

Universidade Federal de Goiás

Redes Neurais Profundas - Deep Learning

Aula 2 - Redes Neurais Perceptron Multicamadas

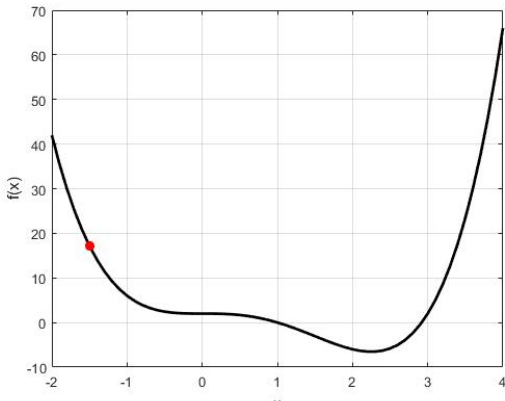
Anderson da Silva Soares
www.inf.ufg.br/~anderson/
www.deeplearningbrasil.com.br

Sumário

- 1 Compreendendo o gradiente
- 2 Funções de ativação
- 3 Superfícies de decisão
- 4 Treinamento de rede multicamadas
 - Retropropagação e regra da cadeia
- 5 Considerações sobre a função de custo
- 6 Considerações - Problemas e limitações
- 7 Dicas de implementações
- 8 Conclusão

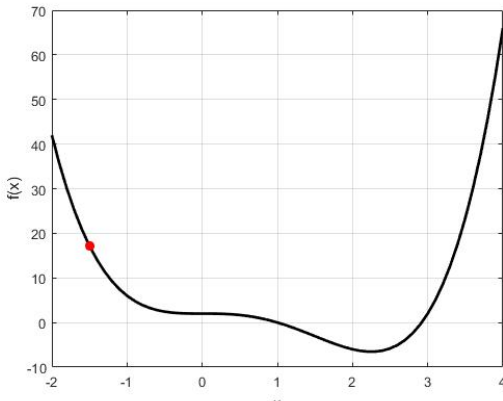
Entendendo o conceito de gradiente:

- Vamos considerar a função $y = f(x) = x^4 - 3x^3 + 2$
- Queremos “descobrir” qual o valor de x que $f(x)$ seja mínimo
- Considere que temos o ponto $x = -1,5$ definido aleatoriamente



Entendendo o conceito de gradiente:

- Vamos considerar a função $y = f(x) = x^4 - 3x^3 + 2$
- Queremos “descobrir” qual o valor de x que $f(x)$ seja mínimo
- Considere que temos o ponto $x = -1,5$ definido aleatoriamente



Entendendo o conceito de gradiente:

- Queremos “descobrir” qual o valor de x que $f(x)$ seja mínimo
- Precisamos da derivada da função, lembrando que, a derivada de x^a é igual a $a * x^{a-1}$
- Logo, a derivada da função no ponto -1,5 é dada por

$$\frac{dy(-1,5)}{dx} = 4 * x^3 - 9 * x^2 = 4 * (-1,5)^3 - 9 * (-1,5)^2 = -33,75$$
- Vejam a “mágica” do gradiente que nos indica a direção do mínimo da função
- $x_{novo} = (-1,5) - 0,0001 * 33,75 * (-1,5) = -1,4949$
- em que 0,0001 é a taxa de aprendizado (aula anterior)
- atualizamos x na direção da resposta correta

Entendendo o conceito de gradiente:

Problema dos mínimos locais

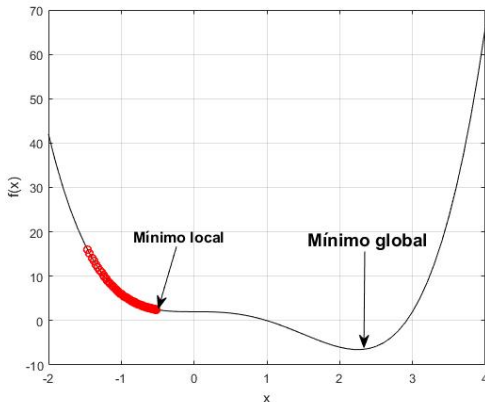


Figura: Soluções encontradas durante o algoritmo.

Relembrando o neurônio individual

Relembrando a última aula: Elementos dos neurônios

- Entradas: x_1, x_2
- Sidas: y_1, y_2
- pesos sinápticos: w_1, w_2, \dots, w_8 (Precisamos determinar por meio do treinamento)
- Funções de ativações de cada neurônio (escolha experimental)

Pergunta: Qual a importância da função de ativação em um neurônio?

Funções de ativação

Importância:

Funções de ativação são funções que recebem um sinal de entrada e o converte para um sinal de saída. Podem ser lineares ou não-lineares, contudo seu principal potencial está na introdução de não lineariedade nas redes neurais.

Funções de ativação:

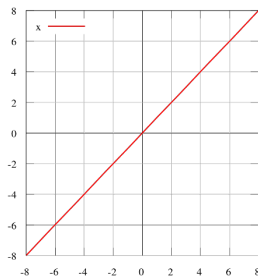


Figura: Função linear.

$$f(x) = x$$

Funções de ativação:

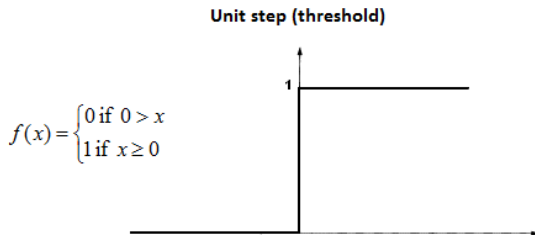


Figura: Função degrau.

Se for um problema de classificação, ótimo. A função entrega de forma direta se o resultado é a classe 0 ou a classe 1.

Funções de ativação:



Como você modelaria um problema de classificação envolvendo 4 classes?

O que aconteceria se mais de um neurônio respondesse 1.

Funções de ativação:

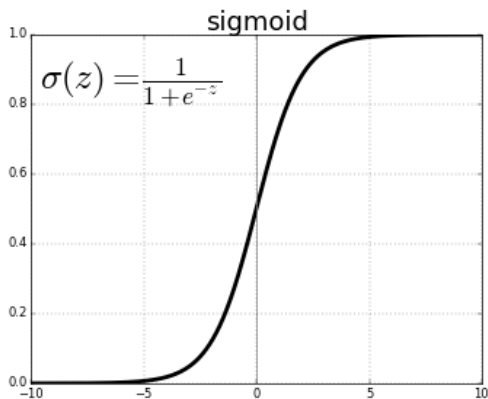


Figura: Função sigmóide.

