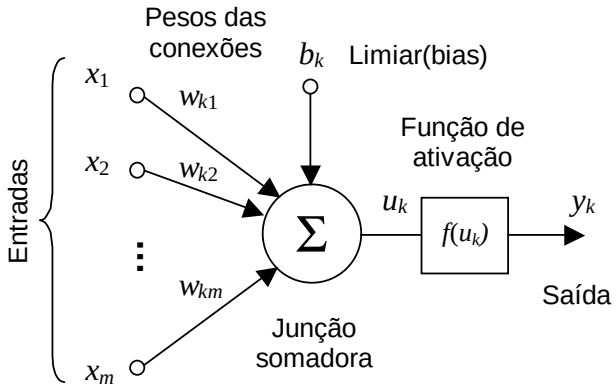
- *Se o conhecimento está armazenado nos pesos das conexões, então o processo de aprendizagem corresponde a identificar um conjunto apropriado de pesos de forma que a rede se comporte como desejado*.
- Esta característica possui duas implicações importantes para as RNAs: a possibilidade de desenvolvimento de técnicas de aprendizagem, e a *representação distribuída* de conhecimento.
- Existem tipos de redes neurais cujo treinamento (ou projeto) é mais complicado do que a simples determinação de conjuntos apropriados de pesos sinápticos.
- Uma rede neural artificial pode ser projetada através de:
  1. Uma definição ou escolha de um conjunto de neurônios artificiais;
  2. A definição ou escolha de um padrão de conectividade entre os neurônios, ou seja, de uma *arquitetura* para a rede; e
  3. A definição de um método de determinação dos parâmetros livres da rede, denominado de *algoritmo de aprendizagem* ou *treinamento*.

- Embora seja possível projetar uma rede neural a partir da definição do papel (computação global) que ela deve desempenhar, combinando-se os efeitos individuais de todos os neurônios, uma rede neural usualmente se adapta para atingir a funcionalidade desejada a partir de uma ou mais estratégias de aprendizado, as quais vão atuar junto a parâmetros configuráveis da rede neural.
- É fundamental, portanto, que a rede neural possua meios de interagir com o ambiente.
- Cada rede neural artificial representa uma arquitetura de processamento específica, havendo uma família de arquiteturas, cada qual adequada para funcionalidades específicas.
- A diversidade de arquiteturas tem aumentado muito, sendo que as últimas gerações de redes neurais já não podem ser caracterizadas por apresentarem um grande número de unidades e conexões, com funções de transferência simples e idênticas para todas as unidades.

2.1. O Neurônio Genérico em RNAs

- No neurônio biológico, os sinais de entrada chegam através de canais localizados nas sinapses, permitindo a entrada e saída de íons. Um potencial de membrana aparece como resultado da integração dos sinais de entrada, que irão determinar se o neurônio irá produzir um sinal de saída (*spike*, pulso, ou potencial de ação) ou não. O potencial de ação resulta na liberação de neurotransmissores na sinapse sempre que o potencial de membrana for superior a um determinado limiar (*threshold*).
- O efeito líquido de todos estes processos biológicos que ocorrem nas sinapses é representado por um *peso* associado.
- O elemento computacional básico empregado na maioria das RNAs é um integrador. Trata-se de um elemento processador de informações que é fundamental para a operação das RNAs.
- As principais partes do neurônio artificial genérico são:

- as sinapses, caracterizadas pelos seus pesos associados;
- a junção somadora; e
- a função de ativação.



- Nesta representação, o primeiro subscrito k do peso sináptico w_{kj} corresponde ao neurônio pós-sináptico, e o segundo subscrito corresponde à sinapse ligada a ele.

- A junção somadora soma todos os sinais de entrada ponderados pelos pesos das conexões. Assumindo os vetores de entrada e de pesos como sendo vetores coluna, esta operação corresponde ao produto interno do vetor de entradas **x** pelo vetor de pesos $\mathbf{w}_k$, mais o limiar b_k. Genericamente, trata-se de uma combinação linear das entradas pelos pesos associados, mais o limiar b_k.
- A função de ativação é geralmente utilizada com dois propósitos: limitar a saída do neurônio e introduzir não-linearidade no modelo.
- O limiar b_k tem o papel de aumentar ou diminuir a influência do valor da entrada líquida para a ativação do neurônio k.
- Matematicamente, a saída do neurônio k pode ser descrita por:

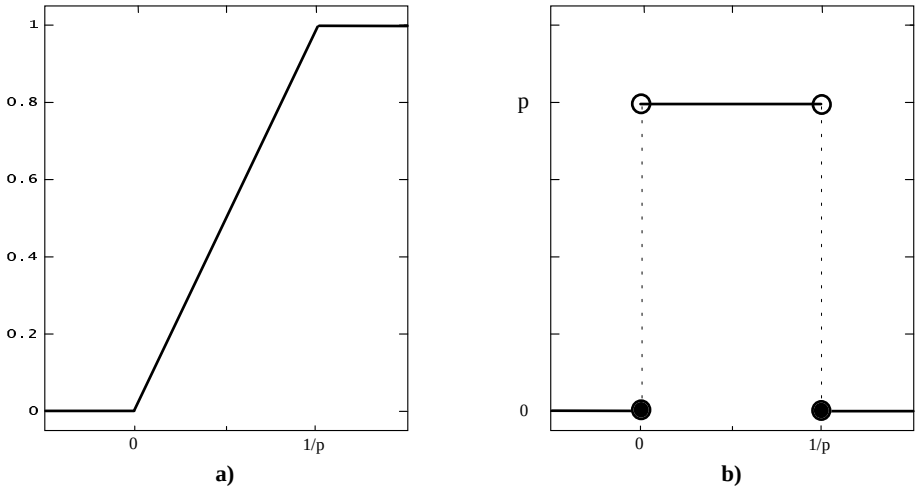
$$y_k = f(u_k) = f\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right) \quad \text{ou} \quad y_k = f(u_k) = f\left(\sum_{j=0}^{m} w_{kj} x_j\right),$$

onde x_0 é um sinal de entrada de valor 1 e peso associado $w_{k0} = b_k$:

- Exemplos de função de ativação:

$$f(\mathbf{u}_k) = \begin{cases} 1 \text{ se } p\mathbf{u}_k \geq 1 \\ p\mathbf{u}_k \text{ se } 0 < p\mathbf{u}_k < 1 \\ 0 \text{ se } p\mathbf{u}_k \leq 0 \end{cases}$$

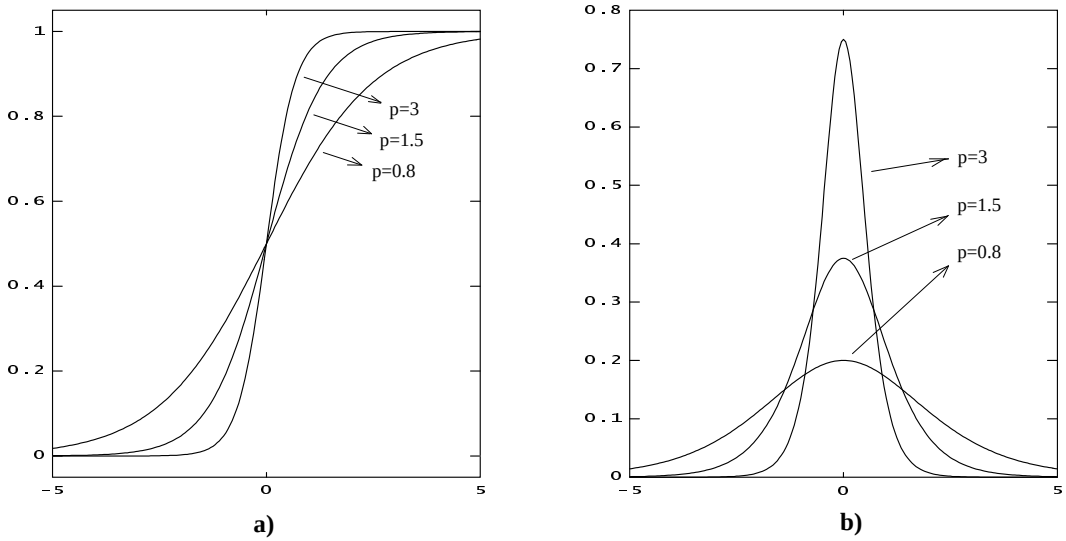
com p constante e positivo.



a) **b)**

Função semi-linear (a) e sua derivada em relação à entrada interna (b)

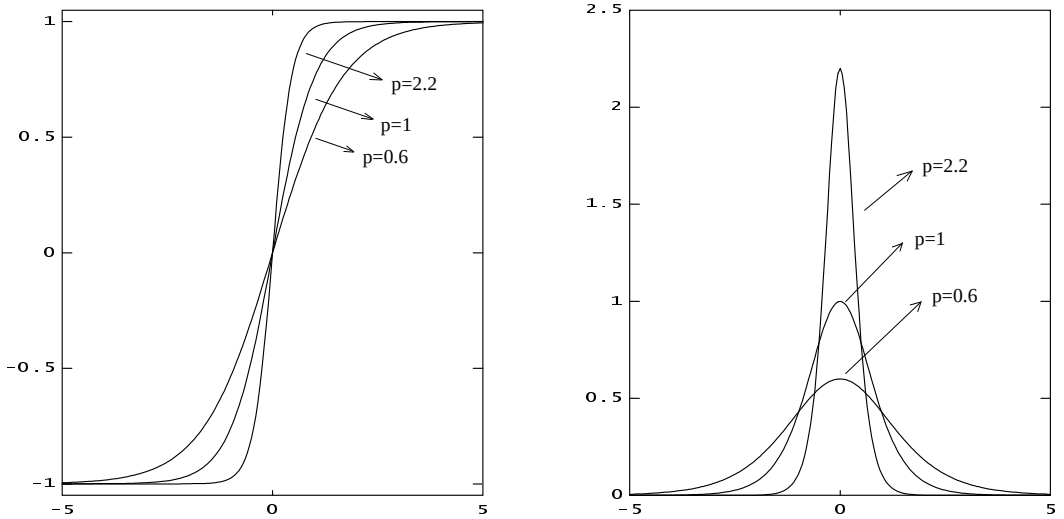
$$y = f(\mathbf{u}_k) = \frac{e^{p\mathbf{u}_k}}{e^{p\mathbf{u}_k} + 1} = \frac{1}{1 + e^{-p\mathbf{u}_k}} \qquad \frac{\partial y}{\partial \mathbf{u}_k} = p\mathbf{u}_k(1 - \mathbf{u}_k) > 0$$



a) **b)**

Função logística (a) e sua derivada em relação à entrada interna (b)

$$y = f(\mathbf{u}_k) = \tanh(p\mathbf{u}_k) = \frac{e^{p\mathbf{u}_k} - e^{-p\mathbf{u}_k}}{e^{p\mathbf{u}_k} + e^{-p\mathbf{u}_k}} \qquad \frac{\partial y}{\partial \mathbf{u}_k} = p\left(1 - \mathbf{u}_k^2\right) > 0$$

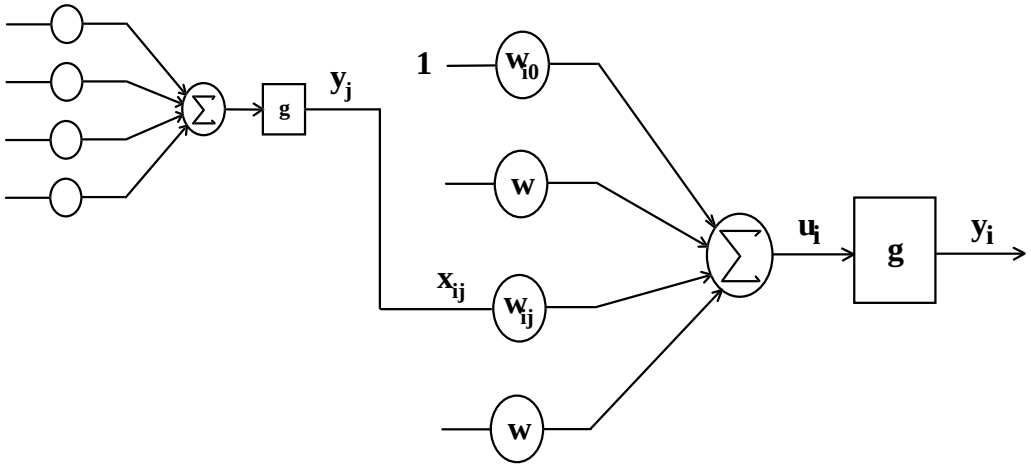


a) **b)**

Função tangente hiperbólica (a) e sua derivada em relação à entrada interna (b)

