Aprendizagem Profunda com Matlab
Aprendizagem de Máquina, Redes Neurais e Inteligência Artificial

Professor Marco H. Terra
terra@sc.usp.br

Departmento de Engenharia Elétrica e de Computação
Universiade de São Paulo em São Carlos

Parte 2

September 8, 2020

MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence
Phil Kim - Seoul, Soul-t'ukpyolsi, Korea (Republic of)
ISBN-13 (pbk): 978-1-4842-2844-9
ISBN-13 (electronic): 978-1-4842-2845-6
DOI 10.1007/978-1-4842-2845-6

Sumário

Rede Neural

- Largamente utilizada como modelo para aprendizagem de máquina
- Grande interesse atual em rede neural (RN): aprendizagem profunda
- Como RN está relacionada com aprendizagem de máquina
- Aprendizagem de RN com uma única camada

Rede Neural

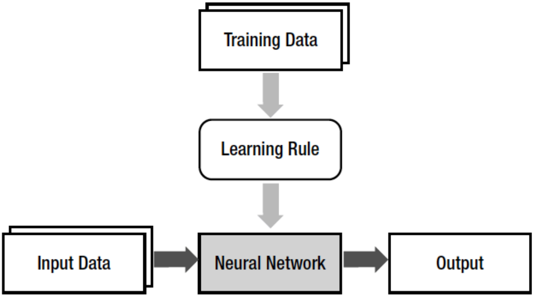

Figure 2: Relação entre aprendizagem de máquina e rede neural.

Nós de uma Rede Neural

- Neurônio não tem a capacidade de armazenar informação
- Apenas transmite sinais para outro neurônio
- O cérebro é uma rede gigante de neurônios
- A associação de neurônios forma uma informação específica
- A RN reproduz a associação dos neurônios, mecanismo mais importante do cérebro, usando os pesos

Nós de uma Rede Neural

Cérebro	**Rede Neural**
Neurônio	Nó
Conexão de neurônios	Peso da conexão

Table 1: Resumo da analogia entre cérebro e rede neural

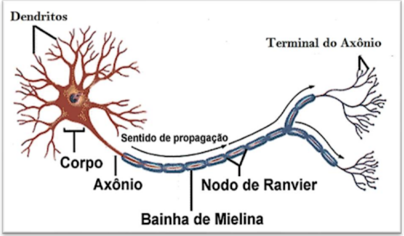

Figure 3: Neurônio (Fonte: MT Ciências, circuito itinerante.)

Nós de uma Rede Neural

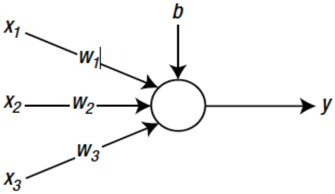

Figure 4: Um nó que recebe três entradas.

Nós de uma Rede Neural

- *Node of Ranvier, periodic gap in the insulating sheath (myelin) on the axon of certain neurons that serves to facilitate the rapid conduction of nerve impulses*
- Neurônio artificial (Figura 4): círculo e seta denotam o nó e o fluxo de sinal
- x_1, x_2 e x_3 são os sinais de entrada
- w_1, w_2 e w_3 são os pesos para os sinais correspondentes
- b é um fator de polarização

Nós de uma Rede Neural

- A soma poderada do neurônio da Figura 4 é calculada da seguinte maneira

$$v = (w_1x_1) + (w_2x_2) + (w_3x_3) + b \tag{1}$$

$$v = wx + b \tag{2}$$

$$w = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3}$$

- Este exemplo ilustra a função dos pesos na RN: imita como o cérebro altera a associação dos neurônios

Nós de uma Rede Neural

- Função de ativação: determina o comportamento dos nós

$$y = \phi(v) = \phi(wx + b) \tag{4}$$

Camadas da Rede Neural

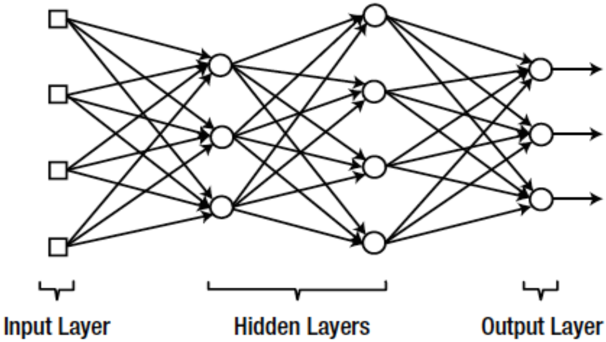

Figure 5: Uma estrutura por camadas dos nós.

