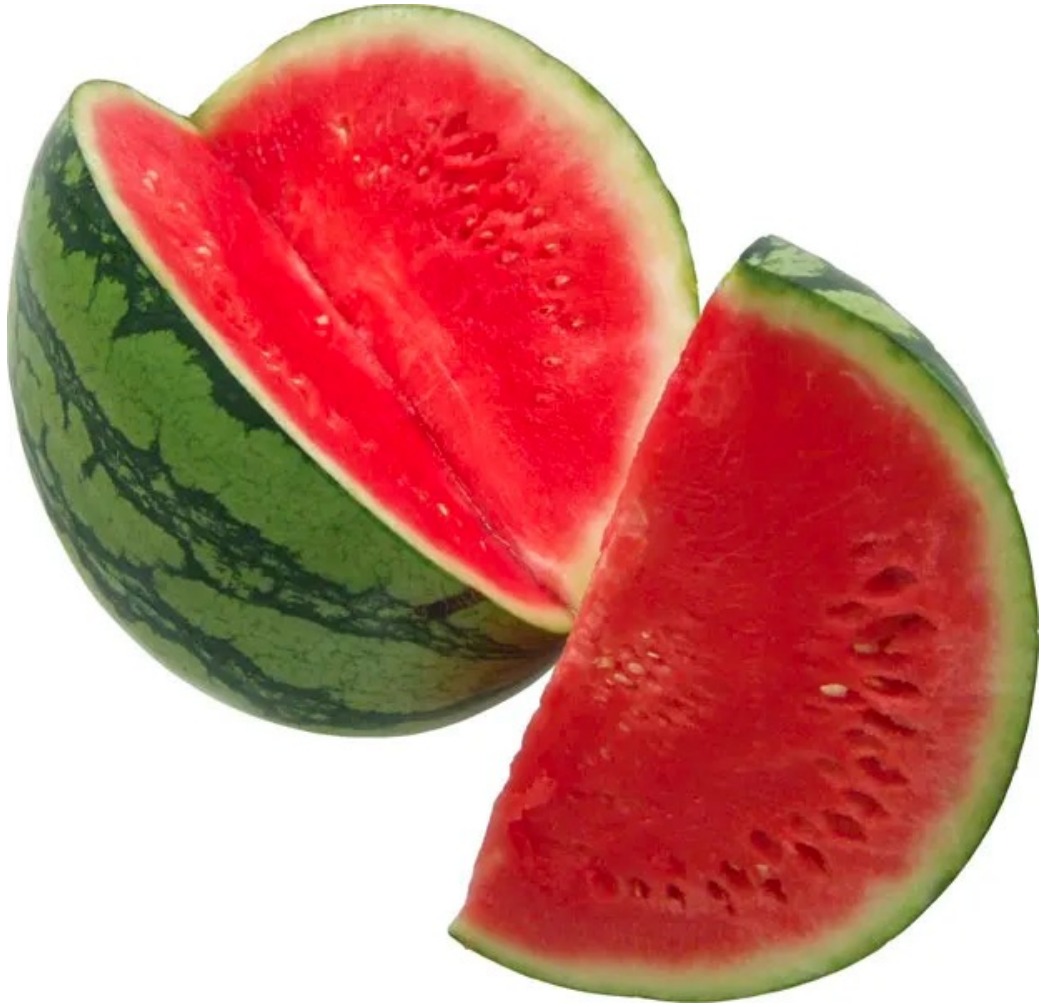
Neural Network Workshop

Khayyam Salehi, Ph.D.
Assistant Professor of Computer Science
Shahrekord University

2023

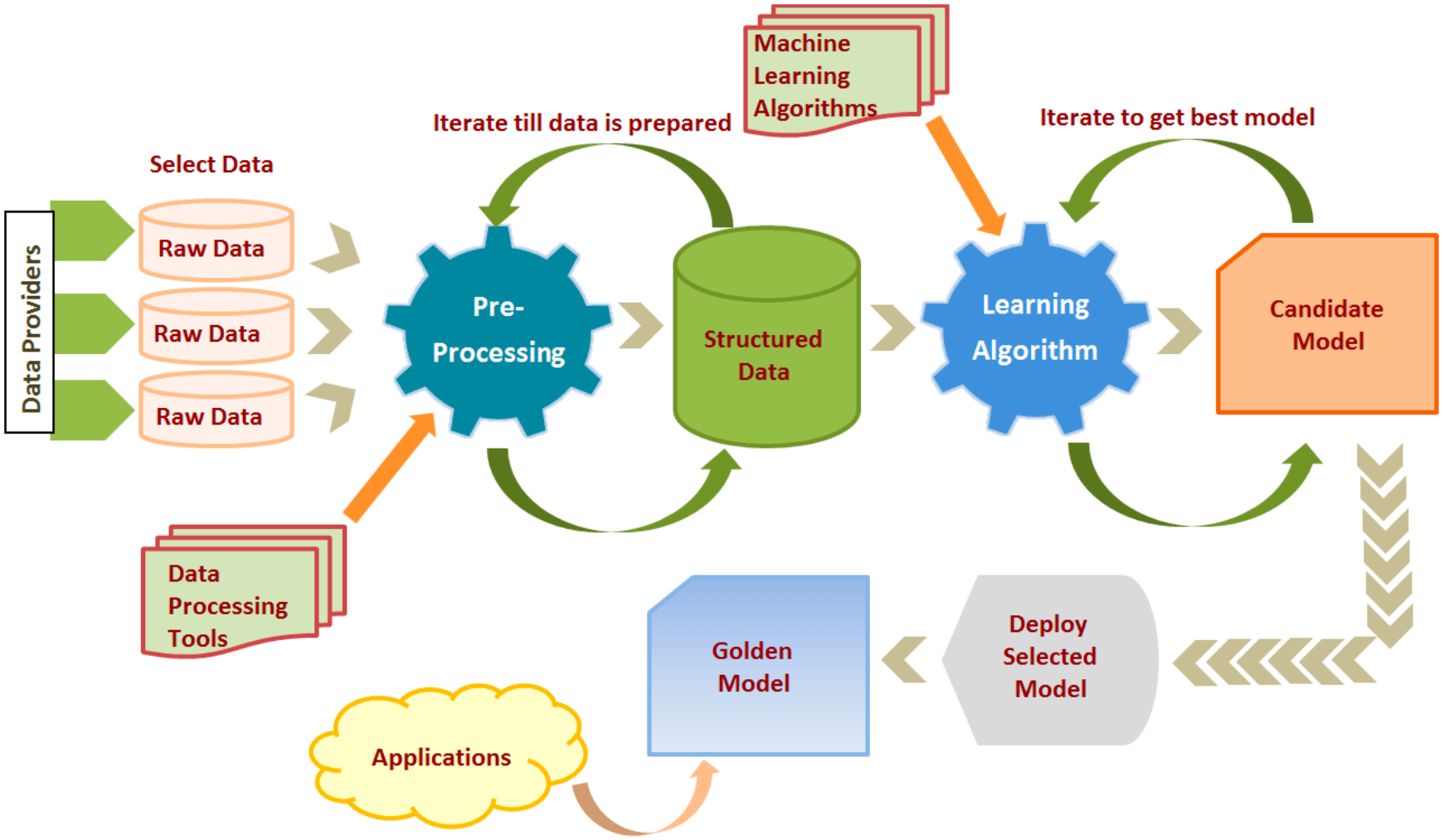

بسم الله الرحمن الرحيم

Fruit of the Slide



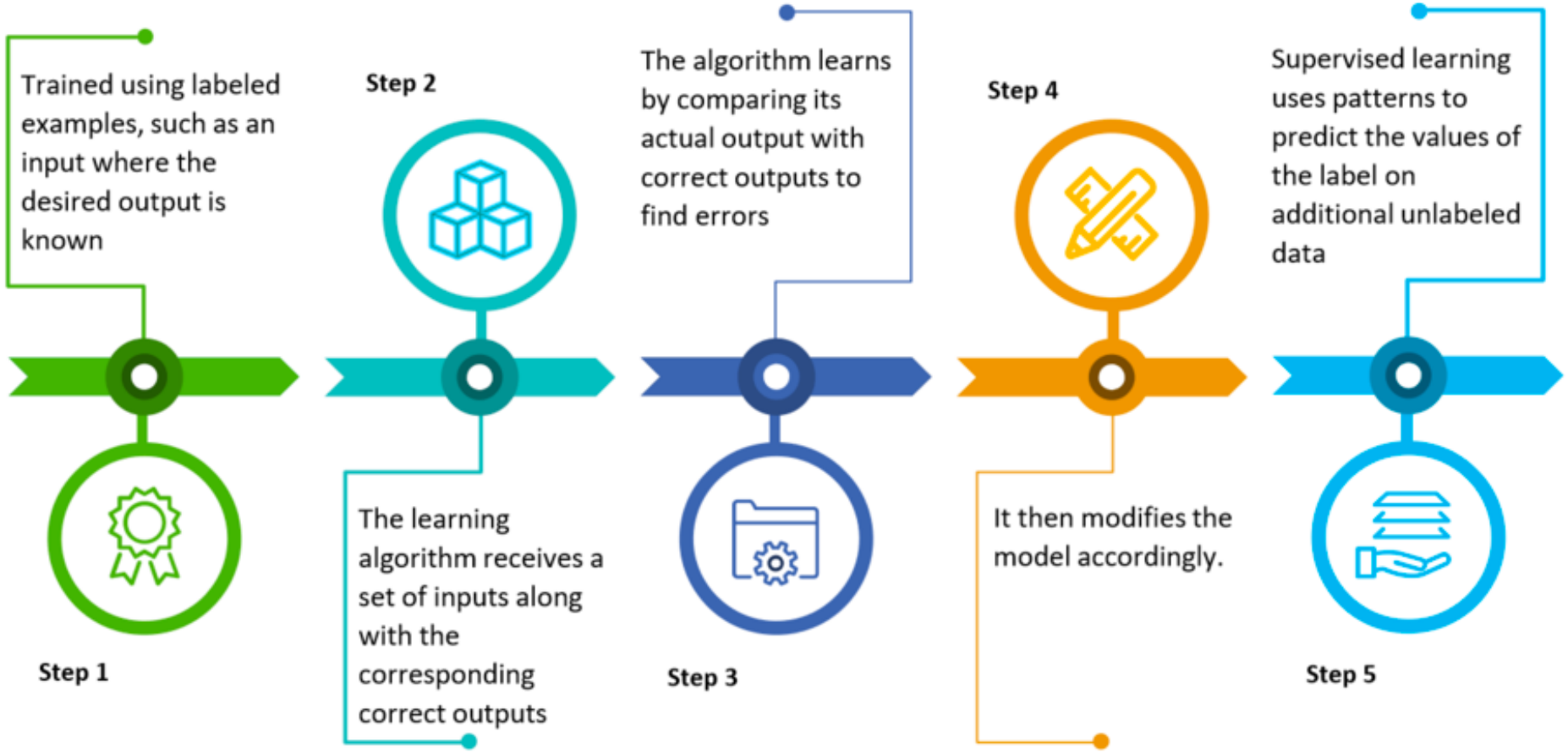
Instructor: Khayyam Salehi, Ph.D.

