DL intro

April 2022

Hello!

I am Maria Tikhonova

Graduated from Mech-Math
MSU & YSDA

Work in R’n’D NLP team in SBER DEVICES

Teach in ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

About course

Syllabus

1. Introduction to DL and backpropagation
2. Deep learning libraries
3. Convolutional neural networks
4. Modern architectures of convolutional neural networks
5. Metric learning
6. Autoencoders and generative models
7. Introduction to text processing
8. Recurrent neural networks, seq2seq tasks
9. Attention mechanisms and transformers
10. Basics of sound processing
11. Recommendation systems
12. Graph models
13. Reinforcement Learning
14. Additional deep Learning topics

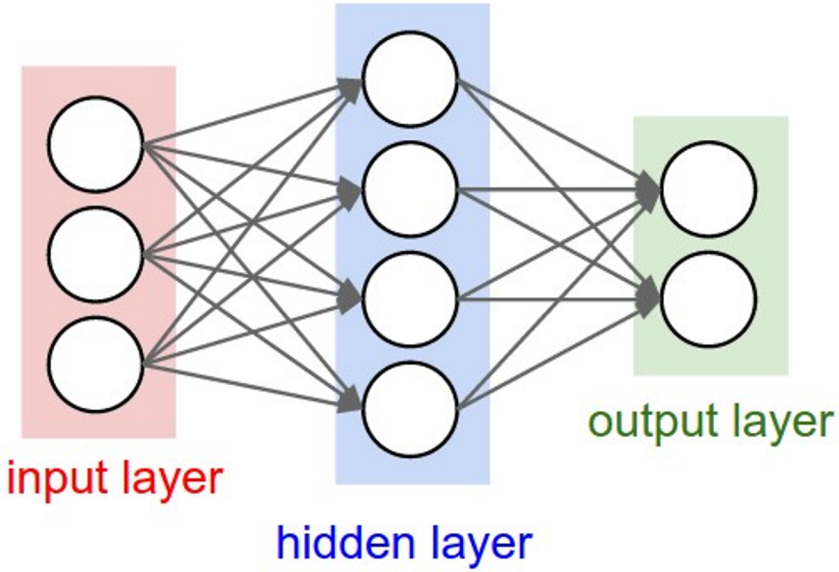
Grading

0.3 * hw_1 + 0.3 * hw_2 + 0.3 * hw_3 + 0.1 * exam

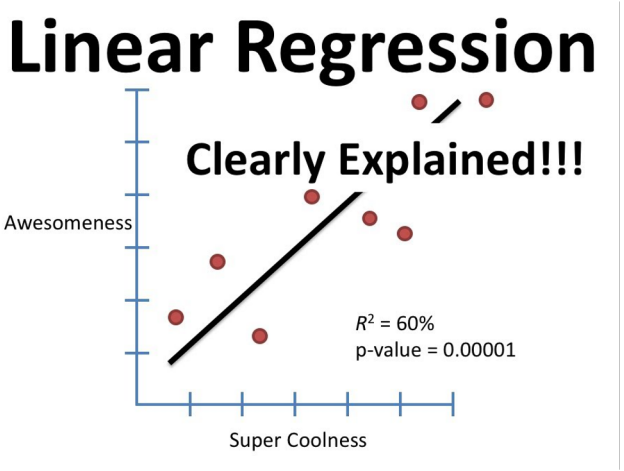
Github

https://github.com/pet67/hse_mlds_deep_learning_course

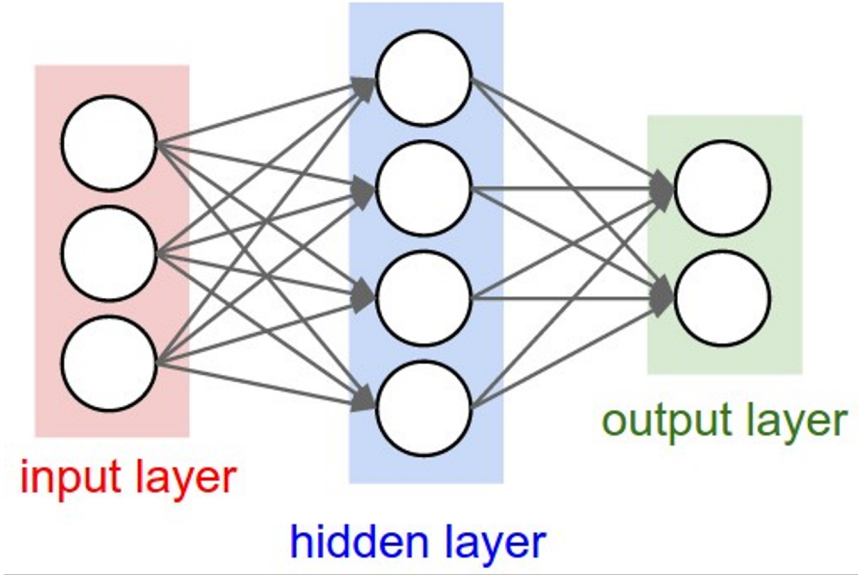
tldr: simple **deep** NN



LR vs 1-FC layer net



vs.

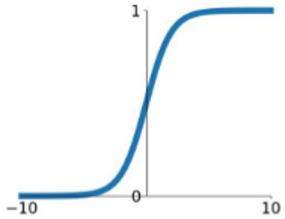


Which is better? Why?

