Redes Neurais Artificiais

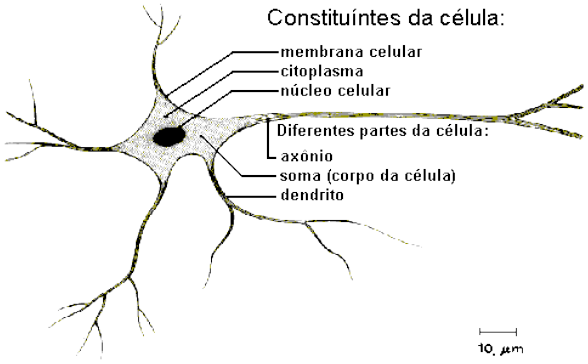
Anderson de Oliveira Marques,
Rodrigo Cesar Lira da Silva,
Sergio Ferreira Ribeiro

I. Introdução

O cérebro humano é considerado mais fascinante processador baseado em carbono existente, sendo composto por aproximadamente 10 bilhões neurônios. Todas as funções e movimentos do organismo estão relacionados ao funcionamento destas pequenas células. Os neurônios estão conectados uns aos outros através de sinapses, e juntos formam uma grande rede, chamada Rede Neural. As sinapses transmitem estímulos através de diferentes concentrações de Na+ (Sódio) e K+ (Potássio), e o resultado disto pode ser estendido por todo o corpo humano. Esta grande rede proporciona uma fabulosa capacidade de processamento e armazenamento de informação.

O sistema nervoso é formado por um conjunto extremamente complexo de neurônios. Nos neurônios a comunicação é realizada através de impulsos, quando um impulso é recebido, o neurônio o processa, e passado um limite de ação, dispara um segundo impulso que produz uma substância neurotrasmissora o qual flui do corpo celular para o axônio (que por sua vez pode ou não está conectado a um dendrito de outra célula). O neurônio que transmite o pulso pode controlar a frequência de pulsos aumentando ou diminuindo a polaridade na membrana pós sináptica. Eles tem um papel essencial na determinação do funcionamento, comportamento e do raciocínio do ser humano. Ao contrário das redes neurais artificiais, redes neurais natural não transmitem sinais negativos, sua ativação é medida pela frequência com que emite pulsos, frequência esta de pulsos contínuos e positivos. As redes naturais não são uniformes como as redes artificiais, e apresentam uniformidade apenas em alguns pontos do organismo. Seus pulsos não são síncronos ou assíncronos, devido ao fato de não serem contínuos, o que a difere de redes artificias. Os principais componentes dos neurônios são:

- Os dendritos, que tem por função receber os estímulos transmitidos pelos neurônios;
- O corpo de neurônio, também chamado de soma, que é responsável por coletar e combinar informações vindas de outros neurônios;
- E finalmente o axônio, que é constituído de uma fibra tubular que pode alcançar até alguns metros, e é responsável por transmitir os estímulos para outras células.



II. Breve histórico das Redes Neurais Artificiais

A primeira versão de um neurônio artificial foi proposto por McCullosh (neurofisiologista) e Pitts (matemático) [**?**] que apresentaram um estudo em 1943 sugerindo a construção de uma máquina inspirada

no cérebro humano. Em função destes trabalhos pioneiros credita-se a eles estabelecimento das bases da neuro computação.

Em seguida ao trabalho de McCulloch e Pitts surge a regra de aprendizagem proposta por Donald Hebb que se constitui na base de todas as regras de aprendizagem. Em seu famoso livro de 1949, The Organization of Behavior, o psicólogo Donald Hebb [**?**] procurou encontrar um mecanismo neural capaz de explicar como as informações podem ser armazenadas e recuperadas nos neurônios. A sua grande contribuição foi formular uma regra de aprendizagem enunciada da seguinte forma: "Quando um neurônio recebe um estímulo de outro neurônio, e se ambos estão altamente ativos, o peso entre estes deve ser fortalecido, caso contrário enfraquecido".

Entretanto, só em 1958 a primeira aplicação prática foi realizada por Frank Rosenblatt [**?**] que desenvolveu uma rede neural Perceptron que era capaz de realizar reconhecimento de padrões através de uma regra de aprendizagem o que gerou um grande interesse nas redes neurais. No ano de 1960, Widrow e Hoff [**?**] apresentaram uma regra de aprendizagem para uma extensão de Perceptron chamada de ADALINE (ADAptive LInear NEuron). Esta regra baseada no método dos mínimos quadrados ficou conhecido como regra delta e é utilizada até os dias atuais.