Machine Learning Scenarios



https://elearningindustry.com/machine-learning-process-and-scenarios

Learning Process



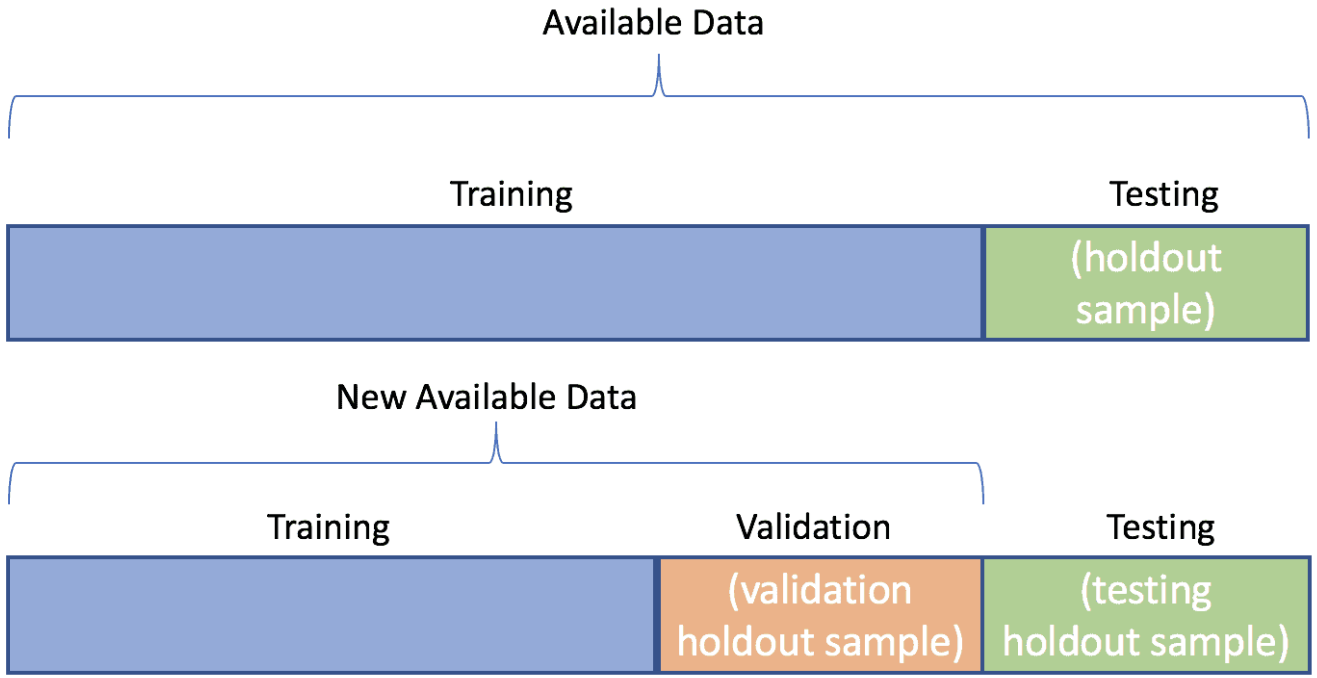
https://www.clariba.com/machine-learning-for-business

Machine Learning Principles

- Using gathered data can predict unseen data.
- Unseen data are the same distribution as the gathered data.

Machine Learning Principles

- Using gathered data can predict unseen data.
- Unseen data are the same distribution as the gathered data.
  - We should testing our model. --> Splitting data into training and testing data.

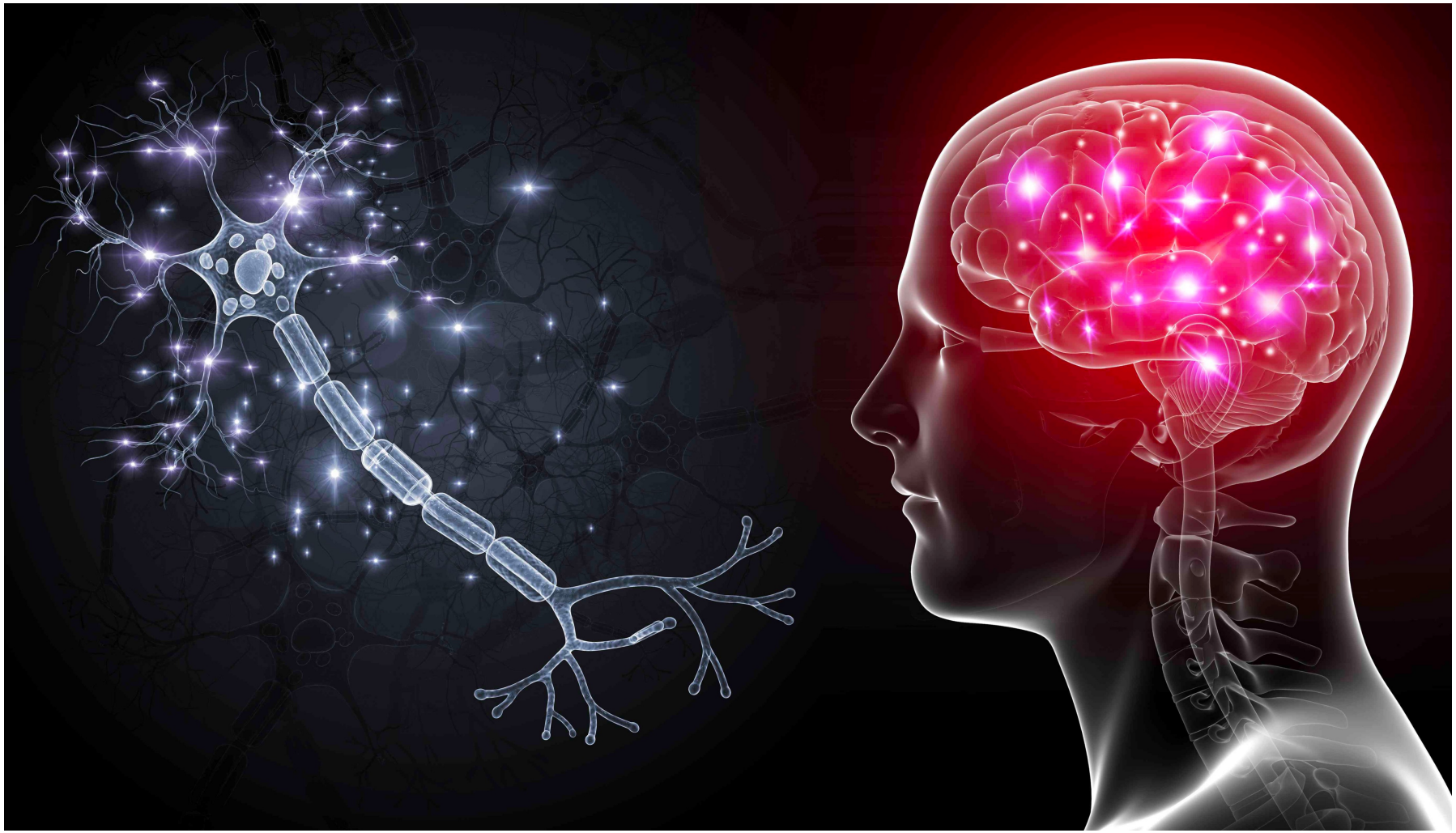


https://algotrading101.com/learn/train-test-split/

Machine Learning Principles

- Using gathered data can predict unseen data.
- Unseen data are the same distribution as the gathered data.
  - We should testing our model. --> Splitting data into training and testing data.
- We have mistakes! --> Accuracy, Precision, Recall ...
- The simpler model, the better model.

How brain cells communicate with each other?

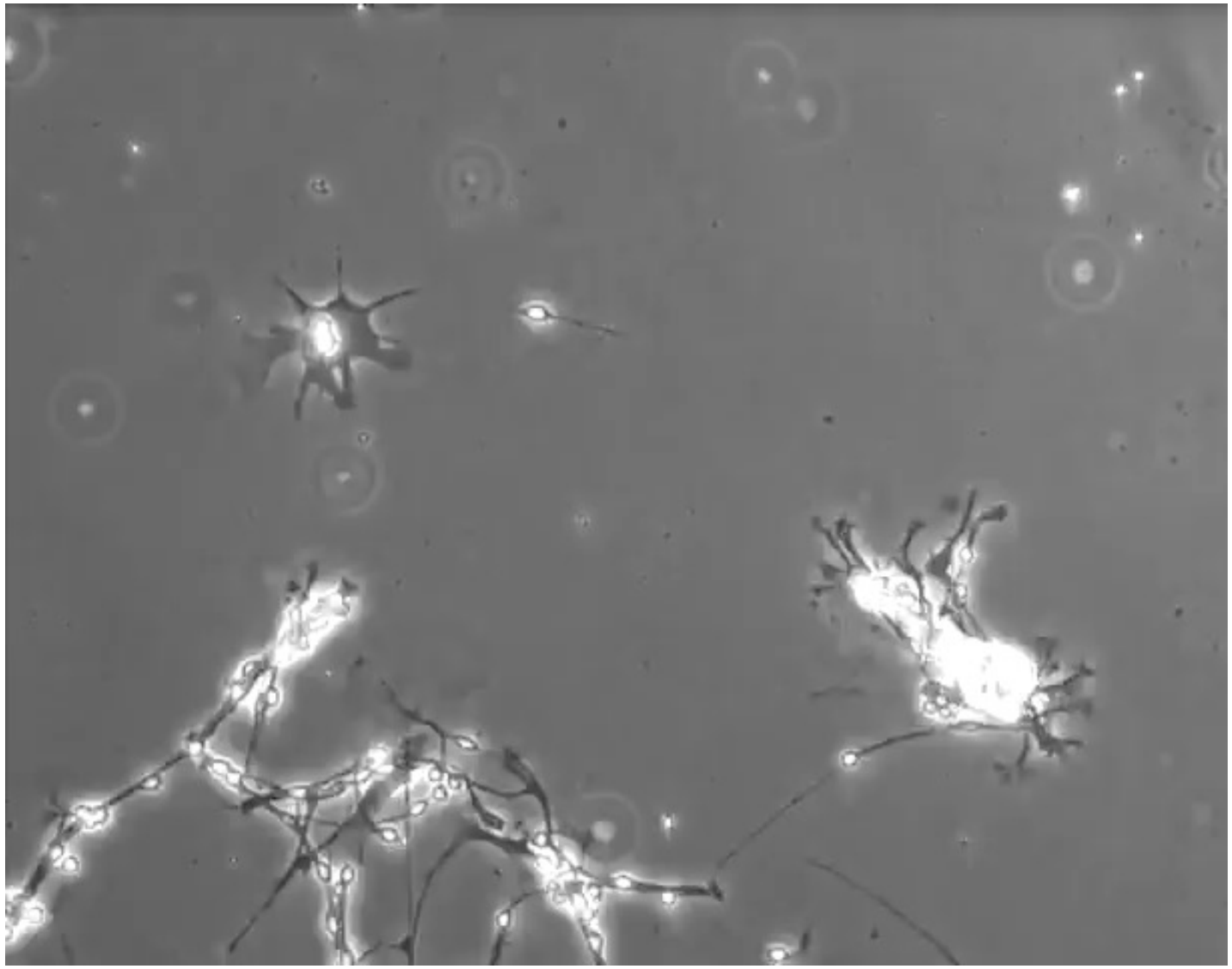


https://www.verywellmind.com/how-brain-cells-communicate-with-each-other-2584397

Single brain cell looking for connections:

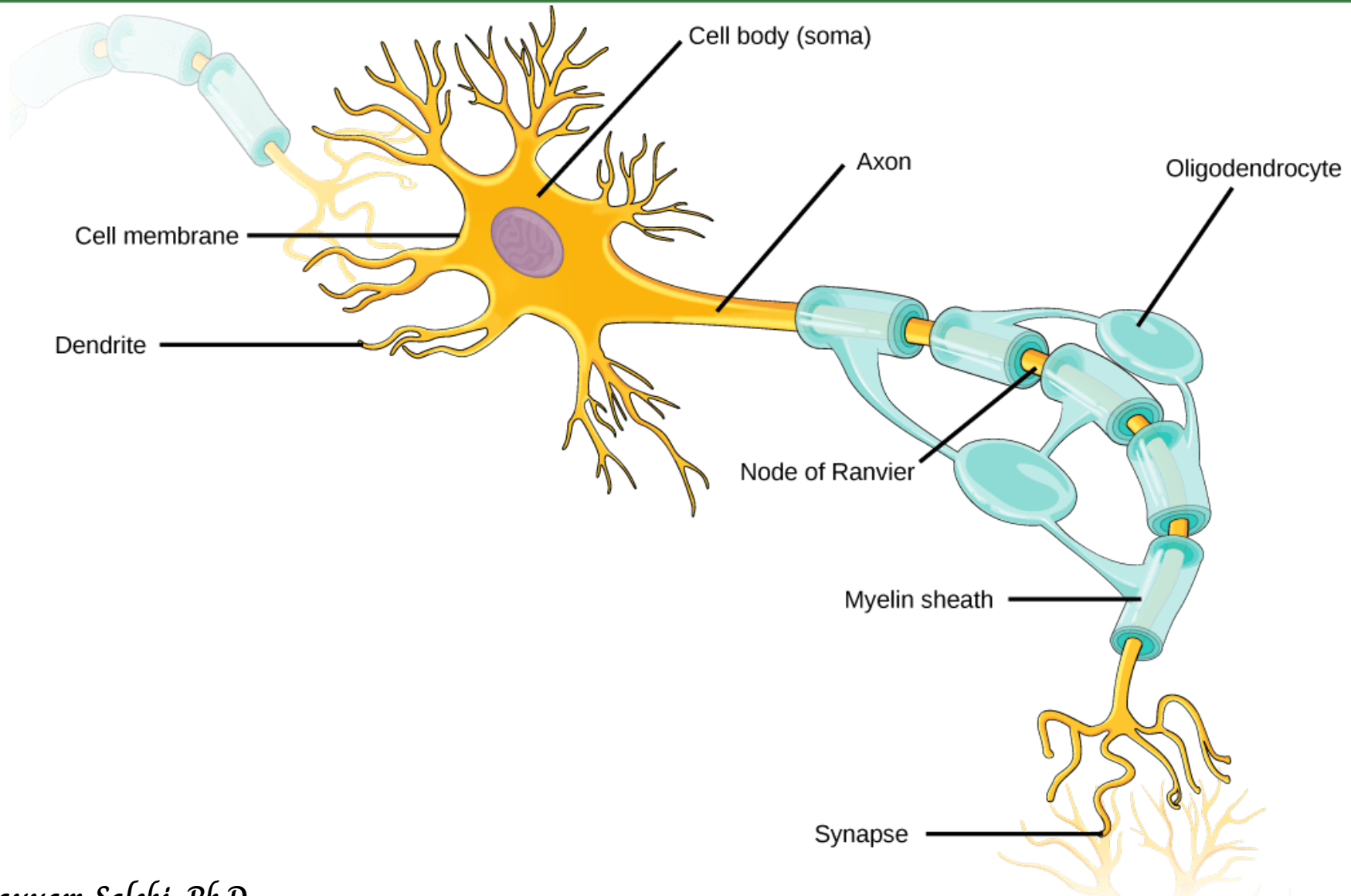
https://www.linkedin.com/posts/slava-bobrov_biology-neuroscience-medicine-activity-6988847118174023680-xZW5?utm_source=share&utm_medium=member_desktop