2.2. Arquiteturas de Rede

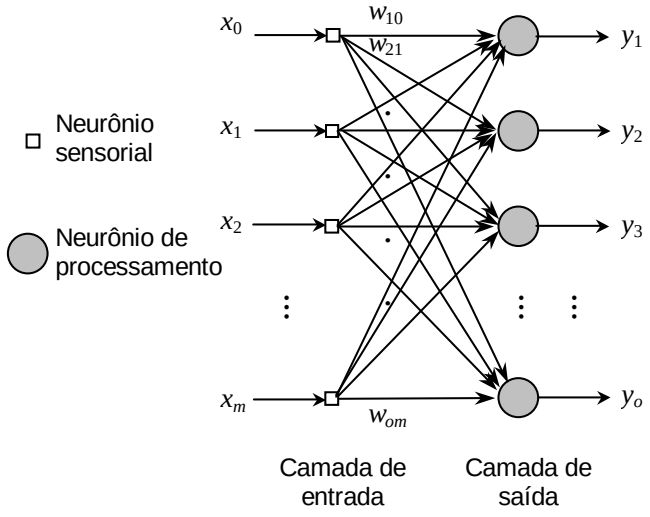
- Muito pouco é sabido sobre os padrões de conexão entre os neurônios biológicos.
- Entretanto, a maioria das RNAs utilizam arquiteturas padronizadas, projetadas especialmente para resolver algumas classes de problemas.
- O processo de conexão entre neurônios artificiais leva à geração de sinapses e à construção de redes neurais artificiais.



- Existem basicamente três camadas em uma rede neural artificial: uma *camada de entrada*, uma *camada intermediária*, e uma *camada de saída*. Entretanto, nem todas as RNAs possuem camadas intermediárias.
- A forma pela qual os neurônios estão interconectados está intimamente relacionada ao algoritmo a ser utilizado no seu treinamento.
- Existem, basicamente, três tipos principais de arquitetura em RNAs: *redes feedforward de uma única camada*, *redes feedforward de múltiplas camadas*, e *redes recorrentes*.

Rede Feedforward com Uma Única Camada

- Este caso mais simples de rede em camadas consiste em uma camada de entrada e uma camada de saída.
- Geralmente os neurônios de entrada são lineares, ou seja, eles simplesmente propagam o sinal de entrada para a próxima camada. São também denominados de neurônios sensoriais.



- Esta rede é denominada *feedforward* porque a propagação do sinal ocorre apenas da entrada para a saída, ou seja, é apenas no sentido positivo.

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{o0} & w_{o1} & \cdots & w_{om} \end{bmatrix}$$

$y_i = f(\mathbf{w}_i.\mathbf{x}) = f(\Sigma_j\, w_{ij}.x_j)$, $j = 1,\ldots,m$.

Note que a primeira coluna de **W** corresponde ao vetor de bias.

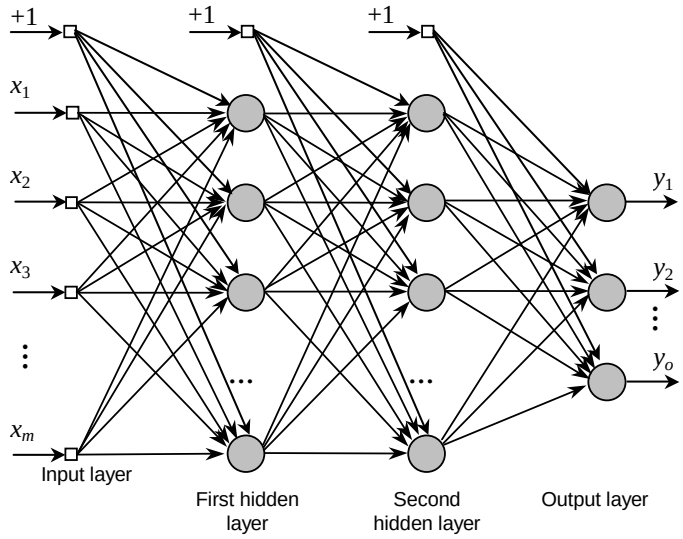
- Em forma matricial:

  $\mathbf{y} = f(\mathbf{W}.\mathbf{x})$,

  onde $\mathbf{W} \in \Re^{o\times m}$, $\mathbf{w}_i \in \Re^{1\times m}$, $i = 1,\ldots,o$, $\mathbf{x} \in \Re^{m\times 1}$, e $\mathbf{y} \in \Re^{o\times 1}$.

Rede Feedforward de Múltiplas Camadas

- As redes de múltiplas camadas possuem uma ou mais camadas intermediárias ou escondidas. Adicionando-se camadas intermediárias não-lineares é possível aumentar a capacidade de processamento de uma rede feedforward.
- A saída de cada camada intermediária é utilizada como entrada para a próxima camada.
- Em geral o algoritmo de treinamento para este tipo de rede envolve a retropropagação do erro entre a saída da rede e uma saída desejada conhecida.



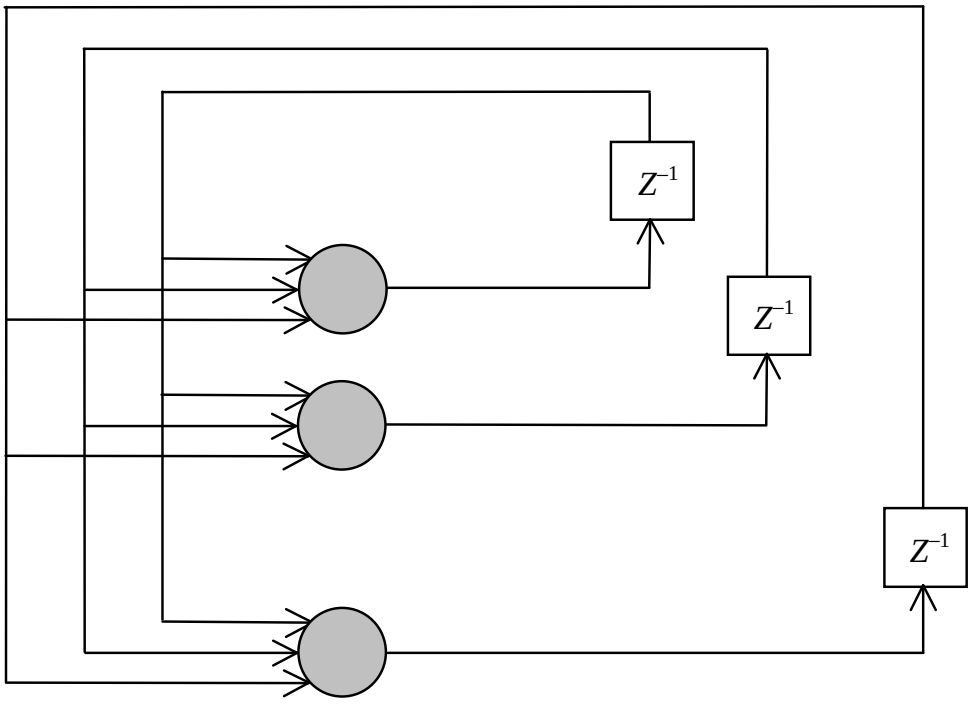
- Seja $\mathbf{W}^k$ a matriz de pesos da camada *k*, contadas da esquerda para a direita.
- w_{ij}^k corresponde ao peso ligando o neurônio pós-sináptico *i* ao neurônio pré-sináptico *j* na camada *k*.
- Em notação matricial, a saída da rede é dada por:

$\mathbf{y} = \mathbf{f}^3(\mathbf{W}^3\, \mathbf{f}^2(\mathbf{W}^2\, \mathbf{f}^1(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$

- Note que $\mathbf{f}^k$, $k = 1,..., M$ (M = número de camadas da rede) é uma matriz quadrada $\mathbf{f}^k \in \Re^{l \times l}$, onde l é o número de neurônios na camada k.
- O que acontece se as funções de ativação das unidades intermediárias forem lineares?

Redes Recorrentes

- O terceiro principal tipo de arquitetura de RNAs são as denominadas de redes recorrentes, pois elas possuem, pelo menos, um laço realimentando a saída de neurônios para outros neurônios da rede.



Rede neural recorrente de Hopfield.

2.3. Paradigmas de Aprendizagem

- A capacidade de "aprender" associada a uma rede neural é uma das mais importantes qualidades destas estruturas.
- Trata-se da habilidade de adaptar-se, de acordo com regras pré-existentes, ao seu ambiente, alterando seu desempenho ao longo do tempo.
- Sendo assim, considera-se "aprendizado" o processo que adapta o comportamento e conduz a uma melhoria de desempenho.
- No contexto de redes neurais artificiais, *aprendizagem* ou *treinamento* corresponde ao processo de ajuste dos parâmetros livres da rede através de um mecanismo de apresentação de estímulos ambientais, conhecidos como padrões (ou dados) de entrada ou de treinamento:

$$\text{estímulo} \rightarrow \text{adaptação} \rightarrow \text{novo comportamento da rede}$$

- Nas RNAs mais simples e tradicionais, os parâmetros livres da rede correspondem apenas aos pesos sinápticos. Toda a estrutura da rede, incluindo os tipos de neurônios e suas funções de ativação, é pré-definida.
- O objetivo do aprendizado em redes neurais é a obtenção de um modelo implícito do sistema em estudo, por ajuste dos parâmetros da rede.
- Dada uma rede neural artificial, seja $w(t)$ um peso sináptico de um dado neurônio, no instante de tempo t. O ajuste $\Delta w(t)$ é aplicado ao peso sináptico $w(t)$ no instante t, gerando o valor corrigido $w(t+1)$, na forma:

$$w(t+1) = w(t) + \Delta w(t)$$

- A obtenção de $\Delta w(t)$ pode ser feita de diversas formas. O tipo de aprendizado é determinado pela técnica empregada no processo de ajuste dos pesos sinápticos (parâmetros da rede neural).

- Um conjunto bem definido de regras para obtê-los é denominado um algoritmo de aprendizagem ou treinamento. Exemplos de alguns algoritmos: regra de Hebb, algoritmo de *backpropagation*, estratégias de competição, máquina de Boltzmann.
- A maneira pela qual o ambiente influencia a rede em seu aprendizado define o paradigma de aprendizagem. Exemplos de paradigmas: aprendizado supervisionado, aprendizado por reforço e aprendizado não-supervisionado (ou auto-organizado).
- Existem basicamente três paradigmas de aprendizado: *aprendizado supervisionado*, *aprendizado não-supervisionado* e *aprendizado por reforço*.

