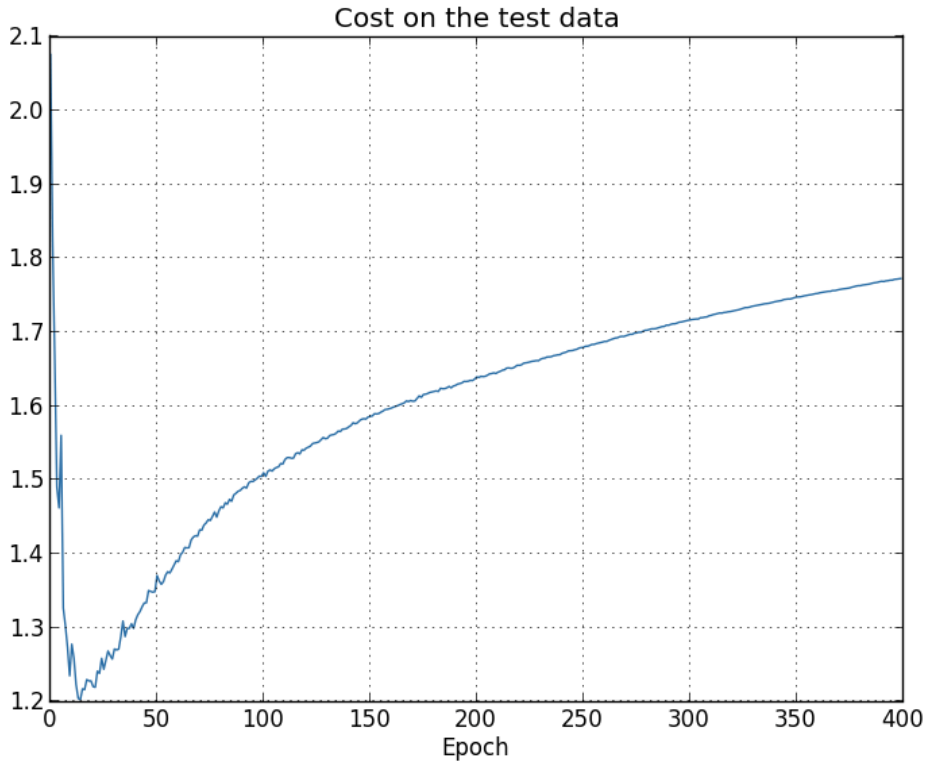
OTIMIZAÇÃO DE REDES NEURAIS ARTIFICIAIS

Prof. Valmir Macário Filho

DC - UFRPE

OVERFITTING

- Acontece quando o modelo está "super adaptado" aos dados. Para de melhorar na avaliação do conjunto de teste.

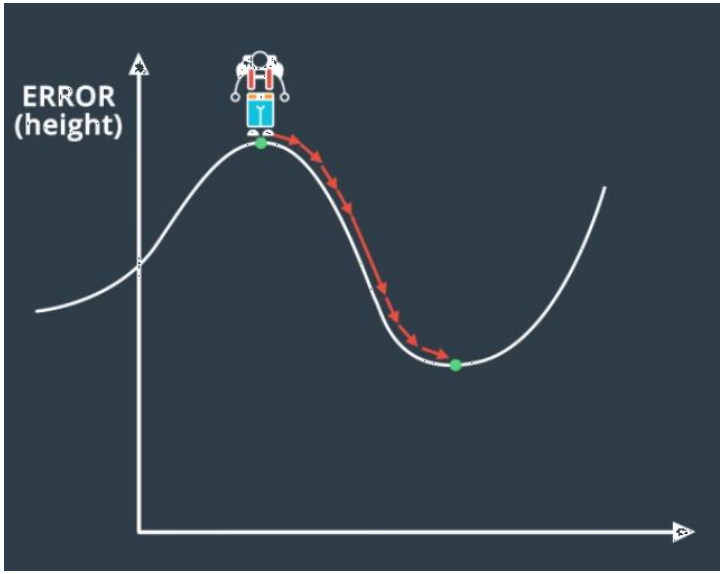


COMO EVITAR OVERFITTING

- Existem algumas técnicas para evitar o overfitting
  - Mudar forma de atualizar pesos para Lotes
  - Reinicio aleatório
  - Aumentar o conjunto de treinamento
  - Utilizar conjunto de validação
  - Técnicas de regularização
  - Dropout
  - Mudança na função de ativação
  - Momento

ATUALIZAÇÃO DOS PESOS

- Cada passo em direção ao mínimo é chamado de época
- Cada época do treinamento representa uma apresentação de todos os dados de treinamento



ATUALIZAÇÃO LOCAL

- A atualização dos pesos é realizada após a apresentação de cada amostra do treinamento de forma aleatória ou estocástica.
- Uma época representa a apresentação de todas amostras do treinamento
- Como as amostras são apresentadas aleatoriamente, o uso da atualização padrão por padrão torna a busca no espaço de conexões estocástica por natureza, reduzindo a possibilidade do algoritmo ficar preso em um mínimo local.