Neurons forming connections



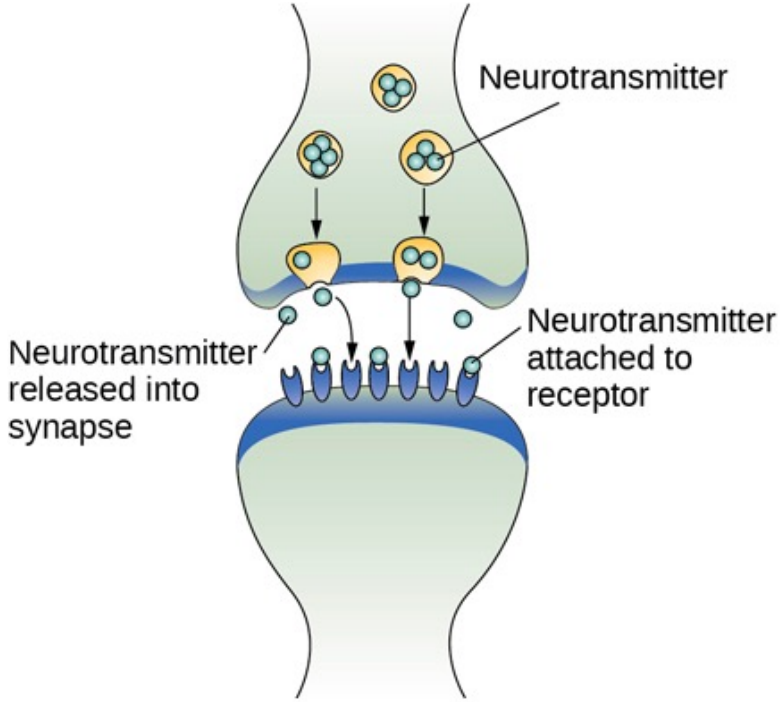
https://www.linkedin.com/posts/slava-bobrov_neuroscience-activity-6856189155949465601-GOHH?utm_source=share&utm_medium=member_desktop

Neurons Structure

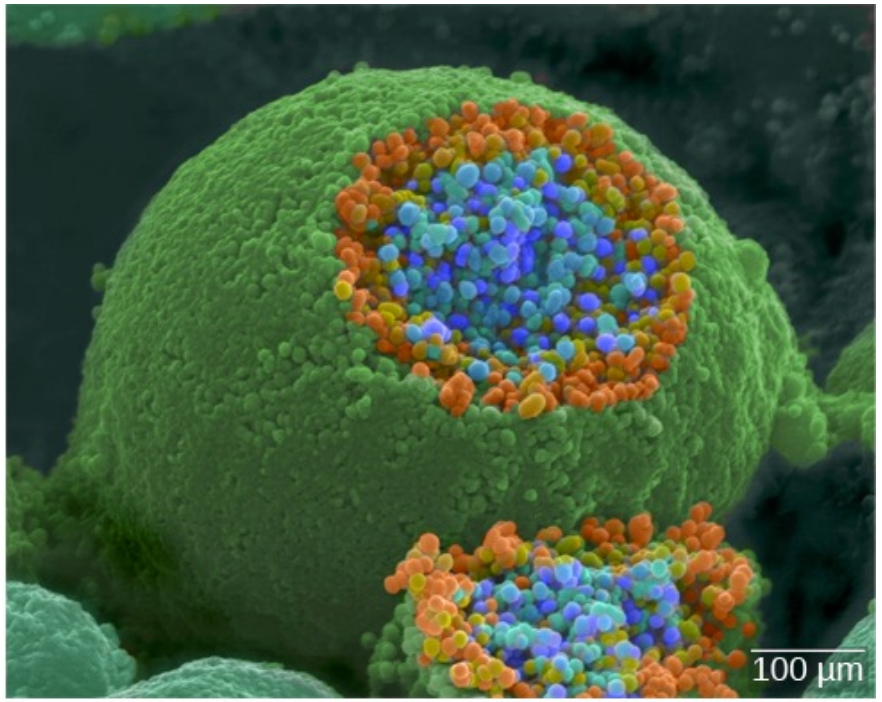


https://organismalbio.biosci.gatech.edu/chemical-and-electrical-signals/neurons/

Communicating



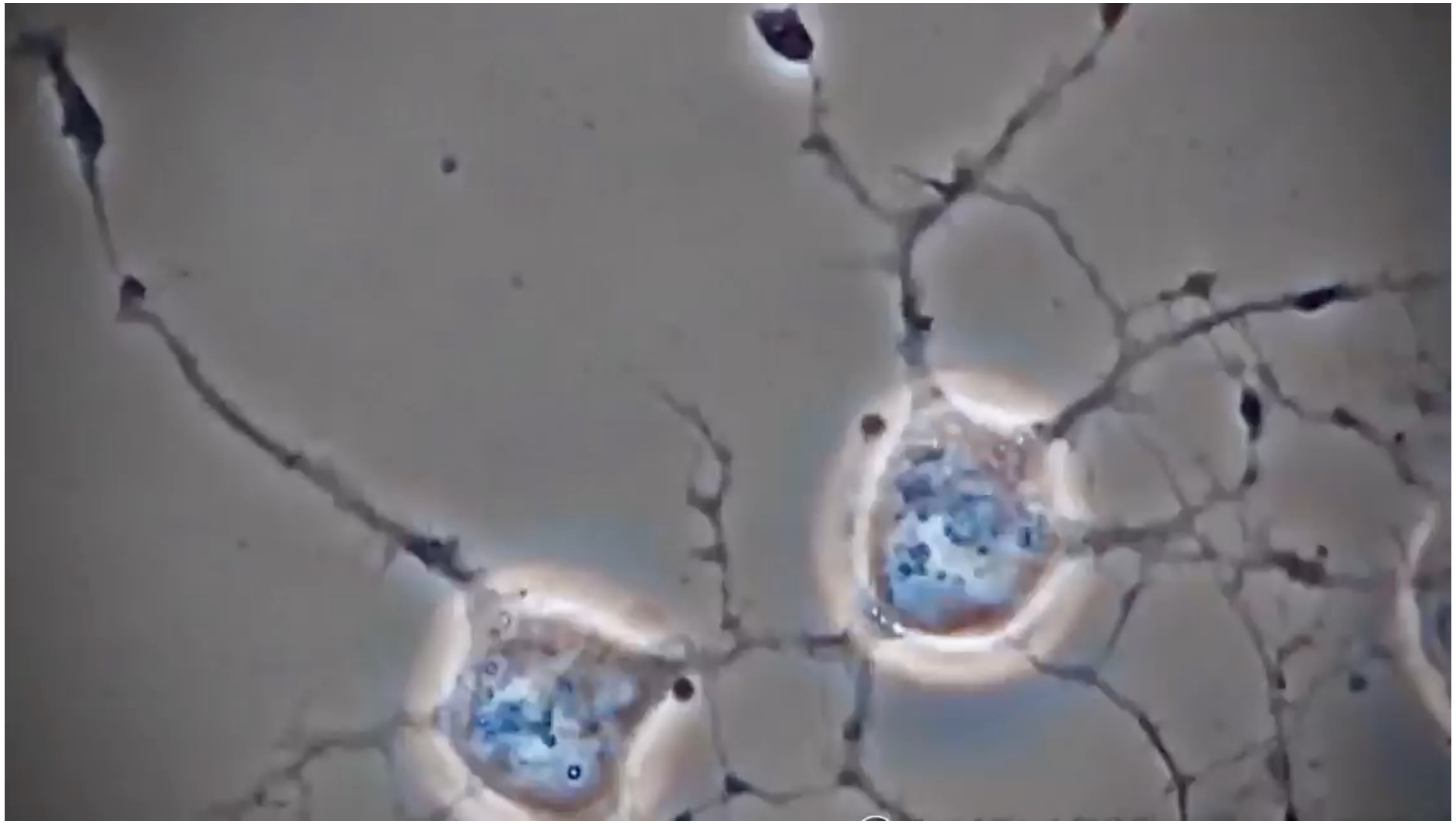
(a)



(b)

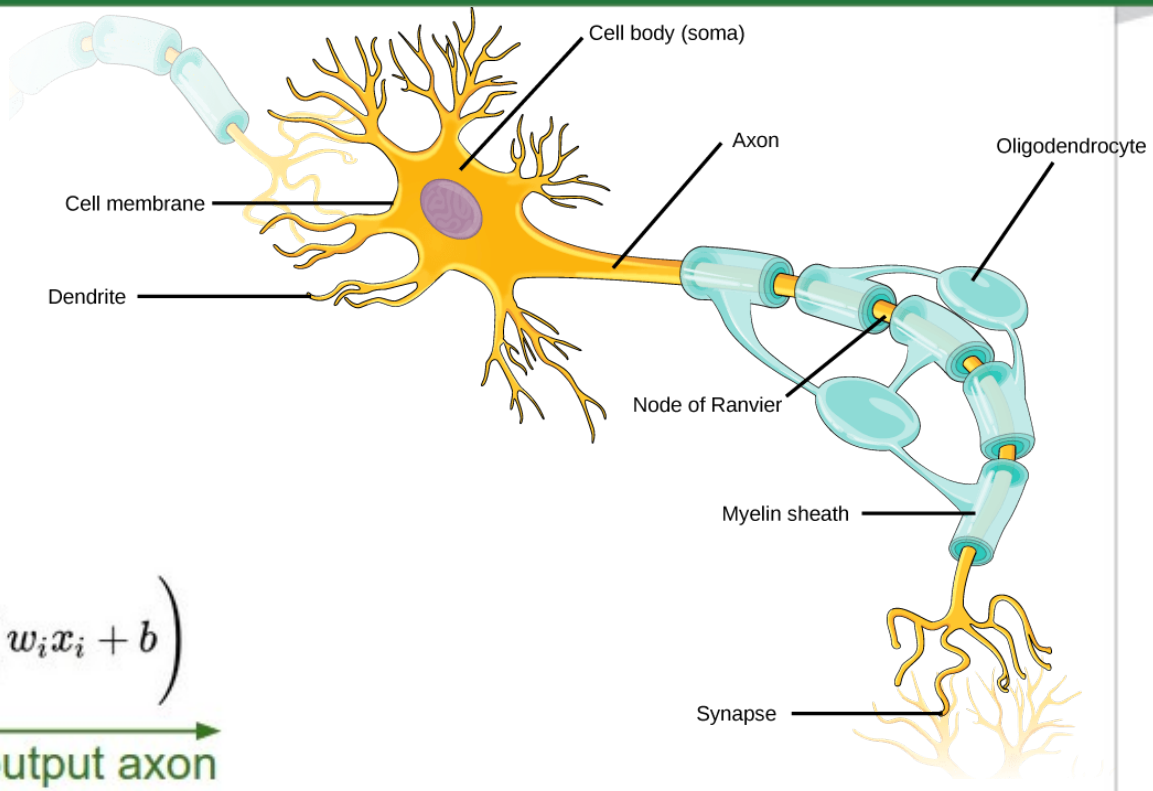
https://organismalbio.biosci.gatech.edu/chemical-and-electrical-signals/neurons/

