Redes Neurais Artificiais

Luís Fabrício W. Góes

Sumário

- Motivação
- Conceito
- Histórico
- Neurônio Perceptron
- Estratégia Simples de Treinamento
- Exemplo
- Perceptron Multicamadas
- Referências

Motivação

- **Constatação que o cérebro processa informações de forma diferente dos computadores convencionais**

- **Cérebro:** velocidade 1 milhão de vezes mais lenta que qualquer porta digital. Neurônio gasta 2ms para processar uma “instrução” ou sinal de entrada

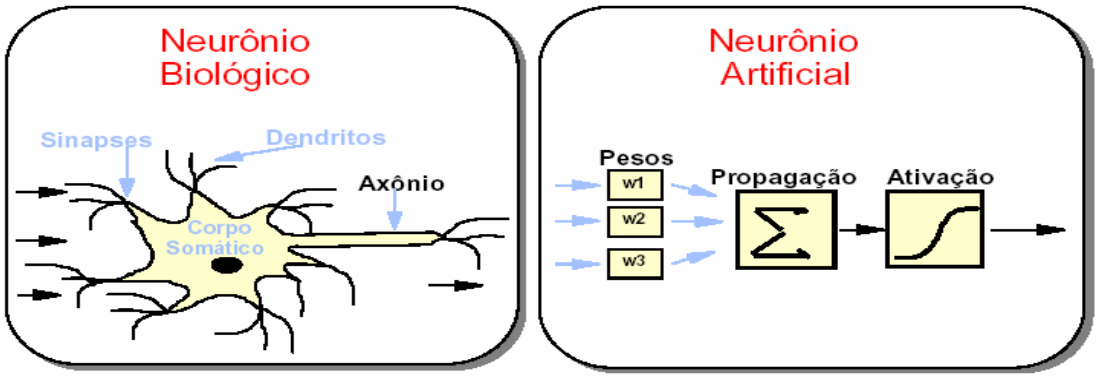
- **Computador:** processamento extremamente rápido e preciso na execução de seqüência de instruções. Um computador gasta 2 ns para processar uma instrução

Conceito

- Redes Neurais Artificiais (RNAs) são sistemas inspirados nos neurônios biológicos e na estrutura massivamente paralela do cérebro, com capacidade de adquirir, armazenar e utilizar conhecimento experimental

Neurônio Biológico x Neurônio Artificial

- Neurônio Biológico
  - Dendritos e Sinapses: entradas de sinais
  - Corpo Somático: processa os sinais de entrada
  - Axônio: saída do neurônio (ativado ou não)
- Neurônio Artificial
  - Entradas e Pesos: entradas de dados com pesos para cada entrada
  - Propagação: somatório da multiplicação dos pesos pelas entradas
  - Saída: o somatório passa por uma função de ativação



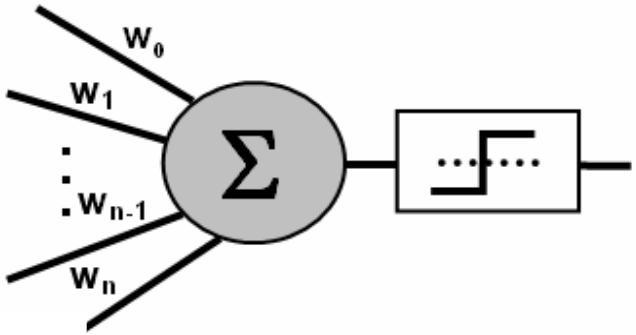
Histórico

- McCulloch & Pitts (1943): modelo computacional para o neurônio artificial. Não possuía capacidade de aprendizado
- Hebb (1949): modelo de aprendizado (***Hebbian Learning Rule)***
- Rosenblatt (1957): Perceptron, com grande sucesso em certas aplicações de problemas em outras aparentemente similares
- Minsly & Papert (Perceptrons 1969): prova matemática de que as redes Perceptron são incapazes de solucionar problemas simples tipo OU-EXCLUSIVO
- Rumelhart (início de 80): novos modelos que superaram os problemas dos Perceptrons. Ex: redes perceptron multicamadas etc.