ATUALIZAÇÃO EM LOTES (BATCH)

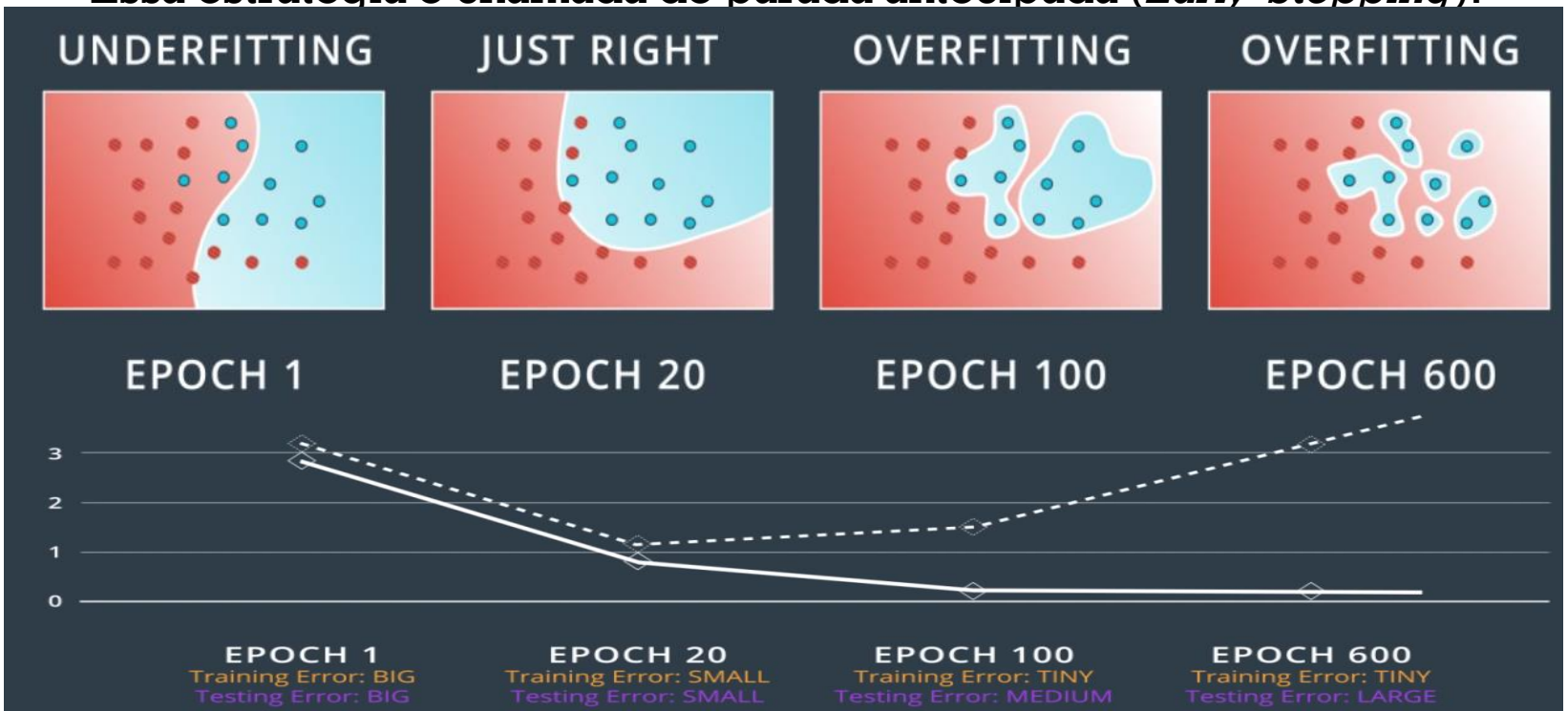
- No treinamento em lotes, os dados são divididos em partes igualmente distribuídas (lotes).
- Cada atualização é realizada quando um lote é processado
- A utilização do método em lote fornece uma estimativa mais precisa do vetor gradiente.
- É usualmente mais utilizada que a atualização local
- A escolha do tamanho do lote é um parâmetro que afetará a velocidade do treinamento
- É usual escolher um múltiplo de 2: 64, 128, 256 e 512