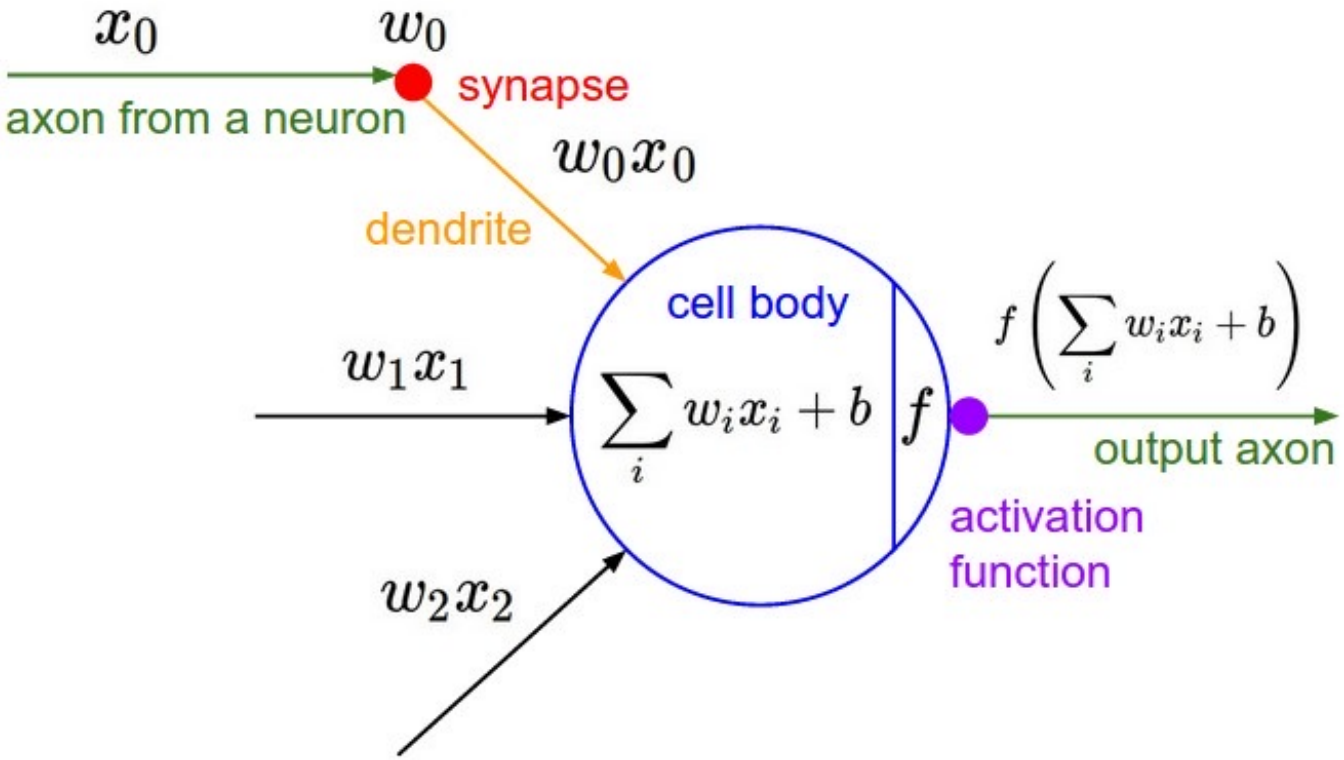
Two brain cells having a chat:



https://www.linkedin.com/posts/slava-bobrov_neuroscience-biology-medicine-activity-7060589883249020928-L06e?utm_source=share&utm_medium=member_desktop

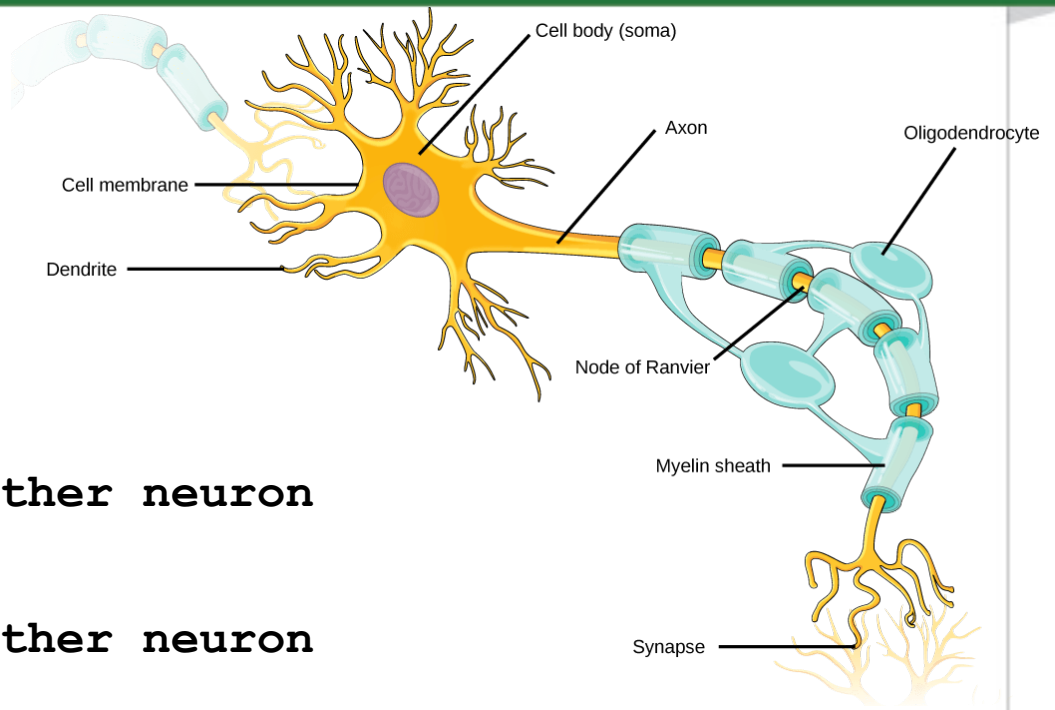
Modeling a Neuron (Perceptron)

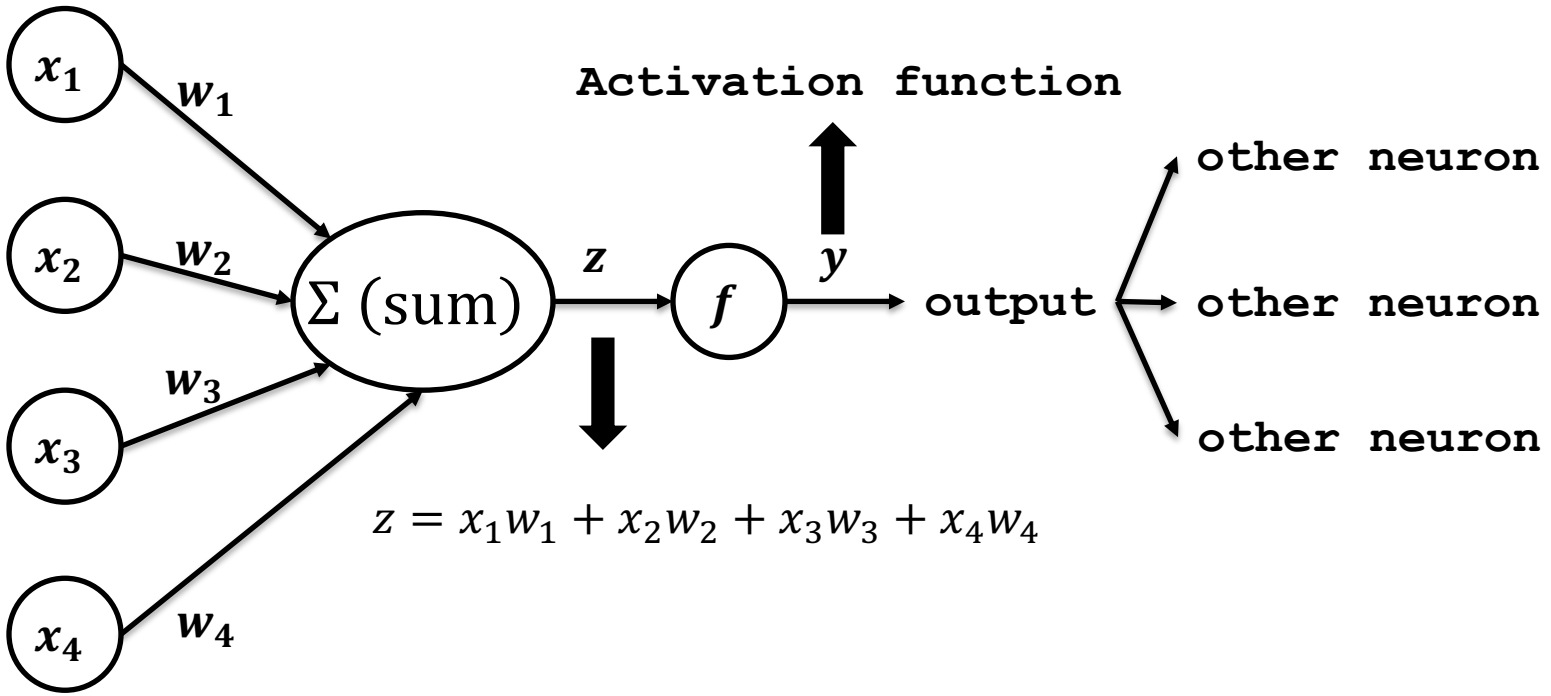




https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/supervised_learning/neural_networks

Modeling a Neuron (Perceptron)

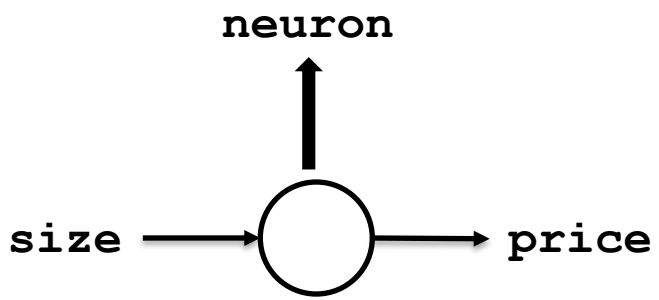




$$z = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4$$

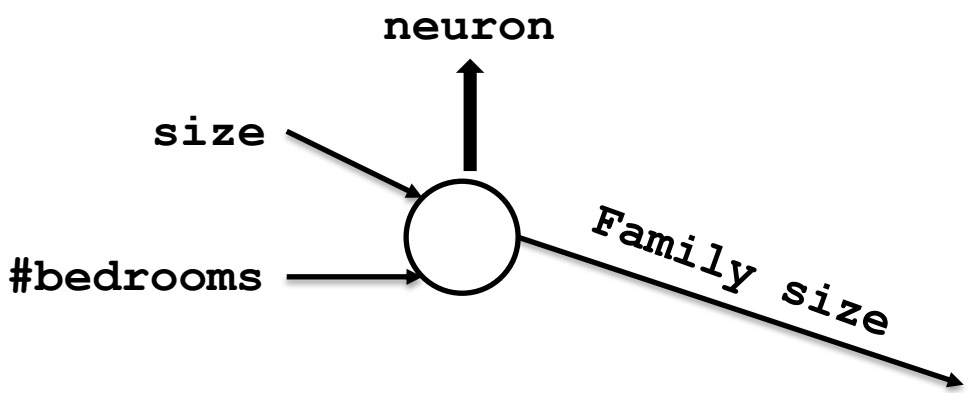
What is a Neural Network?

Predicting a price of house just based on the size (feature):



What is a Neural Network?