Activation functions

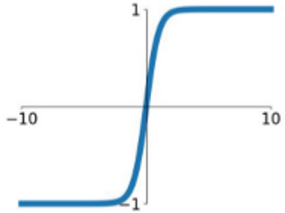
Sigmoid

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



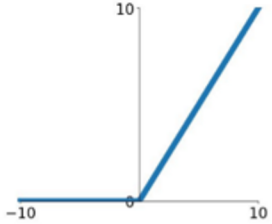
tanh

$$\tanh(x)$$



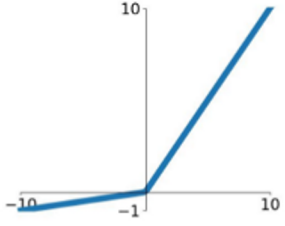
ReLU

$$\max(0, x)$$



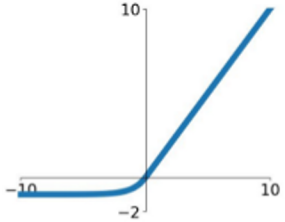
Leaky ReLU

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

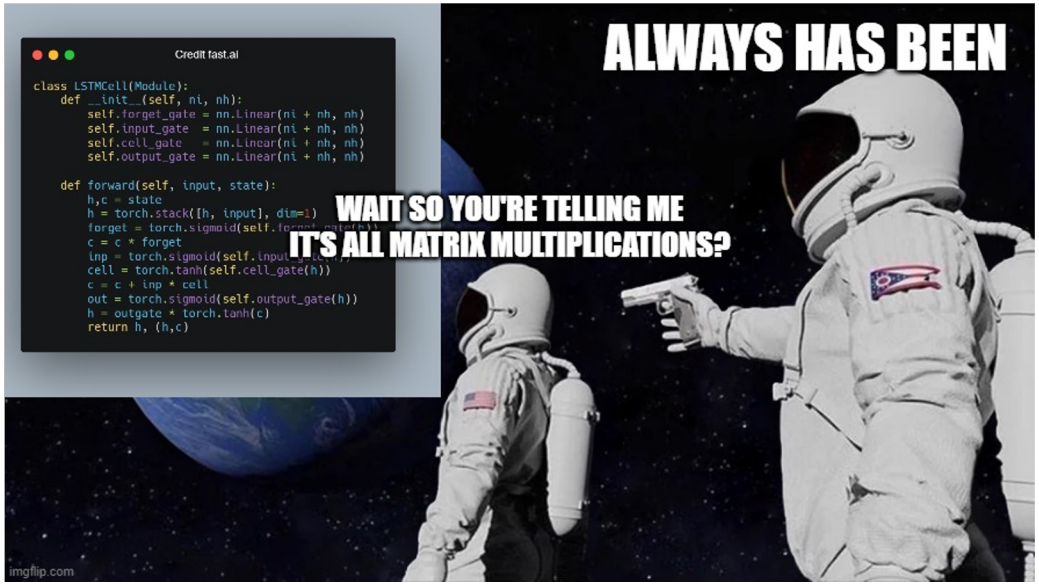


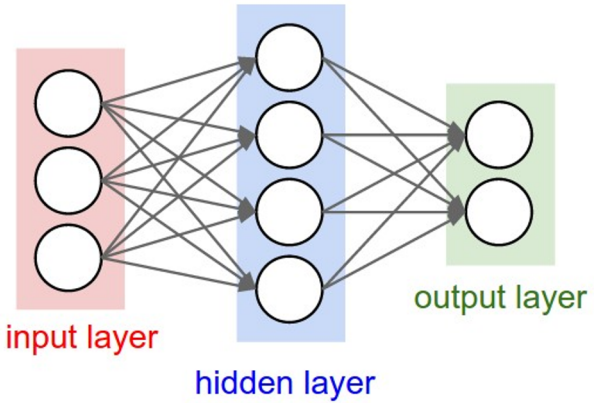
ELU

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



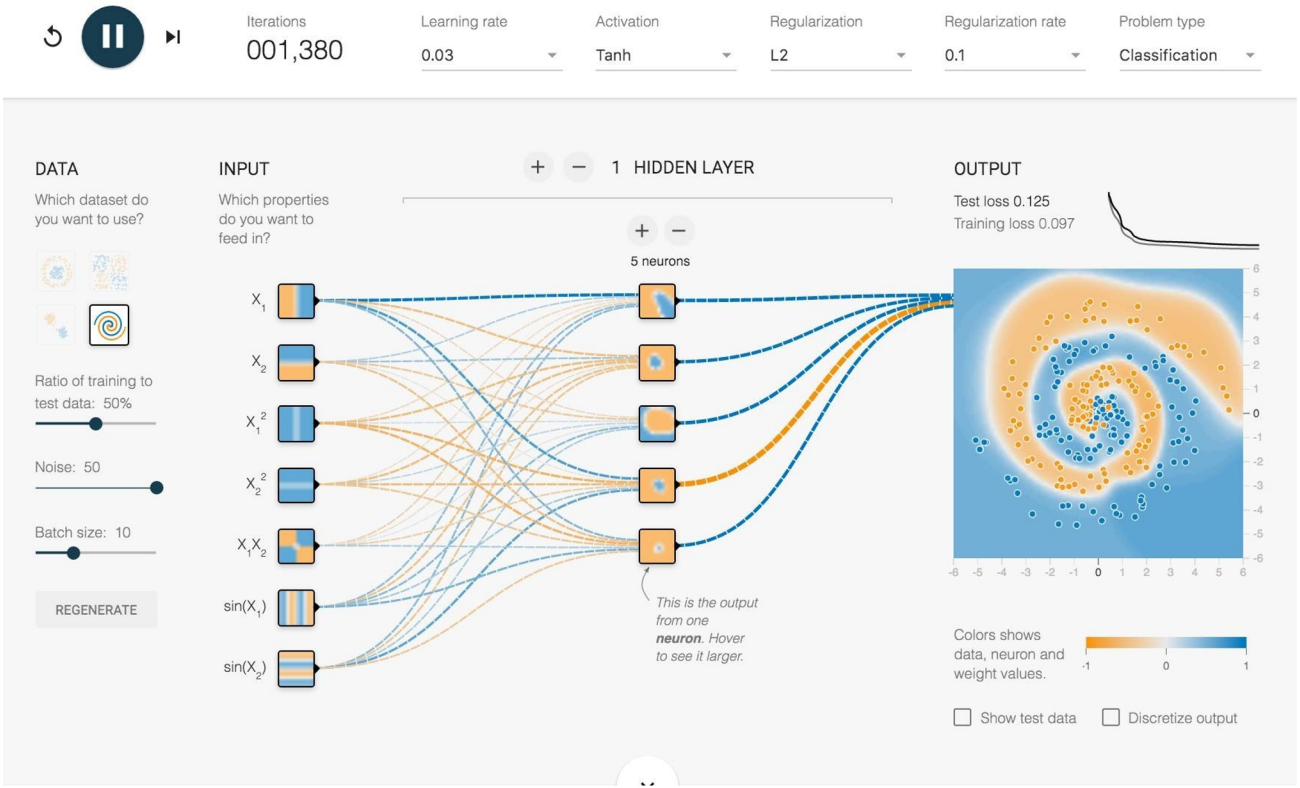
FC layer matrix form





$$Y = X * W + b$$