Nós de uma Rede Neural

- RN com apenas camadas de entrada e de saída são chamadas de RN com camada simples
- Quando há uma camada escondida: *shallow or vanilla NN*
- Quando há duas ou mais camadas escondidadas: RN profunda

Camadas da Rede Neural

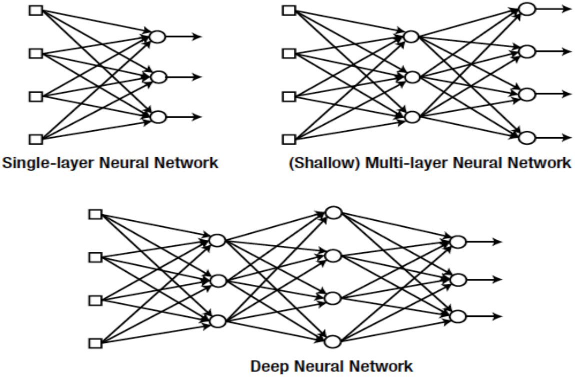

Figure 6: As definições de uma RN dependem da arquitetura das camadas.

Camadas da Rede Neural

(source: NASA Astronomy Picture of the Day)

Figure 7: Via Láctea.

Camadas da Rede Neural

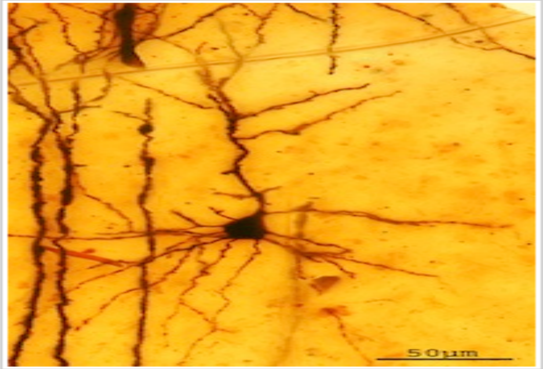

Figure 8: Neurônio.

Camadas da Rede Neural

- O cérebro é uma rede gigante de neurônios
- A RN é uma rede de nós
- Quantos neurônios, aproximadamente, temos no cérebro?

Camadas da Rede Neural

- O cérebro é uma rede gigante de neurônios
- A RN é uma rede de nós
- Quantos neurônios, aproximadamente, temos no cérebro?
- 86 bilhões de neurônios
- Quantas estrelas temos na Via Láctea?

Camadas da Rede Neural

- O cérebro é uma rede gigante de neurônios
- A RN é uma rede de nós
- Quantos neurônios, aproximadamente, temos no cérebro?
- 86 bilhões de neurônios
- Quantas estrelas temos na Via Láctea?
- Entre 200 e 400 bilhões

Camadas da Rede Neural

- Considere a seguinte RN com uma camada escondida:

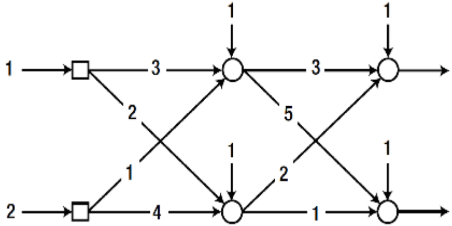

Figure 9: Rede neural com uma única camada escondida.

Camadas da Rede Neural

- Função de ativação:

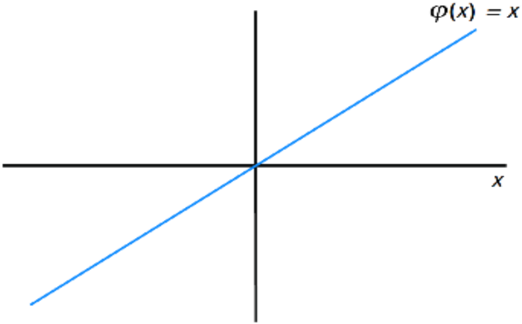

Figure 10: Função de ativação linear de cada nó.

Camadas da Rede Neural

- Calcule as saídas da seguinte rede neural:

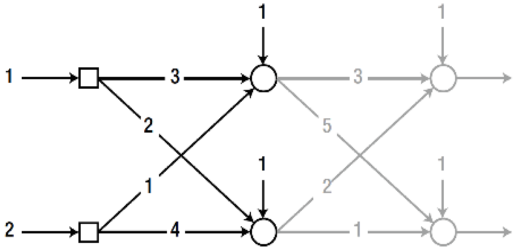

Figure 11: Cálculo da saída a partir da camada escondida.

