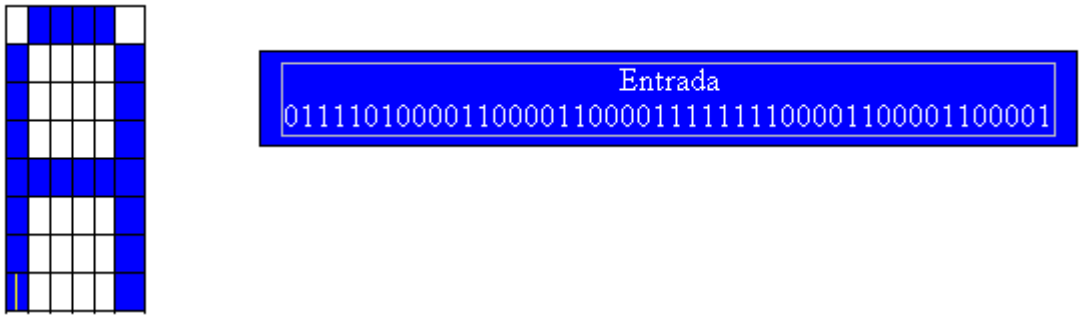
Algoritmo BackPropagation
Disciplina: Inteligência Artificial 2
Profa.Carine Webber
Data: 23/10/2019

Projeto de Redes Neurais

Objetivo do trabalho: Implementar uma Rede Neural para reconhecimento de caracteres da Tabela ASCII. Pode-se usar qualquer linguagem de programação. Implementações utilizando pacotes não serão aceitas. Você deve construir dois conjuntos de dados: um conjunto de dados para treinamento e outro para testes. Após o treinamento, deverá aplicar métodos de avaliação dos resultados e indicar a taxa de acerto da rede implementada.

Conjunto de dados de treinamento

Construa um *dataset* contendo os dígitos e caracteres maiúsculos da Tabela ASCII (48 ao 57, 65 ao 90). Utilize uma representação binária para a entrada e para a saída. Codifique as entradas e saídas conforme o método que você julgar o mais apropriado. Para a entrada, utilize a codificação binária de 48 bits, conforme o exemplo a seguir:



Conjunto de dados de teste

Construia um *dataset* contendo uma sequencia aleatória de 100 caracteres ASCII. Destes 100, 34% devem vir do conjunto de treinamento; 20% deve conter tipo mínimo de ruído (2 bits de diferença em relação ao original), outros 20% devem conter tipo médio de ruído (6 bits de diferença), outros 20% devem conter tipo avançado de ruído (12 bits de diferença) e 6% não devem fazer do conjunto original de treinamento.

Você deve iniciar o trabalho construindo as bases de treinamento e teste. Projete a rede neural (neurônios da camada de entrada, intermediária e de saída) e realize o treinamento.

Ao término de cada execução completa (treinamento e teste), o programa deve produzir um relatório com informações sobre as etapas de treinamento e teste, incluindo a configuração da rede neural, quantas iterações foram necessárias para a aprendizagem dos caracteres, quantos caracteres formam corretamente aprendidos, quantos e quais deles foram incorretamente reconhecidos, apresentando o resultado esperado e o resultado obtido.

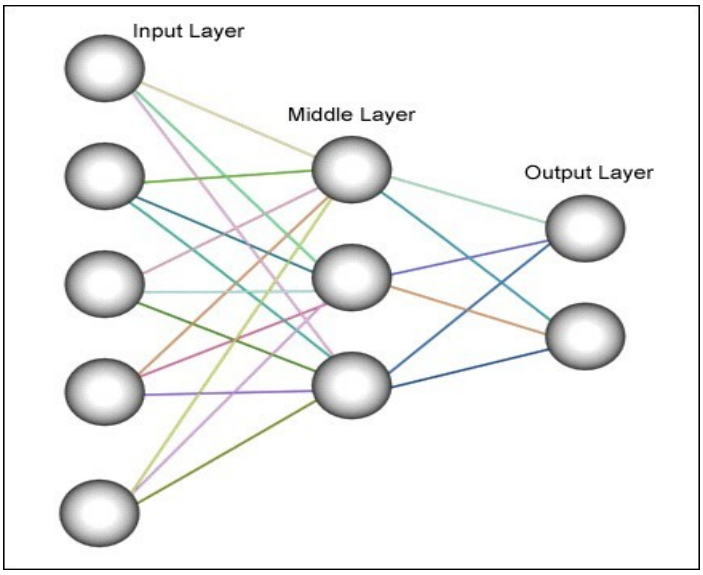
O programa deve permitir que os pesos aprendidos sejam salvos para avaliação futura.