Infelizmente, com a publicação de um trabalho de Minsky e Papert [**?**] em 1969 verificou-se que o Perceptron era capaz de distinguir apenas padrões linearmente separáveis o que levou a um desinteresse nas redes neurais e, consequentemente, uma redução nas verbas de pesquisa.

Apesar destas limitações, alguns pesquisadore continuaram interessados nas redes neurais e em 1970 destaca-se a rede neural auto-organizável desenvolvida por Teuvo Kohonen K̃oh82K̃oh88 (Rede de Kohonen).

O ressurgimento das redes neurais é atribuído ai trabalho do John Hopfield [**?**] publicado em 1982, sobre as propiedades associativas das redes neurais (rede de Hopfield) e ao desenvolvimento do algoritmo Backpropagation que teve seus passos iniciais dados por Paul Werbos em 1970 [**?**] na sua tese de doutorado e posteriormente popularizado através da publicação feita por Rumelhart e McClelland em 1986 [**?**].

III. Perceptron

O modelo mais simples de Rede Neural, no qual várias unidades de processamento estão conectadas unicamente a uma unidade de saída, através dos pesos sinápticos é conhecida como *Perceptron* [**?**].

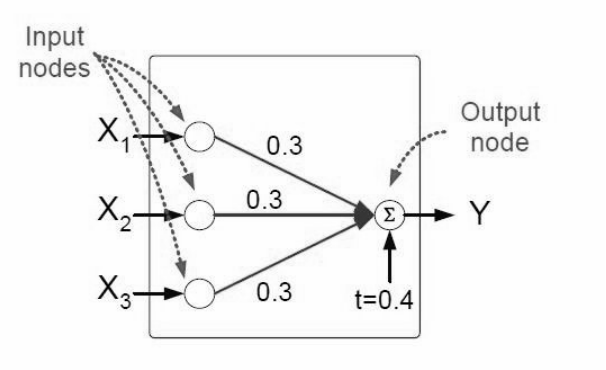
Dentre as importantes características de similaridade destas redes com o cérebro humano, está a capacidade de aprender. Portanto, essas redes neurais artificiais possuem alguma forma de regra de aprendizagem que são responsáveis pela modificação dos pesos sinápticos, em função dos exemplos de entrada que são repetidamente apresentados. Assim pode-se dizer que as Redes Neurais Artificiais aprendem por exemplos.

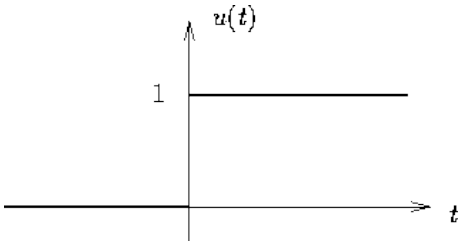
- Aprendizado supervisionado - o aprendizado da rede é feito como o conhecimento prévio do resultado desejado, ou seja, são fornecidos, para a rede, o conjunto de exemplos de entrada e as respectivas respostas. As redes *Perceptron*, as redes multicamadas (MLP) e Adaline utilizam esse tipo de aprendizado [**?**].
- Aprendizado não supervisionado - nesse tipo de aprendizado, a rede aprede com os próprios dados de entrada (somente os exemplos de entrada são mostrados a rede), ou seja, este algoritmo não requer conhecimento de saídas. Um exemplo que utiliza esse aprendizado é a rede de Kohonen [**?**].

A Perceptron apresentado por Frank Rosenblatt em 1958 é uma rede neural simples:

A Perceptron é composta de uma camada de entrada e uma camada de saída estando cada entrada ligada à saída através de uma conexão representada por uma ponderação chamada de peso sináptico em analogia como o neurônio biológico.