Camadas da Rede Neural

- Soma ponderada, primeiro nó da camada escondida

$$\begin{aligned} \text{Soma ponderada:} \quad v &= (3x1) + (1x2) + 1 = 6 \qquad (5) \\ \text{Saída:} \quad y &= \phi(v) = v = 6 \qquad (6) \end{aligned}$$

- Segundo nó da camada escondida

$$\begin{aligned} \text{Soma ponderada:} \quad v &= (2x1) + (4x2) + 1 = 11 \qquad (7) \\ \text{Saída:} \quad y &= \phi(v) = v = 11 \qquad (8) \end{aligned}$$

Nós de uma Rede Neural

- A soma poderada: relação matricial

$$
\begin{aligned}
v &= \begin{bmatrix} 3x1 + 1x2 + 1 \\ 2x1 + 4x2 + 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \end{bmatrix} && (9) \\
v &= Wx + b && (10)
\end{aligned}
$$

-

$$
W = \begin{bmatrix} \text{pesos do primeiro nó} \\ \text{pesos do segundo nó} \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \qquad (11)
$$

Nós de uma Rede Neural

- Cálculo das saídas da próxima camada:

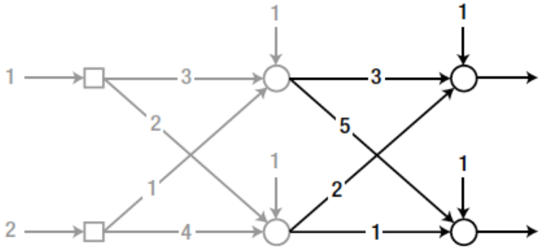

Figure 12: Camada oculta.

Nós de uma Rede Neural

- A soma poderada:

$$v = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 11 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 41 \\ 42 \end{bmatrix} \tag{12}$$

- Saída da camada oculta:

$$v = \phi(v) = v = \begin{bmatrix} 41 \\ 42 \end{bmatrix} \tag{13}$$

- Simplicidade algébrica para tratar problemas complexos

Aprendizagem de máquina

- Consequência de se adotar a função de ativação linear

$$
\begin{aligned}
v &= \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 11 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} && (14) \\
&= \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \left(\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} && (15) \\
&= \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} && (16) \\
&= \begin{bmatrix} 13 & 11 \\ 17 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 6 \\ 7 \end{bmatrix} && (17)
\end{aligned}
$$

Aprendizagem de máquina

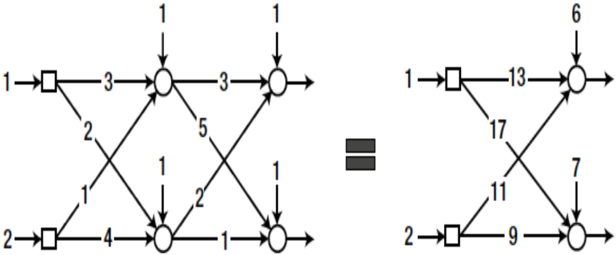

Figure 13: Equivalente a uma RN sem camada escondida.

Aprendizagem supervisionada

Procedimento para treinar uma RN supervisionada

1. Inicialize os pesos com valores adequados
2. Considere a entrada a partir dos dados de treinamento entrada, saída desejada. Insira esses dados na RN e calcule o erro da saída desejada
3. Ajuste os pesos para diminuir o erro
4. Repita os passos 2 e 3 para todos os dados de treinamento

Aprendizagem supervisionada

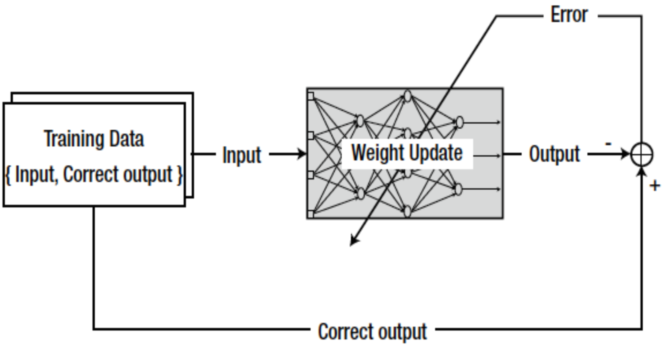

Figure 14: O conceito de rede supervisionada.

