UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Deep Perceptrons

Gisele L Pappa

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Feedforward Deep Network

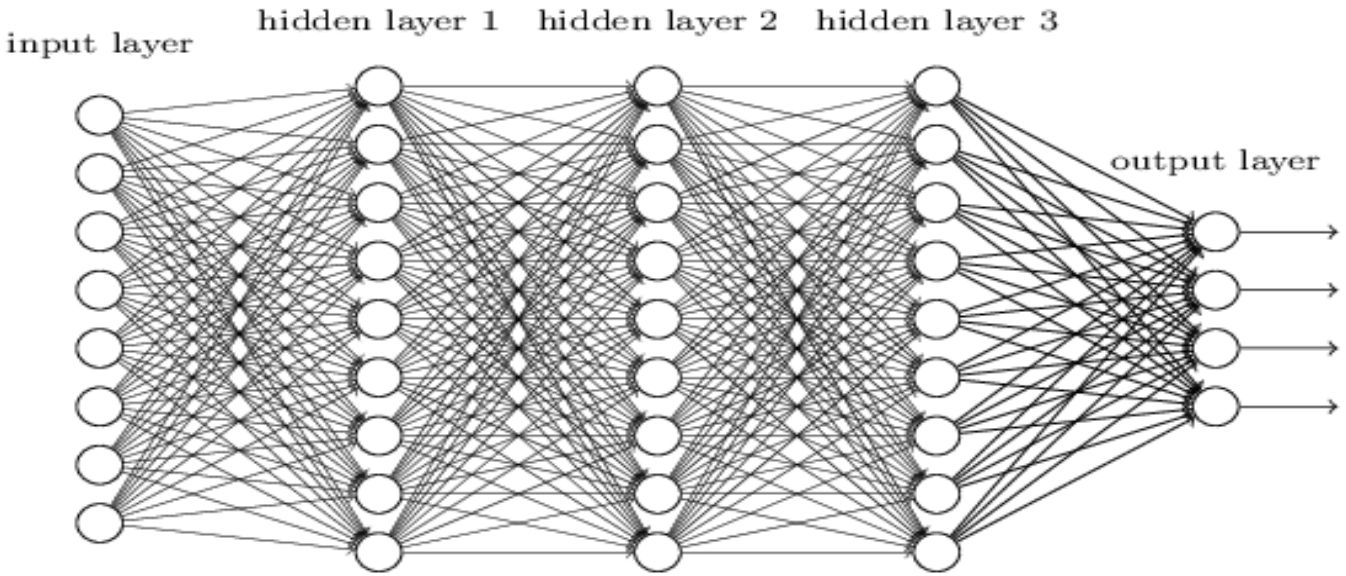
UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

- Os *deep perceptrons*, ou *feedforward deep networks*, são similares a redes perceptron de múltiplas camadas (MLPs) mas, por definição, têm pelo menos 3 camadas escondidas
- Em relação ao MLP clássico temos:
  - Redes maiores treinadas com mais exemplos
  - Função de ativação das camadas escondidas muda devido ao *"vanishing gradient problem"*
  - Pequenas modificações algorítmicas no BP