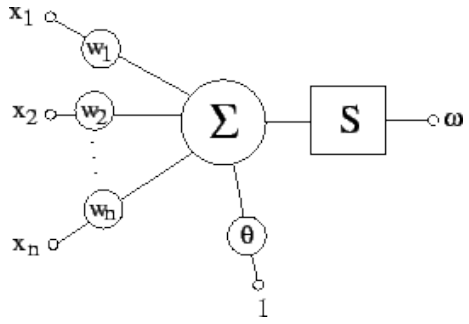
Como a Perceptron utiliza a formulação do neurônio proposta por McCullock e Pitts, a saída da Perceptron obedece a Lei do Tudo ou Nada, logo, a saída da *Perceptron* é estabelecida por uma função degrau que define quando o neurônio estará ativo ou em repouso.





$$u(t) = \begin{cases} 1 & \text{se} \quad x > 0 \\ 0 & \text{se} \quad x = 0 \\ -1 & \text{se} \quad x < 0 \end{cases}$$

Portanto, a *Perceptron* pode ser definido como uma rede neural que é composta por uma camada de entrada onde estão as informações que alimentam a rede (as variáveis de entrada), uma função soma (entrada líquida) que pondera estas entradas através de pesos (chamados pesos sinápticos) e uma função saída (função degrau), também chamada de função de ativação que fornece sinal emitido pelo neurônio da camada de saída. Para entender os termos utilizados para caracterizar uma *Perceptron*, faz-se uso sem perda de generalidade da figura abaixo que apresenta uma *Perceptron* com 3 neurônios na camada de entrada e um neurônio na camada de saída. As entradas são representadas por x_1, x_2 e x_n e a saída por ω e os pesos sinápticos por w_1, w_2, w_n e θ.



O peso θ representa o limiar (bias) para uma entrada com valor +1. A função de saída S, é a função degrau apresentada anteriormente. A entrada líquida para esta rede net_1 é dada por:

$$net_1 = w_1x_1 + w_2x_2 + ... + w_nx_n + 1 * \theta \tag{1}$$

A grande importância do trabalho de Frank Rosenblatt [**?**] sobre o conhecimento de padrões consistiu em montrar que uma Perceptron é capaz de aprender a partir de um conjunto de exemplos. Ele estabeleceu

uma regra de aprendizagem para a rede Perceptron que é dada por:

$$\Delta w_i j = \alpha(d_i - y_i)x_j, \tag{2}$$

$$\Delta w_i j = w_i j(novo) - w_i j(antigo), \tag{3}$$

$$w_i j = w_i j(antigo) + \alpha(d_i - y_i)x_j, \tag{4}$$

onde x_j = sinal de entrada; d_i = valor alvo (desejado); $w_i j$ são os pesos sinápticos; y_i = f(net) é o valor calculado para a saída; α é a taxa de aprendizagem (valor entre 0 e 1) e o valor (d_i - y_i) é o erro na saída representado por e_i

Treinar uma rede neural consiste em ajuste os pesos através de uma regra de aprendizagem até que está forneça respostas satisfatórias ao problema analisado. Para realizar o treinamento do *Perceptron* faz-se necessário um conjunto de exemplos com os valores de entradas e suas respectivas saídas (valores desejados). Logo, o aprendizado do Perceptron é dito supervisionado também chamado de aprendizado com o professor, pois são apresentados à rede os exemplos de entrada e respectivas saídas. O treinamento supervisionado é aquele que utiliza um conjunto de exemplos, de tal maneira que para cada exemplo de entrada é fornecido um exemplo de saída desejado. Este processo é continuamente repetido para todo conjunto de exemplos, até que o erro esteja dentro de um valor considerado satisfatório.

Durante o treinamento, ao finalizar a apresentação de todos os exemplos do conjunto diz-se que se conclui um ciclo. Portanto, um ciclo (ou época) consiste na apresentação completa de um conjunto de exemplos. O treinamento do Perceptron pode ser resumido pelos seguintes passos:

1) Inicialização dos pesos - os pesos são gerados aleatoriamente com valores pequenos;
2) Um dado exemplo é apresentado à rede (uma entrada cuja saída é conhecida);
3) Adota-se uma taxa de aprendizagem, calcula-se a entrada líquida neti e o sinal de saída y_i = f(net_i);
4) Calcula-se o erro (d_i - y_i);
5) Faz-se o ajustamento dos pesos;
6) Testa-se o atendimento ao critério de parada (erro satisfatório e/ou número máximo de ciclos). Se o critério de parada não for satisfatório retorna-se ao passo 2. Os passos de 2 a 6 são repetidos para todos os exemplos e por vários ciclos até encontrar uma solução satisfatória.