Funções de ativação:

$$\sigma(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

$$\tanh(x) = 2 * \text{sigmoid}(2x) - 1$$

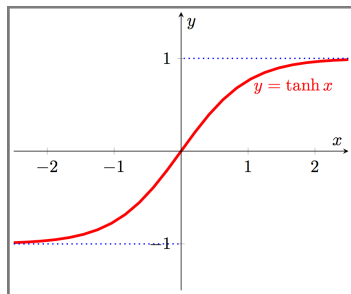


Figura: Função tangente hiperbólica.

Relembrando o neurônio individual

Saída do corpo:

$$x * w$$

Função de
ativação: $\sigma(x * w)$

Note relação não
linear entre a saída
do corpo do
neurônio e a saída
da função de
ativação.

$x * w$	$\sigma(x * w)$	$\text{var}(x * w)$	$\text{var}(\sigma(x * w))$
-2	0,12	-	-
-1,5	0,18	25%	50%
-1	0,27	33%	50%
-0,5	0,38	50%	41%
0	0,50	100%	32%
0,5	0,62	-	-
1	0,73	100%	18%
1,5	0,82	50%	12%
2	0,88	33%	7%

Relembrando o xor:

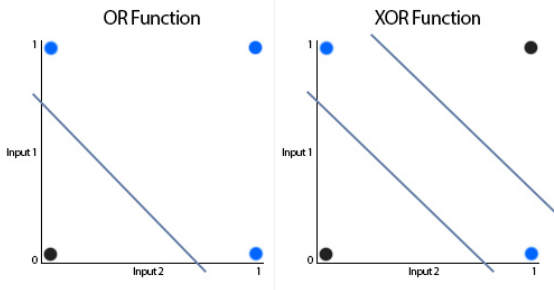


Figura: Exemplo Xor

Solução para o xor:

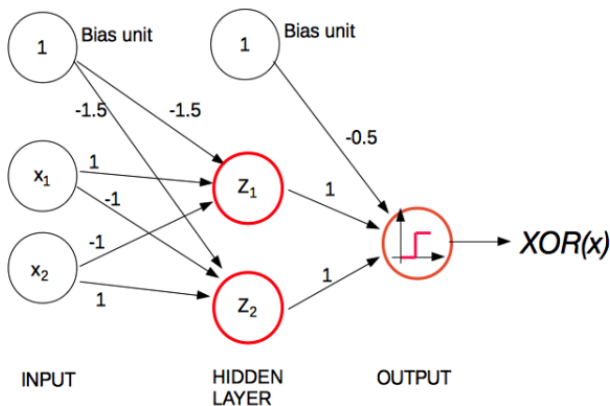


Figura: Solução para o problema xor (não linear)

Relembrando o xor:

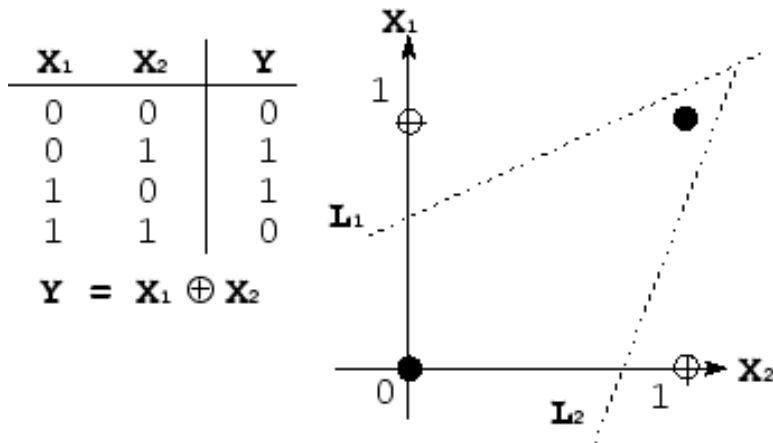


Figura: Exemplo Xor não-linear

Relembrando o xor:


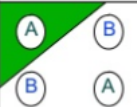



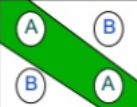






Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

Figura: Exemplo de superfícies construídas por diferentes arquiteturas

Como treinar uma rede multi-layer perceptron

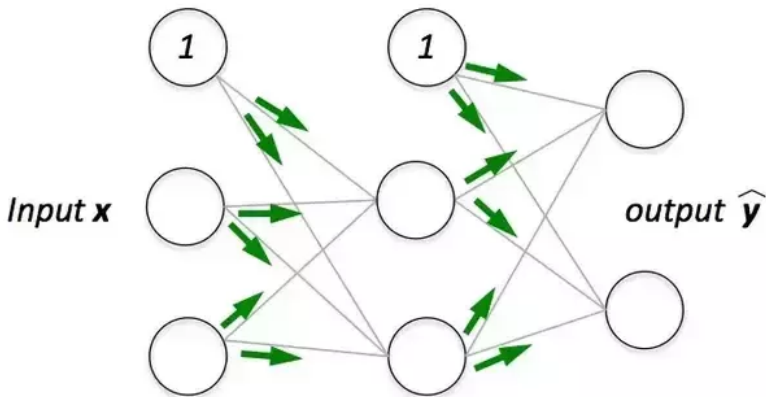


Figura: Fase 1 do Backpropagation

Como treinar uma rede multi-layer perceptron

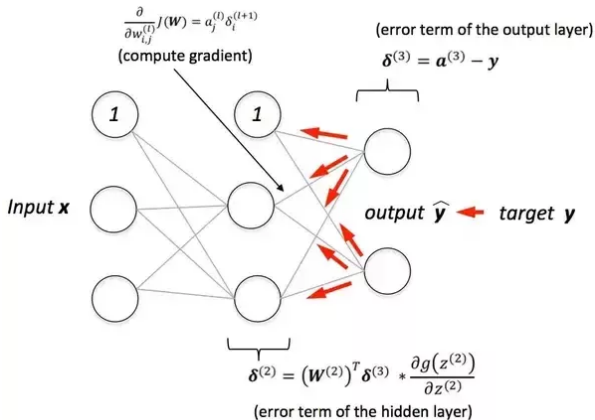


Figura: Fase 2 do Backpropagation

Como treinar uma rede multi-layer perceptron

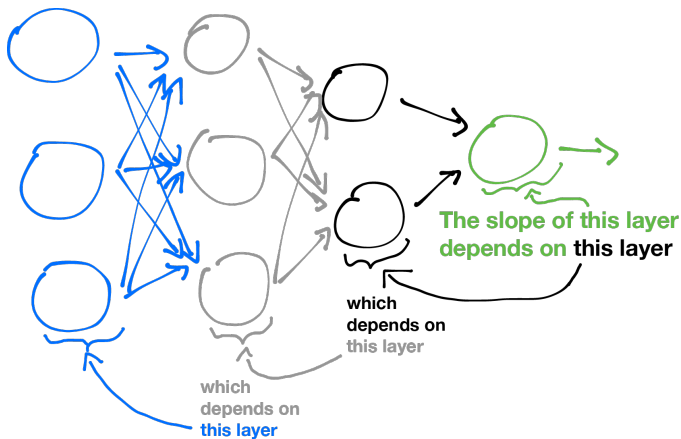


Figura: Fase 3 do Backpropagation

Fase 1 - Propagação - Primeira camada

Tabela: Padrões

Entradas		Saída desejada	
x_1	x_2	y_1	y_2
0,05	0,1	0,01	0,99

Precisamos treinar a rede neural para fornecer as saídas desejadas

Relembrando a última aula: Elementos dos neurônios

- Entradas: x_1, x_2
- Saídas: y_1, y_2
- pesos sinápticos: w_1, w_2, \dots, w_8 (Precisamos determinar por meio do treinamento)
- Funções de ativações de cada neurônio (escolha experimental)

Fase 1 - Propagação - Primeira camada

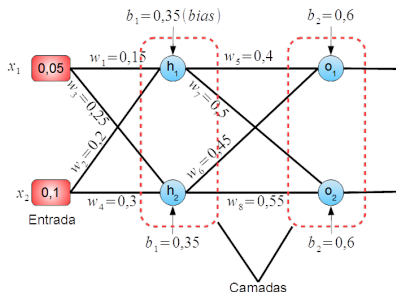
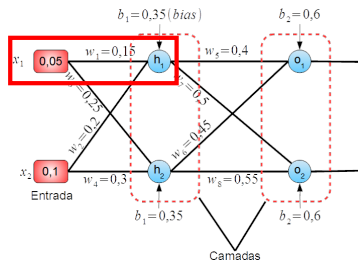


Figura: Rede Neural com pesos inicializados randomicamente.

Parametrizando:

- Usaremos em todos os neurônios a função de ativação hiperbólica;

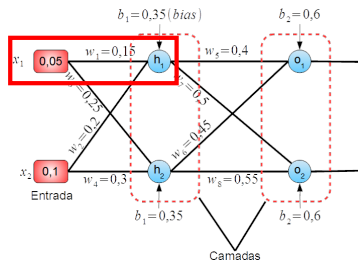
Fase 1 - Propagação - Primeira camada



Propagando a entrada na primeira camada

- $u_{h1} = x_1 * w_1 + x_2 * w_2 + b_1 * 1$
- $u_{h1} = 0,05 * 0,15 + 0,1 * 0,2 + 0,35 * 1 = 0,3775$

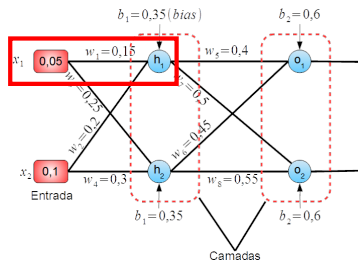
Fase 1 - Propagação - Primeira camada



Propagando a entrada na primeira camada

- $u_{h1} = x_1 * w_1 + x_2 * w_2 + b_1 * 1$
- $u_{h1} = 0,05 * 0,15 + 0,1 * 0,2 + 0,35 * 1 = 0,3775$
- $u_{h2} = x_1 * w_3 + x_2 * w_4 + b_1 * 1$
- $u_{h2} = 0,05 * 0,25 + 0,1 * 0,3 + 0,35 * 1 = 0,39250$

Fase 1 - Propagação - Primeira camada

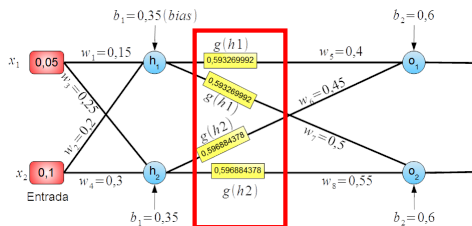


Propagando a entrada na primeira camada - função de ativação

$$\bullet g(h1) = g(u_{h1}) = \frac{1}{1 + e^{u_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269992$$

$$\bullet g(h2) = g(u_{h2}) = \frac{1}{1 + e^{u_{h2}}} = \frac{1}{1 + e^{-0.3925}} = 0.596884378$$

Fase 1 - Propagação - Primeira camada

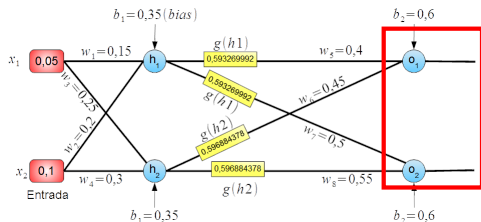


Propagando a entrada na primeira camada - função de ativação

$$\bullet \quad g(h_1) = g(u_{h1}) = \frac{1}{1 + e^{u_{h1}}} = \frac{1}{1 + e^{-0,3775}} = 0,593269992$$

$$\bullet \quad g(h_2) = g(u_{h2}) = \frac{1}{1 + e^{u_{h2}}} = \frac{1}{1 + e^{-0,3925}} = 0,596884378$$

Fase 1 - Propagação - Segunda camada



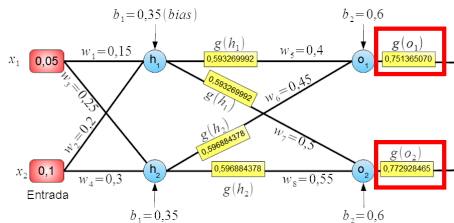
Propagando a entrada na segunda camada

- $$u_{o1} = g(h1) * w_5 + g(h2) * w_6 + b_2 * 1 =$$

$$0,593269992 * 0,4 + 0,596884378 * 0,45 + 0,6 * 1 = 1,105905967$$
- $$u_{o2} = g(h1) * w_7 + g(h2) * w_8 + b_2 * 1 =$$

$$0,593269992 * 0,5 + 0,596884378 * 0,55 + 0,6 * 1 = 1,224921404$$
- $$g(o1) = g(u_{o1}) = \frac{1}{1 + e^{u_{o1}}} = \frac{1}{1 + e^{-1,105905967}} = 0,751365070$$
- $$g(o2) = g(u_{o2}) = \frac{1}{1 + e^{u_{o2}}} = \frac{1}{1 + e^{-1,224921404}} = 0,772928465$$