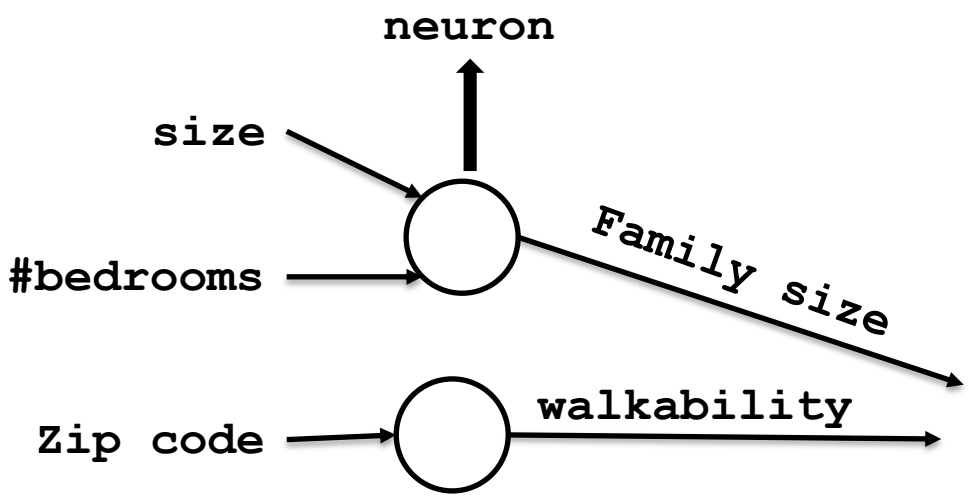
Predicting a price of house just based on some features:



Thanks to Andrew NG, deeplearning.ai

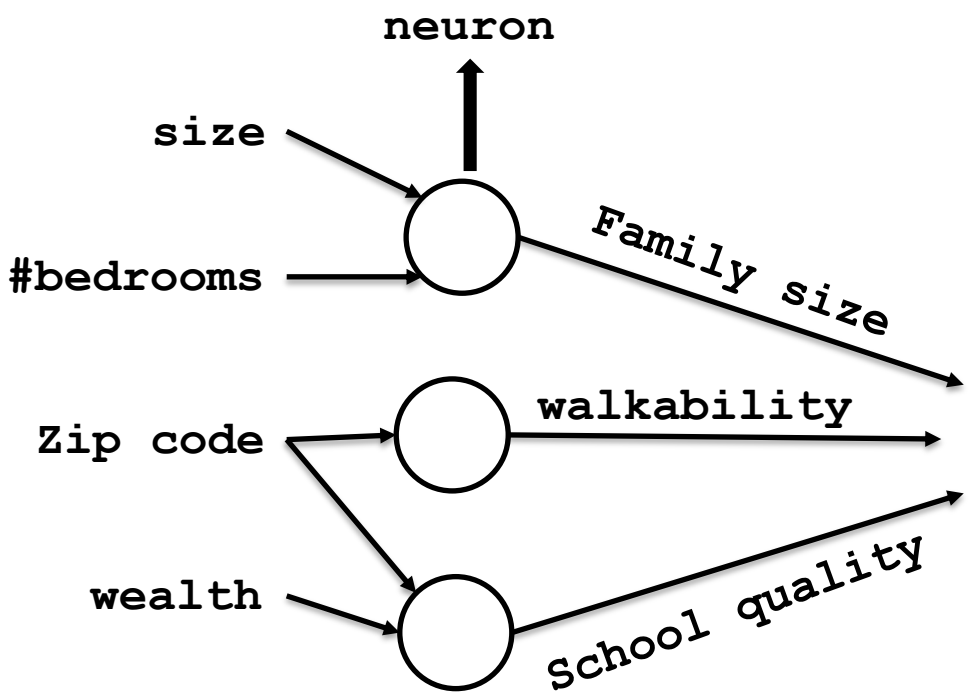
What is a Neural Network?

Predicting a price of house just based on some features:



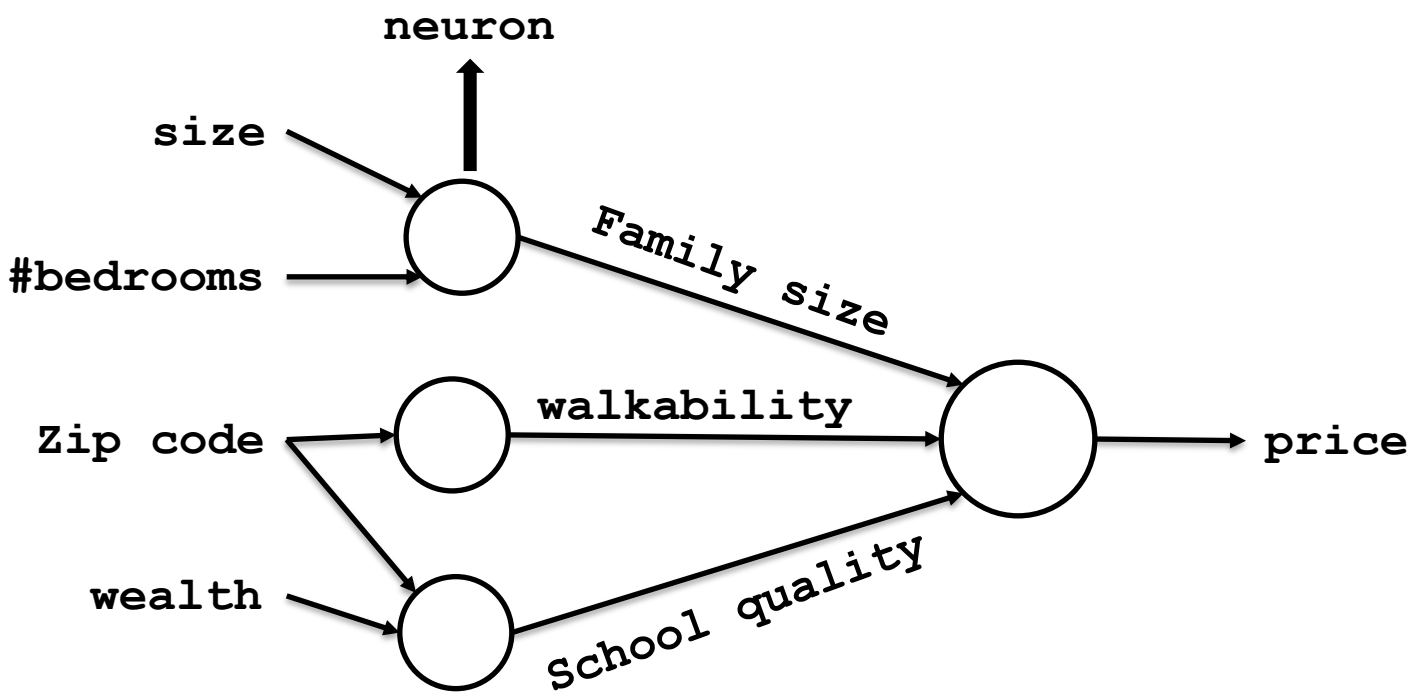
What is a Neural Network?

Predicting a price of house just based on some features:



What is a Neural Network?

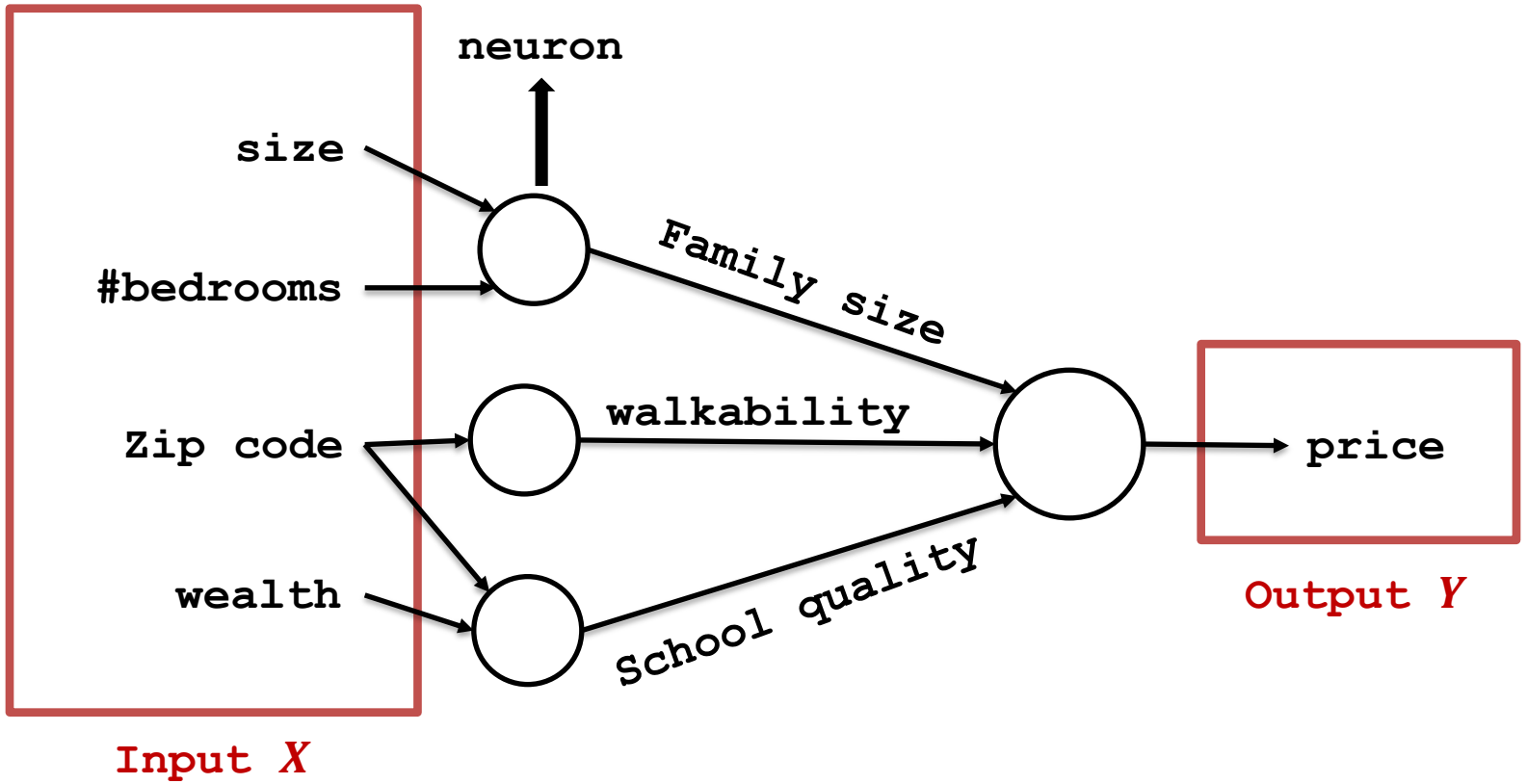
Predicting a price of house just based on some features:



Thanks to Andrew NG, deeplearning.ai

What is a Neural Network?

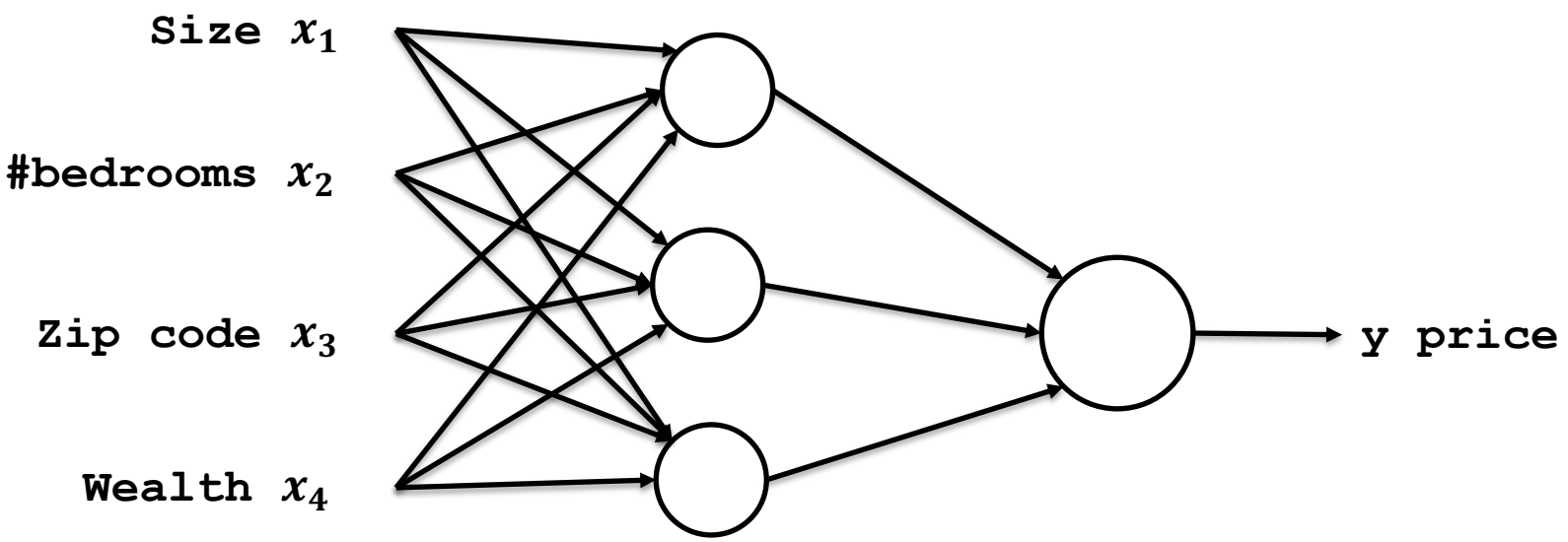
Predicting a price of house just based on some features:



Thanks to Andrew NG, deeplearning.ai

What is a Neural Network?