Neurônio Perceptron

- Uma rede neural perceptron tradicional possui uma camada com vários neurônios artificiais.

Neurônio Artificial



$$a = \sum_{i=1}^{N} x \quad w_i \qquad F(a) = \begin{cases} 1, & \text{se } a \geq \theta \\ 0, & \text{se } a < \theta \end{cases}$$

- Ao receber as entradas, o núcleo do neurônio multiplica cada entrada pelo peso correspondente e realiza um somatório. Depois o resultado passa por uma função de ativação (F(a)), onde θ é um limiar e w é um peso.

Parâmetros

- **Taxa de Aprendizado**: velocidade na qual o sistema está caminhando para encontrar a solução, ou seja, o tamanho do passo.
  - Uma taxa muito baixa vai fazer com que necessite de muitas épocas de treinamento do algoritmo
  - Uma taxa de aprendizado alta pode tornar o algoritmo oscilante, demora para convergir.

- **Erro Global**: média dos erros para todo o conjunto de dados de teste.

- **Época**: cada período de treinamento, no qual todo o conjunto de dados já foi usado.

Estratégia Simples de Treinamento

- Dado um conjunto de dados (entradas e saída)
  - (Passo 1) Iniciar os valores dos pesos aleatoriamente
  - (Passo 2) Para cada conjunto de dados, calcular o erro na saída no neurônio

    (erro = saída do neurônio – saída desejada)
  - (Passo 3) Baseado no erro, atualizar o valor de cada peso segundo a seguinte equação:

$$W_x = W_x + \text{Erro} \times \text{TaxadeAprendizado} \times \text{Entrada}$$

  - (Passo 4) Repetir a partir do passo 2, até que se complete um número de épocas ou o erro global esteja abaixo de um limiar

Exemplo

- Vamos ensinar um neurônio, com dois pesos de entrada e sem função de ativação, a somar dois números iguais, utilizando o seguinte conjunto de dados (1+1 = 2):

1	1	2
2	2	4
3	3	6
4	4	8
5	5	10

- Além disso, vamos utilizar uma taxa de aprendizado igual a 0.05 e apenas uma época (5 iterações)

Execução do Treinamento

- Iniciando pesos aleatoriamente
  - W0 = 0.000000
  - W1 = 0.800000
- Treinando com a linha 0
  - Erro = 1.200000 na linha 0
  - Peso W0 atualizado para 0.060000
  - Peso W1 atualizado para 0.860000
- Treinando com a linha 1
  - Erro = 2.160000 na linha 1
  - Peso W0 atualizado para 0.276000
  - Peso W1 atualizado para 1.076000
- Treinando com a linha 2
  - Erro = 1.944000 na linha 2
  - Peso W0 atualizado para 0.567600
  - Peso W1 atualizado para 1.367600
- Treinando com a linha 3
  - Erro = 0.259200 na linha 3
  - Peso W0 atualizado para 0.619440
  - Peso W1 atualizado para 1.419440
- Treinando com a linha 4
  - Erro = -0.194400 na linha 4
  - Peso W0 atualizado para 0.570840
  - Peso W1 atualizado para 1.370840
- Fim do Treinamento

Teste

- Para verificar a saída da rede é necessário realizar o somatório da multiplicação das entradas pelos pesos.
  Ex: 1x0.570840 + 1x1.370840 = 1.94.
- Na tabela abaixo notamos que a rede aprendeu a somar dois números iguais com uma margem de erro de 0.02. O erro poderia ser menor se tivéssemos a treinado por mais épocas. Além disso, ela consegue generalizar, somando números não treinados anteriormente (ex:10+10)