Fase 1 - Propagação - Segunda camada



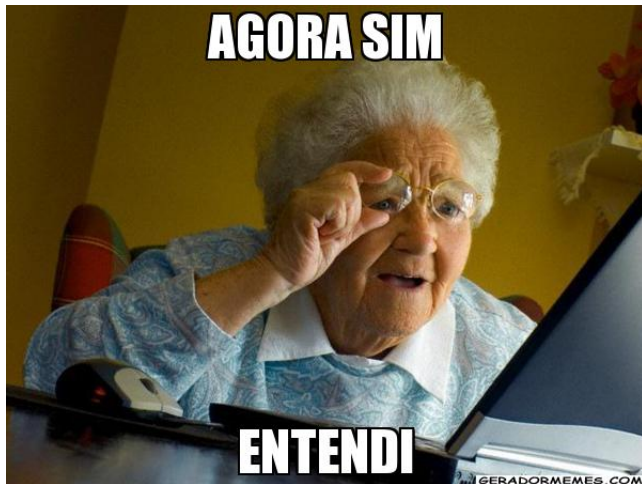
Propagando a entrada na segunda camada

- $$u_{o1} = g(h1) * w_5 + g(h2) * w_6 + b_2 * 1 =$$

$$0,593269992 * 0,4 + 0,596884378 * 0,45 + 0,6 * 1 = 1,105905967$$
- $$u_{o2} = g(h1) * w_7 + g(h2) * w_8 + b_2 * 1 =$$

$$0,593269992 * 0,5 + 0,596884378 * 0,55 + 0,6 * 1 = 1,224921404$$
- $$g(o1) = g(u_{o1}) = \frac{1}{1 + e^{u_{o1}}} = \frac{1}{1 + e^{-1,105905967}} = 0,751365070$$
- $$g(o2) = g(u_{o2}) = \frac{1}{1 + e^{u_{o2}}} = \frac{1}{1 + e^{-1,224921404}} = 0,772928465$$

Fase 1 - Propagação



Fim da Fase 1 - Chega de brincadeira, vamos a parte boa.

Fase 2 - Retropropagação

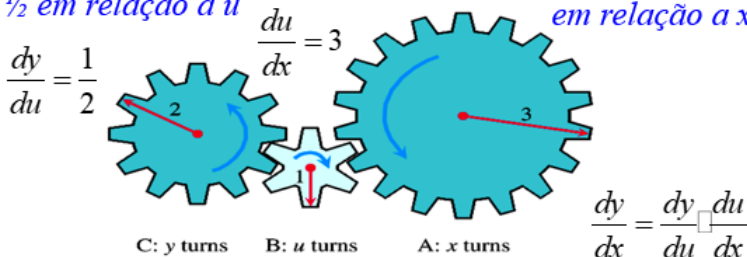
First of all... vamos compreender a regra da cadeia

A faz x girar, B faz u girar e C faz y girar.

u gira 3 vezes em relação a x

Logo, y gira 3/2 em relação a x

y gira 1/2 em relação a u



Fase 2 - Retropropagação

First of all... vamos compreender a regra da cadeia

A regra da cadeia (chain rule) é usada quando temos uma composição de duas funções.

Fase 2 - Retropropagação

First of all... vamos compreender a regra da cadeia

A regra da cadeia (chain rule) é usada quando temos uma composição de duas funções.

Se $y = f(u)$ e $u = g(x)$ então $y = f(g(x))$

Fase 2 - Retropropagação

First of all... vamos compreender a regra da cadeia

A regra da cadeia (chain rule) é usada quando temos uma composição de duas funções.

Se $y = f(u)$ e $u = g(x)$ então $y = f(g(x))$

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$$

$$\frac{dy}{dx} = \frac{d}{dx}(f(g(x))) = f'(g(x))g'(x)$$

Fase 2 - Retropropagação

First of all... vamos compreender a regra da cadeia

Calcule os exemplos abaixo:

- $f(x) = (3x - 5x^2)^7$
- $f(x) = \sqrt{(x^2 - 1)^2}$
- $f(x) = \frac{-7}{(2t - 3)^2}$

Fase 2 - Retropropagação

First of all... vamos compreender a regra da cadeia

Calcule os exemplos abaixo:

- $f(x) = (3x - 5x^2)^7$
 $u = 3x - 5x^2 \rightarrow \frac{du}{dx} = 3 - 10x$
 $y = u^7 \rightarrow \frac{dy}{du} = 7u^6$

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$$
$$\frac{dy}{dx} = 7u^6(3 - 10x) \frac{du}{dx} = 7(3x - 5x^2)^6(3 - 10x)$$

Fase 2 - Retropropagação

Primeiro, precisamos quantificar o quanto a rede errou - diferença entre a saída obtida e a desejada

$$\bullet E_{total} = \frac{1}{2} \sum_{k=1}^N (d_k - g(ok))^2 = E_{o1} + E_{o2} =$$

$$\frac{1}{2} \sum_{k=1}^N (d_1 - g(o1))^2 + \frac{1}{2} \sum_{k=1}^N (y_2 - g(o2))^2$$

$$\bullet E_{o1} = \frac{1}{2} (y_1 - g(o1))^2 = \frac{1}{2} * (0,01 - 0,751365070)^2 = 0,274811083$$

Fase 2 - Retropropagação

Primeiro, precisamos quantificar o quanto a rede errou - diferença entre a saída obtida e a desejada

$$\bullet E_{total} = \frac{1}{2} \sum_{k=1}^N (d_k - g(ok))^2 = E_{o1} + E_{o2} =$$

$$\frac{1}{2} \sum_{k=1}^N (d_1 - g(o1))^2 + \frac{1}{2} \sum_{k=1}^N (y_2 - g(o2))^2$$

$$\bullet E_{o1} = \frac{1}{2} (y_1 - g(o1))^2 = \frac{1}{2} * (0,01 - 0,751365070)^2 = 0,274811083$$

$$\bullet E_{o2} = \frac{1}{2} (y_2 - g(o2))^2 = \frac{1}{2} * (0,99 - 0,772928465)^2 = 0,023560026$$