Este trabalho será iniciado na aula do dia 23/10 e a implementação final deve ser entregue até dia 20/11. Juntamente com o código fonte cada dupla deverá apresentar um pequeno relatório descrevendo a configuração da Rede Neural e o seu desempenho no reconhecimento de cada caractere. A avaliação estará baseada em critérios de codificação do problema e desempenho da rede neural. A rede que obteve o melhor resultado deve ser guardada em memória.

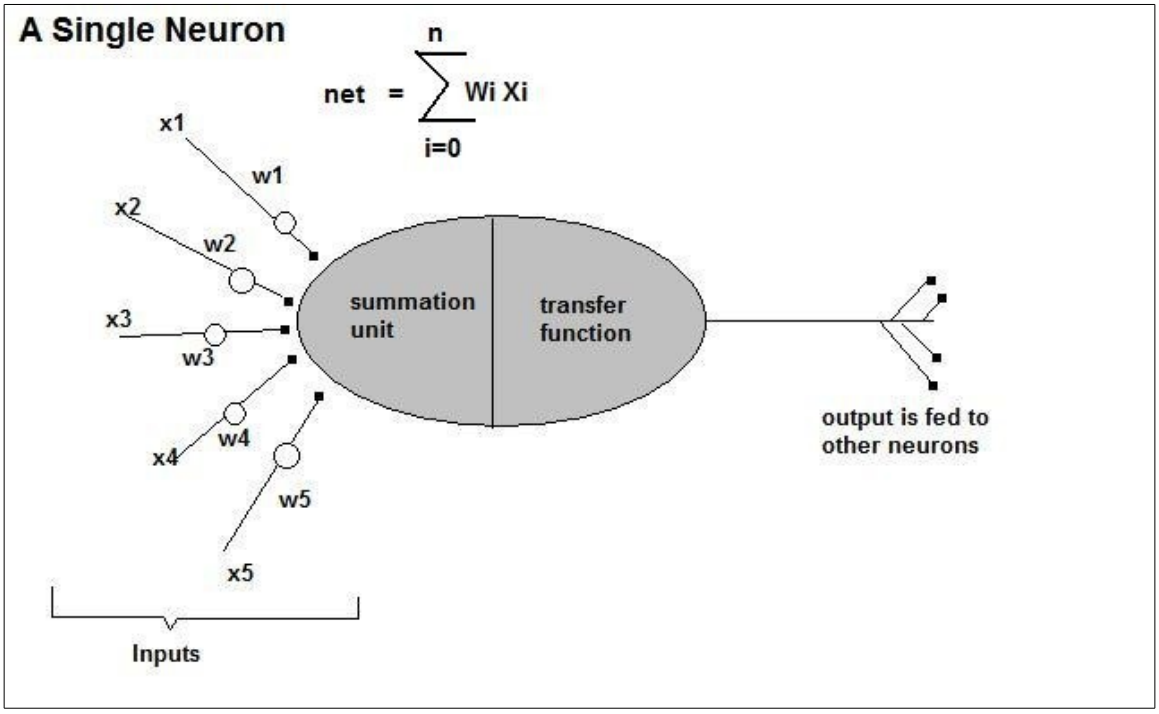
Bom trabalho!

Algoritmo BackPropagation -

Exemplo de uma Rede Neural Multicamadas



Exemplo de um neurônio artificial



Neurônios

O processamento de um neurônio passa por duas fases

1) Somatório das entradas multiplicadas pelos pesos (*summation unit*)
2) Geração da saída do neurônio (*transfer function*)

1) Somatório das entradas multiplicadas pelos pesos (*summation unit*)

Veja na figura do neurônio as entradas (x1, x2, x3, x4, x5) e os pesos (w1, w2, w3, w4, w5). As entradas do neurônio e os pesos de cada conexão devem estar armazenadas em vetores. O somatório é calculado da seguinte forma:

```
somatorio=0
for i=0 to neuronio.entradas.count-1
   somatorio=somatorio + x(i) * w(i)
next
```