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Deep Feedforward Networks

- Mesma estrutura das redes MLP clássicas



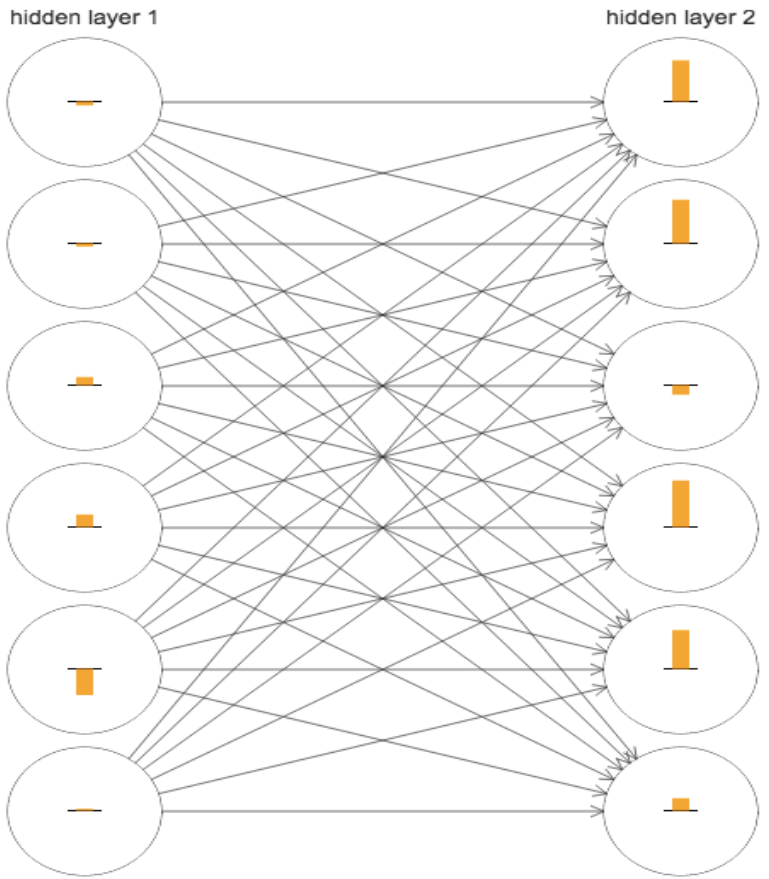
O papel de mais camadas escondidas

- A ideia de adicionar camadas escondidas é que elas consigam extrair representações dos dados em diferentes níveis de abstração
  - Intuitivamente, adicionar uma camada deveria reduzir o erro da rede, ou pelo menos manter o erro
- Problema: se considerarmos a MLP convencional vista anteriormente, erro pode aumentar ao se adicionar mais camadas escondidas
  - Por que?

Velocidade de Aprendizado

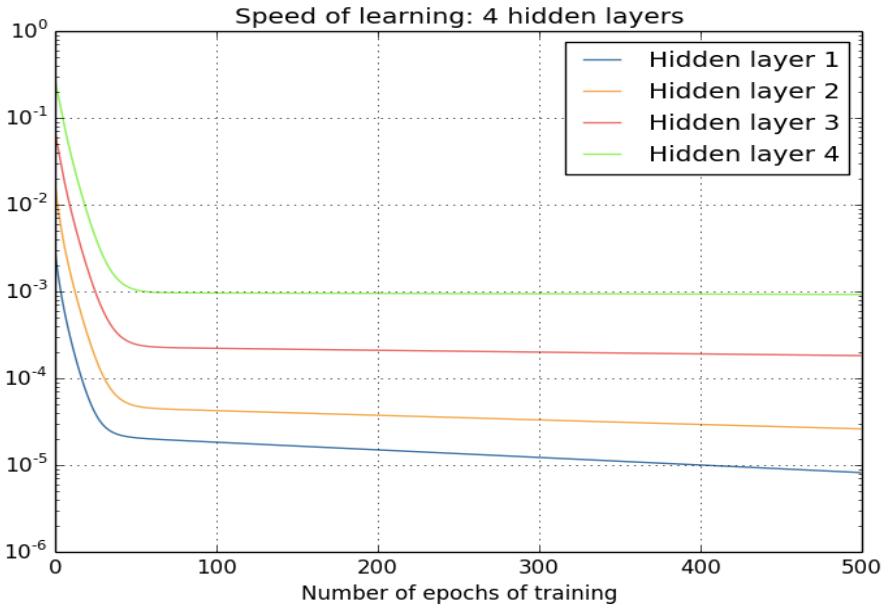
UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

- A figura mostra a velocidade com que cada neurônio das camadas escondidas de uma rede (com pesos inicializados aleatoriamente) atualiza seus pesos
- Quanto maior a barra, mais rápido a rede muda os valores de pesos e bias
- A barra corresponde ao gradiente da função de erro
- Observe que a segunda camada aprende mais rápido