Entradas		Saída
1	1	1.94
2	2	3.88
3	3	5.8
4	4	7.76
5	5	9.7
6	6	11.65
7	7	13.59
10	10	19.41

Exemplo Feliz/Infeliz

- Voltando ao Precisaremos normalizar os valores para que cada entrada tenha um mesmo peso, ou seja, todos os valores de entrada devem estar entre 0 e 1
  - Depois de normalizada a tabela fica assim (0 p/ infeliz e 1 p/ feliz):

Calorias	Horas Sono	Estado
0.9	0.25	0
0.66	0.15	0
0.83	0.55	1
0.86	0.63	1
0.16	0.2	0
0.1	0.65	0
0.33	0.8	1
0.53	0.87	1
0.6	0.46	0
0.23	1	1

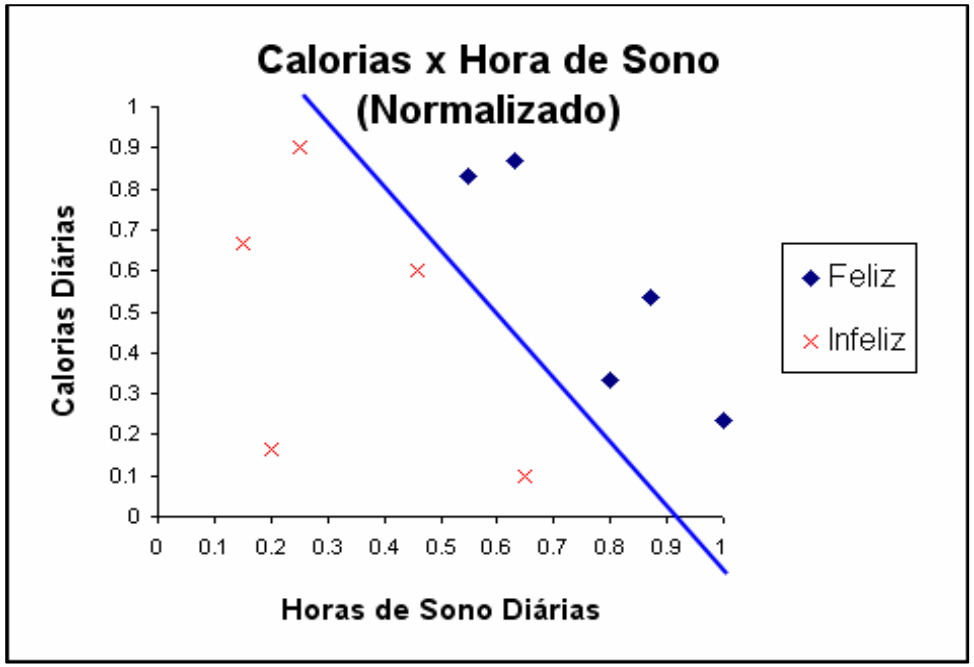
Treinamento

- Parou quando atingiu um erro global de 0.0001
- Durou 30 épocas (ou 300 iterações)
- Utilizamos uma taxa de aprendizado de 0.01
- O limiar da função de ativação foi 0.5, ou seja, uma saída acima de 0.5 vira 1 e menor que 0.5 vira 0
- Pesos finais: W0 = 0.416882

  W1= 0.507391
- Tempo de treinamento = menos de 1 s

Classificador

- Foi encontrada uma função que separa perfeitamente os dois grupos de dados (felizes e infelizes)



- Desvantagem
  - O neurônio perceptron somente resolve problemas lineares (separar dois grupos)