REINICIO ALEATÓRIO

- Reiniciar o treinamento utilizando pesos escolhidos aleatoriamente
- Aumenta as chances de encontrar o mínimo global ou um bom mínimo local
- Uma abordagem comum de inicialização dos pesos é utilizar:
  - Distribuição gaussiana com média 0 e desvio padrão 1
  - Multiplicar o valore gerado por $\sqrt{\frac{2}{n_i}}$, onde n_i é o número nós de entrada na camada considerada.

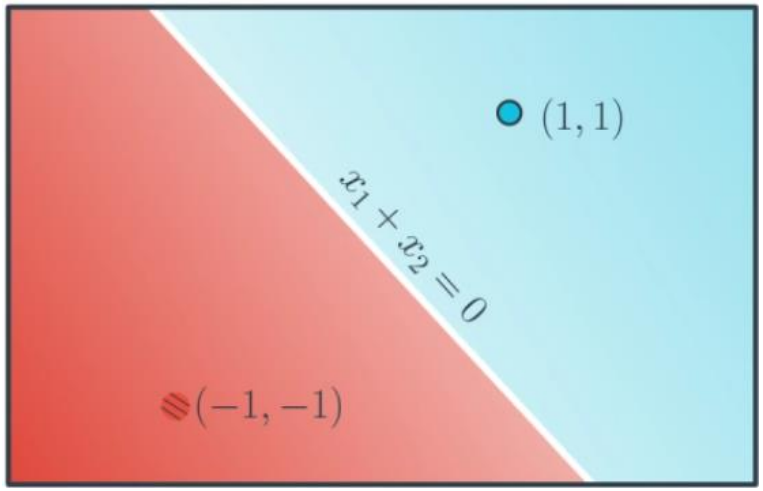
CONJUNTO DE VALIDAÇÃO

- Usar conjunto de validação:
  - Calculamos a precisão da classificação nos dados de validação no final de cada época.
  - Quando a precisão da classificação nos dados de validação estiver saturada, paramos de treinar.
  - Essa estratégia é chamada de parada antecipada (*Early-Stopping*).



REGULARIZAÇÃO

- Qual a melhor solução?



Predição: z=f(net), onde

net = $w_1 x_1 + w_2 x_2 + b$

- **Solução 1**: $x_1 + x_2$

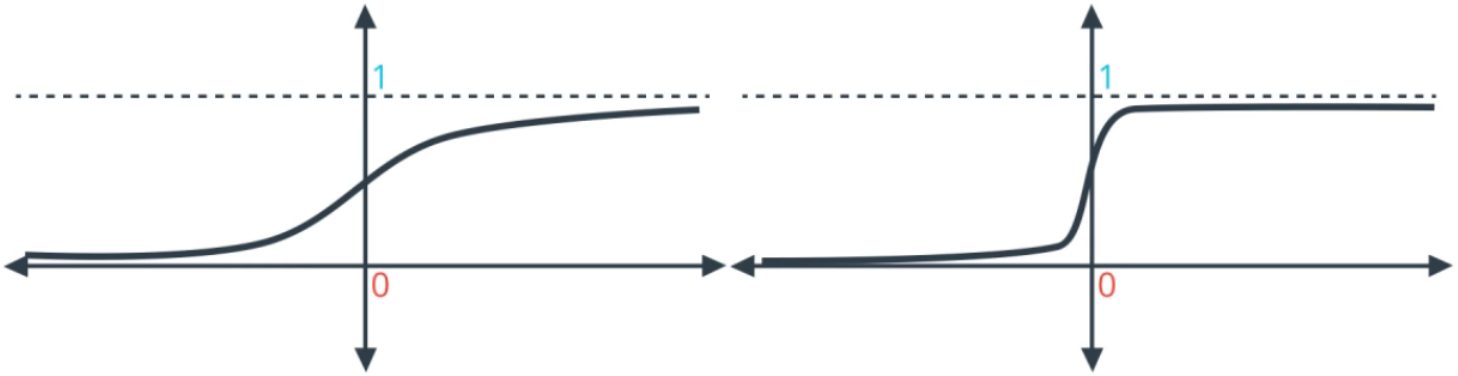
  Predição usando sigmóide:
  f(1+1)=0.88
  f(-1-1)=0.12