Velocidade de aprendizado

- Primeira camada aprende 100x mais lentamente que a última camada.
- Por que isso acontece? O gradiente tende a diminuir quando se move pra trás nas camadas ocultas



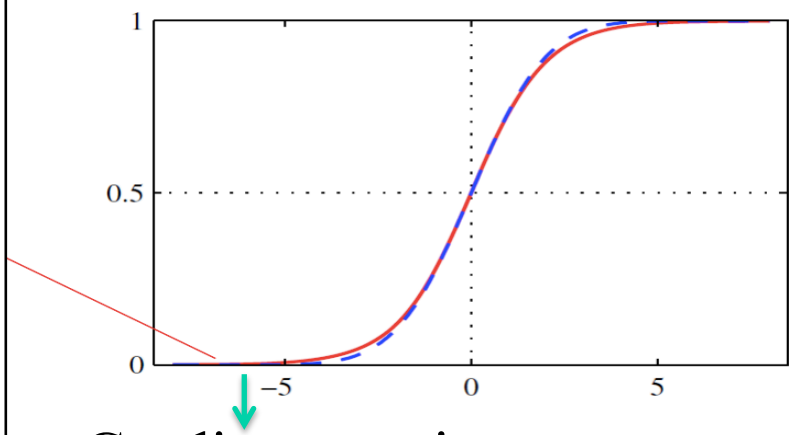
- Esse problema é conhecido como *vanishing gradient*

Referência: http://www.deeplearningbook.org/

Vanishing gradient problem

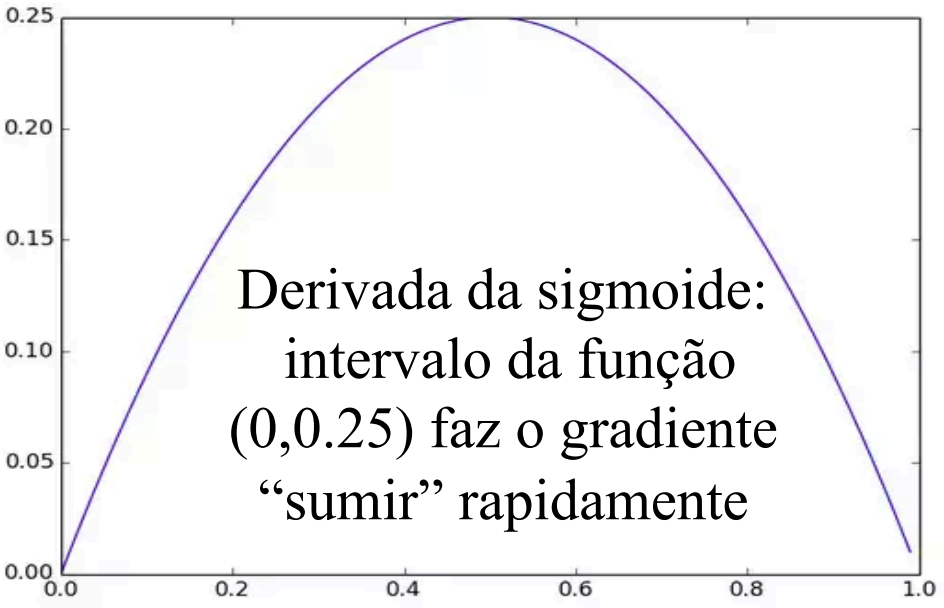
- O gradiente em redes profundas é instável, podendo sumir ou explodir nas primeiras camadas
- Quando a função de ativação usada é uma sigmoide, ela obrigatoriamente mapeia todas as entradas em um intervalo entre 0 e 1
- Para entradas muito próximas a 0 ou 1, o gradiente estará próximo de 0, fazendo com que a rede não aprenda