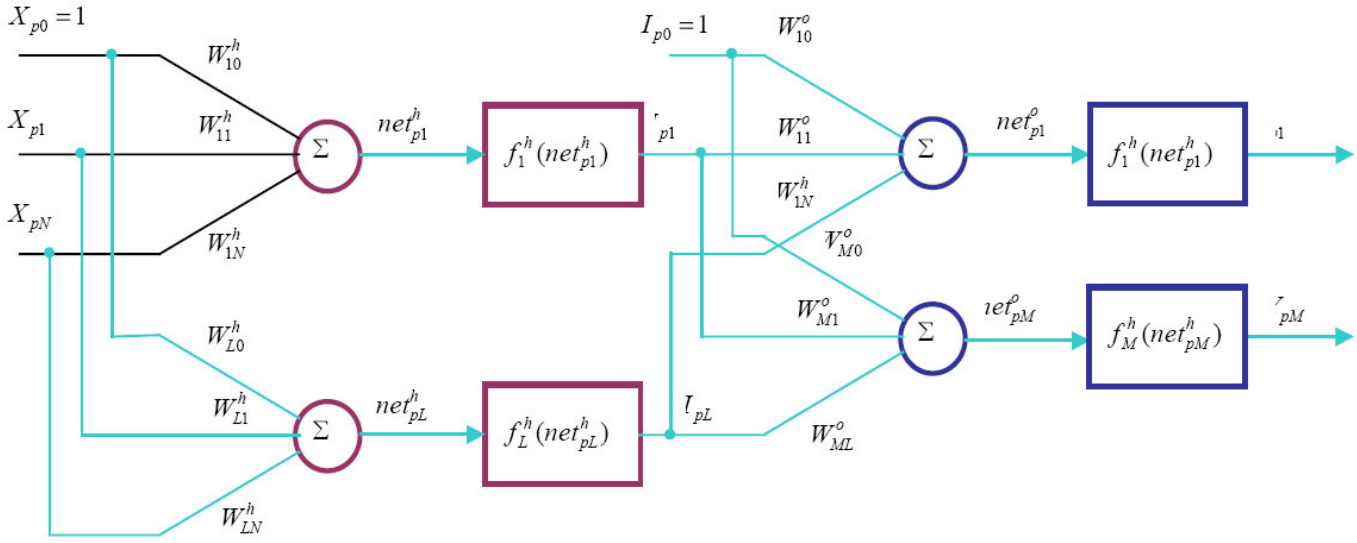
Perceptron Multicamadas

- **Motivação**
  - Superar as limitações das redes de camada única
    - Regressão linear
    - Classificação de problemas linearmente separáveis

- **Arquitetura da Rede**
  - Uma MLP é composta de várias camadas: as entradas, camada escondida e camada de saída.
  - Cada camada possui vários neurônios. Neurônios de uma mesma camada são independentes entre si (exploração do paralelismo).
  - As saídas dos neurônios de uma camada anterior são ligados as entradas dos neurônios da camada da frente

Arquitetura

- Dado um número suficiente de nodos escondidos, uma MLP com apenas uma camada escondida pode aproximar qualquer função contínua (Cybenko, 1989)
- Portanto, não é necessário utilizar mais de uma camada escondida. Apenas ajustar o número de neurônios da camada escondida



Treinamento

- Utiliza o algoritmo back-propagation, no qual o erro na camada de saída é propagado para as camadas anteriores, para ajustar os pesos

- **Problemas**
  - Não existe uma regra de treinamento eficiente. O treinamento é bastante experimental e as vezes demorado
  - É necessário treinar a rede várias vezes com parâmetros diferentes até encontrar um classificador ideal
  - **Overfitting**: uma rede hiper treinada (treinou por muito tempo), ou possui mais neurônios do que precisa, acaba ficando bitolada em um grupo específico de dados, diminuindo a sua generalização

Referências

- Aula baseada no material da disciplina de Inteligência Artificial do professor:
  - Talles Medeiros (PUC-Minas)

- Russel, S. & Norvig, P., "**Inteligência Artificial**", Elsevier, 2ª edição, 2004.

- Luger, G. F., "**Inteligência Artificial**", Bookman, 4ª edição, 2004.

- Han, J. & Kamber, M., "Data Mining: Concepts and Techniques", Morgan Kaufmann, 2001.