Aprendizagem Supervisionada

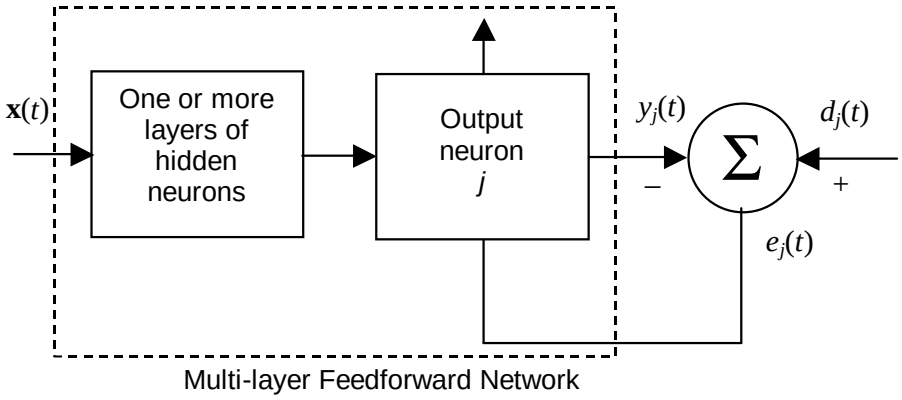
- Este curso vai se ocupar com o desenvolvimento de técnicas para aprendizado supervisionado e não-supervisionado em redes neurais artificiais.
- Pelo fato de serem mais intuitivas, técnicas de aprendizado supervisionado serão abordadas primeiro.

- Idéia intuitiva: controle de processos (ex. pouso e decolagem de aviões)
- Exemplos de problemas de engenharia que podem ser apresentados na forma de um problema de aprendizado supervisionado:
  - classificação e reconhecimento de padrões
  - predição de séries temporais
  - identificação de sistemas
  - controle de processos
  - projeto de filtros em processamento de sinais

Formalização do processo de aprendizado supervisionado

- Seja $d_j(t)$ a resposta desejada para o neurônio j no instante t e $y_j(t)$ a resposta observada do neurônio j no instante t, obtida através de um estímulo $x(t)$ presente na entrada da rede neural.

- $x(t)$ e $d_j(t)$ constituem um EXEMPLO de par estímulo-resposta apresentado ao neurônio no instante t, possivelmente extraídos de um ambiente ruidoso, cujas distribuições de probabilidade são desconhecidas.
- $e_j(t) = d_j(t) - y_j(t)$ é o sinal de erro observado na saída do neurônio j no instante t. Observe que, em ambiente ruidoso, $e_j(t)$ é uma variável aleatória.
- O processo de aprendizado supervisionado tem por objetivo corrigir este erro observado (em todos os neurônios), e para tanto busca minimizar um critério (função objetivo) baseado em $e_j(t)$, j=1,2,…, o, onde o é o número de neurônios da rede neural, de maneira que, para t suficientemente alto, $y_j(t)$, j=1,2,…, o, estejam próximos de $d_j(t)$, j=1,2,…, o, no sentido estatístico.



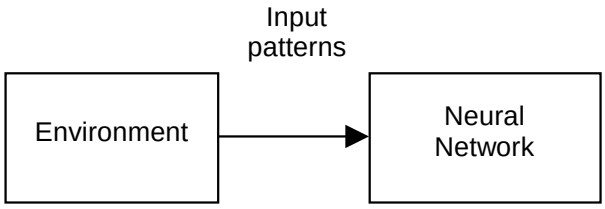
- Um critério muito utilizado é o de erro quadrático médio: $J = E\left[\frac{1}{o}\sum_{j=1}^{o} e_j^2(t)\right]$.
- Um conceito que está implícito em toda esta análise é a hipótese de estacionariedade dos processos aleatórios presentes.
- Além disso, para minimizar J é necessário conhecer as características estatísticas do sistema.

- Uma aproximação para o critério é utilizar o valor instantâneo do erro quadrático médio: $J \cong J(k) = \frac{1}{o}\sum_{j=1}^{o} e_j^2(k)$.
- Nesta fase do curso, vamos considerar que a minimização de $J(t)$ é realizada apenas em relação aos pesos sinápticos da rede neural. Mais adiante, serão apresentados procedimentos para ajuste das funções de ativação dos neurônios e da arquitetura da rede neural.

Aprendizagem Não-Supervisionada

- No paradigma não-supervisionado ou auto-organizado não existe nenhum supervisor para avaliar o desempenho da rede em relação aos dados de entrada.
- Nenhuma medida de erro é utilizada para realimentar a rede.
- Os dados são ditos não-rotulados, no sentido de que as classes às quais eles pertencem ou as saídas desejadas da rede são desconhecidas.

- A rede se adapta a regularidades estatísticas nos dados de entrada, desenvolvendo uma capacidade de criar representações internas que codificam as características dos dados de entrada, tornando-se portanto capaz de identificar a quais classes novos padrões pertencem.



- Geralmente as redes auto-organizadas empregam um *algoritmo competitivo de aprendizagem*.
- Na aprendizagem competitiva, os neurônios de saída da rede competem entre si para se tornarem ativos, com um único neurônio sendo o vencedor da competição.

- Esta propriedade é que faz com que o algoritmo seja capaz de descobrir regularidades estatísticas no conjunto de dados.
- Neurônios individuais aprendem a se especializar a conjuntos (grupos ou clusters) de padrões similares. Eles se tornam detectores ou extratores de características para diferentes classes dos dados de entrada.
- Idéia intuitiva: agrupamento de dados (ex. balões coloridos)

Formalização do processo de aprendizado competitivo

- Para que um neurônio *i* seja o vencedor, a distância entre este o vetor de pesos $\mathbf{w}_i$ deste neurônio e um determinado padrão de entrada $\mathbf{x}$ deve ser a menor dentre todos os outros neurônios da rede, dada uma métrica de distância $\|\cdot\|$ (geralmente utiliza-se a distância Euclidiana).
- A idéia é encontrar o neurônio cujo vetor de pesos seja o mais parecido ao padrão de entrada, ou seja:

$$i = \arg\min_i \|\mathbf{x} - \mathbf{w}_i\|, \ \forall i.$$

- Se um neurônio não responde a um padrão de entrada, ou seja, não é o vencedor, então nenhuma adaptação é sofrida por este neurônio.
- Entretanto, o neurônio *i* que ganhou a competição sofre um ajuste $\Delta\mathbf{w}_i$ no seu vetor de pesos na direção do vetor de entrada:

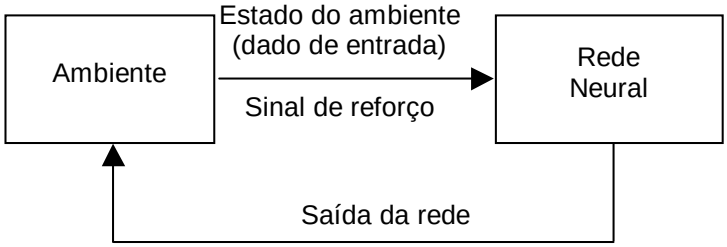
$$\Delta\mathbf{w}_i = \begin{cases} \alpha(\mathbf{x} - \mathbf{w}_i) & \text{se } i \text{ ganha a competição} \\ 0 & \text{se } i \text{ perde a competição} \end{cases}$$

onde α indica o tamanho do passo a ser dado na direção de $\mathbf{x}$. O parâmetro α é conhecido como taxa de aprendizagem.

Aprendizagem Por Reforço

- A aprendizagem por reforço é distinta das outras abordagens, pois neste caso não existe uma interação direta com um supervisor ou modelo explícito do ambiente.
- Geralmente, a única informação disponível é um valor escalar que indica a qualidade do desempenho da RNA.

- Na aprendizagem por reforço existe um objetivo a ser alcançado. Durante o processo de aprendizagem, a rede "tenta" algumas ações (saídas) e recebe um sinal de reforço (estímulo) do ambiente que permite avaliar a qualidade de sua ação.
- O sistema em aprendizagem seletivamente retém as ações que levam a uma maximização dos sinais de reforço.
- Idéia intuitiva: ex. ensinar animais circenses.
- A cada iteração t, o sistema em aprendizagem recebe uma entrada $\mathbf{x}(t)$ (representando o estado do ambiente), fornece uma saída $y(t)$, e no próximo passo recebe um escalar de reforço $r(t+1)$ e um novo estado do ambiente $\mathbf{x}(t+1)$.
- Portanto, os dois conceitos básicos por trás da aprendizagem por reforço são: *busca por tentativa e erro* e *reforço retardado*.



3. Alguns Algoritmos de Aprendizado Supervisionado

3.1. Perceptron

- Rosenblatt introduziu o perceptron como a arquitetura mais simples de rede neural capaz de classificar padrões linearmente separáveis.
- O algoritmo de treinamento do perceptron foi o primeiro modelo de treinamento supervisionado, embora alguns perceptrons fossem auto-organizados.
- Basicamente, o perceptron consiste em uma única camada de neurônios com pesos sinápticos e bias ajustáveis.

- Se os padrões de entrada forem linearmente separáveis, o algoritmo de treinamento do perceptron possui convergência garantida, ou seja, é capaz de encontrar um conjunto de pesos que classifica corretamente os dados.
- Os pesos dos neurônios que compõem o perceptron serão tais que as superfícies de decisão produzidas pela rede neural estarão apropriadamente posicionadas no espaço.
- Os neurônios do perceptron são similares ao neurônio de McCulloch & Pitts (função de ativação tipo degrau), mas possuem pesos associados, incluindo o bias.

Perceptron Simples para Classificação de Padrões

- O algoritmo do perceptron funciona como a seguir.
  - Para cada padrão de treinamento (dado de entrada) $\mathbf{x}_i$, a saída da rede y_i é calculada.
  - Em seguida, é determinado o erro e_i entre a saída desejada para este padrão d_i e a saída da rede y_i, $e_i = d_i - y_i$.

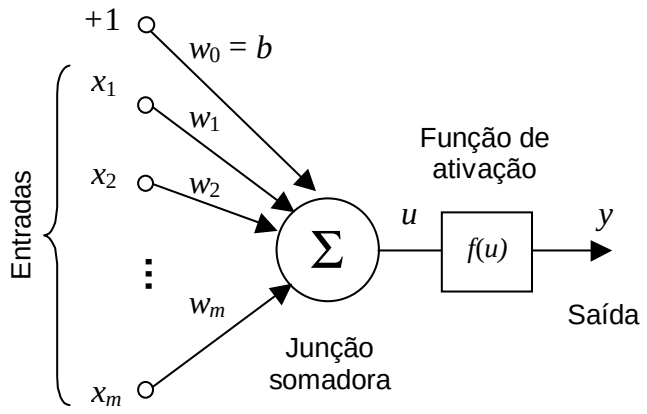
  - O vetor de pesos conectando as entradas (neurônios pré-sinápticos) a cada saída (neurônios pós-sinápticos) e o bias do neurônio são atualizados de acordo com as seguintes regras:

    $\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha\, e_i\, \mathbf{x}_i$,

    $b(t+1) = b(t) + \alpha\, e_i$,

    onde $\mathbf{w} \in \Re^{1\times m}$, $\mathbf{x} \in \Re^{1\times m}$, e $b \in \Re^{1\times 1}$.

- Considere agora o caso mais simples do perceptron com um único neurônio.



- O objetivo desta rede, mais especificamente deste neurônio, é classificar alguns padrões de entrada como pertencentes ou não pertencentes a uma dada classe.
- Considere o conjunto de dados de entrada como sendo formado por N amostras $\{\mathbf{x}_1,d_1\}$, $\{\mathbf{x}_2,d_2\}$, .., $\{\mathbf{x}_N,d_N\}$, onde $\mathbf{x}_j$ é o vetor j de entradas, e d_j sua saída desejada (classe) correspondente.