Vanishing gradient problem



Gradiente muito pequeno

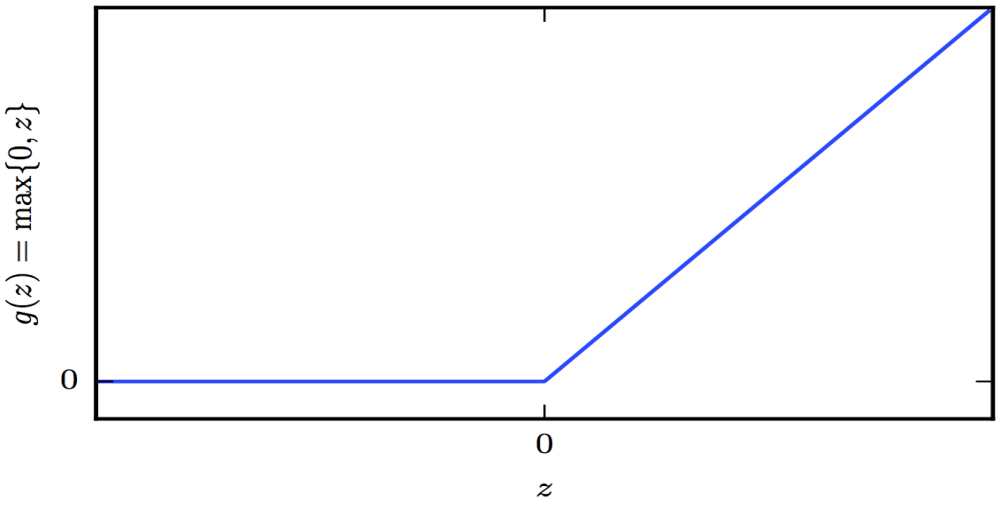
Função sigmoide



Derivada da sigmoide: intervalo da função (0,0.25) faz o gradiente "sumir" rapidamente

Vanishing gradient problem

- Pode ser contornado trocando a função de ativação dos neurônios
- Função mais popular: ReLU (Rectified linear unit)

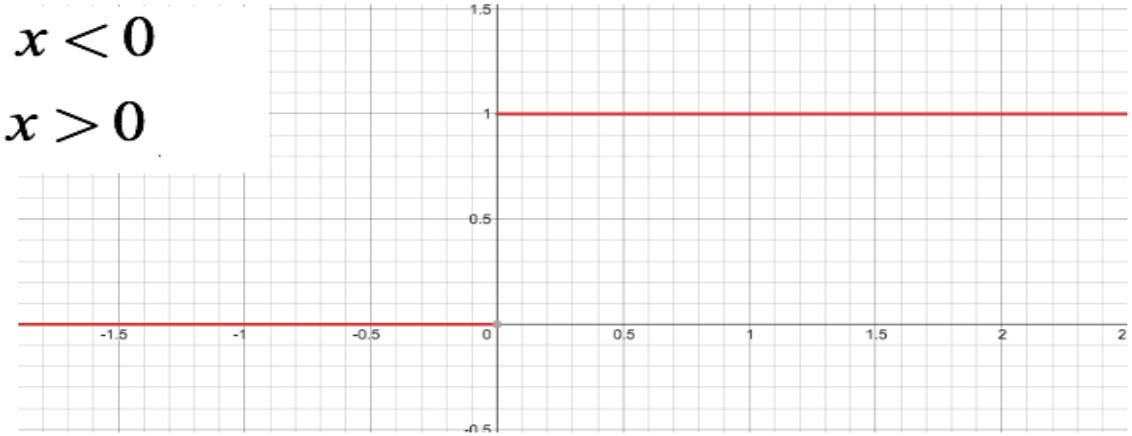


$$RELU(x) = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x >= 0 \end{cases}$$

ReLU

- Função próxima de uma função linear
  - Preserva propriedades interessantes de funções lineares que fazem com que a otimização com descida do gradiente seja fácil
- Derivada da ReLU é uma função degrau, não diferenciável em 0

$$\frac{d}{dx} RELU(x) = \begin{cases} 0 & if \quad x < 0 \\ 1 & if \quad x > 0 \end{cases}$$