Treinamento de uma RN com uma camada

- Regra de aprendizagem é vital na pesquisa de RN
- Regra Delta: regra de aprendizagem representativa de uma RN com uma camada

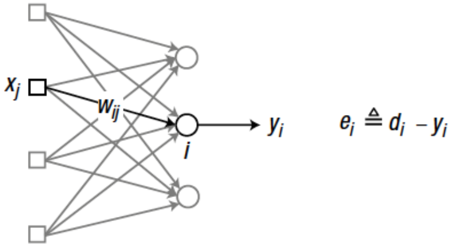

Figure 15: RN com uma camada

Treinamento de uma RN com uma camada

"Se um nó de entrada contribue para o erro do nó de saída, o peso entre os dois nós é ajustado em proporção ao valor da entrada, x_j e o erro de saída, e_i."

$$w_{ij} \leftarrow w_{ij} + \alpha e_i x_j \tag{18}$$

- x_j = saída do nó de entrada j, (j=1,2,3)
- e_i = erro do nó de saída i
- w_{ij} = peso entre o nó de saída i e o nó de entrada j
- α = taxa de aprendizagem $(0 < \alpha \leq 1)$
  (baixa, aprendizagem lenta / alta, aprendizagem rápida)

Exemplo

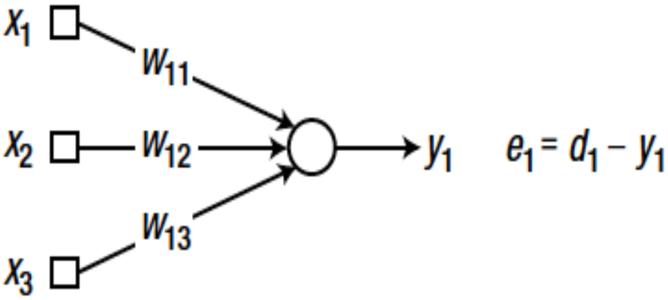

Figure 16: Rede neural com uma única camada com três nós de entrada e um nó de saída.

Exemplo

Regra delta

$$w_{11} \leftarrow w_{11} + \alpha e_1 x_1 \tag{19}$$

$$w_{12} \leftarrow w_{12} + \alpha e_1 x_2 \tag{20}$$

$$w_{13} \leftarrow w_{13} + \alpha e_1 x_3 \tag{21}$$

Resumo do processo de treinamento usando a regra delta para uma rede com uma única camada

- 1. Inicialize os pesos com valores adequados
- 2. Considere x_i, d_i e calcule

$$e_i = d_i - y_i$$

- 3. Calcule as taxas de ajuste dos pesos

$$\Delta w_{ij} = \alpha e_i x_j$$

- 4. Ajuste os peso com

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \tag{22}$$

- 5. Execute os passos 2-4 para todos os dados de treinamento
- 6. Repita os passos 2-5 até que o erro alcance um nível de tolerância aceitável

Gradiente descendente

- Época: número de iterações de treinamento
- Exemplo: época $= n$ significa que a rede neural repete o processo de treinamento n vezes com o mesmo conjunto de dados
- A regra delta é o método do gradiente descendente

Regra delta generalizada

$$w_{ij} \leftarrow w_{ij} + \alpha_i s_j \tag{23}$$

$$\delta_i = \dot{\phi}(v_i) e_i \tag{24}$$

- e_i = erro do nó de saída i
- v_i = soma ponderada do nó de saída i
- $\dot{\phi}$ = derivada da função de ativação ϕ do nó de saída i

Gradiente descendente

- Considerando a função de ativação linear $\phi(x) = x$, $\dot{\phi}(x) = 1$, então

$$\delta_i = e_i$$

- Função de ativação sigmoidal

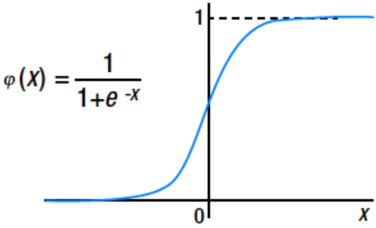

Figure 17: Função sigmoidal

Gradiente descendente

- Derivada da função sigmoidal

$$\dot{\phi}(x) = \frac{0 * (1 + e^{-x}) - (0 - e^{-x})}{(1 + e^{-x})^2} \quad (25)$$

$$= \frac{1}{(1 + e^{-x})} \times \left(1 - \frac{1}{(1 + e^{-x})}\right) \quad (26)$$

$$= \phi(x)(1 - \phi(x)) \quad (27)$$

- Regra delta e ajustes dos pesos

$$\delta_i = \dot{\phi}_i(v_i)e_i = \phi(v_i)(1 - \phi(v_i))e_i$$

$$w_{ij} \leftarrow w_{ij} + \phi(v_i)(1 - \phi(v_i))e_i x_j$$

Método Gradiente Descendente Estocástico (GDE)