- Seja $\mathbf{X} \in \Re^{m \times N}$, a matriz de dados de entradas com N padrões de dimensão m cada (colunas de $\mathbf{X}$), e $\mathbf{d} \in \Re^{1 \times N}$ o vetor de saídas desejadas. O algoritmo abaixo pode ser utilizado para treinar o perceptron de um único neurônio:

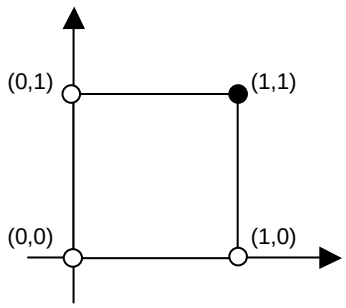
```
procedure [w] = perceptron(max_it,E,α,X,d)
  initialize w     // por simplicidade, inicialize com 0
  initialize b     //por simplicidade, inicialize com 0
  t ← 1
  while t < max_it & E > 0 do,
    for i from 1 to N do,  //para cada padrão de entrada
      y_i ← f(w x_i + b)          //determine a saída para x_i
      e_i ← d_i − y_i             //determine o erro para x_i
      w ← w + α e_i x_i           //atualize o vetor de pesos
      b ← b + α e_i               //atualize o bias
    end for
    E ← sum(e_i)
    t ← t + 1
  end while
end procedure
```

Algo 1: Simple perceptron learning algorithm. (The function $f(\cdot)$ is the threshold function)

Exemplo de Aplicação e Motivação Geométrica

- Considere o problema de utilizar o perceptron com um único neurônio para representar a função lógica AND.

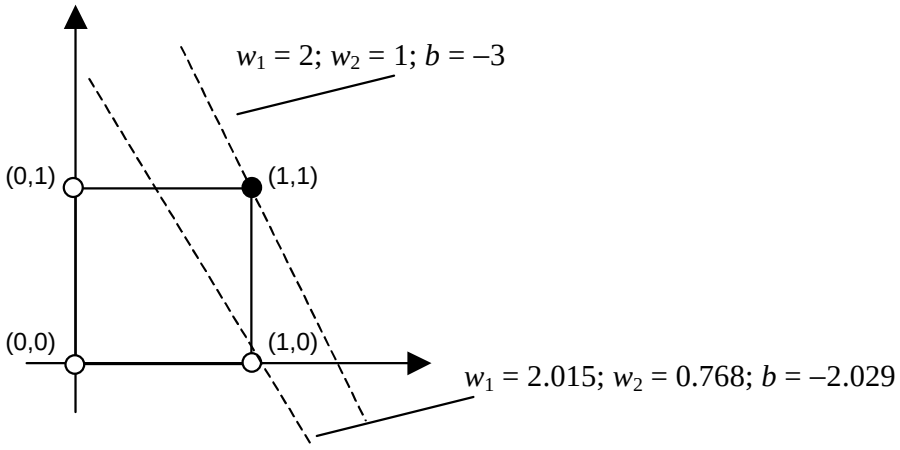
Entradas		Saídas
x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1



$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \mathbf{d} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

- A saída y_i do neurônio para o vetor de dados $\mathbf{x}_i$ pode ser representada na forma:
  $y_i = f(\mathbf{w}\mathbf{x}_i + b)$
- Para quaisquer valores de $\mathbf{w}$ e b, a função $f(\mathbf{u})$ separa o espaço de entradas em duas regiões, sendo que a curva de separação (*superfície de decisão*) é uma linha reta.

- A equação desta reta é dada por:
  $w_1 x_1 + w_2 x_2 + b = 0$
- Se a função de ativação do tipo sinal (degrau) possui $\theta = 0$, então
  $w_1 x_1 + w_2 x_2 + b \geq 0$
  resultará em uma saída positiva da rede.
- Inicializando todos os pesos e o limiar em zero $\mathbf{w} = [0\ 0]$ e $b = 0$, e definindo $\alpha = 1$, o algoritmo de treinamento do perceptron fornece o seguinte:
  $w_1 = 2$; $w_2 = 1$; $b = -3$, portanto $2x_1 + 1x_2 - 3 = 0$.
- Obs.: note que os pesos do perceptron também poderiam ter sido inicializados com valores aleatórios pequenos.
  - Neste caso, a superfície de decisão obtida seria diferente.
  - Considere para efeitos ilustrativos: $w_1 = 0.015$; $w_2 = 0.768$; $b = 0.971$.
  - A figura a seguir mostra as duas superfícies de decisão e os pesos e bias determinados pelo algoritmo de treinamento do perceptron.



Perceptron com Múltiplos Neurônios

- Note que a regra de aprendizagem do perceptron é do tipo supervisionada, empregando a aprendizagem por correção de erro.
- Esta regra pode ser facilmente estendida para atualizar os pesos de uma rede de neurônios em uma única camada.

- Neste caso, para cada vetor de entrada $\mathbf{x}_i$ haverá um vetor de saídas da rede:

$\mathbf{y}_i = f(\mathbf{W}\mathbf{x}_i + \mathbf{b})$; $\mathbf{W} \in \Re^{o \times m}$, $\mathbf{x}_i \in \Re^{m \times 1}$, $i = 1,..,N$, $\mathbf{y}_i \in \Re^{o \times 1}$, e $\mathbf{b} \in \Re^{o \times 1}$, $\mathbf{D} \in \Re^{o \times N}$.

- Existe agora um vetor de erros para cada padrão de entrada: $\mathbf{e}_i = \mathbf{d}_i - \mathbf{y}_i$.

```
procedure [W] = perceptron(max_it,α,X,D)
  initialize W //for simplicity set it to zero
  initialize b //for simplicity set it to zero
  t ← 1
  while t < max_it do,
    for i from 1 to N do,  //para cada padrão de entrada
      y_i ← f(W x_i + b)          //determine a saída da rede para x_i
      e_i ← d_i − y_i             //determine o vetor de erros para x_i
      W ← W + α e_i x_i^T         //atualize a matriz de pesos
      b ← b + α e_i               //atualize o vetor de bias
    end  for
    t ← t + 1
  end while
end procedure
```

Algo 2: Learning algorithm for the perceptron with multiple outputs.

Exemplo de Aplicação: Reconhecimento de Caracteres

- Considere o problema de aplicar o perceptron com múltiplas saídas ao problema de classificação (reconhecimento) dos seguintes caracteres binários:

- Cada um destes oito padrões de entrada possui uma resolução de 12×10 pixels e as classes a que eles pertencem (0,1,2,3,4,6,■,9) estão pré-definidas.
- Vamos projetar um perceptron com oito neurônios de saída, onde cada neurônio irá corresponder a uma classe. Temos então $\mathbf{X} \in \Re^{120\times 8}$ e $\mathbf{D} \in \Re^{8\times 8}$ (matriz diagonal).
- O algoritmo de treinamento do perceptron será responsável então por definir uma hipersuperfície de decisão em um espaço de dimensão 120 capaz de classificar os dados corretamente.

Aspectos Práticos do Treinamento do Perceptron

- **Condição inicial**: verificou-se que diferentes conjuntos iniciais de pesos para o perceptron podem levar a diferentes superfícies de decisão.
  - Na verdade, o problema de ajuste supervisionado de pesos pode ser visto como um processo de busca por um conjunto de pesos que otimizam uma determinada superfície de erro.
  - Sendo assim, uma escolha inadequada da condição inicial da rede pode levar o algoritmo a uma convergência para ótimos locais desta superfície de erro.
- **Critério de convergência**: no caso do perceptron, é possível garantir que, dado um conjunto de padrões linearmente separáveis, o algoritmo é capaz de encontrar uma superfície de decisão capaz de classificar corretamente os dados.
  - Sendo assim, é possível utilizar como critério de convergência para o perceptron simples (classificação binária) a determinação de erro igual a zero para todos os padrões de entrada.

- Outro critério que pode ser utilizado é a adaptação por uma quantidade finita de iterações, denominadas de *épocas de treinamento*.

- **Parâmetros de treinamento**: o algoritmo de treinamento do perceptron possui basicamente o parâmetro de treinamento α que deve ser definido pelo usuário.
- **Treinamento versus aplicação da rede**: é importante diferenciar entre o processo de treinamento e aplicação da rede.
  - O treinamento da rede corresponde ao processo de ajuste de pesos.
  - Após treinada, a rede poderá ser aplicada ao mesmo problema, de forma a verificar a qualidade do aprendizado, ou a outro problema, de forma a verificar sua *capacidade de generalização*.

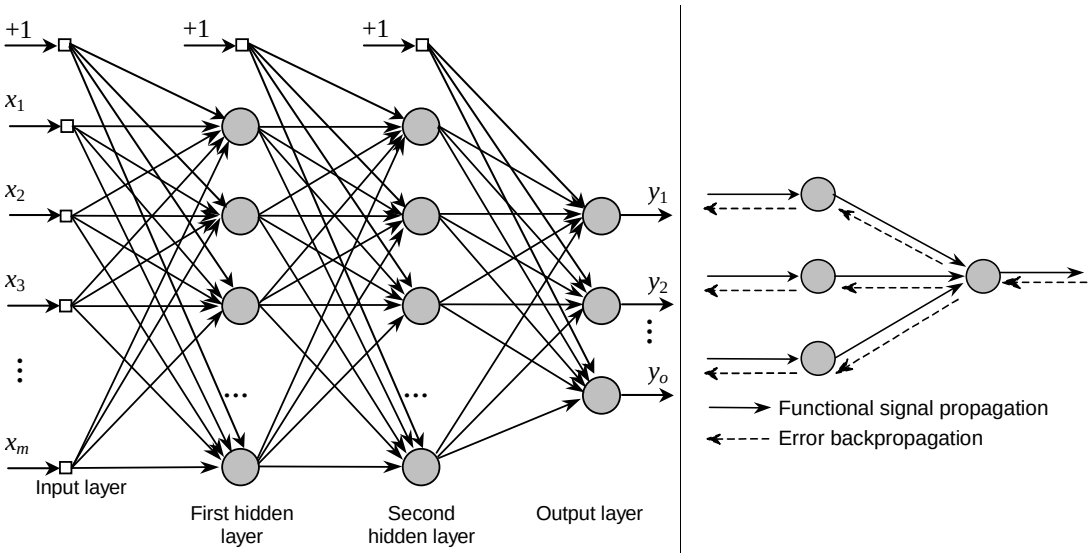
```
procedure [y] = perceptron(W,b,Z)
  for i from 1 to N do,    //para cada padrão de entrada x_i
     y_i ← f(Wx_i + b)     //determine as saídas da rede
  end  for
end procedure
```

Algo 3: Algorithm used to run the trained single-layer perceptron network.

3.2. Perceptron de Múltiplas Camadas

- O perceptron de múltiplas camadas (*multi-layer perceptron*) é uma rede do tipo perceptron com, pelo menos, uma camada intermediária.
- Trata-se de uma generalização do perceptron simples estudado anteriormente.
- O treinamento da rede MLP foi feito originalmente utilizando-se um algoritmo denominado de retropropagação do erro, conhecido como *backpropagation*.
- Este algoritmo consiste basicamente de dois passos:
  - *Propagação positiva do sinal funcional*: durante este processo todos os pesos da rede são mantidos fixos; e
  - *Retropropagação do erro*: durante este processo os pesos da rede são ajustados baseados no erro.
- O sinal de erro é propagado em sentido oposto ao de propagação do sinal funcional, por isso o nome de retro-propagação do erro.
- Uma rede MLP típica possui três características principais:

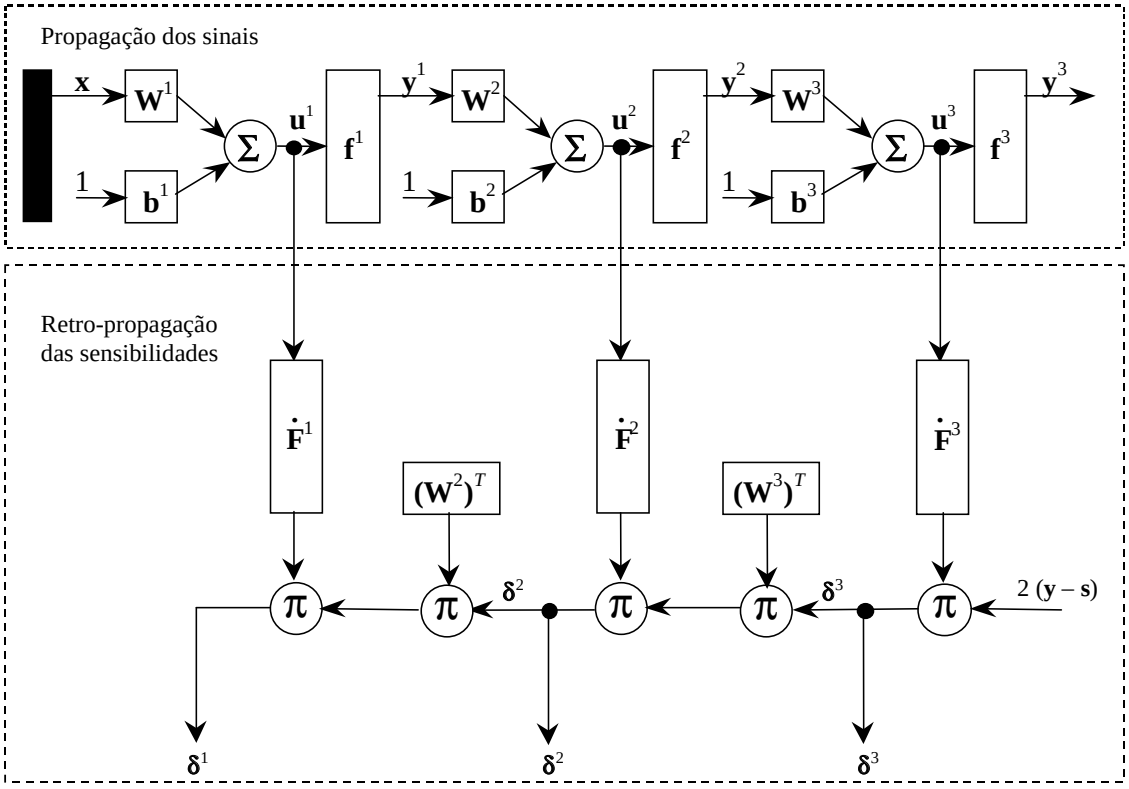
- Os neurônios das camadas intermediárias (e de saída) possuem uma função de ativação não-linear do tipo sigmoidal (p. ex. função logística ou tangente hiperbólica).
- A rede possui uma ou mais camadas intermediárias.
- A rede possui altos graus de conectividade.



- A regra delta generalizada será utilizada para ajustar os pesos e limiares (bias) da rede de forma a minimizar o erro entre a saída da rede e a saída desejada para todos os padrões de treinamento.
- Para isso, serão utilizadas informações sobre o gradiente do erro em relação aos pesos e limiares da rede, pois já sabemos que atualizando os pesos da rede na direção oposta ao vetor gradiente em relação aos pesos estamos minimizando o erro entre a saída da rede e a saída desejada.
- Entretanto, o erro é calculado diretamente apenas para a camada de saída. A pergunta que o algoritmo de retropropagação do erro responde é como determinar a influência do erro nas camadas intermediárias da rede.

$$w_{i,j}^m(t+1) = w_{i,j}^m(t) - \alpha \frac{\partial \Im(t)}{\partial w_{i,j}^m}, \qquad b_i^m(t+1) = b_i^m(t) - \alpha \frac{\partial \Im(t)}{\partial b_i^m} \quad \textbf{?}$$

- A derivação do algoritmo de backpropagation está fora do escopo deste curso, porém ela pode ser encontrada em diversas fontes.




```
procedure [W] = backprop(max_it,min_err,α,X,D)
  for m from 1 to M do,
    initialize W^m    //small random values
    initialize b^m    //small random values
  end for
  t ← 1
  while t < max_it & MSE > min_err do,
    for i from 1 to N do,   //for each training pattern
      //forward propagation of the functional signal
      y^0 ← x_i
      y_i^{m+1} ← f^{m+1}(W^{m+1} y_i^m + b^{m+1}), m = 0, 1, .., M – 1,
      //backpropagation of sensitivities
      δ_i^M ← –2 Ḟ^M(u_i^M)(d_i – y_i)
      δ_i^m ← Ḟ^m(u_i^m)(W^{m+1})^T δ_i^{m+1}, m = M – 1, ..,2,1,
      //update weights and biases
      W^m ← W^m – α δ_i^m (y_i^{m–1})^T, m = 1, .., M,
      b^m ← b^m – α δ_i^m, m = 1, .., M,
      E_i ← e_i^T e_i = (d_i – y_i)^T (d_i – y_i) //calculate the error for pattern i
    end for
    MSE ← 1/N.sum(E_i)                  //MSE
    t ← t + 1
  end while
end procedure
```

3.3. Capacidade de Aproximação Universal

- Uma arquitetura do tipo MLP pode ser vista como uma ferramenta prática geral para fazer um *mapeamento não-linear de entrada-saída*.
- Especificamente, seja *m* o número de entradas da rede e *o* o número de saídas. A relação entrada-saída da rede define um mapeamento de um espaço euclidiano de entrada *m*-dimensional para um espaço euclidiano de saída *o*-dimensional, que é infinitamente continuamente diferenciável.
- CYBENKO (1989) foi o primeiro pesquisador a demonstrar rigorosamente que uma rede MLP com uma única camada intermediária é suficiente para aproximar uniformemente qualquer função contínua que encaixe em um hipercubo unitário.
- O *teorema da aproximação universal* aplicável a redes MLP é descrito abaixo:

- **Teorema:** *Seja f*(.) *uma função contínua não-constante, limitada, e monotonicamente crescente. Seja* $\boldsymbol{I}_m$ *um hipercubo unitário m-dimensional* $(0,1)^m$. *O espaço das funções contínuas em* $\boldsymbol{I}_m$ *é denominado* $\boldsymbol{C}(\boldsymbol{I}_m)$. *Então, dada qualquer função* $g \in \boldsymbol{C}(\boldsymbol{I}_m)$ *e* $\varepsilon > 0$, *existe um inteiro M e conjuntos de constantes reais* α_i *e* w_{ij}, *onde* $i = 1, \ldots, M$ *e* $j = 1, \ldots, m$, *tais que pode-se definir*

$$F(x_1, x_2, ..., x_m) = \sum_{i=1}^{M} \alpha_i f\left(\sum_{j=1}^{m} w_{ij} x_j - w_{0i} \right)$$

como uma aproximação da função g(.) *tal que,*

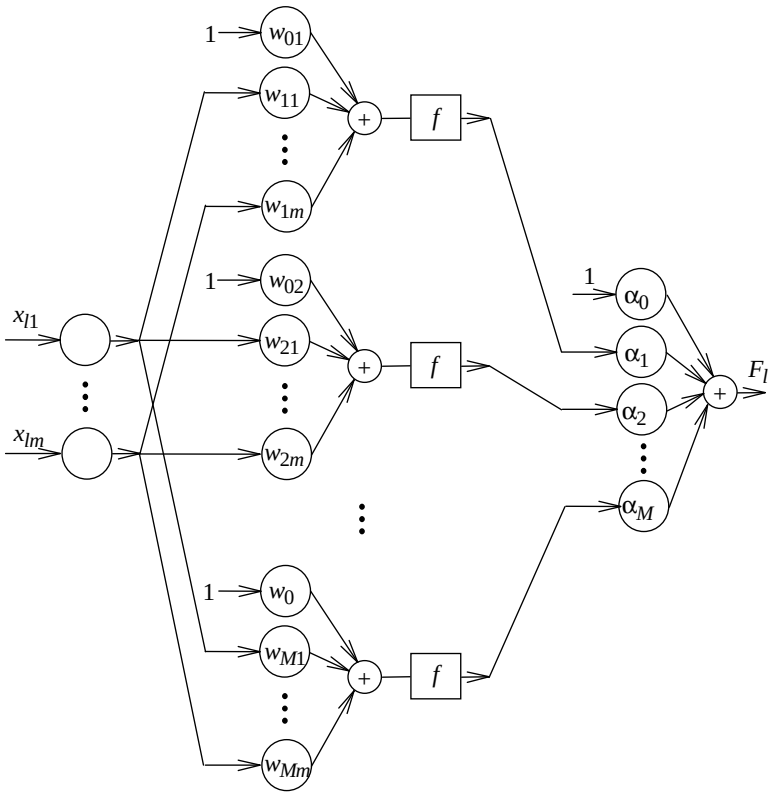
$$\left| F(x_1, x_2, ..., x_m) - g(x_1, x_2, ..., x_m) \right| < \varepsilon$$

Para todo $\{x_1, \ldots, x_m\} \in \boldsymbol{I}_m$.

- Este teorema é diretamente aplicável aos perceptrons de múltiplas camadas:
  - Percebemos que a função logística, ou tangente hiperbólica, utilizada como a não-linearidade de um neurônio é contínua, não-constante, limitada, e monotonicamente crescente; satisfazendo as condições impostas à função *f*(.).

- Verificamos também que a equação para a função $F(\cdot)$ representa as saídas de uma rede MLP descrita como segue:
  - a rede possui m nós de entrada e uma única camada intermediária com M unidades; o conjunto de entradas é $\{x_1,\ldots, x_m\}$
  - o neurônio intermediário i possui pesos $w_{i1}, \ldots, w_{im}$ e limiar w_{0i}
  - a saída da rede é uma combinação linear das saídas das unidades intermediárias, com $\alpha_1, \ldots, \alpha_M$ definindo os coeficientes dessa combinação.

- O teorema afirma que *um perceptron de múltiplas camadas com uma única camada intermediária é capaz de realizar uma aproximação uniforme, dado um conjunto de treinamento suficientemente significativo para representar a função*.
- Note que este teorema é um teorema de existência. Nada garante que um MLP com uma única camada é ótimo no sentido de tempo de processamento, facilidade de implementação e eficiência na representação.



3.4. Aspectos Práticos do Treinamento de Redes MLP

Treinamento Local ou em Lote

- Em aplicações práticas do algoritmo de retro-propagação, o aprendizado é resultado de apresentações repetidas de todas as amostras do conjunto de treinamento ao MLP.
- Cada apresentação de todo o conjunto de treinamento durante o processo de aprendizagem é chamada de *época*.
- O processo de aprendizagem é repetido época após época, até que um determinado critério de parada seja atingido (próxima seção).
- É uma boa prática fazer com que a ordem de apresentação das amostras seja feita aleatoriamente de uma época para a outra. Esta aleatoriedade tende a fazer com que a busca no espaço de pesos tenha um caráter estocástico ao longo dos ciclos de treinamento.

- Para um dado conjunto de treinamento, a atualização dos pesos pode ser feita de duas maneiras básicas:
  - **Atualização local**: neste método, a atualização dos pesos é feita imediatamente após a apresentação de cada amostra de treinamento. Especificamente, considere uma época consistindo de *N* padrões de treinamento arbitrariamente ordenados $\{(\mathbf{x}_1,\mathbf{s}_1),...,(\mathbf{x}_N,\mathbf{s}_N)\}$. A primeira amostra $(\mathbf{x}_1,\mathbf{s}_1)$ da época é apresentada à rede, e são efetuados o passo positivo e a retro-propagação do erro, resultando em um ajuste dos pesos das conexões. Em seguida, a próxima amostra $(\mathbf{x}_2,\mathbf{s}_2)$ é apresentada à rede e o passo positivo e a retro-propagação do erro são efetuados, resultando em mais um ajuste do conjunto de pesos. Este processo é repetido até que a última amostra seja apresentada à rede. Este método também é conhecido como método de atualização *on-line* ou *padrão a padrão*.
  - **Atualização em lote**: no método em lote, a atualização dos pesos só é feita após a apresentação de todas as amostras de treinamento que constituem uma época.

O ajuste relativo a cada apresentação de um par de entrada é acumulado (somado). Este método também é conhecido como método de atualização *off-line* ou *batch*.

Critérios de Parada

- O processo de minimização do MSE (ou da função custo), em geral, não tem convergência garantida e não possui um critério de parada bem definido.
- Um critério de parada não recomendável, por não levar em conta o estado do processo iterativo de treinamento, é interromper o treinamento após um número fixo (definido previamente) de iterações.
- Serão discutidos aqui critérios de parada que levam em conta alguma informação a respeito do estado do processo iterativo, cada qual com seu mérito prático.
- Para formular tal critério, deve-se considerar a possibilidade de existência de mínimos locais.