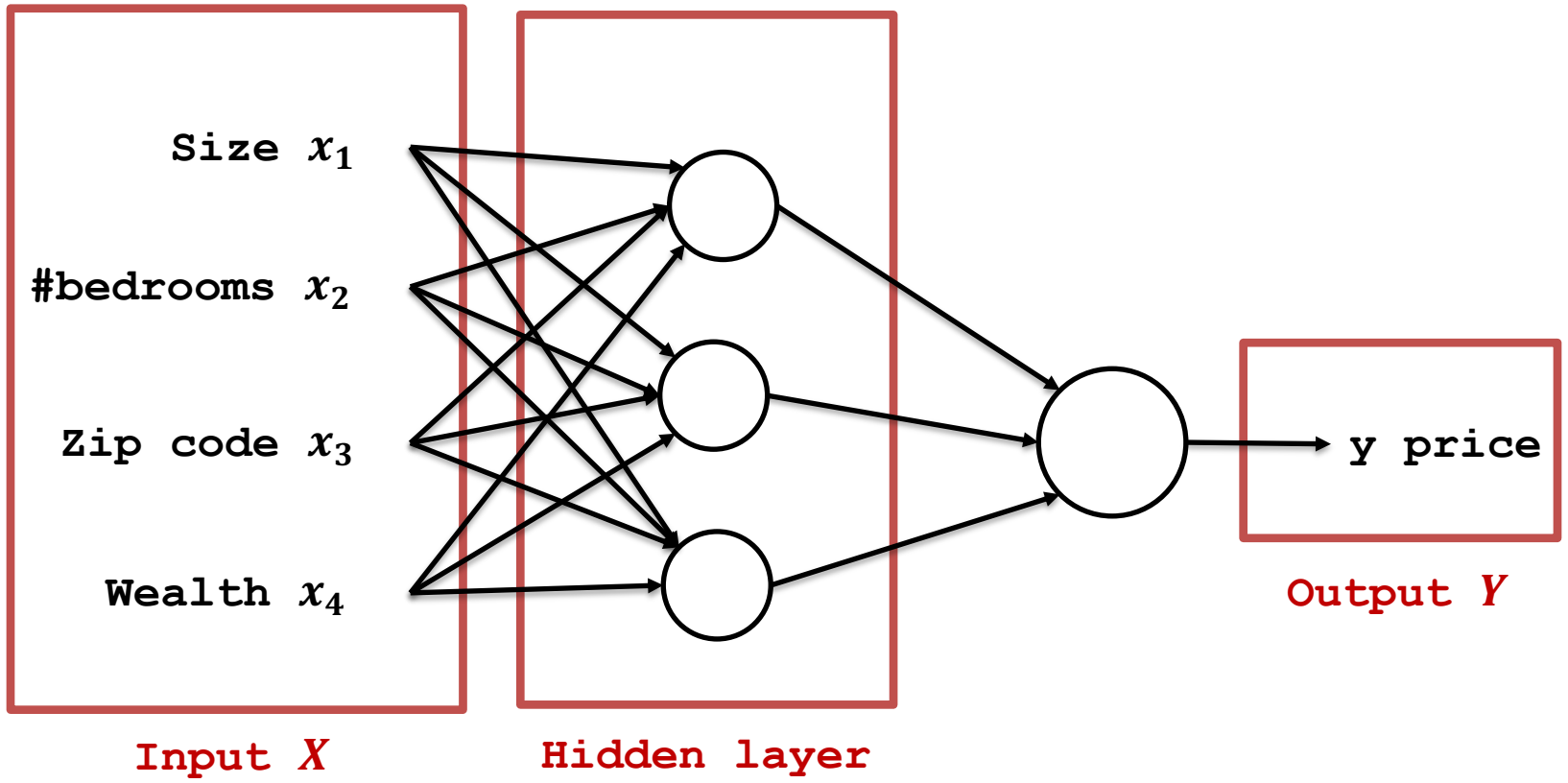
Predicting a price of house just based on some features:



Thanks to Andrew NG, deeplearning.ai

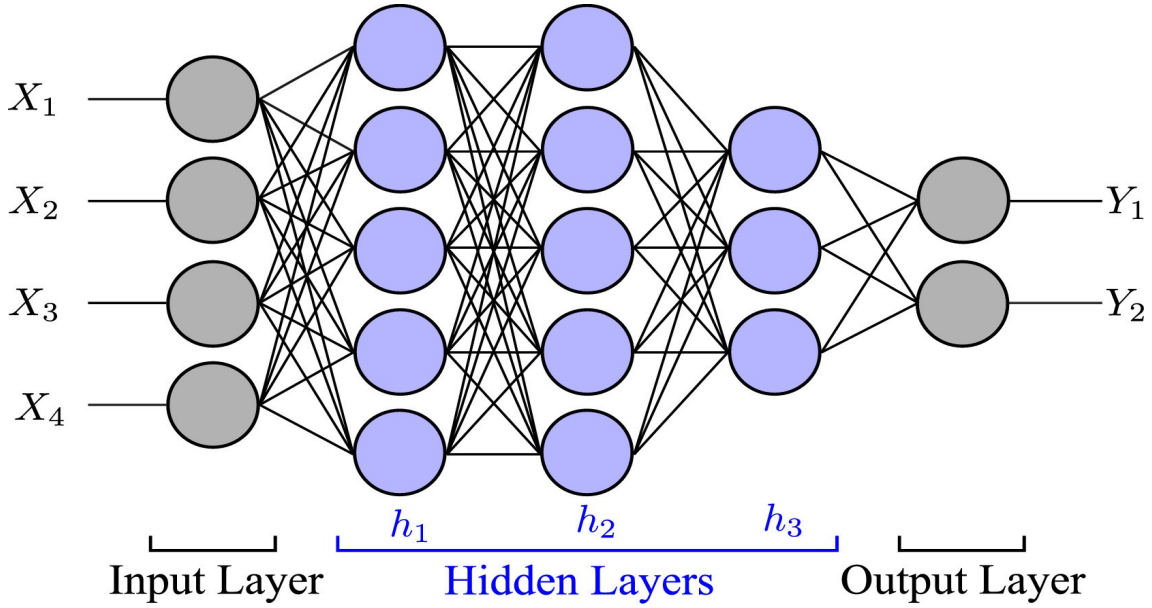
Multi Layer Perceptron (MLP)

Predicting a price of house just based on some features:

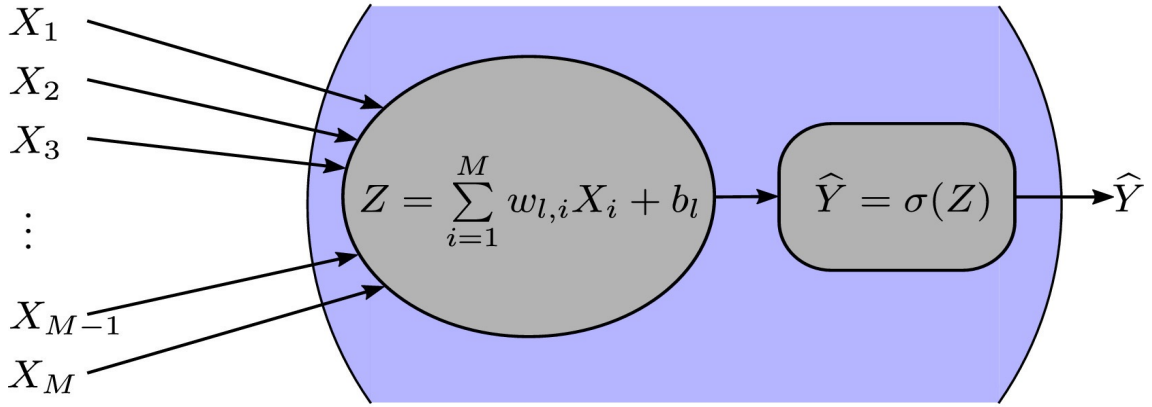


Thanks to Andrew NG, deeplearning.ai

Schematic of a fully connected neural network



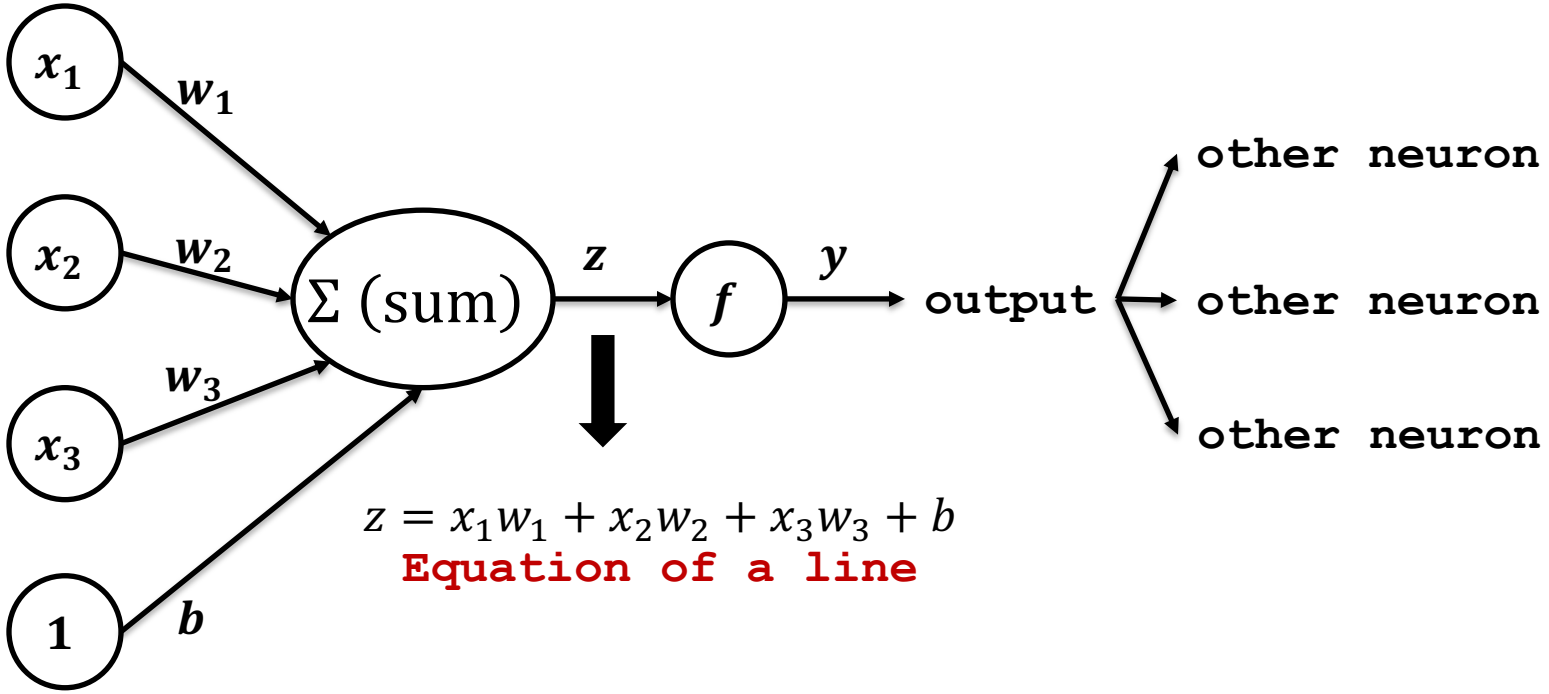
(a) Network architecture.