- Calcula o erro para cada dado de treinamento e ajusta os pesos imediatamente
- Se tivermos n pontos de dados de treinamento o GDE ajusta os pesos n vezes

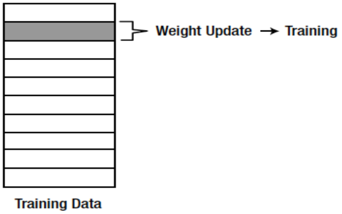

Figure 18: Como o ajuste dos pesos do GDE está relacionado com o conjunto dos dados de treinamento

Método Gradiente Descendente Estocástico (GDE)

- O GDE calcula os ajustes dos pesos com

$$\Delta w_{ij} = \alpha \delta_i x_j$$

Método *batch*

- No método *batch*, cada ajuste de peso é calculado para todos os erros dos dados de treinamento
- A média dos ajustes dos pesos é usada para ajustar os pesos

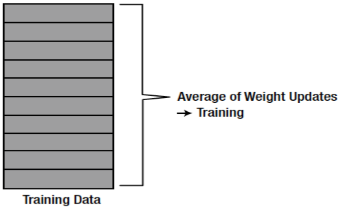

Figure 19: O método *batch* calcula o ajuste dos pesos e processo de trainamento.

Método *batch*

- Utiliza todos dados de treinamento e ajusta os pesos uma úncia vez
- O cálculo dos pesos é feito da seguinte maneira:

$$\Delta w_{ij} = \frac{1}{N} \sum_{k=1}^{N} \Delta w_{ij}(k)$$

  sendo $\Delta_{ij}(k)$ é o ajuste do peso para o k-ésimo dado de treinamento e n é o número total dos dados de treinamento
- Cálculo da média dos ajustes dos pesos: maior consumo de tempo

Método *batch*

- Utiliza todos dados de treinamento e ajusta os pesos uma úncia vez
- O cálculo dos pesos é feito da seguinte maneira:

$$\Delta w_{ij} = \frac{1}{N} \sum_{k=1}^{N} \Delta w_{ij}(k)$$

  sendo $\Delta_{ij}(k)$ é o ajuste do peso para o k-ésimo dado de treinamento e n é o número total dos dados de treinamento
- Cálculo da média dos ajustes dos pesos: maior consumo de tempo

Exemplo 1

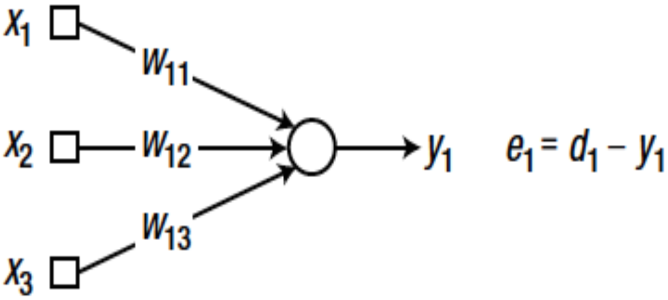

Figure 20: Rede neural com uma única camada com três nós de entrada e um nó de saída.

Exemplo 1

x_1	x_2	x_3	d_i
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

- Regra delta - função sgmoidal

$$\begin{aligned} \delta_i &= \phi(v_i)(1 - \phi(v_i))e_i \\ \Delta w_{ij} &= \alpha \delta_i x_j \\ w_{ij} &\leftarrow w_{ij} + \Delta w_{ij} \end{aligned}$$

- Aplicação dos métodos GDE e *batch*

Exemplo 1

Método do Gradiente Descendente Estocástico

- This program calls in the function DeltaBatch.m for supervised training of a neural network.

clear all clc

X = [0 0 1; 0 1 1; 1 0 1; 1 1 1]; D = [0; 0; 1; 1];

W = 2*rand(1, 3) - 1;