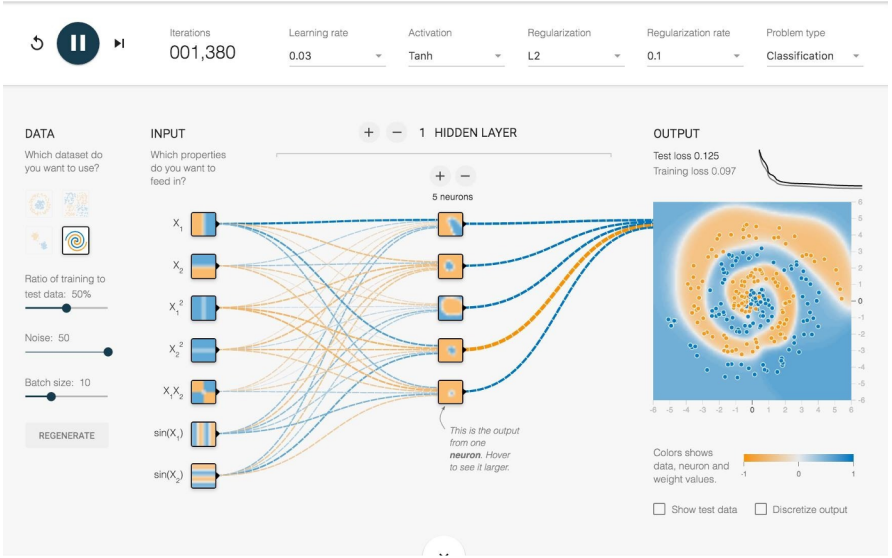
How strong are FC-nets?

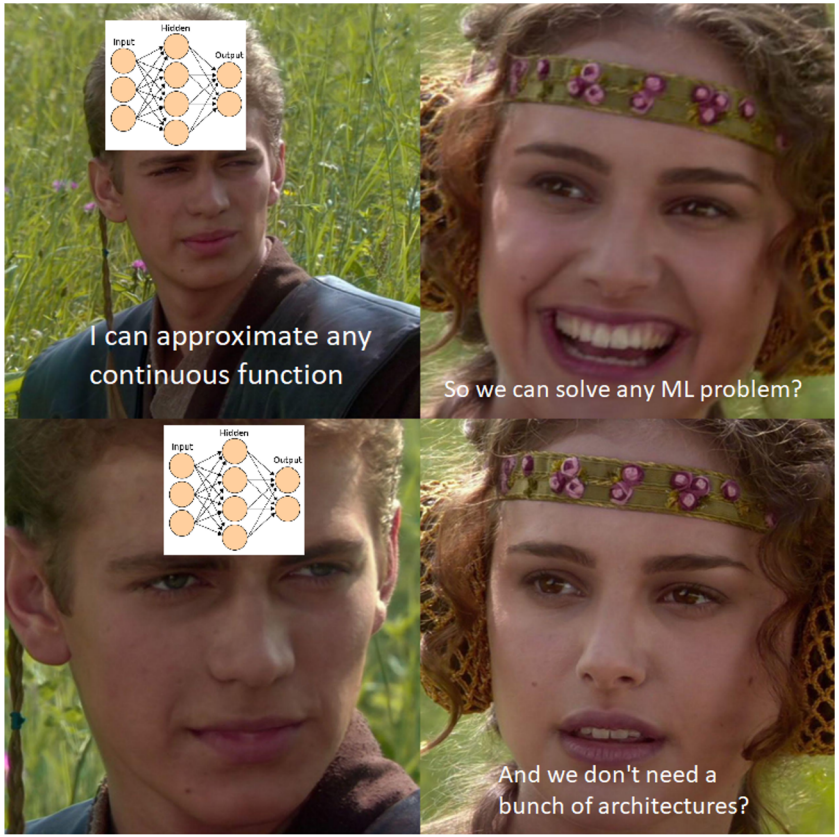


http://playground.tensorflow.org/

Universal approximation theorem

In simple words: *the universal approximation theorem says that neural networks can approximate any continuous function.*