- Seja θ^* o vetor de parâmetros (pesos) que denota um ponto de mínimo, seja ele local ou global.
- Uma condição necessária para que θ^* seja um mínimo, é que o vetor gradiente $\nabla\Im(\theta)$ (ou seja, a derivada parcial de primeira ordem) da superfície de erro em relação ao vetor de pesos θ seja zero em $\theta = \theta^*$.
- Sendo assim, é possível formular critérios de convergência (ou parada) como segue:
  - *é considerado que o algoritmo de retro-propagação convergiu quando a norma Euclideana da estimativa do vetor gradiente* ($||\nabla\Im(\theta)||$) *atingiu um valor suficientemente pequeno.*

    Obs.: o problema deste critério de parada é que, para simulações bem sucedidas, o tempo de treinamento pode ser muito longo. Este critério também requer o cálculo da norma do vetor gradiente. O Tópico 12 fará um estudo sobre técnicas de aceleração da convergência deste algoritmo.

- *é considerado que o algoritmo de retro-propagação convergiu quando a variação do erro quadrático de uma época para a outra atingir um valor suficientemente pequeno*.

  Obs.: esse critério é proposto devido a uma outra propriedade única de um mínimo: o fato de que a função custo, ou medida de erro, $\Im_{med}(\theta)$ é estacionária no ponto $\theta = \theta^*$.

- *é considerado que o algoritmo de retro-propagação convergiu quando o erro quadrático médio atingir um valor suficientemente pequeno, ou seja,* $\Im_{med}(\theta) \leq \varepsilon$, *onde* ε *é um valor suficientemente pequeno*.

  Obs.: este critério é uma variação do critério anterior.

- Se o critério de parada é, por exemplo, um valor mínimo para o MSE, então não podemos garantir que o algoritmo será capaz de atingir o valor desejado.
- Por outro lado, ao tomarmos como critério de parada um valor mínimo para a norma do vetor gradiente devemos estar conscientes de que o algoritmo,

provavelmente, irá produzir como resultado o mínimo local mais próximo da condição inicial.

- Atualmente existe uma grande quantidade de pesquisadores procurando funções de erro (custo) alternativas, com o objetivo de melhorar as características de convergência dos perceptrons de múltiplas camadas.
- Um exemplo disso é a função de custo chamada de *entropia cruzada* (*cross-entropy*) (SHEPHERD, 1997), dada por:

$$\Im = -\sum_{n=1}^{N}\sum_{i=1}^{S^m} \ln\left[\left(y_{i,n}^{M}\right)^{s_{i,n}}\left(1-y_{i,n}^{M}\right)^{1-s_{i,n}}\right]$$

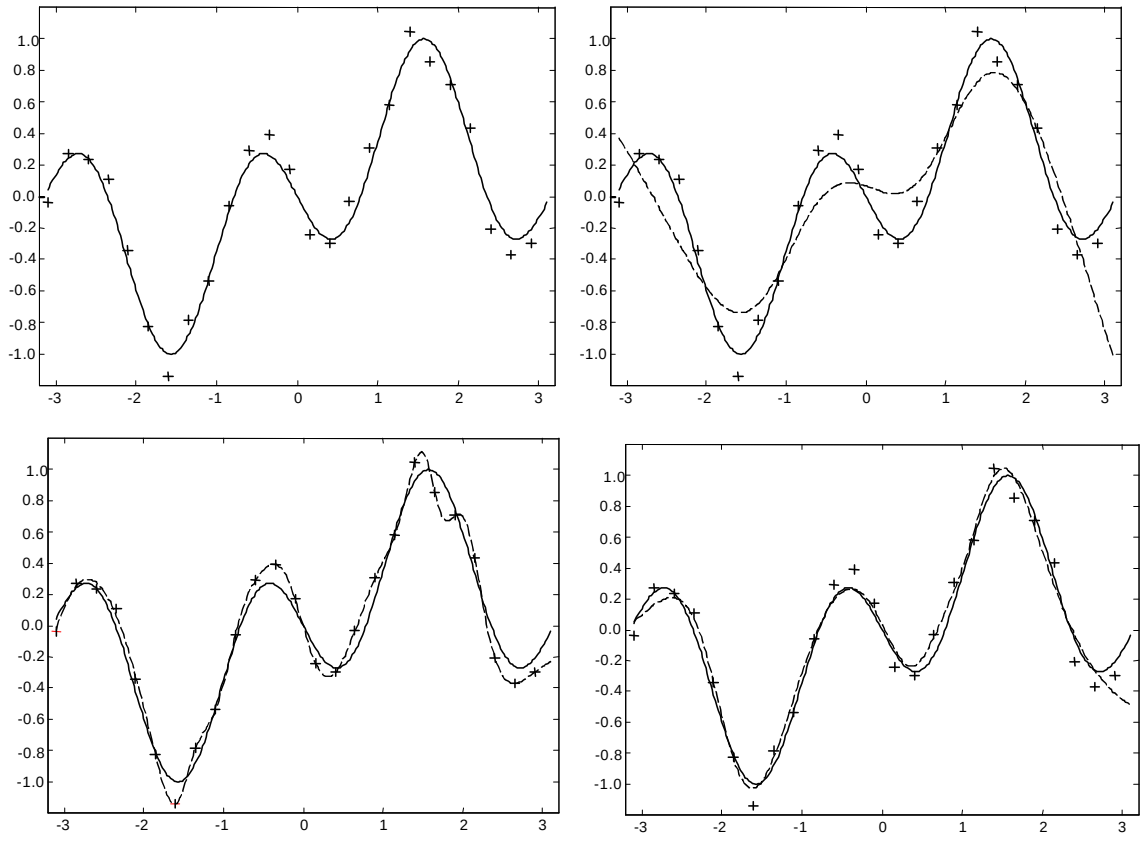
onde N é o número de amostras, M é o número de camadas, S^m é a dimensão de cada camada e y são as saídas da rede.

- Outro critério de parada bastante útil, e geralmente utilizado em conjunto com algum dos critérios anteriores, é a avaliação da capacidade de generalização da

rede após cada época de treinamento. O processo de treinamento é interrompido antes que a capacidade de generalização da rede seja deteriorada.

Arquitetura da Rede

- A quantidade de neurônios na camada de entrada da rede é geralmente dada pelo problema ser abordado.
- Entretanto, a quantidade de neurônios nas camadas intermediárias e de saída são características de projeto.
- Aumentando-se a quantidade de neurônios na camada intermediária aumenta-se a capacidade de mapeamento não-linear da rede MLP.
- Porém, é preciso tomar cuidado para que o modelo não se sobre-ajuste aos dados, pois se houver ruído nas amostras de treinamento, então a rede estará sujeita ao sobre-treinamento (*overfitting*).
- Uma rede com poucos neurônios na camada intermediária pode não ser capaz de aproximar o mapeamento desejado, causando um *underfitting*.



- O *underfitting* também pode ser causado caso o treinamento seja interrompido prematuramente.

Normalização dos Dados de Entrada

- Uma das características importantes das funções sigmoidais é a presença de saturação, ou seja, para valores muito grandes de seu argumento elas estarão operando na região saturada da curva.
- É importante portanto, fazer com que os valores dos atributos dos dados de entrada estejam contidos em um intervalo pré-definido, por exemplo, [0,1] ou [−1,1].

Inicialização dos Vetores de Pesos e Limiares (Bias)

- A qualidade e eficiência do aprendizado supervisionado em redes multicamadas depende fortemente da especificação de arquitetura da rede, função de ativação dos neurônios, regra de aprendizagem e valores iniciais do vetor de parâmetros (pesos).

- Valores ótimos destes itens são desconhecidos *a priori*, pois dependem principalmente do conjunto de treinamento e da natureza da solução (THIMM & FIESLER, 1997).
- Assumimos aqui que a arquitetura da rede, as funções de ativação dos neurônios e a regra de aprendizado já foram determinadas adequadamente, embora não necessariamente de maneira ótima. Sob essas considerações, um processo de treinamento bem sucedido passa a depender somente de uma boa definição do conjunto inicial de pesos, ou seja, um conjunto que guie o processo de treinamento para uma solução satisfatória, fora de mínimos locais pobres e problemas de instabilidade numérica.
- Logo, o conjunto inicial de pesos a ser utilizado no treinamento supervisionado de redes multicamadas possui grande influência na velocidade do aprendizado e na qualidade da solução obtida após a convergência.

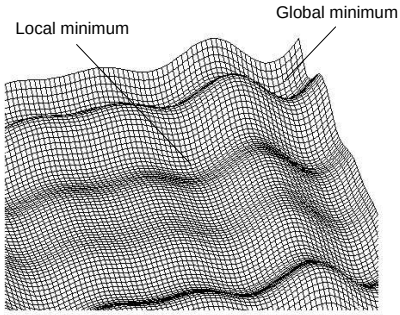
- Uma escolha inicial inadequada para os valores dos pesos pode fazer com que o treinamento não conduza a modelos de aproximação satisfatórios (mesmo que o processo de otimização seja bem-sucedido) ou apresente problemas numéricos que de outro modo poderiam ser evitados.
- Atualmente, existem várias técnicas propostas no sentido de definir adequadamente o conjunto de pesos iniciais.

3.5. Superfícies de Erro

- Seja uma rede neural com *n* pesos a serem ajustados. Este conjunto de pesos pode ser visto como um ponto em um espaço *n*-dimensional, denominado de espaço de pesos (*weight space*).
- Se a rede neural é utilizada para classificar um conjunto de padrões, para cada um destes padrões a rede irá gerar um determinado sinal de erro.

- Isso significa que cada conjunto de pesos e bias possui um valor associado de erro. Os valores de erro para todos os conjuntos possíveis de pesos e bias definem uma superfície no espaço de pesos denominada de *superfície de erro*.
- A questão que resulta então é qual o papel do algoritmo de treinamento. A forma que boa parte dos algoritmos supervisionados operam é através da minimização de uma função de custo baseada no erro entre as saídas da rede e as saídas desejadas.
- Três conclusões importantes devem ser salientadas:
  - A visualização do treinamento de RNAs via minimização do erro permite a interpretação do problema como um problema de otimização, que é geralmente não-linear e irrestrito. Isso permite a aplicação de diversas técnicas de otimização não-linear irrestrita para o treinamento de RNAs feedforward.
  - A superfície de erro possui, potencialmente, uma grande quantidade de mínimos locais, sugerindo que os algoritmos de treinamento estão sujeitos a ficarem presos em mínimos locais da superfície de erro.

- Outros métodos de busca como algoritmos evolutivos também podem ser empregados no treinamento de RNAs.



4. Auto-Organização e Redes Neurais Auto-Organizadas

- A essência da auto-organização está no surgimento de estrutura (formas restritas) e ordem (organização) sem que estas sejam impostas de fora do sistema. Isto implica que este fenômeno é interno ao sistema, resulta da interação de seus componentes e, em essência, não depende da natureza física destes componentes.
- A organização pode se dar no espaço, no tempo, ou em ambos.

- O que se busca são regras gerais para o crescimento e evolução de estruturas sistêmicas. Com isso, espera-se poder prever a organização futura que irá resultar de alterações promovidas junto aos componentes de um dado sistema, além de poder estender estes resultados a outros sistemas semelhantes.
- Em geral, os mecanismos estabelecidos pelos componentes de um dado sistema capaz de expressar auto-organização são: realimentação positiva, realimentação negativa e interação local.
- É importante buscar respostas (direta ou indiretamente) para as seguintes questões:
  - Uma vez caracterizadas as regras de interação que irão definir o comportamento individual de cada componente de um sistema, ao longo do tempo, e que supostamente permitem desenvolver e manter um processo auto-organizado, ocorrerá efetivamente algum tipo de auto-organização do comportamento espaço-temporal?