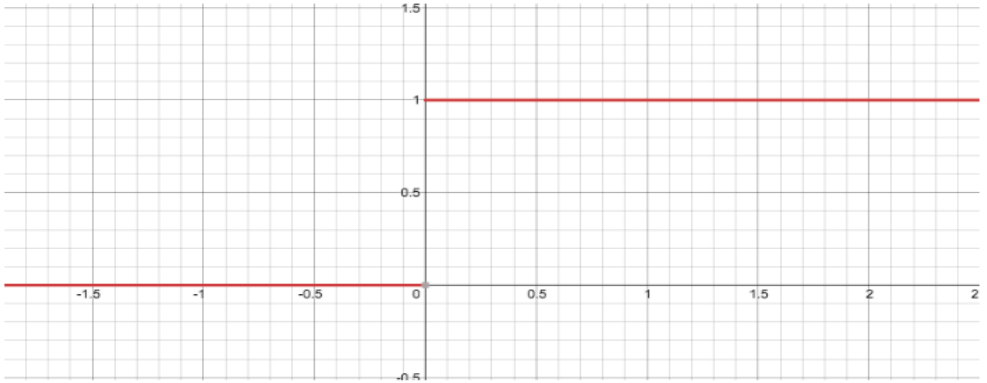


ReLU

- Agora as derivadas não estão mais em um intervalo de 0 a 1, e não vão sumir.
- Porém, ReLUs tem saída zero quando a entrada é negativa
  - Isso pode bloquear o backpropagation porque os gradientes passam a ser 0

    depois de uma

    entrada negativa



Generalizations of ReLU

- Esse problema de valores negativos podem ser resolvidos por generalizações do ReLU, como o Leaky-ReLU ou Parametric-ReLU

$$gRELU(x) = \max\{x,0\} + \alpha \min\{x,0\}$$

ReLU

- Leaky-RELU(x) = max{x,0} + 0.01 min{x,0}
- Parametric-ReLU(x) = max{x,0} + α min{x,0}, onde o valor de α é aprendido junto com os outros parâmetros da rede.

Função de custo

- O algoritmo de back-propagation é baseado na otimização de uma função de erro através do método da descida do gradiente
- Nas aulas de MLP, a função de erro utilizada foi o erro quadrático médio
- A velocidade de treinamento da rede depende do valor do gradiente da função de erro
  - Se esse valor for muito pequeno, a rede demora pra ser treinada

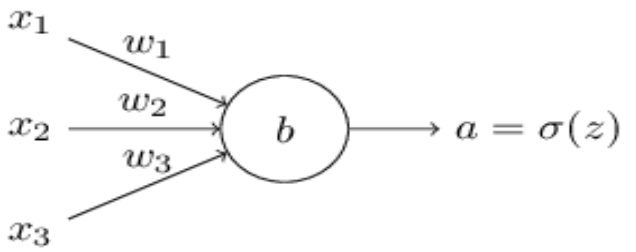
Função de custo

- Pode-se melhorar a velocidade de treinamento simplesmente mudando a função de erro
- Funções utilizadas atualmente são baseadas no princípio de máxima verossimilhança
  - Método para estimar os parâmetros de um modelo estatístico
  - *Cross-entropy (negative log-likelihood)* entre os dados de treinamento e as saídas do modelo.

Funções de custo

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

- Cross-entropy



$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)] ,$$

$$z = \sum_j w_j x_j + b$$

N é o número de exemplos de treinamento
y é a saída esperada

- Calculando a derivada parcial da cross-entropy com relação aos pesos da rede, tem-se:

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j (\sigma(z) - y)$$

A taxa com que os pesos são aprendidos é controlada pelo erro da saída
Quanto mais alto o erro, mais rápido o aprendizado.