(b) Representation of a neuron.

https://www.sciencedirect.com/science/article/pii/S0360128522000193

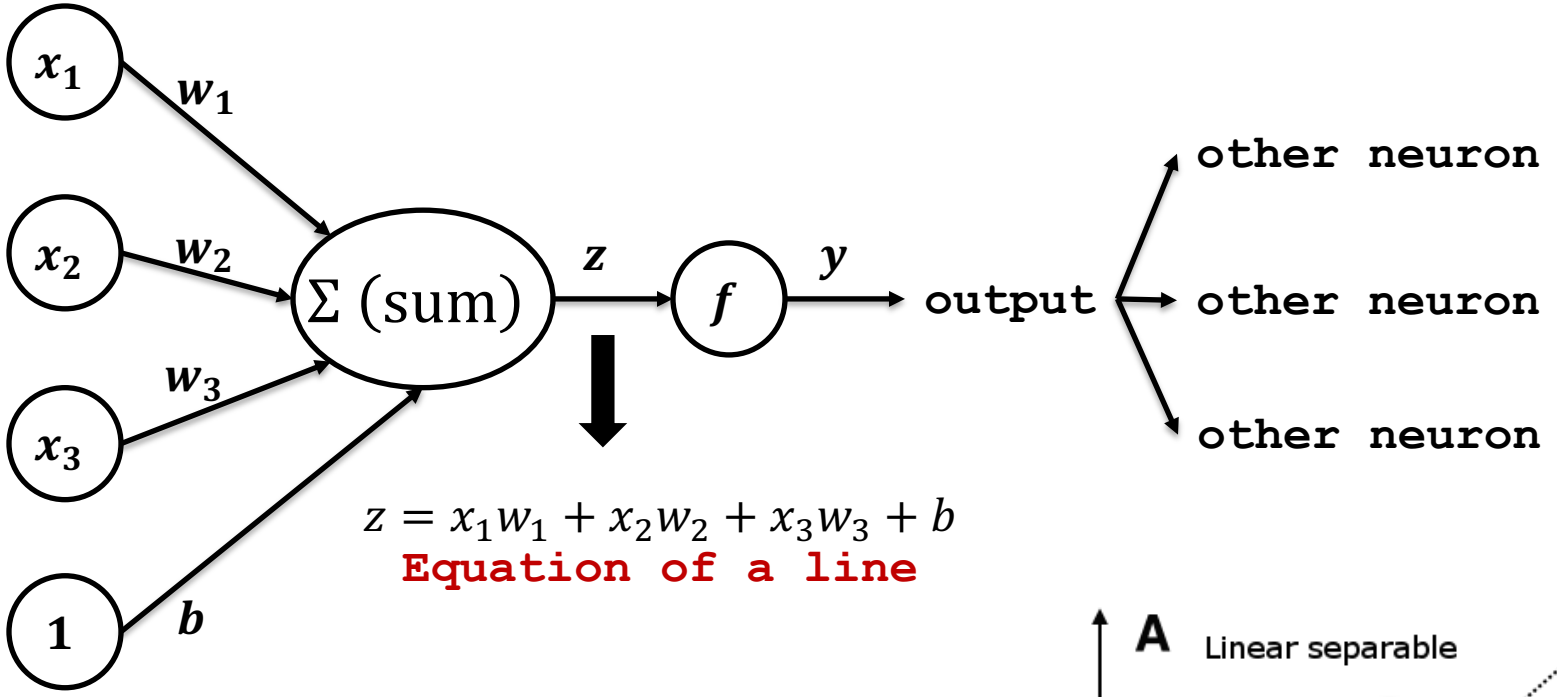
The need for MLPs



$$z = x_1w_1 + x_2w_2 + x_3w_3 + b$$

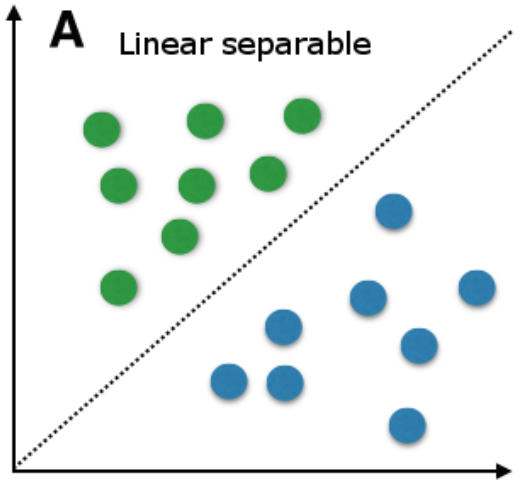
Equation of a line

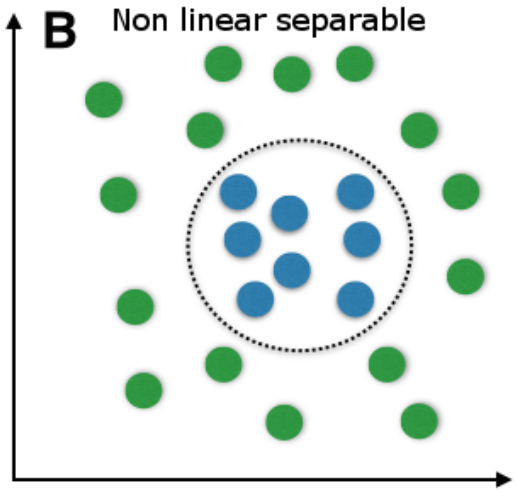
The need for MLPs



$$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

Equation of a line





https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/supervised_learning/linear_classification

XOR Problem

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

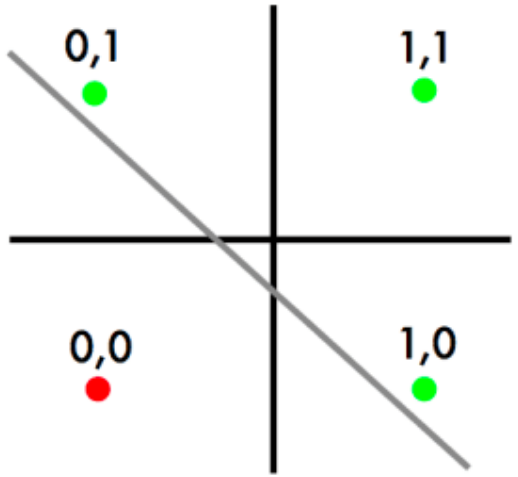
XOR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

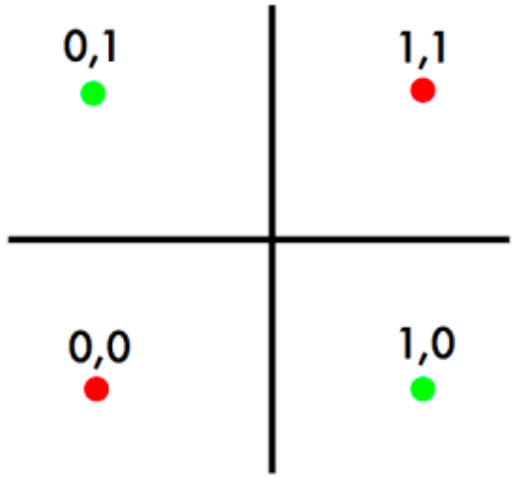
NOT Truth Tabl

A	B
0	1
1	0

The XOR problem



OR



XOR

Forward Propagation

	Feature 1	Feature 2
Samples	0.5	-0.3
	0.4	0.1
	0.6	0.8

Output 1	Output 2
0.9	0.1
0.9	0.8
0.1	0.1

Forward Propagation

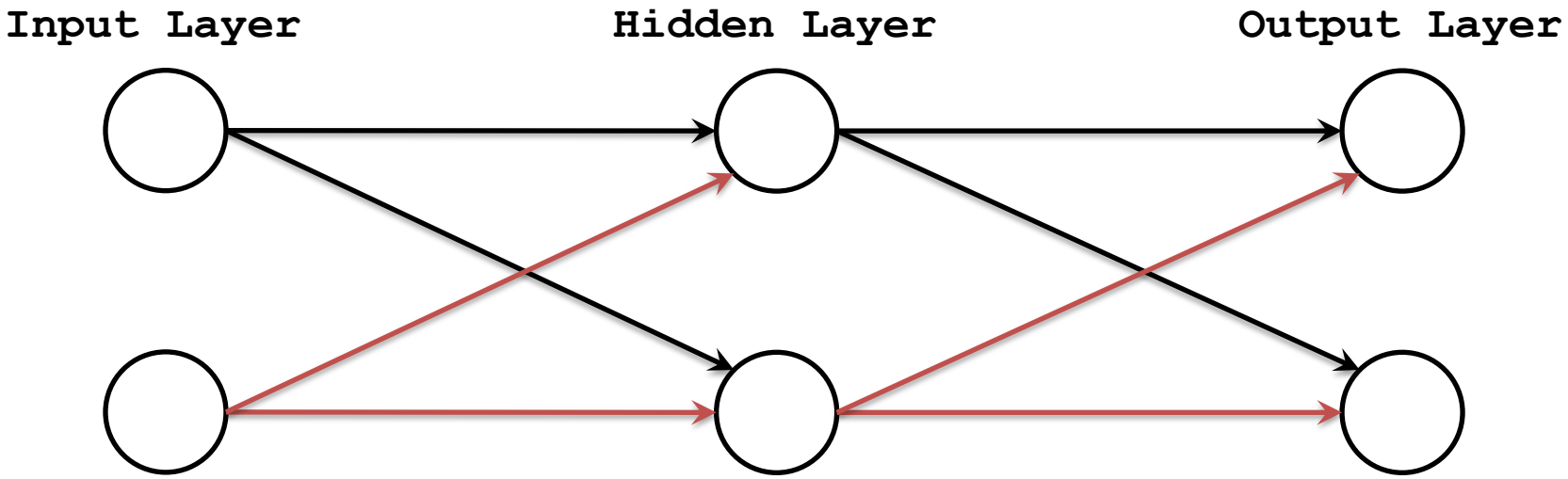
	Feature 1	Feature 2
Samples	0.5	-0.3
	0.4	0.1
	0.6	0.8

Output 1	Output 2
0.9	0.1
0.9	0.8
0.1	0.1

Forward Propagation

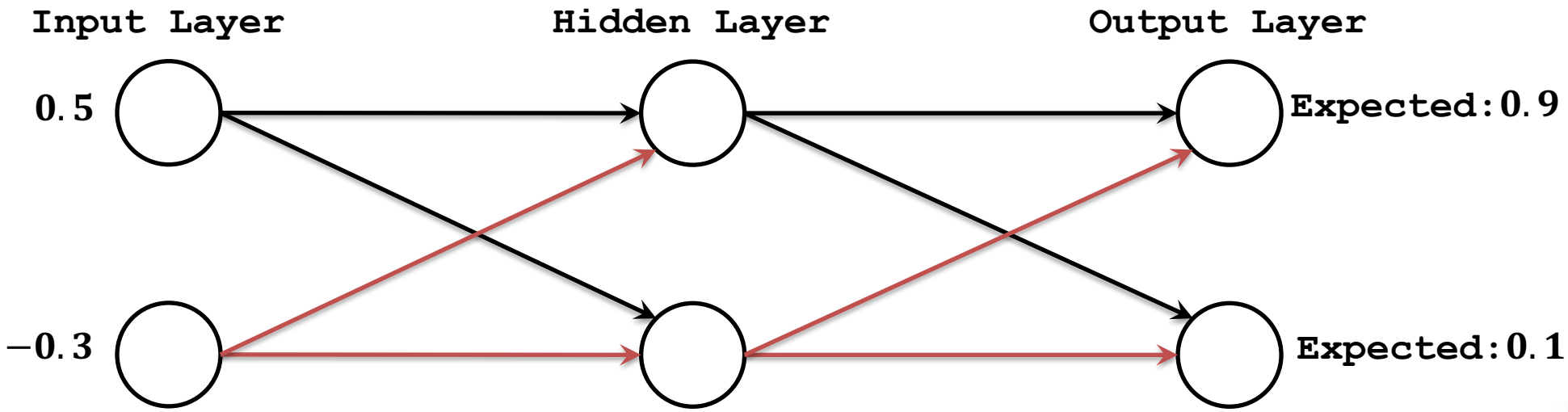
	Feature 1	Feature 2
Samples	0.5	-0.3
	0.4	0.1
	0.6	0.8