Fase 2 - Retropropagação

Primeiro, precisamos quantificar o quanto a rede errou - diferença entre a saída obtida e a desejada

$$\bullet E_{total} = \frac{1}{2} \sum_{k=1}^N (d_k - g(ok))^2 = E_{o1} + E_{o2} =$$

$$\frac{1}{2} \sum_{k=1}^N (d_1 - g(o1))^2 + \frac{1}{2} \sum_{k=1}^N (y_2 - g(o2))^2$$

$$\bullet E_{o1} = \frac{1}{2} (y_1 - g(o1))^2 = \frac{1}{2} * (0,01 - 0,751365070)^2 = 0,274811083$$

$$\bullet E_{o2} = \frac{1}{2} (y_2 - g(o2))^2 = \frac{1}{2} * (0,99 - 0,772928465)^2 = 0,023560026$$

$$\bullet E_{total} = 0,274811083 + 0,023560026 = 0,298371109$$

Fase 2 - Retropropagação

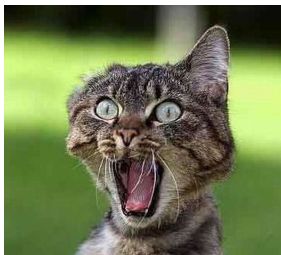
Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial w_5}$: Gradiente com respeito a w_5
- Segundo a regra da cadeia:
$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$$

Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial w_5}$: Gradiente com respeito a w_5
- Segundo a regra da cadeia: $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$
- Agora basta aplicar essa regra das derivadas parciais e está tudo resolvido. Lista de exercícios para a próxima semana.



Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

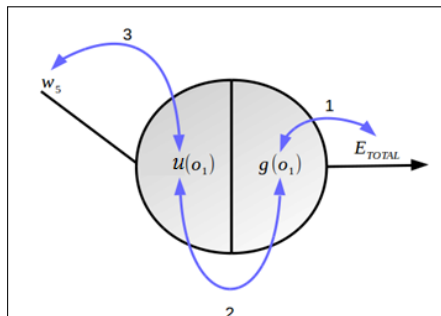
- $\frac{\partial E_{total}}{\partial w_5}$: Gradiente com respeito a w_5
- Segundo a regra da cadeia: $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$
- Vamos ilustrar para ver o quanto pode ser fácil...



Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial w_5}$: Gradiente com respeito a w_5
- $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$

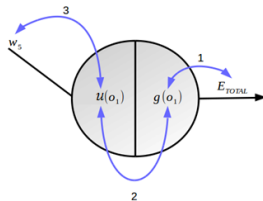


O ponto principal na retropropagação é “desmontar” ou “quebrar” o processo usando a regra da cadeia para torná-lo o mais modular possível e usar a descida do gradiente para atualizar os pesos.

Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

- $E_{total} = \frac{1}{2} * (y_1 - g_{o1})^2 + \frac{1}{2} * (y_2 - g_{o2})^2$
- $\frac{\partial E_{total}}{\partial g_{o1}} = 2 * \frac{1}{2} (y_{o1} - g_{o1})^{2-1} * -1 + 0$ (Constante na derivada em relação a g_{o1})
- $\frac{\partial E_{total}}{\partial g_{o1}} = -(y_1 - g_{o1}) = 0,01 - 0,75136507 = 0,74136507$



Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

Agora a parcela $\frac{\partial g_{o1}}{\partial u_{o1}}$

- $g_{o1} = \frac{1}{1 + e^{-u_{o1}}} = \text{saída da função de ativação}$
- $\frac{\partial g_{o1}}{\partial u_{o1}} = g_{o1}(1 - g_{o1}) = 0,75136507 * (1 - 0,75136507) =$
0,186815602

Agora a parcela $\frac{\partial u_{o1}}{\partial w_5}$

- $u_{o1} = w_5 * g_{h1} + w_6 * g_{h2} + b_2 * 1$
- $\frac{\partial u_{o1}}{\partial w_5} = 1 * g_{h1} * w_5^{1-1} + 0 + 0 = g_{h1} =$ 0,59326992

Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

Finalmente nós temos todas as parcelas

- $$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$$
- $$\frac{\partial E_{total}}{\partial w_5} = 0,74136507 * 0,186815602 * 0,593269992 = 0,0822167041$$

Simplificando em termos de algoritmo:

- $$\frac{\partial E_{total}}{\partial w_5} = -(y_1 - g_{o1}) * g_{o1} * (1 - g_{o1}) * g_{h1} =$$

$$0,74136507 * 0,186815602 * 0,593269992 = 0,0822167041$$

Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

Finalmente nós temos todas as parcelas

- $$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$$
- $$\frac{\partial E_{total}}{\partial w_5} = 0,74136507 * 0,186815602 * 0,593269992 = 0,0822167041$$

Simplificando em termos de algoritmo:

- $$\frac{\partial E_{total}}{\partial w_5} = -(y_1 - g_{o1}) * g_{o1} * (1 - g_{o1}) * g_{h1} =$$

$$0,74136507 * 0,186815602 * 0,593269992 = 0,0822167041$$
- $$\delta_{o1} = -(y_1 - g_{o1}) * g_{o1} * (1 - g_{o1}) = \text{Derivada parcial } (o1).$$

Fase 2 - Retropropagação

Corrigindo w_5 : queremos estimar o quanto w_5 afeta o erro total (E)

Finalmente nós temos todas as parcelas

- $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial w_5}$
- $\frac{\partial E_{total}}{\partial w_5} = 0,74136507 * 0,186815602 * 0,593269992 = 0,0822167041$

Simplificando em termos de algoritmo:

- $\frac{\partial E_{total}}{\partial w_5} = -(y_1 - g_{o1}) * g_{o1} * (1 - g_{o1}) * g_{h1} =$
 $0,74136507 * 0,186815602 * 0,593269992 = 0,0822167041$
- $\delta_{o1} = -(y_1 - g_{o1}) * g_{o1} * (1 - g_{o1}) =$ Derivada parcial (o1).
- Logo, considerando a taxa de aprendizado $\eta = 0,5$
 $w_5(t + 1) = w_5(t) - \eta \frac{\partial E_{total}}{\partial w_5} = 0,4 - 0,5 * 0,0822167041 = 0,35891648$

Fase 2 - Retropropagação

Mesma regra para atualizar os outros pesos sinápticos da camada

- $w_6(t + 1) = 0,408666186$
- $w_7(t + 1) = 0,511301270$
- $w_8(t + 1) = 0,561370121$

Fase 2 - Retropropagação - Primeira camada

Agora, vamos atualizar os pesos da primeira camada.

Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial w_1}$: Gradiente com respeito a w_1

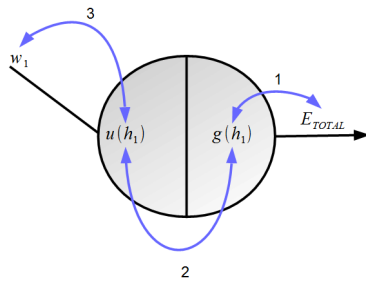
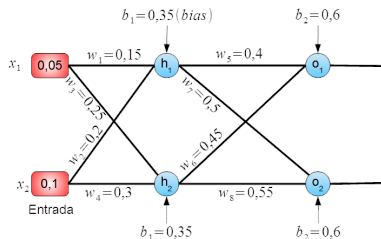
- $$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial g_{h1}} * \frac{\partial g_{h1}}{\partial u_{h1}} * \frac{\partial u_{h1}}{\partial w_1}$$

Fase 2 - Retropropagação - Primeira camada

Agora, vamos atualizar os pesos da primeira camada.

Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial w_1}$: Gradiente com respeito a w_1
- $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial g_{h1}} * \frac{\partial g_{h1}}{\partial u_{h1}} * \frac{\partial u_{h1}}{\partial w_1}$



Fase 2 - Retropropagação - Primeira camada

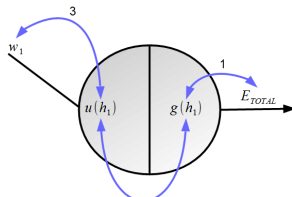
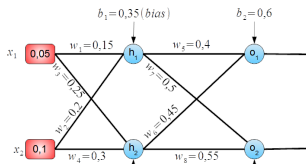
Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial w_1}$: Gradiente com respeito a w_1
- $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial g_{h1}} * \frac{\partial g_{h1}}{\partial u_{h1}} * \frac{\partial u_{h1}}{\partial w_1}$
- Temos três parcelas para resolver a) $\frac{\partial E_{total}}{\partial g_{h1}}$, b) $\frac{g_{h1}}{\partial u_{h1}}$ e, por fim, c) $\frac{\partial u_{h1}}{\partial w_1}$
- Para resolver a), temos que lembrar que $E_{total} = E_{o1} + E_{o2}$
- $E_{o1} = \frac{1}{2} * (d_1 - g_{o1})^2$ e $E_{o2} = \frac{1}{2} * (d_2 - g_{o2})^2$
- $\frac{\partial E_{total}}{\partial g_{h1}} = \boxed{\frac{\partial E_{o1}}{\partial g_{h1}}} + \boxed{\frac{\partial E_{o2}}{\partial g_{h1}}}$, não temos essas parcelas :(

Fase 2 - Retropropagação - Primeira camada

Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $$\frac{\partial E_{total}}{\partial g_{h1}} = \boxed{\frac{\partial E_{o1}}{\partial g_{h1}}} + \boxed{\frac{\partial E_{o2}}{\partial g_{h1}}}, \text{ não temos essas parcelas :}$$
- $$\frac{\partial E_{o1}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial g_{h1}}$$
- $$\frac{\partial E_{o1}}{\partial u_{o1}} = \frac{\partial E_{o1}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} = 0,74136507 * 0,186815602 = \boxed{0,138498562}$$
- valores calculados nos slides 38 e 39



Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial g_{h1}} + \frac{\partial E_{o2}}{\partial g_{h1}}$, não temos essas parcelas :(
- $\frac{\partial E_{o1}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial g_{h1}}$
- $\frac{\partial E_{o1}}{\partial u_{o1}} = \frac{\partial E_{o1}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} = 0,74136507 * 0,186815602 = 0,138498562$
- valores calculados nos slides 38 e 39
- $\frac{\partial u_{o1}}{\partial g_{h1}} = w_5$, pois, $u_{o1} = w_5 * g_{h1} + \boxed{w_6 * g_{h2} + b_2 * 1} = 0,4$



Constante, estamos derivando para g_{h1}

Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial g_{h1}} + \frac{\partial E_{o2}}{\partial g_{h1}}$, não temos essas parcelas :(
- $\frac{\partial E_{o1}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial g_{h1}}$
- $\frac{\partial E_{o1}}{\partial u_{o1}} = \frac{\partial E_{o1}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} = 0,74136507 * 0,186815602 = 0,138498562$
- valores calculados nos slides 38 e 39
- $\frac{\partial u_{o1}}{\partial g_{h1}} = w_5$, pois, $u_{o1} = w_5 * g_{h1} + w_6 * g_{h2} + b_2 * 1 = 0,4$
- $\frac{\partial E_{o1}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial g_{h1}} = 0,138498562 * 0,4 = 0,055399425$

Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- $\frac{\partial E_{total}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial g_{h1}} + \frac{\partial E_{o2}}{\partial g_{h1}}$, não temos essas parcelas :(
- $\frac{\partial E_{o1}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial g_{h1}}$
- $\frac{\partial E_{o1}}{\partial u_{o1}} = \frac{\partial E_{o1}}{\partial g_{o1}} * \frac{\partial g_{o1}}{\partial u_{o1}} = 0,74136507 * 0,186815602 = 0,138498562$
- valores calculados nos slides 38 e 39
- $\frac{\partial u_{o1}}{\partial g_{h1}} = w_5$, pois, $u_{o1} = w_5 * g_{h1} + w_6 * g_{h2} + b_2 * 1 = 0,4$
- $\frac{\partial E_{o1}}{\partial g_{h1}} = \frac{\partial E_{o1}}{\partial u_{o1}} * \frac{\partial u_{o1}}{\partial g_{h1}} = 0,138498562 * 0,4 = 0,055399425$
- Usando os mesmos passos, $\frac{\partial E_{o2}}{\partial g_{h2}} = -0,019049119$
- Logo, $\frac{\partial E_{Total}}{\partial g_{h1}} = 0,055399425 - 0,019049119 = 0,036350306$

Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- Agora falta: $\frac{\partial g_{h1}}{\partial u_{h1}}$, sabemos que: $g_{h1} = \frac{1}{1 + e^{-u_{h1}}}$
- $\frac{\partial g_{h1}}{\partial u_{h1}} = g_{h1} * (1 - g_{h1}) = 0,59326999 * (1 - 0,59326999) =$
 $0,241300709$
- Falta, $\frac{\partial u_{h1}}{\partial w_1}$, sabemos que $u_{h1} = w_1 * x_1 + w_2 * x_2 + b_1 * 1$
- Derivando em relação a w_1 , temos que, $\frac{\partial u_{h1}}{\partial w_1} = x_1 = 0,05$
- Finalmente:
- $\frac{\partial E_{total}}{\partial w_1} = 0,036350306 * 0,241300709 * 0,05 = 0,000438568$
- $w_1(t + 1) = w_1(t) - \eta * \frac{\partial E_{total}}{\partial w_1}$