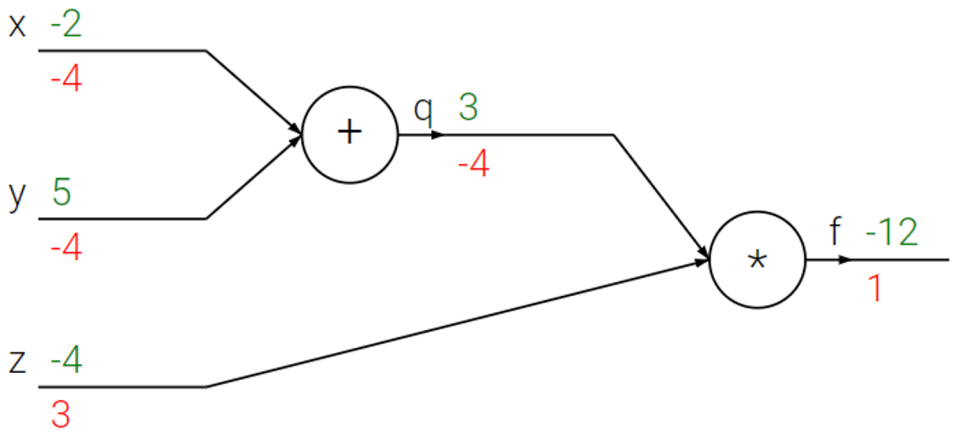
Backprop Basics and Chain Rule

$$f(x, y, z) = (x + y)z.$$

$q = x + y$ and $f = qz$.

chain rule $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$.

Forward and Backward Pass



https://cs231n.github.io/optimization-2/

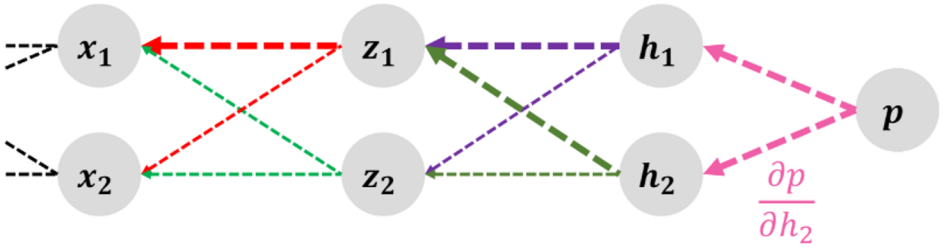
Backprop Basics and total derivative

3: $\frac{\partial p}{\partial h_1}$ $\frac{\partial p}{\partial h_2}$

2: $\frac{\partial p}{\partial z_1} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_1} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_1}$ $\frac{\partial p}{\partial z_2} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_2} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_2}$

1: $\frac{\partial p}{\partial x_1} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_2}\frac{\partial z_2}{\partial x_1} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial x_1}$

$\frac{\partial p}{\partial x_2} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial x_2} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_1}\frac{\partial z_1}{\partial x_2} + \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_2}\frac{\partial z_2}{\partial x_2} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial x_2}$