IV. Adaline

Na mesma época em que Rosenblatt propôs o Perceptron, Widrow e Hoff propuseram um modelo matemático de rede, denominado de Adaline (Adaptive Linear Element), também conhecido como (regra delta), similar ao Perceptron, exceto no seu algoritmo de treinamento utiliza conceitos de mínimos quadrados. As funções de ativação mais utilizadas são a linear (1) , sigmoidal logística (2) e tangente hiperbólica (3) ao invés de função degrau como na Perceptron.

$$y = f(u) = u, \tag{5}$$

$$y = f(net) = \frac{1}{1 + e^{-net}}, \tag{6}$$

$$y = f(net) = \frac{e^{net} - e^{-net}}{e^{net} - e^{-net}}, \tag{7}$$

$$u = \sum_{i=0}^{u} w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n \tag{8}$$

O objetivo do algoritmo de treinamento é minimizar o erro quadrático médio entre a saída e a saída desejada. Assim, a ideia ajuste de pesos modelo Adaline se baseia em minimizar a função erro quadrático para cada padrão de treinamento.

$$E_i = \frac{1}{2}(d_i - y_i)^2 \tag{9}$$

onde d_i é o valor desejado que o neurônio deve gerar como saída para a entrada x_i e y_i é a saida obtida para a entrada x_i , (i = 0,1,2, ... , n).

A regra de aprendizagem desenvolvida por Widrow-Hoff, baseada no método dos mínimos quadrados é conhecida como regra delta e é utilizada para realizar o treinamento da Adaline. Esta regra determina o conjunto de pesos ótimos pela utilização de uma técnica de otimização conhecida como métodos do gradiente e é utilizada para realizar o treinamento da Adaline. Esta regra determina o conjunto de pesos ótimos pela utilização de uma técnica de otimização conhecida como métodos do gradiente descendente, que usa como função custo a ser minimizada o erro médio quadrático. A regra de aprendizagem da rede Adaline é dada por:

$$\Delta w_i j = \alpha(d_i - y_i)x_j f'(net_i) \tag{10}$$

Assim o ajuste a ser aplicado aos pesos é:

$$w_i j(novo) = w_i j(antigo) + \alpha(d_i - y_i)x_j f'(net_i) \tag{11}$$

α a taxa de aprendizagem, $w_i j$ são os pesos, d_i é a resposta desejada, y_i é a saída calculada, x_j é a variável de entrada e f'(net_i) é a derivada da função de saída. A regra delta como se pode observar na equação ANTERIOR possui um termo a mais do que a regra de teinamento do *Perceptron* este termo é a derivada da função de saída com relação à entrada líquida.

Quando a função de saída também conhecida como função de ativação na unidade de saída é a linear dada por:

$$y_i = f(net_i) = net_i \tag{12}$$

A derivada $f'(net_i)$ é:

$$f'(net) = \frac{\partial f(net_i)}{\partial (net)} = \frac{\partial net_i}{\partial net_i} = 1 \tag{13}$$

Logo o ajuste a ser aplicado aos pesos é:

$$w_i j(novo) = w_i j(antigo) + \alpha(d_i - y_i)x_j * 1 \tag{14}$$

$$w_i j(novo) = w_i j(antigo) + \alpha(d_i - y_i)x_j \tag{15}$$

Portanto, a regra de aprendizagem da rede Adaline é igual a do Perceptron quando a função de saída é linear.

Quando a função de saída também conhecida como função de ativação na unidade de saída é a função sigmóide logística dada pela equação abaixo:

$$y_i = f(net_i) = \frac{1}{1 + e^{net_i}} \tag{16}$$

$$f'(net) = y_i(1 - y_i) \tag{17}$$

$$\Delta w_i j = \alpha x_j (d_i - y_i) y_i (1 - y_i) \tag{18}$$

Portanto, neste caso das redes Adaline que não possuem camada escondida, o ajuste dos pesos é feito pela equação ANTERIOR de forma bastante simples, haja vista que todos os termos podem ser facilmente determinados.

A. Regra Delta

Matematicamente, os pesos são corrigidos da seguinte forma:

Seja d_i o valor desejado para o neurônio i na camada de saída e y_i (y_i = f(net)) o valor calculado para o neurônio i na camada de saída.