- **Solução 2:** $10x_1 + 10x_2$

  Predição usando sigmóide:
  f(10+10)=0.9999999979
  f(-10-10)=0.0000000021

REGULARIZAÇÃO

- Sigmóides geradas a partir de cada um dos modelos:



$x_1 + x_2$ $\quad$ $10x_1 + 10x_2$

- O primeiro modelo é aplicado sob valores pequenos, então produzem um ângulo melhor para a função do gradiente descendente
- O segundo modelo produz respostas melhores, porém é mais ajustados aos dados, pode produzir overfitting

REGULARIZAÇÃO

- A ideia da Regularização é adicionar um termo extra à função de custo, um termo chamado termo de regularização.

- Intuitivamente, o efeito da regularização é fazer com que a rede prefira aprender pesos pequenos, e penaliza os demais os tornando iguais

- Pesos grandes só serão permitidos se melhorarem consideravelmente a primeira parte da função de custo.

- Dito de outra forma, a regularização pode ser vista como uma forma de se comprometer entre encontrar pequenos pesos e minimizar a função de custo original.

REGULARIZAÇÃO L1

- Custo da entropia cruzada com regularização L1

$$\mathcal{J}(w) = -\frac{1}{M}\sum_{j=1}^{M} y^j \, ln\left(h_w\left(x^j\right)\right) + \left(1 - y^j\right)\ln\left(1 - h_w\left(x^j\right)\right) + \frac{\lambda}{2M}\sum_w |w|$$

- Custo quadrático com regularização L1

$$\mathcal{J}(w) = -\frac{1}{2M}\sum_{j=1}^{M}\left(y^j - h_w\left(x^j\right)\right)^2 + \frac{\lambda}{2M}\sum_w |w|$$

- Quando λ é pequeno, é minimizado a função de custo original, mas quando λ é grande, preferimos pesos pequenos.

REGULARIZAÇÃO L2

- Custo da entropia cruzada com regularização L2

$$\mathcal{J}(w) = -\frac{1}{M}\sum_{j=1}^{M} y^j \, ln\left(h_w\left(x^j\right)\right) + \left(1 - y^j\right)\ln\left(1 - h_w\left(x^j\right)\right) + \frac{\lambda}{2M}\sum_w w^2$$

- Custo quadrático com regularização L2

$$\mathcal{J}(w) = -\frac{1}{2M}\sum_{j=1}^{M}\left(y^j - h_w\left(x^j\right)\right)^2 + \frac{\lambda}{2M}\sum_w w^2$$

- Quando λ é pequeno, preferimos minimizar a função de custo original, mas quando λ é grande, preferimos pesos pequenos.

REGULARIZAÇÃO L1 VS L2

L1

- Esparsidade (1,0,0,1,0)
- Transforma valores pequenos em zeros
- Pode-se reduzir o número de pesos
- Bom para seleção de características

L2

- Esparsidade: (0.5,0.3,-0.2,0.4,0.1)
- Tenta manter os pesos pequenos
- Normalmente produz resultados melhores em modelos de treinamento.
- Mais usado

REGULARIZAÇÃO L1 VS L2

- $\mathcal{J}(w) = -\frac{1}{2}\sum(y_j - z_j)^2 + Penalidade$
- Derivada em relação ao peso $\frac{\partial \mathcal{J}(w)}{\partial w_{ji}}$
- **Atualização dos pesos com L1**
  - $\Delta w_{ji}(t) = \eta\left[x_i * (z_j - y_j)z_j(1 - z_j)\right] + \lambda$
- **Atualização dos pesos com L2**
  - $\Delta w_{ji}(t) = \eta\left[x_i * (z_j - y_j)z_j(1 - z_j)\right] + \lambda w_{ji}$