Função de custo

- Funções de custo também consideram um fator de regularização
- Regularização é uma técnica utilizada para tentar resolver o problema de overfitting.
- Uma forma tradicional de regularização é inserir um termo de penalização na função de custo
  - A intuição é que modelos com pesos menores são mais simples que modelos com pesos maiores
  - Essas penalizações mantêm os valores dos pesos baixos ou zero

Regularização

- Termo regularizador
  - L2 (ou weight decay): encoraja pesos com valores pequenos
  - Função de custo de *cross-entropy* regularizada

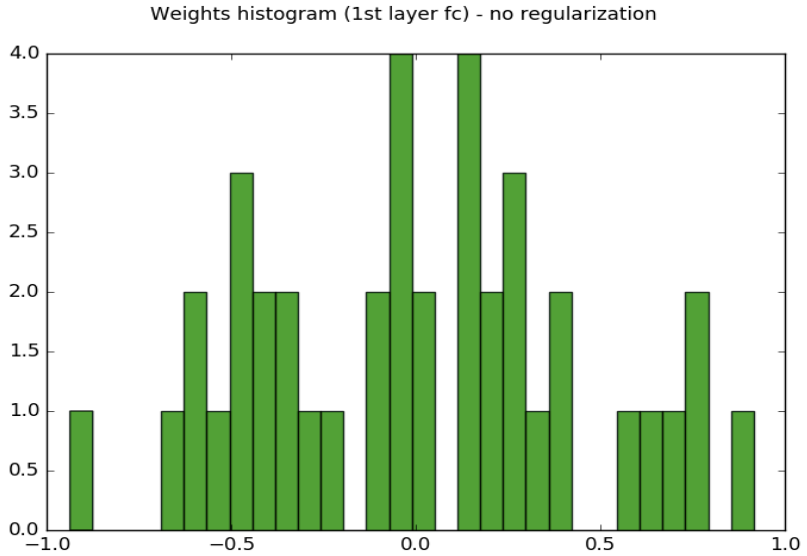
$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] + \frac{\lambda}{2n}\sum_{w} w^2.$$

Termo regularizador

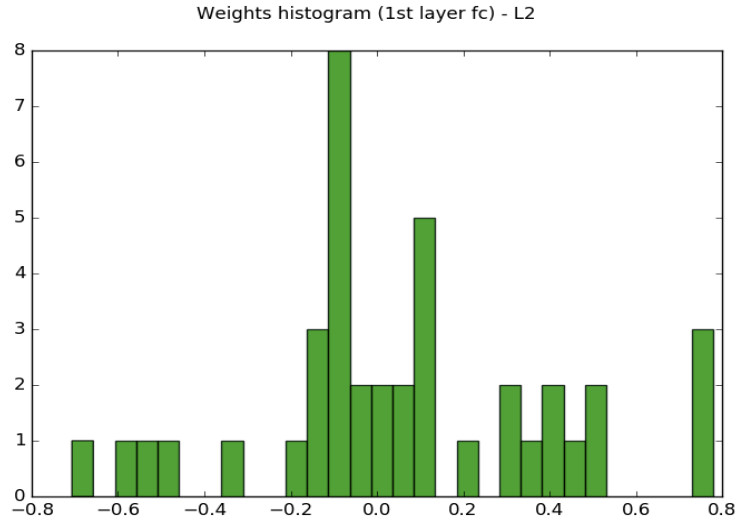
  - Termo de regularização não inclui o bias
  - λ é um parâmetro de regularização

Regularização

Sem regularização



Norma L2



- Histograma de distribuição dos pesos da rede sem/com regularização

Outras técnicas de Regularização

- Dataset augmentation
  - Aumentar o número de exemplos do dataset
- Parar o processo de aprendizado antes do overfitting
- Dropout
  - Usa um ensemble de redes
  - Para cada passo de atualização de pesos:
    - Amostre aleatoriamente uma máscara para todos os neurônios de entrada e camada escondida
    - Multiplique a máscara com o neurônio e atualize o peso normalmente