Considere a função objetivo como sendo erro médio quadrático. Define-se a função erro para o exemplo n como:

$$e_i^2(n) = \frac{1}{2}(d_i - y_i)^2 \tag{19}$$

Pelo método do gradiente descendente, o ajuste nos pesos deve ser proporcional ao sentido contrário ao gradiente da função erro com relação aos pesos. Portanto, define-se este ajuste como sendo:

$$\Delta w_i j = -\alpha \frac{\partial e_i^2(n)}{\partial w_i j} \tag{20}$$

O cálculo de 20 é feito pelo uso da regra da cadeia, ou seja:

$$\frac{\partial e_i^2(n)}{\partial w_i j} = \frac{\partial e_i^2(n)}{\partial y_i} \frac{\partial y_i}{\partial net_i} \frac{\partial net_i}{\partial w_i j} \tag{21}$$

sendo: net_i = $\sum_{j=0}^{N} w_{ij} x_j$

Por outro lado, a derivada da entrada net_i com relação aos pesos é:

$$\frac{\partial net_i}{\partial w_i j} = x_j \tag{22}$$

$$\frac{\partial y_i}{\partial net_i} = f'(net_i) \tag{23}$$

$$\frac{\partial e_i^2}{\partial y_i} = 2\frac{1}{2}(d_i - y_i)(-1) = -(d_i - y_i) \tag{24}$$

Assim o ajuste $\Delta w_i j$ a ser aplicado aos pesos é:

$$\Delta w_i j = \alpha (d_i - y_i) x_i f'(net_i), \tag{25}$$

$$w_{ij}(novo) = w_{ij}(antigo) + \alpha (d_i - y_i) x_j f'(net_i), \tag{26}$$

sendo: α a taxa de aprendizagem.

Quando o conjunto de treinamento é grande a atualização padrão tende a ser mais rápida uma vez que não necessita do armazenamento dos ajustes para cada exemplo. O método padrão também em função da sua natureza estocástica é menos suscetível ao problema de mínimos locais. A utilização de um dos métodos para treinamento é função do problema que se deseja resolver.

Quando aos critérios de parada durante o treinamento estes serão detalhados no capítulo que trata das redes MLP onde se apresentam aplicações passo a passo.

Quando a rede está satisfatoriamente treinada, ela pode ser usada com outro conjunto de dados, graças a sua capacidade de generalização. Portanto, o uso de rede neural, depois de treinada, passa a ser bastante simples, sendo necessário apenas guardar os valores dos pesos sinápticos ótimos.

Este algoritmo garante a minimização do erro ao longo do tempo, através do ajuste dos pesos, ou seja, sua convergência garante que a adaptação dos pesos seja realizada num número finito de iterações. A prova de convergência pode ser encontrada em Beale e Jackson [**?**]. Durante o treinamento o ajuste dos pesos pode ser realizado de duas maneiras:

- Atualização 'on line' (padrão): A atualização dos pesos acontece a cada apresentação à rede de um exemplo do conjunto de treinamento. A atualização dos pesos baseia-se somente no erro do exemplo apresentando naquele momento. Este processo é repetido até que todos os exemplos sejam apresentados à rede. Portanto, se existem N exemplos, em cada ciclo ocorrem N atualizações. Para que este método apresente estabilidade faz-se necessário a utilização de taxas de aprendizagem.
- Atualização 'Batch'(lote): Atualização dos pesos só é feita após a apresentação de todos os exemplos de treinamento que constituem o ciclo. Todos os exemplos do conjunto de treinamento são apresentados à rede, o valor do ajuste Δw_{ij} é calculado para cada exemplo e armazenado, e a partir da soma acumulada de todos os Δw_{ij} para cada exemplo fazem-se às correções dos pesos. Esta atualização que também é conhecida como 'off-line' é mais estável do que o modo padrão por trabalhar com o gradiente médio

O erro quadrático para um exemplo n qualquer é calculado pela expressão:

$$e_i^2 = (d_i - y_i)^2 \tag{27}$$

O cálculo do erro médio quadrático (EMQ) para N exemplos da camada de entrada e um neurônio na camada de saída é dado por:

$$EMQ = \frac{1}{N}\sum_{k=0}^{N} e_i^2(N) \tag{28}$$

No caso de ter-se N exemplos e s neurônios na camada de saída a expressão geral fica:

$$EMQ = \frac{1}{N} * \frac{1}{s}\sum_{n=0}^{N}\sum_{i=0}^{s}(d_i - y_i)^2 \tag{29}$$

V. MLP

Quando Redes Neurais Artificiais de uma só camada são utilizadas os padrões de treinamento apresentados à entrada são mapeados diretamente em um conjunto de padrões de saída da rede, ou seja não é possível a formação de uma representação interna. Neste caso, a codificação proveniente do mundo exterior deve ser suficiente para implementar esse mapeamento.