- Training (adjusting weights):
for epoch = 1:10000
W = DeltaBatch(W, X, D);
end

- Inference:
N = 4;
y = zeros(N,1);
for k = 1:N
x = X(k, :)';
v = W*x;
y(k) = Sigmoid(v); - obtained
output.
end
disp('Results:'); disp(' [desired neuron$_{o}$*utput*]'); *disp*([*Dy*]);

function W = DeltaBatch(W, X, D)
alpha = 0.9;
dWsum = zeros(3, 1);
N = 4;
for k = 1:N

$$\begin{aligned} x &= X(k,:)'; \\ d &= D(k); \\ v &= W * x; \\ y &= Sigmoid(v); \\ e &= d - y; \\ delta &= y * (1 - y) * e; \\ dW &= alpha * delta * x; \\ dWsum &= dWsum + dW; \end{aligned}$$

end
dWavg = dWsum/N;
W(1) = W(1) + dWavg(1);
W(2) = W(2) + dWavg(2);
W(3) = W(3) + dWavg(3);
end

```
function y = Sigmoid(x)
y = 1 / (1 + exp(-x));
end
```

Resultado:

saída real	d_i
0.0102	0
0.0083	0
0.9932	1
0.9917	1

w_1	w_2	w_3
9.5649	-02084	-4.5752

Exemplo 1

Método *Batch*

Exemplo 1

- This program calls in the function DeltaBatch.m for supervised training of a neural network.

```
clear all; clc

X = [0 0 1; 0 1 1; 1 0 1; 1 1 1]; D = [0; 0; 1; 1];

W = 2*rand(1, 3) - 1;

for epoch = 1:40000
W = DeltaBatch(W, X, D);
end
```

- Inference:

```
N = 4;
y = zeros(N,1);
for k = 1:N
x = X(k, :)';
v = W*x;
y(k) = Sigmoid(v);
end

disp('Results:');
disp('[desired neuron-output]');
disp([D y]);
```

Exemplo 1

```
function W = DeltaBatch(W, X, D);
alpha = 0.9;
dWsum = zeros(3, 1);
N = 4;
for k = 1:N
x = X(k, :)';
d = D(k);
v = W*x;
y = Sigmoid(v);
e = d - y;
delta = y*(1-y)*e;
dW = alpha*delta*x;
dWsum = dWsum + dW;
end
dWavg = dWsum/N;
W(1) = W(1) + dWavg(1);
W(2) = W(2) + dWavg(2);
W(3) = W(3) + dWavg(3);
end
```

Resultado:

saída real	d_i
0.0102	0
0.0083	0
0.9932	1
0.9917	1

w_1	w_2	w_3
9.5642	-02088	-4.5746

Comparação entre GDE e *Batch*
Erros médios

Exemplo 1

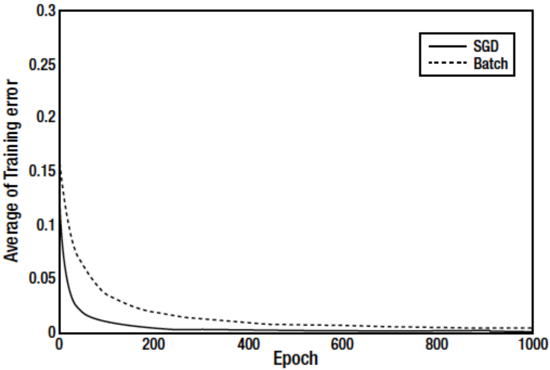

Figure 21: O método Gradiente Descendente Estocástico aprende mais rápido que o método *Batch*.

Exemplo 1

- The weights of both methods are initialized with the same values.

```
X = [0 0 1;
0 1 1;
1 0 1;
1 1 1];
D = [0; 0; 1; 1];
E1 = zeros(1000, 1);
E2 = zeros(1000, 1);
W1 = 2*rand(1, 3) - 1;
W2 = W1;
        for epoch = 1:1000
        W1 = DeltaSGD(W1, X, D);
        W2 = DeltaBatch(W2, X, D);
        es1 = 0;
        es2 = 0;
        N = 4;
            for k = 1:N
            x = X(k, :)';
            d = D(k);
            v1 = W1*x;
            y1 = Sigmoid(v1);
            es1 = es1 + (d - y1)^2;
            v2 = W2*x;
            y2 = Sigmoid(v2);
            es2 = es2 + (d - y2)^2;
            end
        E1(epoch) = es1/N;
        E2(epoch) = es2/N;
       end
```

```
plot(E1, 'r', 'LineWidth', 1.5)
hold on
plot(E2, 'b:', 'LineWidth', 1.5)
xlabel('Epoch')
ylabel('Average of Training error')
legend('SGD', 'Batch')
```