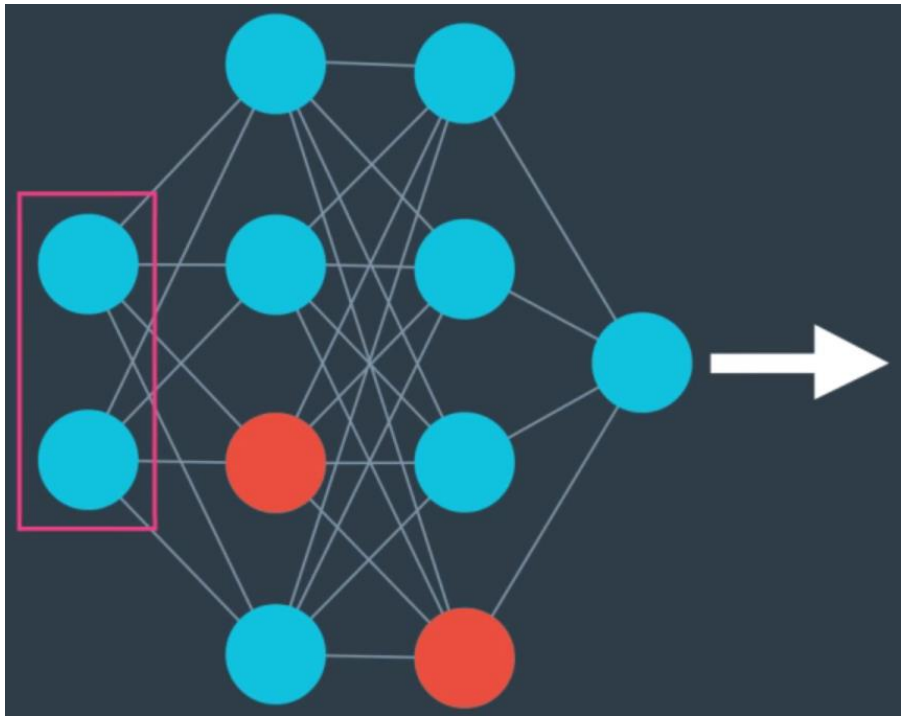
DROPOUT

- No processo de treinamento, muitas vezes temos áreas da rede neural que possui mais peso que outras áreas.
- Também existem áreas que não são muito utilizadas e não são treinadas.
- Seria útil então avaliar estruturas de redes neurais artificiais diferentes
- O dropout é uma técnica utilizada para "desligar" alguns neurônios durante o treinamento.
- É equivalente a treinar redes neurais diferentes.
- O procedimento de remoção de nós é como calcular a média dos efeitos de um grande número de redes diferentes

DROPOUT

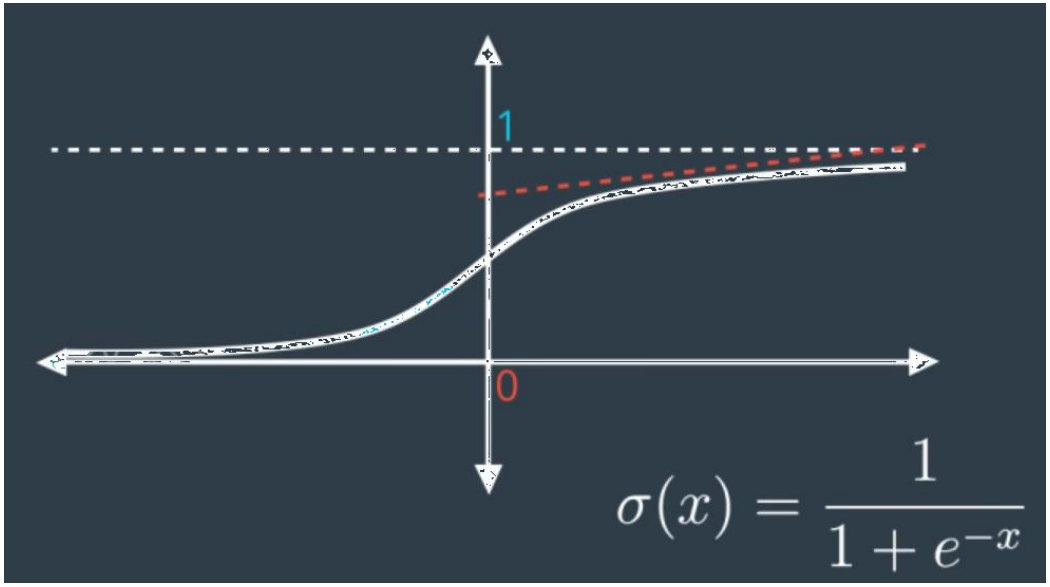
- É dada uma probabilidade para cada nó ser desligado no treinamento em cada época (foward+backpropagation).