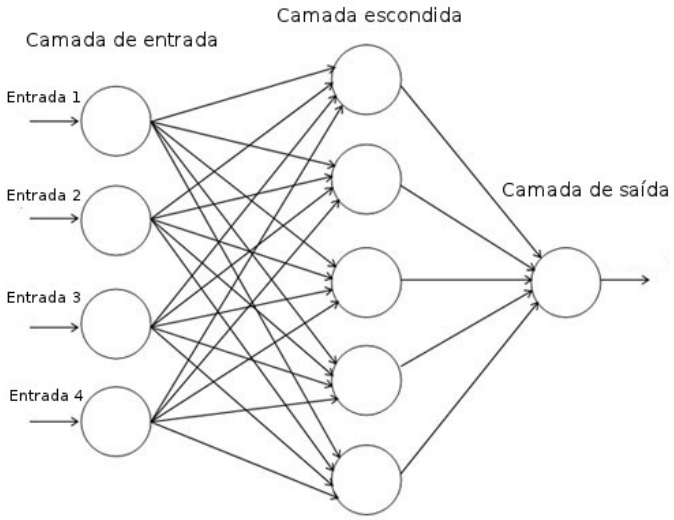
Tal restrição implica que padrões de entrada similares resultem em padrões de saída similares, o que leva o sistema à incapacidade de aprender importantes mapeamentos. Como resultado, padrões de entrada com estruturas similares, fornecidos do mundo externo, que levem a saídas diferentes não são possíveis de serem mapeados por redes sem representações internas, isto é, sem camadas intermediárias. Um exemplo clássico deste caso é a função ou-exclusivo (XOR).

Minsky e Papert analisaram matematicamente o Perceptron e demostraram que redes de uma camada não são capazes de solucionar problemas que não sejam linearmente separáveis. Como não acreditavam na possibilidade de se construir um método de treinamento para redes com mais de uma camada, eles concluíram que as redes neurais seriam sempre suscetíveis a essa limitação.

Contudo, o desenvolvimento do algoritmo de treinamento backpropagation, por Rumelhart, Hinton e Williams em 1986, precedido por propostas semelhantes ocorridas nos anos 70 e 80, mostrou que é possível treinar eficientemente redes com camadas intermediárias, resultando no modelo de Redes Neurais

Artificiais mais utilizado atualmente, as redes Perceptron Multi-Camadas (MLP), treinadas com o algoritmo backpropagation.

Nessas redes, cada camada tem uma função específica. A camada de saída recebe os estímulos da camada intermediária e constrói o padrão que será a resposta. As camadas intermediárias funcionam como extratoras de características, seus pesos são uma codificação de características apresentadas nos padrões de entrada e permitem que a rede crie sua própria representação, mais rica e complexa, do problema.



Se existirem as conexões certas entre as unidades de entrada e um conjunto suficientemente grande de unidades intermediárias, pode-se sempre encontrar a representação que irá produzir o mapeamento correto da entrada para a saída através das unidades intermediária.

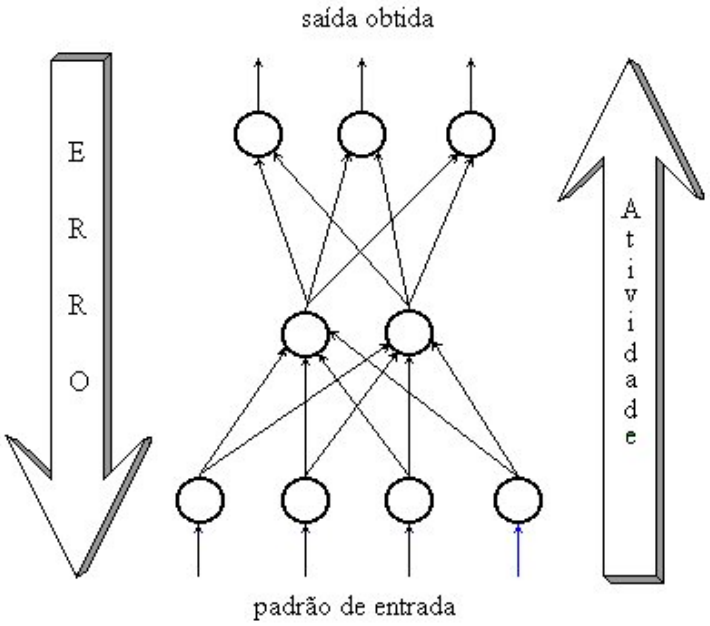
Como provou Cybenko, a partir de extensões do Teorema de Kolmogoroff, são necessárias no máximo duas camadas intermediárias, com um número suficiente de unidades por camada, para se produzir quaisquer mapeamentos. Também foi provado que apenas uma camada intermediária é suficiente para aproximar qualquer função contínua.