2) Geração da saída do neurônio (*transfer function*)

A função de transferência (ou função de ativação) é uma função simples que usa o valor do somatório para gerar a saída do neurônio. Esta saída é então propagada para os neurônios da próxima camada. Tem-se vários tipos de função de transferência:

Função rígida

```
if (somatorio<0.5)
     saida = 0
else
     saida = 1
```

Função Sigmoidal

Gera um valor entre 0 e 1.

```
saida = 1 / (1 + Exp(-somatorio))
```

Treinamento da Rede Neural

Treinamento é o processo a partir do qual os pesos das conexões entre neurônios são ajustados a fim de que a rede produza o resultado (saída) esperado para todas as entradas fornecidas.

Passo 1 – As entradas

Inicialmente as entradas devem ser fornecidas para a Rede Neural. Na primeira camada, a saída dos neurônio será a sua própria entrada.

```
i = 0
For Each neuronio In CamadaEntrada
    neuronio.Saida(i) = Entrada(i)
    i = i + 1
Next
```

Passo 2 – Saída da rede

Calcule a saída da rede:

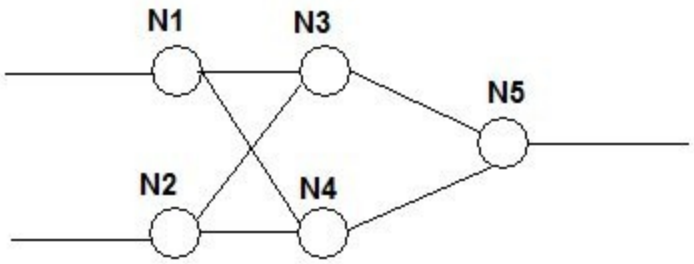
```
 //Calcule a saídas dos neurônios das camadas escondidas
 For cada camada in CamadasEscondidas
    For Each neuronio In camada.Neuronios
        neuronio.UpdateSaida()
    Next
 Next

 // Calcule a saídas dos neurônios da camada de saída
 For cada neuronio In CamadaSaida.Neuronios
        neuronio.UpdateSaida()
 Next
```

A função **UpdateSaida** funciona como o **somatorio** apresentado anteriormente.

```
For cada NeuronioEntrada conectado a EsteNeuronio
       somatorio = somatorio + (Peso (w) Associado com o
NeuronioEntrada* Saida do NeuronioEntrada)
Next
```

Veja um exemplo. Suponha seguinte a rede neural:



somatorio de N3 = (N1.Saida * Peso da conexao de N1 para N3) + (N2.Saida * Peso da conexao de N2 para N3)

somatorio de N4 = (N1.Saida * Peso da conexao de N1 para N4) + (N2.Saida * Peso da conexao de N2 para N4)

Para gerar a saída do neurônio utilize a função sigmoidal.

Função de Transferência ou Ativação

```
Saida do Neuronio = 1/(1+Exp(-somatorio))
```

Passo 3 – Cálculo do Erro

Erro ou Delta é a diferença entre a saída esperada e a saída obtida. Deve ser calculado para os neurônios das camadas intermediárias e de saída.

Primeiro calcula-se o erro da camada de saída. Este valor é utilizado para calcular o erro das camadas intermediárias (retro-propagação do erro).

A equação geral do cálculo do erro delta de um neurônio é:

```
Neuronio.Erro = Neuronio.Saida * (1 - Neuronio.Saida) * FatorErro
```

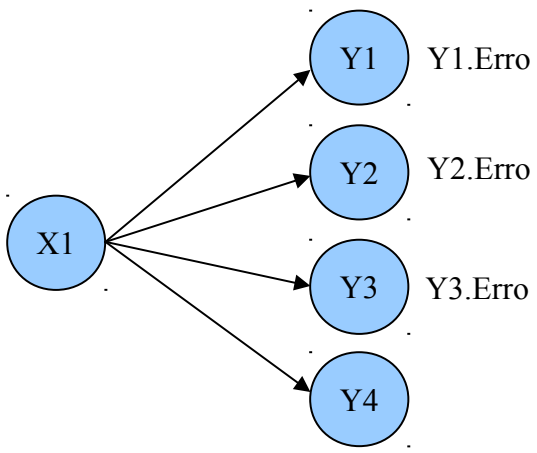
Para um **neurônio da camada de saída**, calcula-se o fator de erro da seguinte maneira:

```
FatorErro de neuronio na camada de saída=SaídaEsperada-SaídaAtualNeuronio
```