- Nesse exemplo, os nós em vermelho são deligados



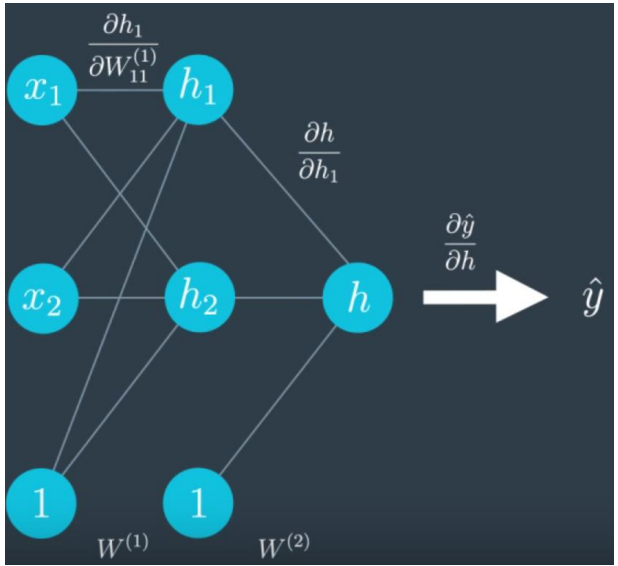
DISSIPAÇÃO DO GRADIENTE

- A curva se achata nas laterais
- Se calcular a derivadas, tanto na esquerda quanto na direita, as derivadas serão valores muito pequenos ou quase zero.
- Não é bom, pois a derivada indica a direção a seguir



DISSIPAÇÃO DO GRADIENTE

- Correção do erro através da derivada da regra da cadeia da função sigmóide
- O resultado é o produto de números pequenos, então esse número pode ser muito pequeno. (ver exemplo da aula anterior)
- Isso pode levar a atualizações muito pequenas nos pesos, na medida que a arquitetura da rede neural cresce



$$\frac{\partial E}{\partial W_{11}^{(1)}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial h_1} \frac{\partial h_1}{\partial W_{11}^{(1)}}$$

muito pequeno

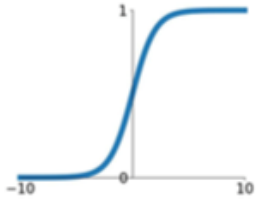
pequeno

SOLUÇÃO PARA DISSIPIÇÃO DO GRADIENTE

- Trocar a função de ativação

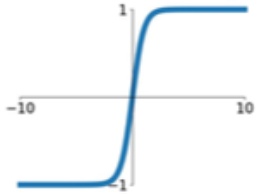
Sigmoid

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



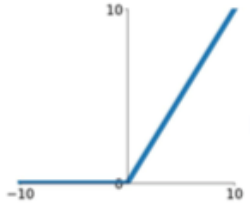
tanh

$$\tanh(x)$$



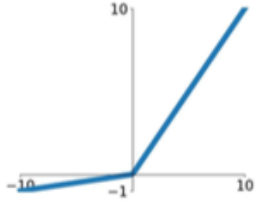
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