$$\frac{dL}{dt} = \frac{d}{dt}L\big(t, x_1(t), \ldots, x_n(t)\big).$$

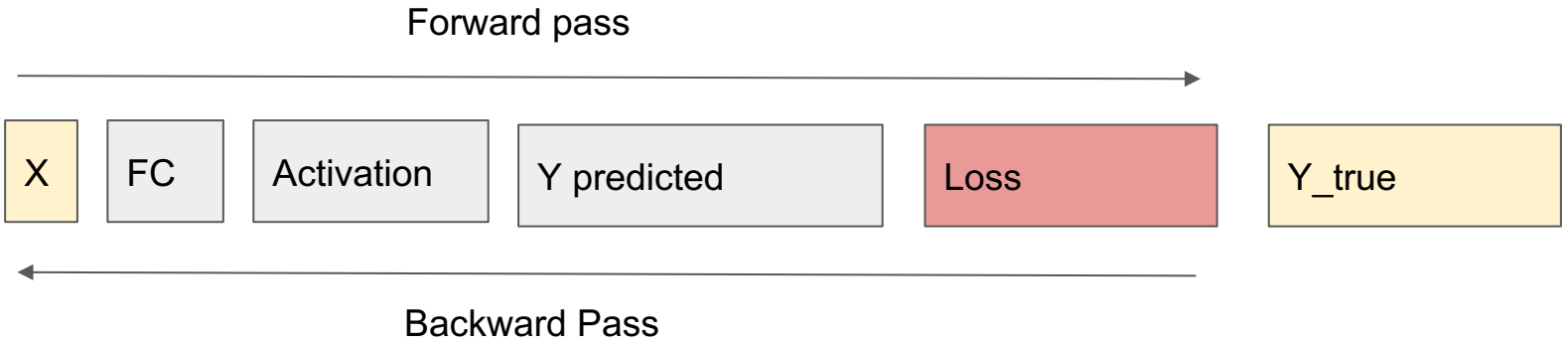
$$\frac{dL}{dt} = \frac{\partial L}{\partial t} + \sum_{i=1}^{n} \frac{\partial L}{\partial x_i}\frac{dx_i}{dt}$$

https://github.com/hse-ds/iad-deep-learning/blob/master/2021/lectures/lecture02-convnets.pdf

Backprop Basics: Simple FC net in matrix form

FC = X * W1 + b1
Activation = your_activation_function(FC)
Y_predicted = Activation * W2 + b2

X [batch size, features]
W [features, outputs]
b [outputs]

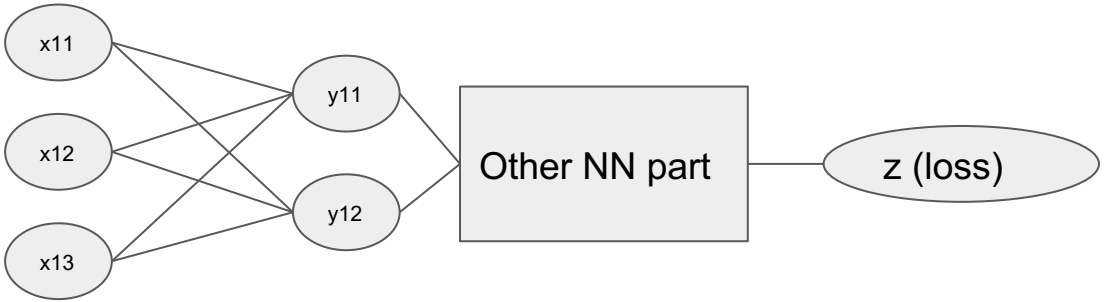


$$\frac{\partial Loss}{\partial W1} = \frac{\partial Loss}{\partial Y predicted} * \frac{\partial Y predicted}{\partial Activation} * \frac{\partial Activation}{\partial W1}$$

$$\frac{\partial Loss(X*W+b))}{\partial X} = \frac{\partial Loss}{\partial X*W+b} * W^T$$ [batch size, features]

$$\frac{\partial Loss(X*W+b))}{\partial W} = X^T * \frac{\partial Loss}{\partial X*W+b}$$ [batch size, outputs]

Backprop Basics: Simple FC net in matrix form



$$\begin{pmatrix} y_{11} \\ y_{12} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} = \begin{pmatrix} w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} \\ w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13} \end{pmatrix}^T$$