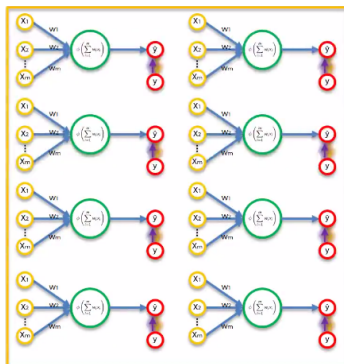
Corrigindo w_1 : queremos estimar o quanto w_1 afeta o erro total (E)

- Atualizando w_1 , $w_1(t+1) = w_1(t) - \eta * \frac{\partial E_{total}}{\partial w_1}$
- $w_2(t+1) = 0,19956143$
- $w_3(t+1) = 0,24975114$
- $w_4(t+1) = 0,29950229$
- Com os pesos sinápticos no instante t , $E = 0,298371109$
- Com os pesos sinápticos no instante $t+1$, $E = 0,291027924$
- Com os pesos sinápticos no instante $t+10000$, $E = 0,000035085$
- Saídas calculadas no instante $t+10000$,
- Para $(y_1 = 0,01)$ temos $0,015912196$
- Para $(y_2 = 0,99)$ temos $0,984065734$

Resumindo:

- $w_6(t+1) = w_6(t) - \delta_{o_1} * g_{h_1}$
- $\delta_{o_1} = -(d_{o_1} - g_{o_1}) * g_{o_1} (1 - g_{o_1})$
- $\delta_{o_2} = -(d_{o_2} - g_{o_2}) * g_{o_2} (1 - g_{o_2})$
- $w_2(t+1) = w_2(t) - \delta_{h_1} * i_1$
- $\delta_{h_1} = (w_5 * \delta_{o_1} + w_2 * \delta_{o_1}) * g_{h_1} * (1 - g_{h_1})$
- $\delta_{h_2} = (w_6 * \delta_{o_1} + w_8 * \delta_{o_2}) * g_{h_2} * (1 - g_{h_2})$

Considerações sobre o cálculo do erro



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \frac{1}{2}(\hat{y} - y)^2$$



Figura: Funcionamento do gradiente descendente.

Considerações sobre o cálculo do erro

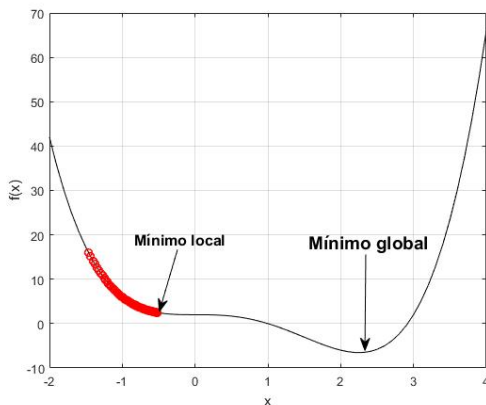


Figura: Funcionamento do gradiente descendente.

Considerações sobre o cálculo do erro

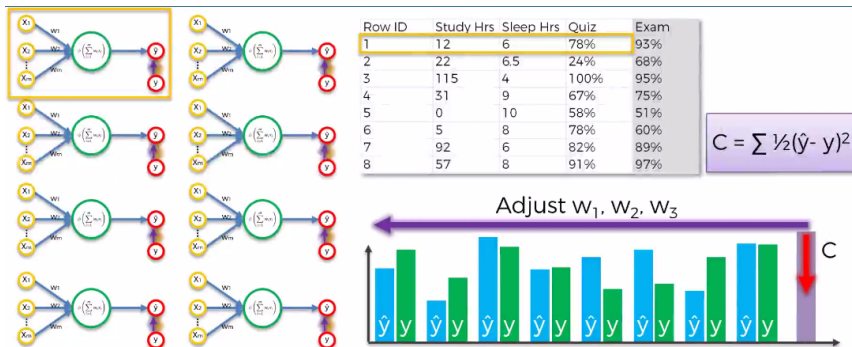
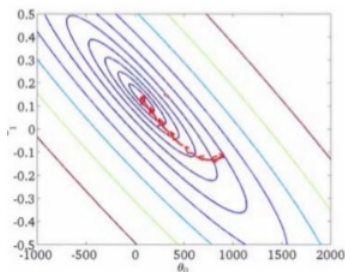


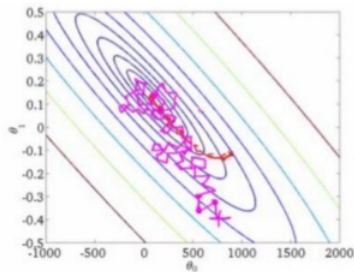
Figura: Funcionamento do gradiente descendente.

Considerações sobre o cálculo do erro



Batch: gradient

$$x \leftarrow x - \eta \nabla F(x)$$



Stochastic: single-example gradient

$$x \leftarrow x - \eta \nabla F_i(x)$$

Figura: Comparação gradiente descendente (Full batch) e estocástico (mini-batch).

Considerações sobre o cálculo do erro

Gradiente descendente: pode exigir muita memória - impraticável para datasets grandes.

SGD: na prática pode ser entendido como gradiente descendente “ruído” porém lento. A escolha é um trade-off difícil.

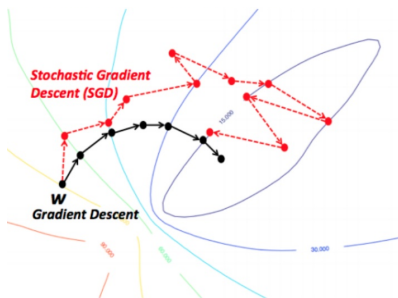


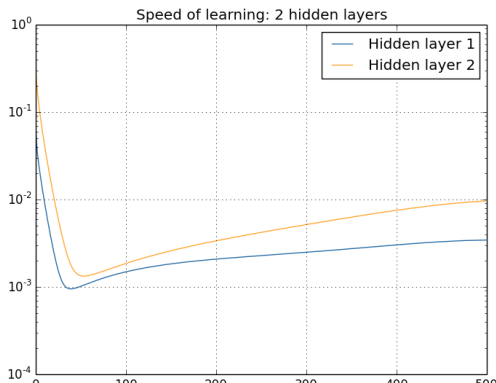
Figura: Comparação gradiente descendente (Full batch) e estocástico (mini-batch).