- Se a resposta à questão acima for afirmativa, qual será o padrão emergente, ou seja, o resultado final da auto-organização?
- Dado que existe, na natureza, a formação de padrões espaço-temporais não devidos a processos auto-organizados, como diferenciá-los de um processo auto-organizado?
- Qual é o nível de complexidade local que cada componente deve apresentar para permitir a emergência da complexidade global observada no sistema?
- O quanto do comportamento observado no agregado é devido ao efeito da ação individual de cada componente e o quanto é devido à dinâmica do meio?

4.1. Motivação para Treinamento Não-Supervisionado

- Dados rotulados são aqueles que assumem valores em um mesmo espaço vetorial multidimensional e que vêm acompanhados da classe a que cada um pertence (rótulo), podendo haver múltiplas classes, com variâncias e número de dados distintos ou não para cada classe.

- Dados não-rotulados são aqueles que assumem valores em um mesmo espaço vetorial multidimensional, e que não se conhece a priori a classe a que cada um pertence, embora cada um pertença a uma classe específica. O número de classes pode ser conhecido a priori ou não. A variância e o número de dados de cada classe pode diferir ou não.
- Como aprender a representar padrões de entrada de modo a refletir a estrutura estatística de toda a coleção de dados de entrada? Que aspectos da entrada devem ser reproduzidos na saída?
- Em contraposição ao treinamento supervisionado e ao treinamento por reforço, não há aqui nenhuma saída desejada explícita ou avaliação externa da saída produzida para cada dado de entrada.
- O treinamento não-supervisionado é predominante no cérebro humano. É sabido que as propriedades estruturais e fisiológicas das sinapses no córtex cerebral são influenciadas pelos padrões de atividade que ocorrem nos neurônios sensoriais.

No entanto, em essência, nenhuma informação prévia acerca do conteúdo ou significado do fenômeno sensorial está disponível.

- Sendo assim, a implementação de modelos computacionais para ajuste de pesos sinápticos via treinamento não-supervisionado deve recorrer apenas aos dados de entrada, tomados como amostras independentes de uma distribuição de probabilidade desconhecida.
- Duas abordagens têm sido propostas para aprendizado não-supervisionado:
  - técnicas para estimação de densidades de probabilidade, que produzem modelos estatísticos explícitos para descrever os fenômenos responsáveis pela produção dos dados de entrada. Ex: redes bayesianas.
  - técnicas de extração de regularidades estatísticas (ou então irregularidades) diretamente dos dados de entrada. Ex: redes de Kohonen.

4.2. Mapas Auto-Organizáveis de Kohonen

- Um mapa de Kohonen é um arranjo de neurônios, geralmente restrito a espaços de dimensão 1 ou 2, que procura estabelecer e preservar noções de vizinhança (preservação topológica).
- Se estes mapas apresentarem propriedades de auto-organização, então eles podem ser aplicados a problemas de clusterização e ordenação espacial de dados.
- Neste caso, vai existir um mapeamento do espaço original (em que os dados se encontram) para o espaço em que está definido o arranjo de neurônios.
- Como geralmente o arranjo de neurônios ocorre em espaços de dimensão reduzida (1 ou 2), vai existir uma redução de dimensionalidade sempre que o espaço original (em que os dados se encontram) apresentar uma dimensão mais elevada.
- Toda redução de dimensionalidade (relativa à dimensionalidade intrínseca dos dados) pode implicar na perda de informação (por exemplo, violação topológica). Sendo assim, este mapeamento deve ser tal que minimiza a perda de informação.

- A informação é uma medida da redução da incerteza, sobre um determinado estado de coisas. Neste sentido, a informação não deve ser confundida com o dado ou seu significado, e apresenta-se como função direta do grau de originalidade, imprevisibilidade ou valor-surpresa do dado (ou conjunto de dados).
- Espaços normados são aqueles que permitem o estabelecimento de propriedades topológicas entre seus elementos. Ex: medida de distância (fundamental para a definição do conceito de vizinhança), ordenamento.

4.2.1. Arranjo unidimensional

- Um mapa de Kohonen unidimensional é dado por uma seqüência ordenada de neurônios lineares, sendo que o número de pesos é igual ao número de entradas.
- Há uma relação de vizinhança entre os neurônios (no espaço unidimensional vinculado ao arranjo), mas há também uma relação entre os pesos dos neurônios no espaço de dimensão igual ao número de entradas. Para entender a funcionalidade dos mapas de Kohonen, é necessário considerar ambas as relações.

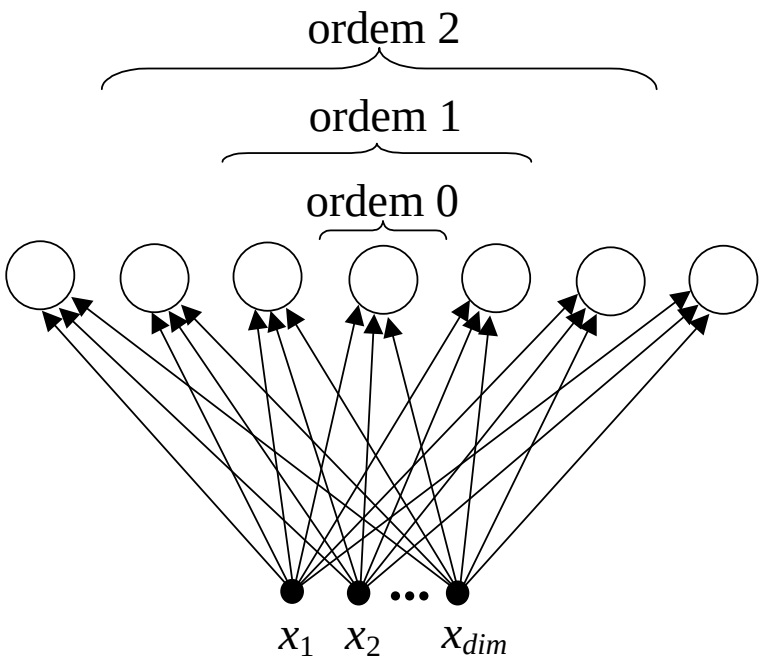


Figura 1 - Rede de Kohonen em arranjo unidimensional: ênfase na vizinhança

- $t_{il} = \begin{cases} +1 & \text{se } i = l \\ -\varepsilon & \text{se } i \neq l, \ \ \varepsilon < \dfrac{1}{n} \end{cases}$

- $\vec{y}(0) = \mathbf{W}\vec{x}$

- $y_i(k+1) = \max\left\{0, \sum_{l=1}^{n} t_{il}\, y_l(k)\right\}$

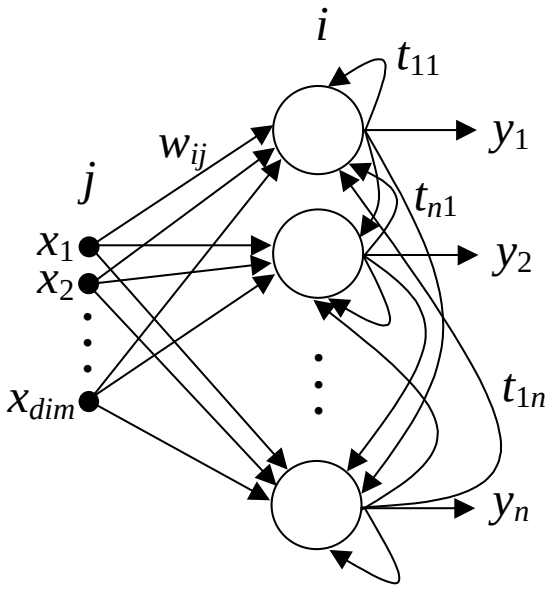


Figura 2 - Rede de Kohonen em arranjo unidimensional: ênfase nas conexões

Exemplo: Para *dim* = 2, considere *n* = 4, sendo que os vetores de pesos são dados na forma:

$$\mathbf{w}_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} -3 \\ 0 \end{bmatrix} \quad \text{e} \quad \mathbf{w}_4 = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

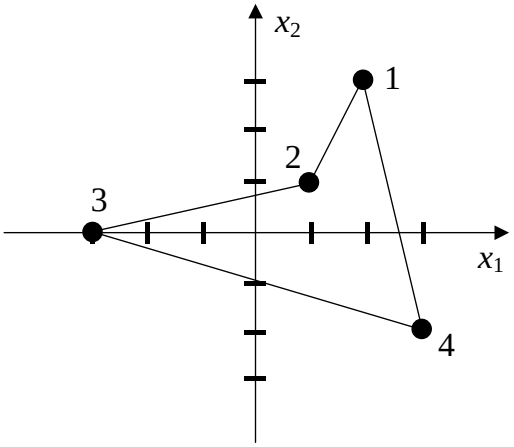


Figura 3 - Mapa de Kohonen em arranjo unidimensional (nesta caso, existe vizinhança entre primeiro e último neurônios)

4.2.2. Arranjo bidimensional

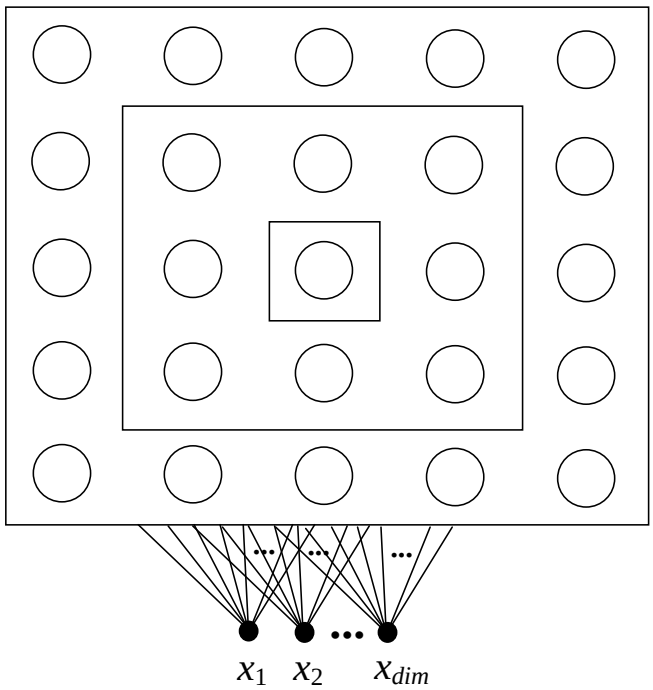
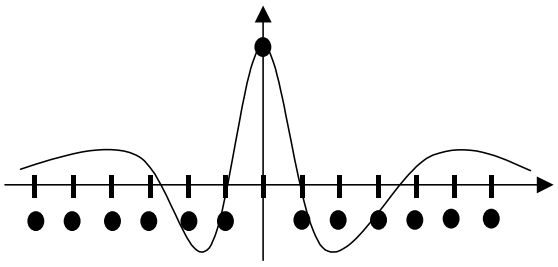


Figura 4 - Rede de Kohonen em arranjo bidimensional: ênfase na vizinhança

4.2.3. Fase de competição

- Três elementos básicos em uma lei de aprendizado competitivo:
  1. um conjunto de neurônios similares, exceto pelo valor de seus pesos;
  2. um limite imposto ao valor que os pesos podem atingir;
  3. um mecanismo de competição.
- A competição vai produzir um único neurônio ativo para cada entrada (winner-takes-all)
- Como implementar? $\rightarrow$ definindo uma regra de influência da ativação de um neurônio junto a todos os demais.



4.2.4. Fase de aprendizado não-supervisionado

- Se os pesos se mantiverem com norma unitária, então a definição do neurônio vencedor pode se dar de duas formas:

  1. produto escalar;

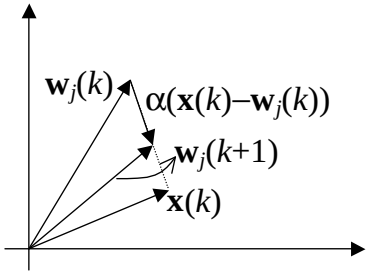
  2. norma euclidiana (é a única válida também no caso de pesos não-normalizados).