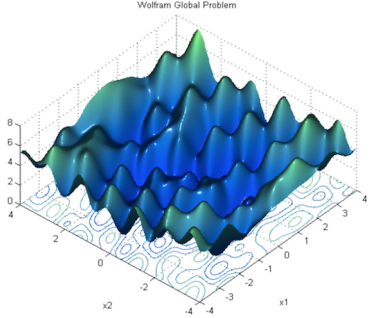
$$\frac{dz}{dx_{11}} = \frac{dz}{dy_{11}} * \frac{dy_{11}}{dx_{11}} + \frac{dz}{dy_{12}} * \frac{dy_{12}}{dx_{11}} = \frac{dz}{dy_{11}} * w_{11} + \frac{dz}{dy_{12}} * w_{12}$$

$$\frac{dz}{dw_{11}} = \sum_j \frac{dz}{dy_{j1}} * \frac{dy_{j1}}{dw_{11}} = \sum_j \frac{dz}{dy_{j1}} * x_{j1}$$

$$\frac{\partial Loss(X * W + b))}{\partial X} = \frac{\partial Loss}{\partial X * W + b} * W^T$$

$$\frac{\partial Loss(X * W + b))}{\partial W} = X^T * \frac{\partial Loss}{\partial X * W + b}$$

Optimisation



1. **Initialization?**

2. **Feature scaling?**

Gradient descent: $\theta = \theta - \eta \cdot \nabla_\theta J(\theta).$

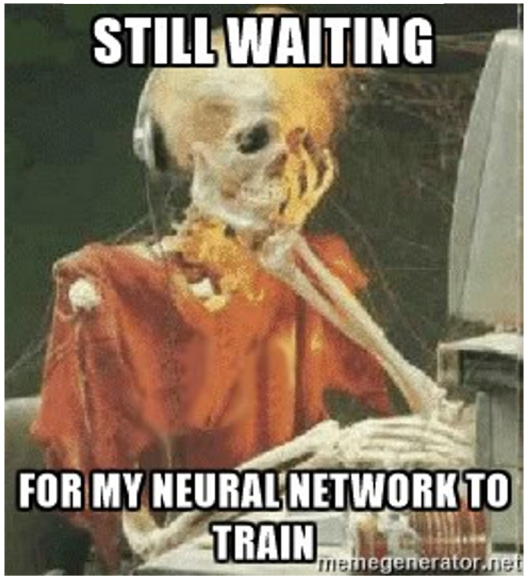
1. **(Batch) GD**
2. **SGD**
3. **Mini-Batch GD**

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

Image 2: SGD without momentum

Image 3: SGD with momentum



*When you forgot to take the required steps

https://ruder.io/optimizing-gradient-descent/index.html

GD optimization algorithms

Adaptive learning rate: smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates)

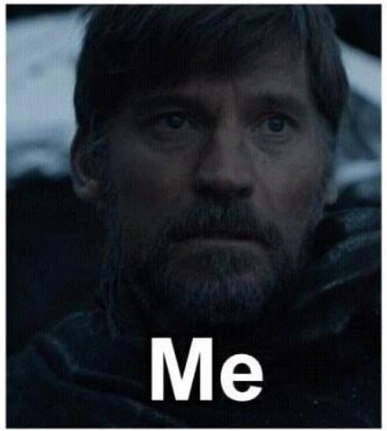
Algorithms:

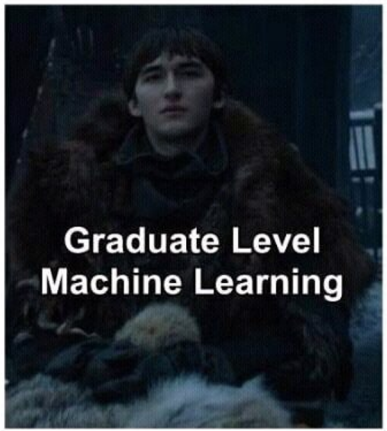
- AdaGrad
- Adadelta
- RMSprom
- AdaMax
- Adam
- etc.

https://ruder.io/optimizing-gradient-descent/index.html

Now, coding :)


Me
Lin Alg, Stats,
Probability Theory
in FreshmanYear


Me


Graduate Level
Machine Learning