A. Backpropagation

Durante o treinamento com o algoritmo backpropagation, a rede opera em uma sequência de dois passos. Primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados conforme o erro é retropropagado.

As redes que utilizam backpropagation trabalham com uma variação da regra delta, apropriada para redes multi-camadas: a regra delta generalizada. A regra delta padrão essencialmente implementa um gradiente descendente no quadrado da soma do erro para funções de ativação lineares. Redes sem camadas intermediárias, podem resolver problemas onde a superfície de erro tem a forma de um parabolóide com apenas um mínimo. Entretanto, a superfície do erro pode não ser tão simples, como a ilustrada na figura abaixo, e suas derivadas mais difíceis de serem calculadas. Nestes casos devem ser utilizadas redes com camadas intermediárias. Ainda assim, as redes ficam sujeitas aos problemas de procedimentos ”hill-climbing”, ou seja, ao problema de mínimos locais.

A regra delta generalizada funciona quando são utilizadas na rede unidades com uma função de ativação semi-linear, que é uma função diferenciável e não decrescente. Note que a função threshold não se enquadra nesse requisito. Uma função de ativação amplamente utilizada, nestes casos, é a função sigmoid.



A taxa de aprendizado é uma constante de proporcionalidade no intervalo [0,1], pois este procedimento de aprendizado requer apenas que a mudança no peso seja proporcional à neta.

Entretanto, o verdadeiro gradiente descendente requer que sejam tomados passos infinitesimais. Assim quanto maior for essa constante, maior será a mudança nos pesos, aumentando a velocidade do aprendizado, o que pode levar à uma oscilação do modelo na superfície de erro. O ideal seria utilizar a maior taxa de aprendizado possível que não levasse à uma oscilação, resultando em um aprendizado mais rápido.

O treinamento das redes MLP com backpropagation pode demandar muitos passos no conjunto de treinamento, resultando um tempo de treinamento considerávelmente longo. Se for encontrado um mínimo local, o erro para o conjunto de treinamento pára de diminuir e estaciona em um valor maior que o aceitável. Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação é modificar a regra delta generalizada para incluir o termo momentum, uma constante que determina o efeito das mudanças passadas dos pesos na direção atual do movimento no espaço de pesos.

IDesta forma, o termo momentum leva em consideração o efeito de mudanças anteriores de pesos na direção do movimento atual no espaço de pesos. O termo momentum torna-se útil em espaços de erro que contenham longas gargantas, com curvas acentuadas ou vales com descidas suaves, como o apresentado na figura acima.

B. Treinamento da rede MLP

O treinamento supervisionado da rede MLP utilizando backpropagation consiste em dois passos. No primeiro, um padrão é apresentado às unidades da camada de entrada e, a partir desta camada as unidades calculam sua resposta que é produzida na camada de saída, o erro é calculado e o no segundo passo, este é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados utilizando a regra delta generalizada. Com isso o erro vai sendo progressivamente diminuído

C. Utilização

Depois que a rede estiver treinada e o erro estiver em um nível satisfatório, ela poderá ser utilizada como uma ferramenta para classificação de novos dados. Para isto, a rede deverá ser utilizada apenas no modo progressivo (forward). Ou seja, novas entradas são apresentadas à camada de entrada, são processadas nas camadas intermediárias e os resultados são apresentados na camada de saída, como no treinamento, mas sem a retropropagação do erro. A saída apresentada é o modelo dos dados, na interpretação da rede.

VI. Conclusão