Considerações - Poder expressivo

- “Two layer network with sigmoid units in the hidden layer and (unthresholded) linear units in the output layer - Any bounded continuous function.” (Cybenko 1989, Hornik et. al. 1989)
- “A network of three layers, where the output layer again has linear units - Any function.” (Cybenko 1988).
- Então podemos concluir que sigmóides empilhadas são a última fronteira no aprendizado supervisionado? Certo?

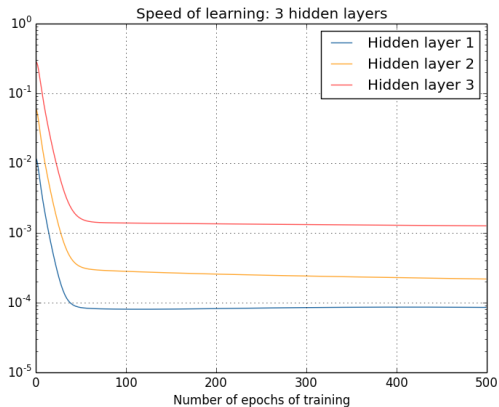
Considerações sobre a quantidade de neurônio e camadas

Gradiente diminui exponencialmente em relação ao número de camadas.



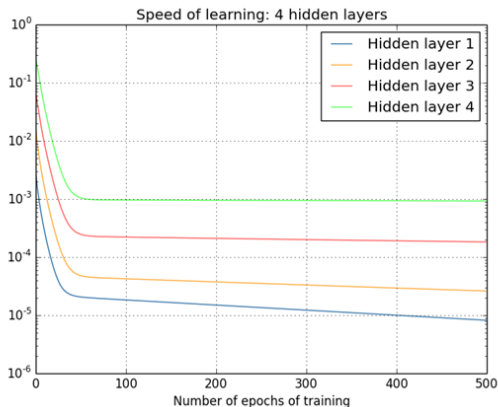
Considerações - Problemas

Gradiente diminui exponencialmente em relação ao número de camadas.



Considerações - Problemas

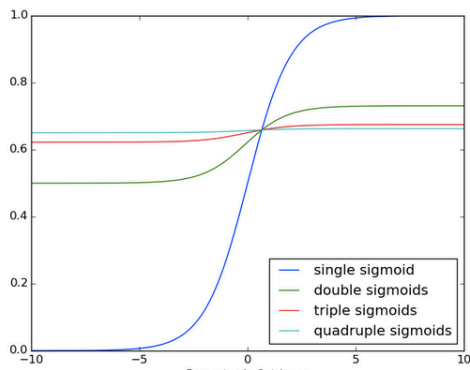
Gradiente diminui exponencialmente em relação ao número de camadas.



Considerações - Problemas

Gradiente diminui exponencialmente em relação ao número de camadas.

Efeitos da aplicação de uma função sigmoide em cascata. Não há inclinação detectável.



Considerações - Playground tensorflow

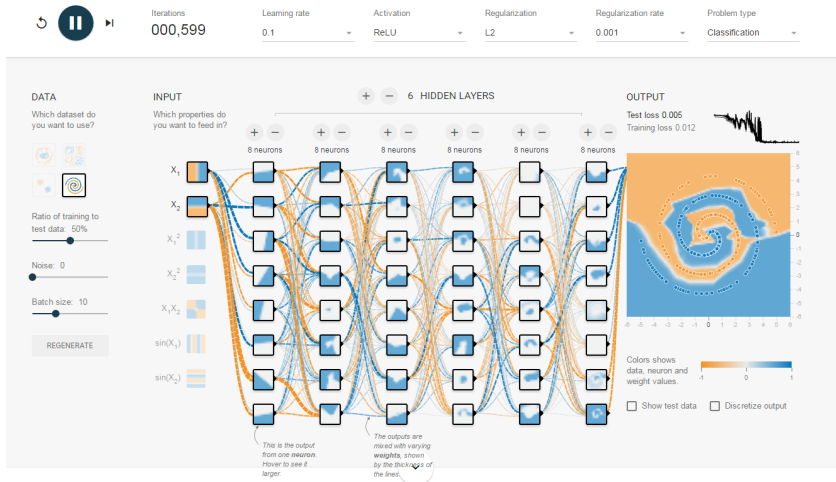


Figura: <http://playground.tensorflow.org>

Considerações - TensorBoard

TensorBoard é um software de visualização gráfica incluído em qualquer instalação TensorFlow.

Nas palavras do time da Google: “The computations you’ll use TensorFlow for - like training a massive deep neural network - can be complex and confusing. To make it easier to understand, debug, and optimize TensorFlow programs, we’ve included a suite of visualization tools called TensorBoard.”

Ferramentas úteis

Meu primeiro tensorboard

```
1 # meuprimeiortensorboard.py
2 import tensorflow as tf
3 a = tf.constant(2)
4 b = tf.constant(3)
5 x = tf.add(a, b)
6 with tf.Session() as sess:
7     print sess.run(x)
8
9 writer = tf.summary.FileWriter('./graphs', sess.graph)
10 writer.close
```

Explicando...

A última linha cria um objeto que grava as operações em um log.

```
1 $ python [meuprimeiortensorboard.py]
2 $ tensorboard --logdir="./graphs"
```


Ferramentas úteis

Abra seu navegador em <http://localhost:6006/>

The screenshot displays the TensorBoard web interface. The top navigation bar is orange and contains the following tabs: SCALARS, IMAGES, AUDIO, GRAPHS, DISTRIBUTIONS, and HISTOGRAMS. The SCALARS tab is currently selected. Below the navigation bar, there is a search bar with the placeholder text "Write a regex to create a tag group" and a close button (X). Below the search bar, there are two checkboxes: "Split on underscores" and "Data download links", both of which are unchecked. Below these checkboxes, there is a dropdown menu for "Tooltip sorting method" with the value "default" selected. Below the dropdown menu, there is a "Smoothing" slider with a value of 0.6. Below the slider, there is a "Horizontal Axis" section with three buttons: "STEP", "RELATIVE", and "WALL". The "STEP" button is currently selected. Below the "Horizontal Axis" section, there is a "Runs" section with a search bar and a "TOGGLE ALL RUNS" button. The search bar contains the text "my_graph". To the right of the TensorBoard interface, there is a diagram of the "Add" operation. The diagram shows two input nodes labeled "Const" and "Const_1" connected by arrows to a single output node labeled "Add".

Conclusão

Backpropagation é um método que nos permite atualizar os pesos de uma rede neural multicamadas. Foi possível demonstrar que a regra da cadeia combinada com a gradiente descendente é capaz de otimizar a função objetivo da rede.

Por hoje é só...