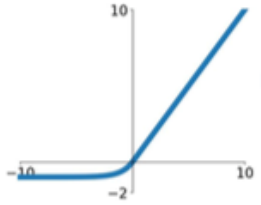


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

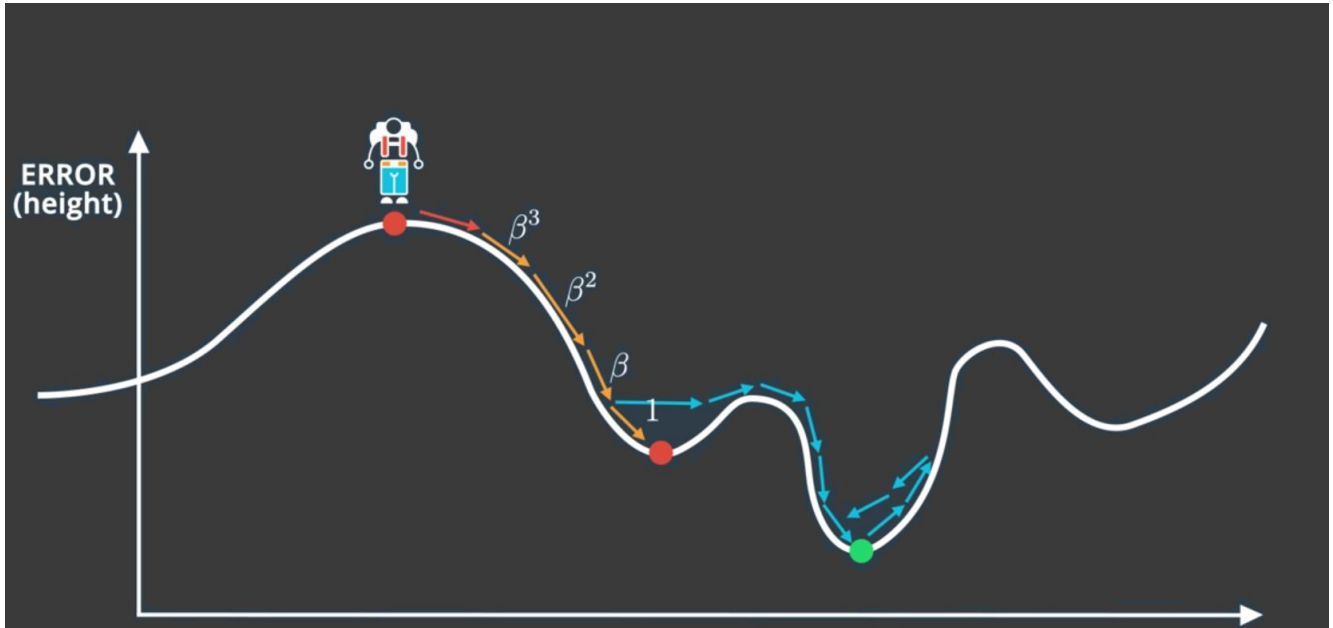


HISTÓRICO DAS FUNÇÕES DE ATIVAÇÃO

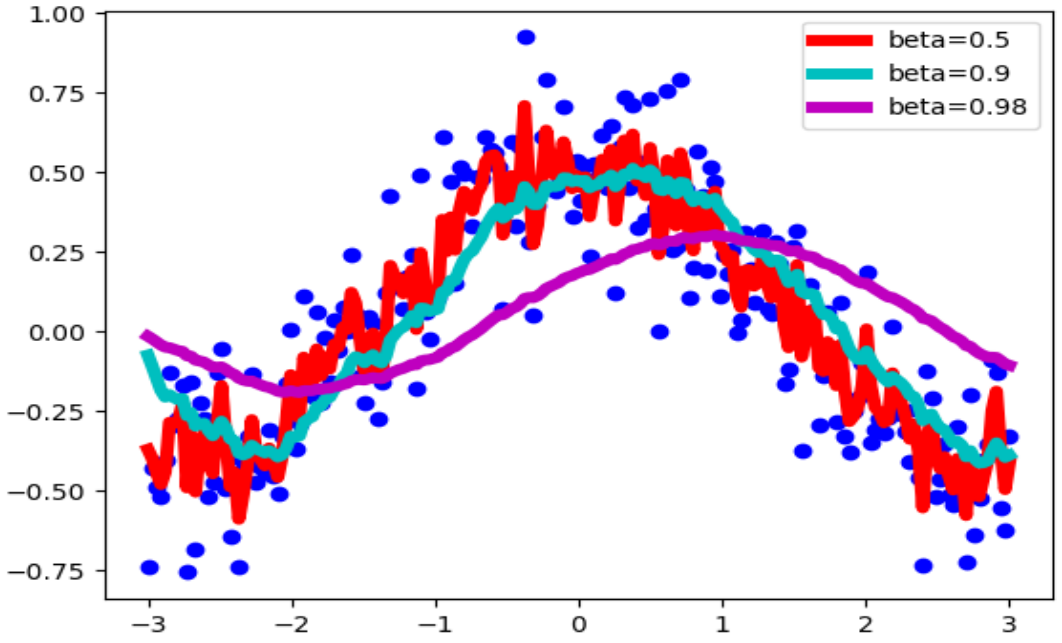
Name	Formula	Year
none	$y = x$	-
sigmoid	$y = \frac{1}{1+e^{-x}}$	1986
tanh	$y = \frac{e^{2x}-1}{e^{2x}+1}$	1986
ReLU	$y = \max(x, 0)$	2010
(centered) SoftPlus	$y = \ln(e^x + 1) - \ln 2$	2011
LReLU	$y = \max(x, \alpha x)$, $\alpha \approx 0.01$	2011
maxout	$y = \max(W_1 x + b_1, W_2 x + b_2)$	2013
APL	$y = \max(x,0) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$	2014
VLReLU	$y = \max(x, \alpha x)$, $\alpha \in 0.1, 0.5$	2014
RReLU	$y = \max(x, \alpha x)$, $\alpha = \text{random}(0.1, 0.5)$	2015
PReLU	$y = \max(x, \alpha x)$, α is learnable	2015
ELU	$y = x$, if $x \geq 0$, else $\alpha(e^x - 1)$	2015