- Seja *j* o neurônio vencedor:

Alternativa 1

- Somente o neurônio *j* é ajustado na forma:

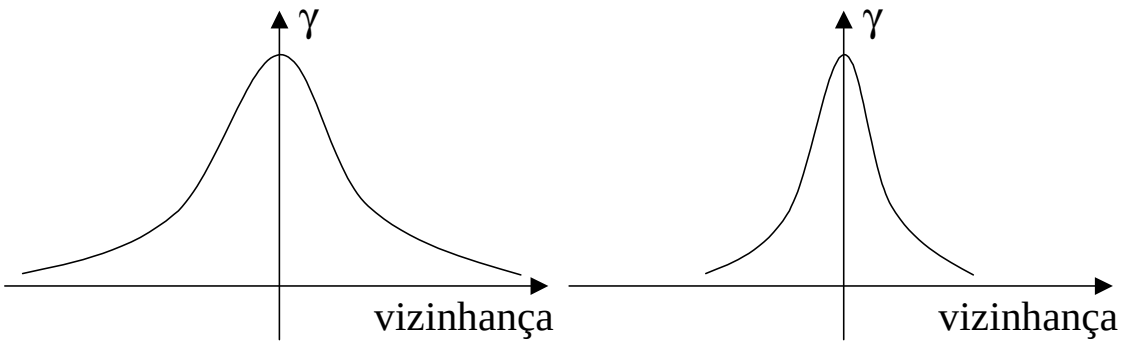
$$\mathbf{w}_j(k+1) = \mathbf{w}_j(k) + \alpha\left(\mathbf{x}(k) - \mathbf{w}_j(k)\right)$$



Alternativa 2

- caso existam múltiplos representantes para cada agrupamento de dados, então é interessante ajustar o neurônio vencedor e seus vizinhos mais próximos.

Implementação



- é importante que a influência de cada neurônio vencedor seja ampla no início do processo e sofra uma redução continuada com o decorrer das iterações.

4.2.5. Algoritmo de ajuste dos pesos

while <condição de parada> é falso,

ordene aleatoriamente os *N* padrões de entrada;

for i=1 até *N*,

$$j = \arg\min_j \left\| \mathbf{x}_i - \mathbf{w}_j \right\|$$

$\forall\, J \in \text{Viz}(j)$ **do**:

$$\mathbf{w}_J = \mathbf{w}_J + \alpha\big(dist(j, J)\big)\big(\mathbf{x}_i - \mathbf{w}_J\big);$$

end do

end for

atualize a taxa de aprendizado γ;

atualize a vizinhança;

avalie a condição de parada;

end while

4.2.6. Discriminação dos agrupamentos

- Dada a conformação final de neurônios (não rotulados), como realizar a clusterização, ou seja, definição de agrupamentos e atribuição do mesmo rótulo a todos os neurônios pertencentes a um dado agrupamento?
- Solução: matriz(vetor)-U (ULTSCH, 1993; COSTA, 1999)
- Aspecto a ser explorado: após o processo de auto-organização, dados de entrada com características semelhantes passam a promover reações semelhantes da rede neural.
- Assim, comparando-se as reações da rede neural treinada, é possível agrupar os dados pela análise do efeito produzido pela apresentação de cada um à rede.
- A matriz-U é uma ferramenta que permite realizar a discriminação dos agrupamentos, a partir de uma medida do grau de similaridade entre os pesos de neurônios adjacentes na rede. O perfil apresentado pelas distâncias relativas entre neurônios vizinhos representa uma forma de visualização de agrupamentos.

- Recebeu a denominação de matriz por ter sido proposta no caso de mapas bidimensionais, sendo que o grau de similaridade é plotado na terceira dimensão gerando uma superfície em relevo em 3D.
- Para o caso de mapas unidimensionais, tem-se o vetor-U.
- Topologicamente, as distâncias entre neurônios vizinhos refletem os agrupamentos, pois uma "depressão" ou um "vale" da superfície de relevo representa neurônios pertencentes a um mesmo agrupamento. Neurônios que têm uma distância grande em relação ao neurônio adjacente, a qual é representada por um pico da superfície de relevo, são neurônios discriminantes de agrupamentos.

4.2.7. Aplicação

- Como utilizar o mapa de Kohonen, após a fase de treinamento não-supervisionado e depois de ter as classes devidamente discriminadas, para classificação de padrões?
- Exemplo de aplicação: Agrupamento de dados (clusterização)

Caso 1: Ausência de agrupamentos

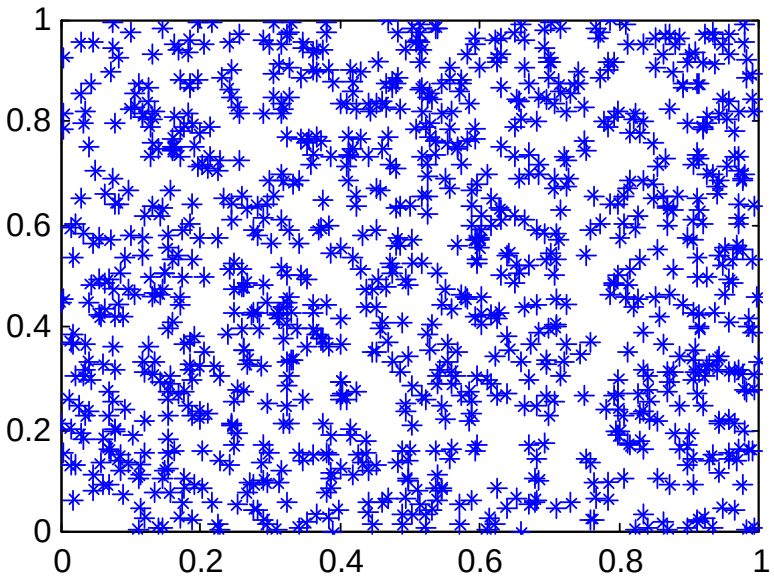


Figura 5 – Padrões de entrada com distribuição uniforme

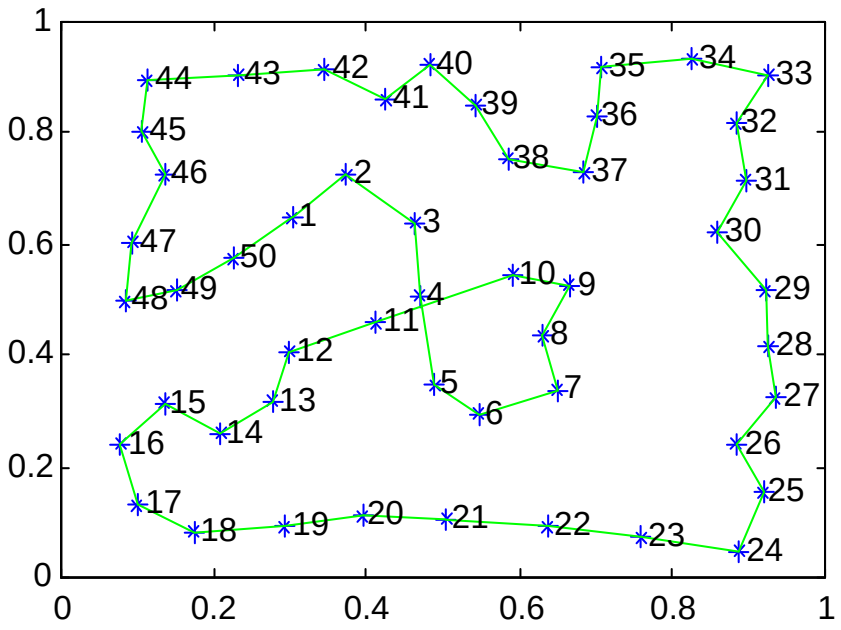


Figura 6 – Configuração final de 50 neurônios em um arranjo unidimensional

Caso 2: Presença de agrupamentos bem distintos

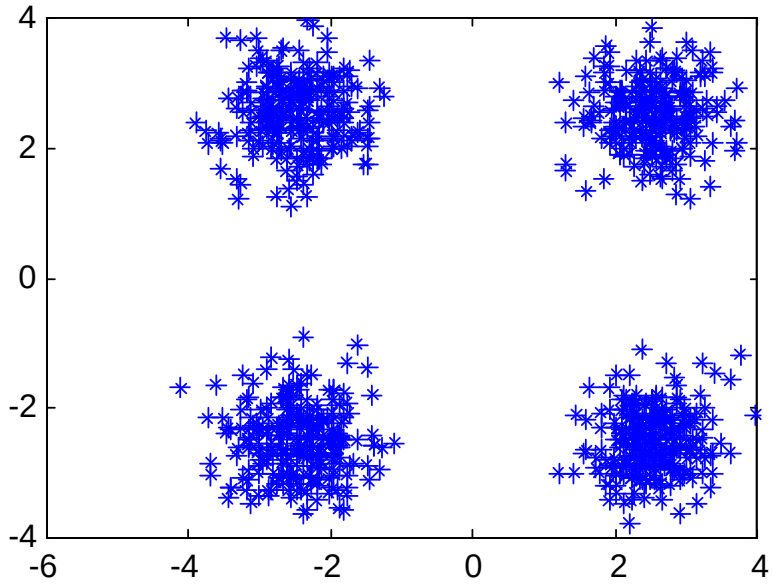


Figura 7 – Padrões de entrada com 4 agrupamentos

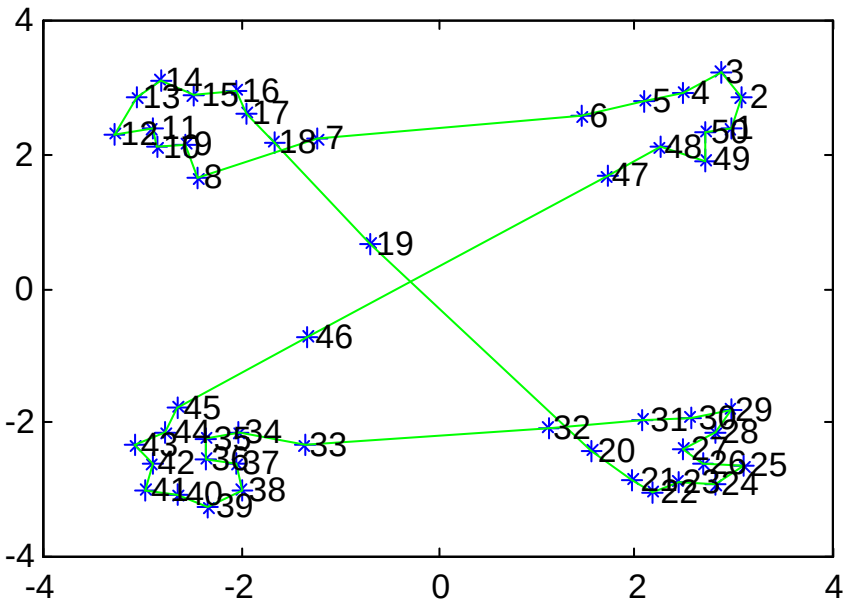


Figura 8 – Configuração final de 50 neurônios em um arranjo unidimensional.

4.2.8. Questões a serem melhor investigadas

- Sintonia de parâmetros
- Neurônios que não vencem nunca (devem ser podados para aumentar eficiência)
- Neurônios que vencem sempre
- Dimensão do arranjo para uma dada aplicação
- Número de neurônios, uma vez definido o arranjo
- Inicialização dos pesos
- Apresentação dos dados à rede (padrão-a-padrão ou em batelada?)
- Interpretação do mapa resultante (análise discriminante)
- Métodos construtivos e de poda
- Outras aplicações e múltiplos mapeamentos simultâneos
- Comparações com ferramentas similares