Para um **neurônio da camada intermediária**, o fator de erro é calculado de forma diferente.

```
 //Calcule o fator de erro de um neuronio da camada intermediária
 FatorErro=0;
 For cada neuronio N para o qual EsteNeuronio Está Conectado
   // Somatorio do erro*peso
      FatorErro = FatorErro + (N.Erro * Peso da Conexão a partir
EsteNeuronio para N)
 Next
```

Para ilustrar, considere um neurônio X1 (EsteNeuronio) que se encontra na camada escondida. X1 está conectado com os neurônios Y1, Y2, Y3 e Y4 (da camada de saída).



Y4.Erro

```
FatorErro de X1 = (Y1.Erro * Peso da conexão de X1 para Y1) + (Y2.Erro *
Peso da conexão de X1 para Y2) + (Y3. Erro * Peso da conexão de X1 para
Y3) + (Y4. Erro * Peso da conexão de X1 para Y4)
```

Agora o erro de X1 pode ser calculado como:

```
X1.Erro = X1.Saida * (1 - X1.Saida) * FatorErro de X1
```

Passo 4 – Ajuste dos Pesos

Após o cálculo dos erros de todos os neurônios de todas das camadas, pode-se corrigir os pesos a fim de que se possa obter saídas mais próximas das esperadas.

Declare uma constante momentum = 0.9

```
// Atualize os pesos dos neurônios das camadas escondidas
 For cada camada in CamadasEscondidas
    For cada neuronio In camada.Neuronios
        neuronio.UpdatePesos()
    Next
 Next

// Atualize os pesos dos neurônios da camada de saída
 For cada neuronio In CamadaSaida.Neuronios
        neuronio.UpdatePesos()
 Next
```

UpdatePesos() corrige cada peso baseado no erro calculado anteriormente. É necessário uma taxa de aprendizagem, por exemplo pode-se utilizar um valor constante: taxa_de_aprendizagem=0.5 ou menor.

Novo_peso=Peso_anterior*momentum+Taxa_aprendizagem*Saída_neurônio_anterior*Erro_neur_posterior

Esta operação deve ser feita para cada conexão entre os neurônios.

```
For cada Neuronio de Entrada N conectado a EsteNeuronio
    Novo_Peso de N = Peso_anterior de N + taxa_de_aprendizagem * N.Saida
* EsteNeuronio.Erro
Next
```

Uma vez que o passo 4 estiver concluído, tem-se uma Rede Neural modificada. Este processo deve ser repetido para todas as entradas por aproximadamente 1000 iterações para que o treinamento da rede seja suficiente.

Para Rodar a Rede Neural

1. Forneça as entradas para a rede como descrito no passo 1;
2. Calcule as saídas como explicado no passo 2;

A rede deve ser treinada com um número suficiente de exemplos e com um número grande de iterações para que o erro seja reduzido e o resultado esperado seja alcançado. Saiba que é impossível obter uma saída 100% correta para uma dada entrada.

Por onde começar?

1. Entenda o problema (página 1);
2. Defina as estruturas de dados necessárias para armazenar entradas, neurônios, pesos, saídas;
3. Implemente os principais métodos descritos nos passos 1 a 4;
4. Selecione um sub-conjunto de caracteres para realizar os testes com um treinamento mais curto e assim mais ágil;
5. Uma vez que a rede estiver reconhecendo alguns caracteres, aumente o conjunto de letras e dígitos e verifique como a rede se comporta, quais os limites de reconhecimento dos caracteres;
6. Se a rede apresentar uma performance baixa, modifique os valores das constantes: taxa de aprendizagem e momentum, reduzindo-a para que os pesos sejam sutilmente alterados a cada correção.
7. A cada execução você poderá observar pequenas mudanças no comportamento da rede, uma vez que os pesos são inicializados aleatoriamente. Entretanto, as execuções sempre devem convergir (com mais ou menos iterações sendo necessárias).