MOMENTO

- O momento acumula pesos anteriores para estabilizar a convergência da rede
- Ajuda a desviar de mínimos locais
- Pode acelerar treinamento em regiões muito planas da superfície de erro



MOMENTO



- **Equações:**
  - $v_t = \beta v_{t-1} + \eta \frac{\partial \mathcal{J}(w)}{\partial w_{ji}}$
  - $w_{ji}(t) = w_{ji}(t-1) - v_t$

  onde $(0 < \beta < 1)$ é a constante de momento
- Normalmente, β é ajustada entre 0,5 e 0,9
- $\frac{\partial \mathcal{J}(w)}{\partial w_{ji}}$, essa derivada depende de qual otimizador e função de ativação está sendo utilizada. É a equação de atualização dos pesos.

MOMENTO

- Exemplo com $\beta = 0.9$ e 3 parcelas:
  - $v_1 = \frac{\partial \mathcal{J}(w)}{\partial w_1}$
  - $v_2 = 0.9 * \frac{\partial \mathcal{J}(w)}{\partial w_2} + \frac{\partial \mathcal{J}(w)}{\partial w_1}$
  - $v_3 = 0.9 * \left(0.9 * \frac{\partial \mathcal{J}(w)}{\partial w_3} + \frac{\partial \mathcal{J}(w)}{\partial w_2}\right) + \frac{\partial \mathcal{J}(w)}{\partial w_1} = 0.81 * \frac{\partial \mathcal{J}(w)}{\partial w_3} + 0.9 * \frac{\partial \mathcal{J}(w)}{\partial w_2} + \frac{\partial \mathcal{J}(w)}{\partial w_1}$

ROOT MEAN SQUARE PROPOGATION (RMSPROP)

- RMSProp foi proposto por Geoffrey Hilton em uma de suas aulas
- Assim como o momento, tenta diminuir oscilações na convergência da Rede Neural.
- RMSprop propõe uma taxa de aprendizado pra cada atualização de peso.

ROOT MEAN SQUARE PROPOGATION (RMSPROP)

- **Equações:**
  - $g_{t,i} = \frac{\partial \mathcal{J}(w_t)}{\partial w_{ti}}$
  - $v_t = \rho v_{t-1} + (1 - \rho) * {g_t}^2$
  - $w_{ji}(t) = w_{ji}(t-1) - \frac{\eta}{\sqrt{v_t + \varepsilon}} * g_t$

Onde:

- $\frac{\partial \mathcal{J}(w_t)}{\partial w_{ti}}$ derivada em relação ao peso i no tempo t
- ρ é um parâmetro com valor sugerido é 0.9
- ε é utilizado pra evitar que tenha divisão por zero e o valor é muito baixo.

ADAPTIVE MOMENT ESTIMATION (ADAM)

- É outro método que calcula os pesos de forma adaptativa
- Guarda além dos valores dos pesos passados, o decaimento exponencial da média de gradientes passados m_t
- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) * g_t$
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) * {g_t}^2$

- m_t e v_t são estimativas do primeiro momento (a média) e do segundo momento (a variância não centralizada) dos gradientes
- β_1 e β_2 são valores próximos de 1

ADAPTIVE MOMENT ESTIMATION (ADAM)

- Para evitar tendência de ter valores próximos de zero nos primeiros momentos, são calculadas estimativas de primeiro e segundo momento corrigidas por viés
- $\widehat{m_t} = \frac{m_t}{1-\beta_1}$
- $\widehat{v_t} = \frac{v_t}{1-\beta_2}$
- $w_{ji}(t) = w_{ji}(t-1) - \frac{\eta}{\sqrt{\widehat{v_t}+\varepsilon}} * \widehat{m_t}$

- Os autores propõe valores 0.9 para β_1, 0.999 para β_2 e 10^{-8} para ε

REFERÊNCIAS

- Ruder S. An overview of gradient descent optimization algorithms. arXiv, 2017.

- http://ruder.io/optimizing-gradient-descent/index.html#rmsprop