Função de ativação da camada de saída

- Está intimamente ligada a função de erro sendo otimizada

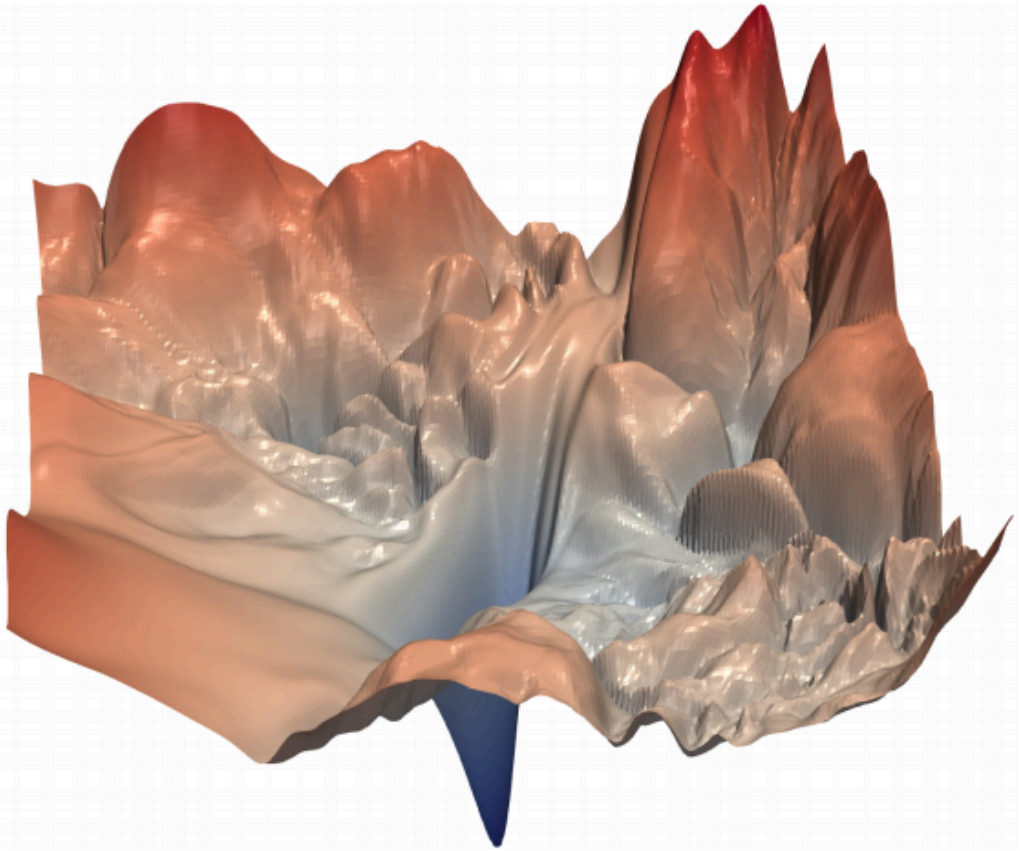
Função de ativação da camada de saída

Output Type	Output Distribution	Output Layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary cross-entropy
Discrete	Multinoulli	Softmax	Discrete cross-entropy
Continuous	Gaussian	Linear	Gaussian cross-entropy (MSE)
Continuous	Mixture of Gaussian	Mixture Density	Cross-entropy
Continuous	Arbitrary	See part III: GAN, VAE, FVBN	Various

Referência: http://www.deeplearningbook.org/

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Treinar uma rede é uma tarefa difícil



