

Введение в АД

Лекция 7

Deep Learning

ФЭФМ МФТИ
Весенний семестр 2023



Так что такое нейронная сеть?

Представим, что у нас был вектор $\mathbf{x}_0 = (1, 2, 3)$

Применим к нему некоторое параметризованное преобразование $\mathbf{f}(\mathbf{a})$, $(1, 2, 3) \rightarrow (0.2, 0.3, 0.1, 0.8) = \mathbf{x}_1$

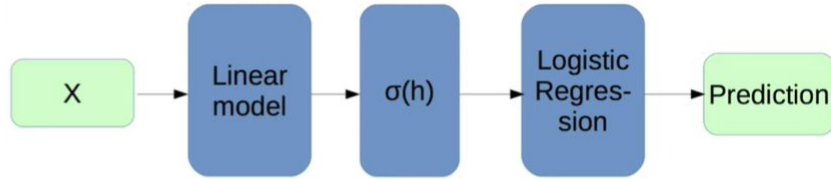
А к \mathbf{x}_1 применим $\mathbf{g}(\mathbf{b})$, $(0.2, 0.3, 0.1, 0.8) \rightarrow (1, 1, 1, 2) = \mathbf{y}$

Ура, мы сделали нейронную сеть! Только непонятно, что она делает.

Нейронная сеть это композиция некоторых преобразований, например линейных в нейронах и нелинейных на выходе из слоев (об этом далее): $NN = f(\text{sigmoid}(g(x_0)))$, например



Нелинейность

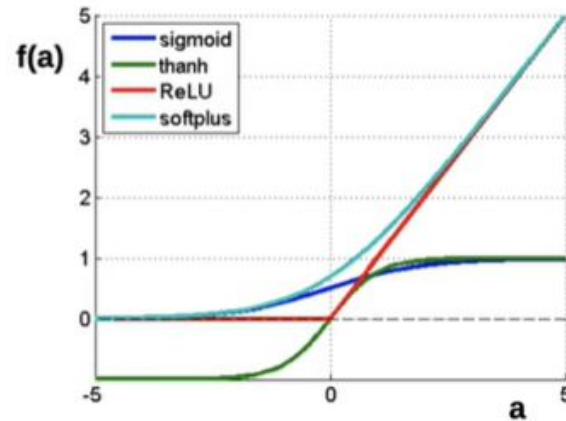


$$f(a) = \frac{1}{1 + e^a}$$

$$f(a) = \tanh(a)$$

$$f(a) = \max(0, a)$$

$$f(a) = \log(1 + e^a)$$





Термины

- Layer/слой – набор нейронов, который принимает на вход вектор размера N , применяет к нему некоторое преобразование и дает на выходе вектор размера M
- Функция активации – функция, которая применяется к вектору, полученному на выходе слоя i перед его передачей в слой $i+1$
- Backpropagation/обратное распространение ошибки – красивое название для дифференцирования сложной функции



A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probablistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

Perceptron (P)



Feed Forward (FF)



Radial Basis Network (RBF)



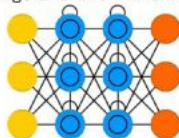
Deep Feed Forward (DFF)



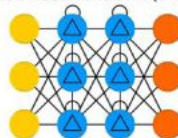
Recurrent Neural Network (RNN)



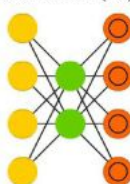
Long / Short Term Memory (LSTM)



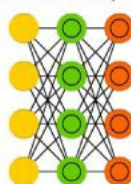
Gated Recurrent Unit (GRU)



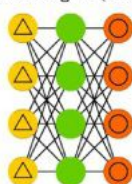
Auto Encoder (AE)



Variational AE (VAE)



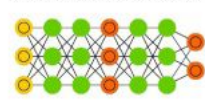
Denosing AE (DAE)



Sparse AE (SAE)



Generative Adversarial Network (GAN)



Liquid State Machine (LSM)



Extreme Learning Machine (ELM)



Echo State Network (ESN)



Deep Residual Network (DRN)



Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)



Markov Chain (MC)



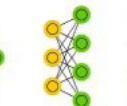
Hopfield Network (HN)



Boltzmann Machine (BM)



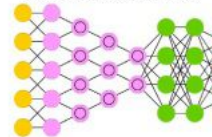
Restricted BM (RBM)



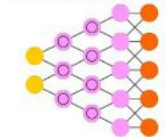
Deep Belief Network (DBN)



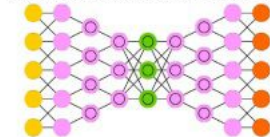
Deep Convolutional Network (DCN)



Deconvolutional Network (DN)



Deep Convolutional Inverse Graphics Network (DCIGN)



ResNet



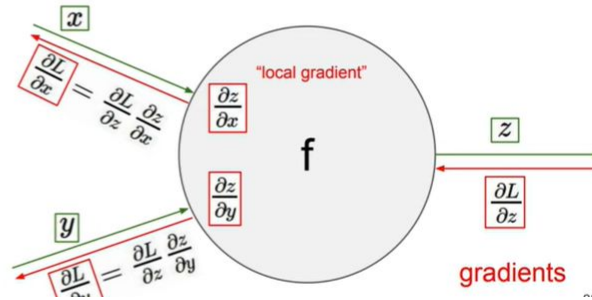
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



Backpropagation

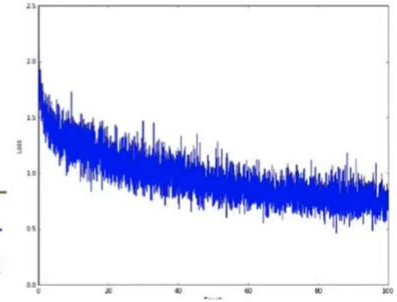
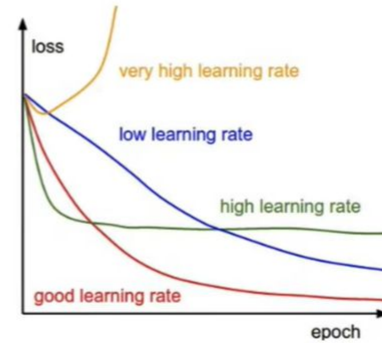
Chain rule is just simple math: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}$

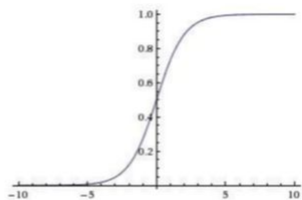
Backprop is just way to use it in NN training.



Stochastic gradient descent (and variations) is used to optimize NN parameters.

$$x_{t+1} = x_t - \text{learning rate} \cdot dx$$





Sigmoid

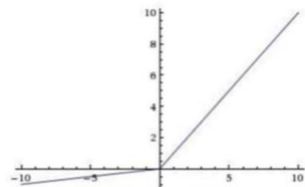
$$f(a) = \frac{1}{1 + e^a}$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. $\exp()$ is a bit compute expensive

playground.tensorflow.org

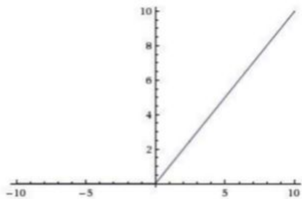


- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not “die”.**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)



ReLU

(Rectified Linear Unit)

$$f(a) = \max(0, a)$$

- Not zero-centered output
- An annoyance:

hint: what is the gradient when $x < 0$?