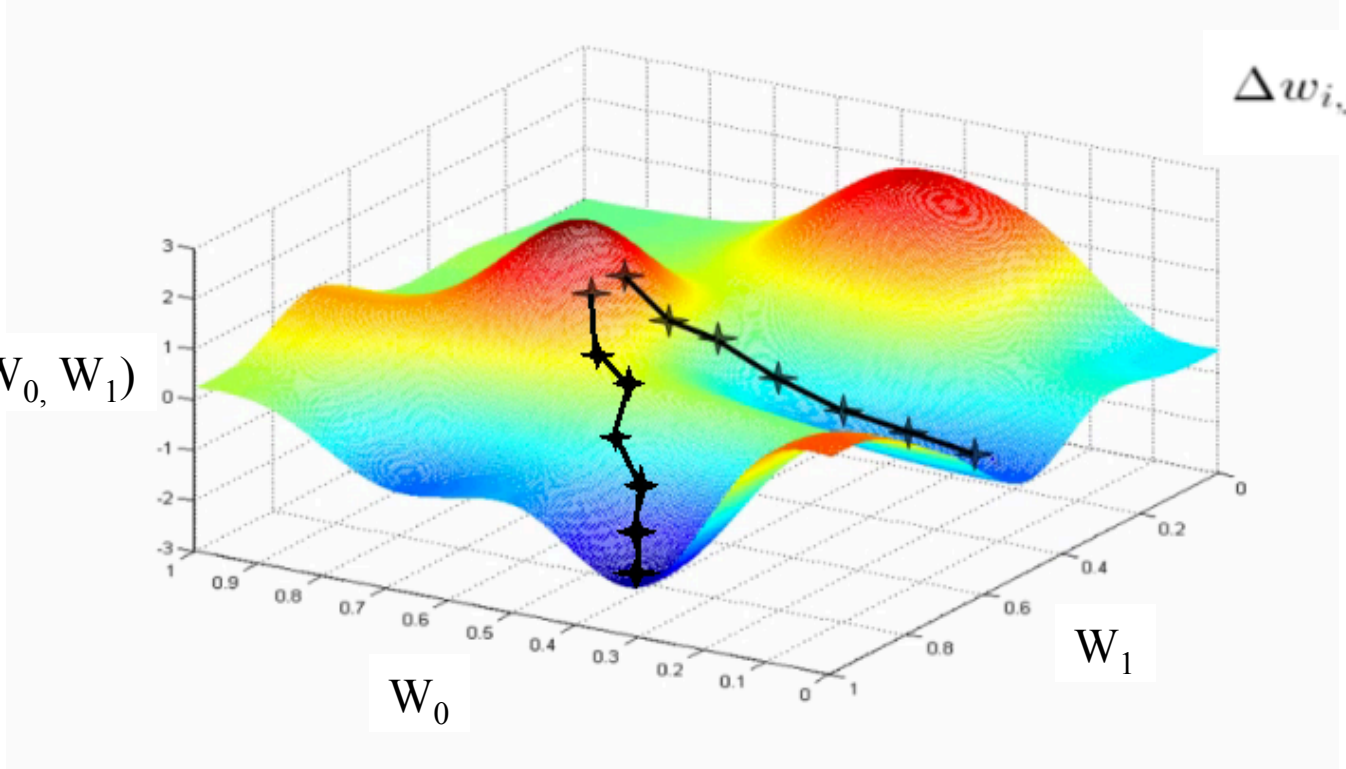
DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Source: https://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Descida do Gradiente

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$



https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/

Learning rate

- Pode ser ajustada:
  - Manualmente
  - De forma adaptativa
    - Muda de acordo com o processo de otimização em si
- Backprop com Adam (*Adaptive Moment Estimation*)
  - Utiliza uma taxa de aprendizado adaptativa

Sumário

- MLP versus Deep Perceptrons
  - Uso da ReLU como função de ativação
  - Funções de perda baseadas na máxima verossimilhança
    - Entropia cruzada
  - Regularização
    - Considerada na própria função de erro
    - Dropout
  - Taxas de aprendizado auto-adaptativas (Adam)

Bibliografia

- 2 livros disponíveis online:
  - http://www.deeplearningbook.org/
  - http://neuralnetworksanddeeplearning.com/index.html
- For optimizers, see https://mlfromscratch.com/optimizers-explained/#/