Output 1	Output 2
0.9	0.1
0.9	0.8
0.1	0.1



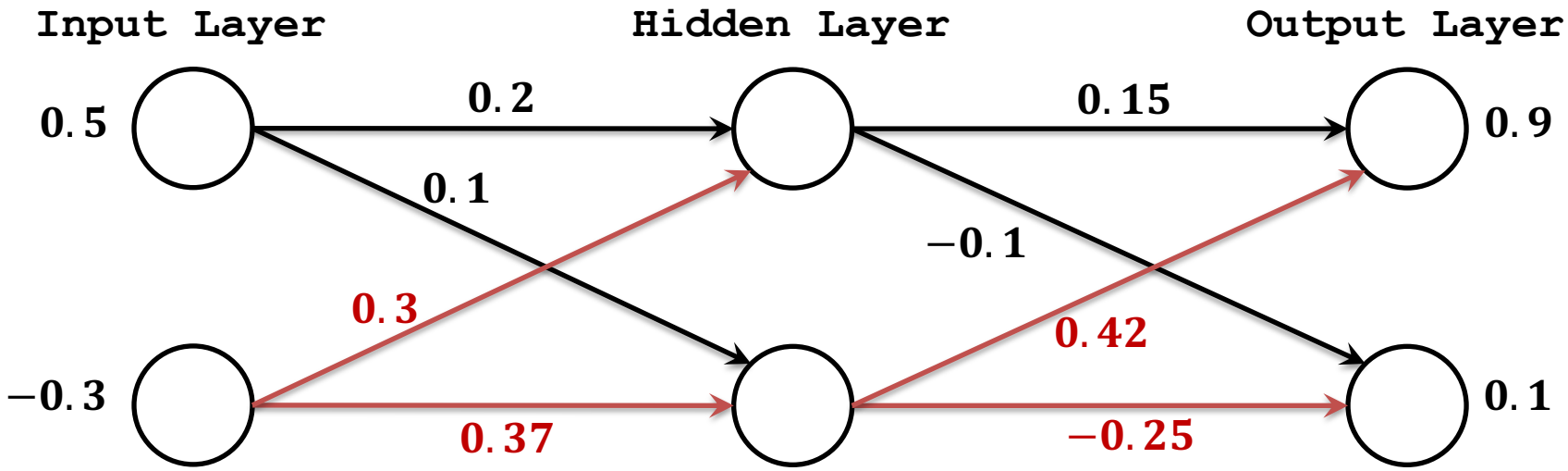
Forward Propagation

	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1



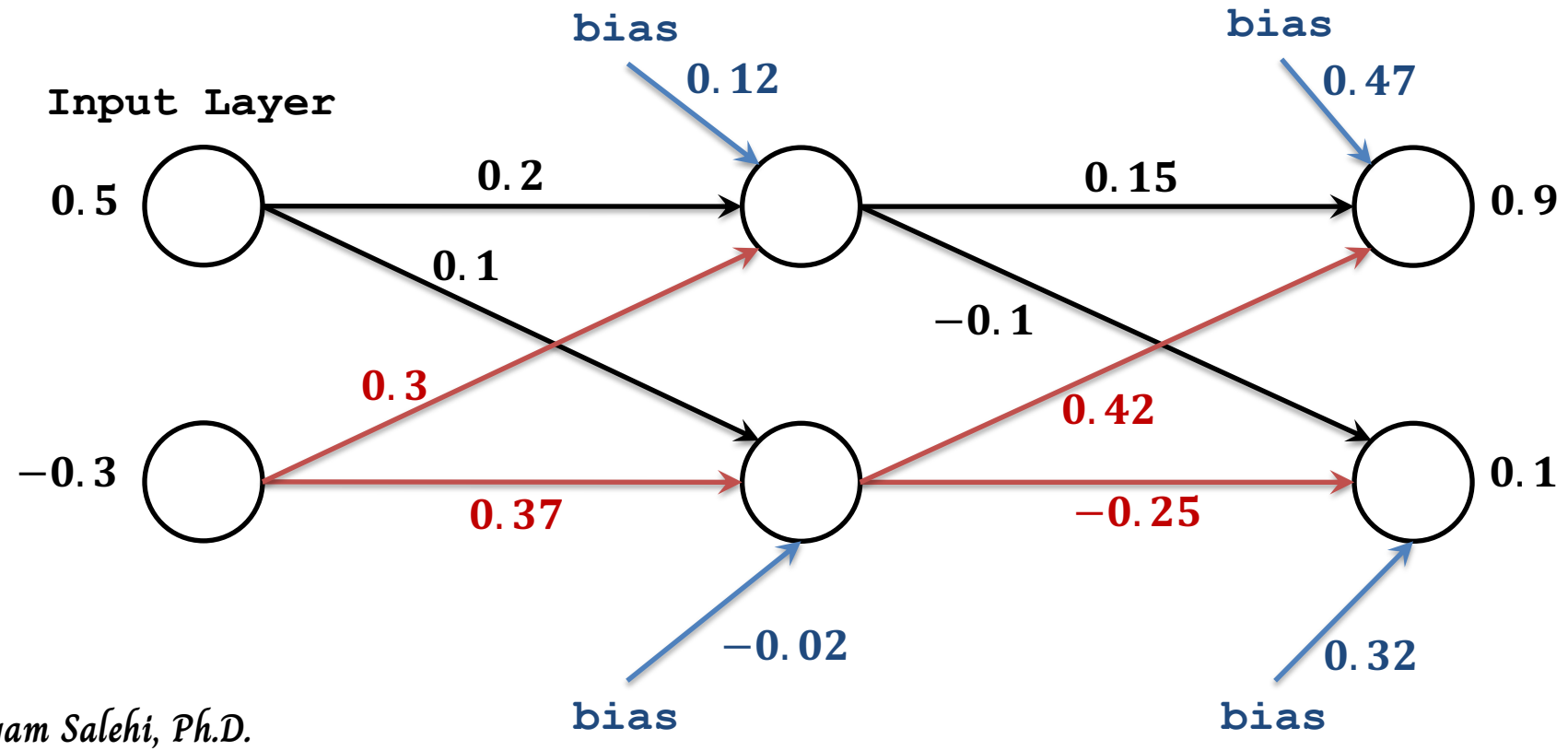
Forward Propagation

	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1



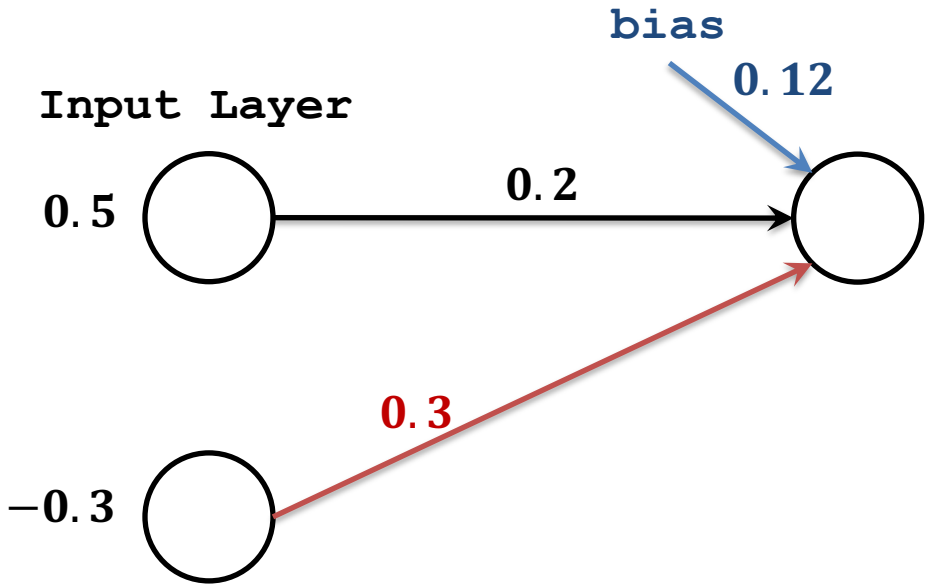
Forward Propagation

	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1



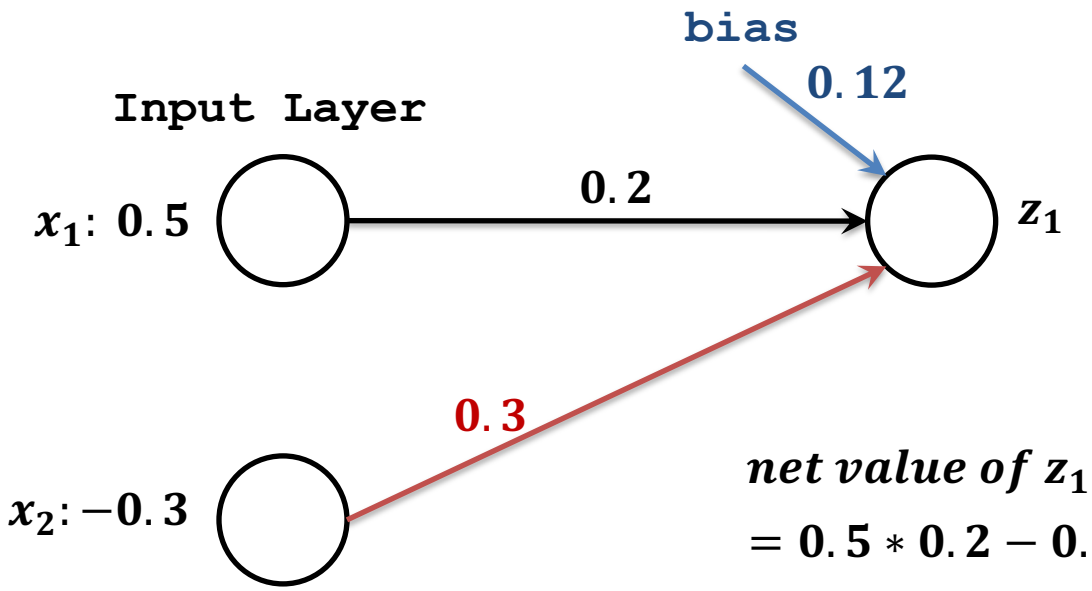
Forward Propagation

	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1



Forward Propagation

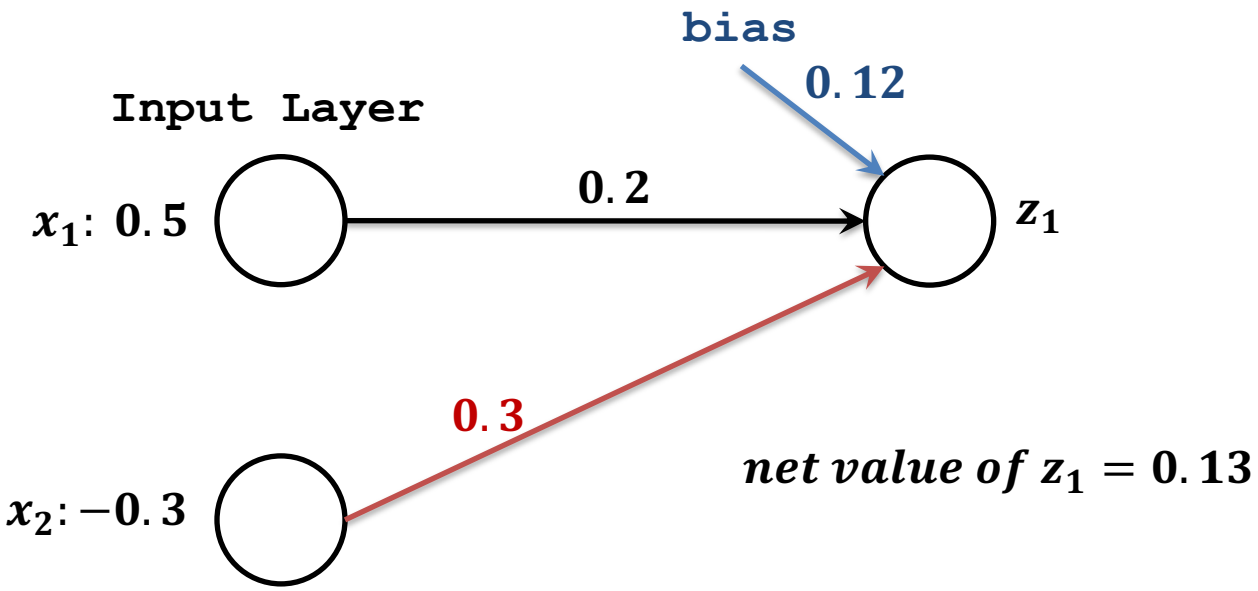
	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1

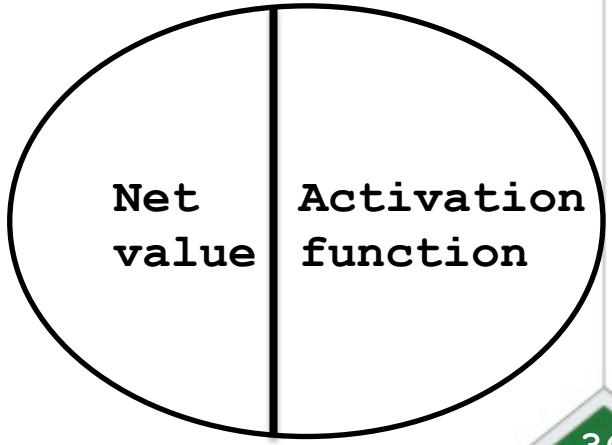


$$net\ value\ of\ z_1 = x_1 w_1 + x_2 w_2 + bias$$
$$= 0.5 * 0.2 - 0.3 * 0.3 + 0.12 = 0.13$$

Forward Propagation

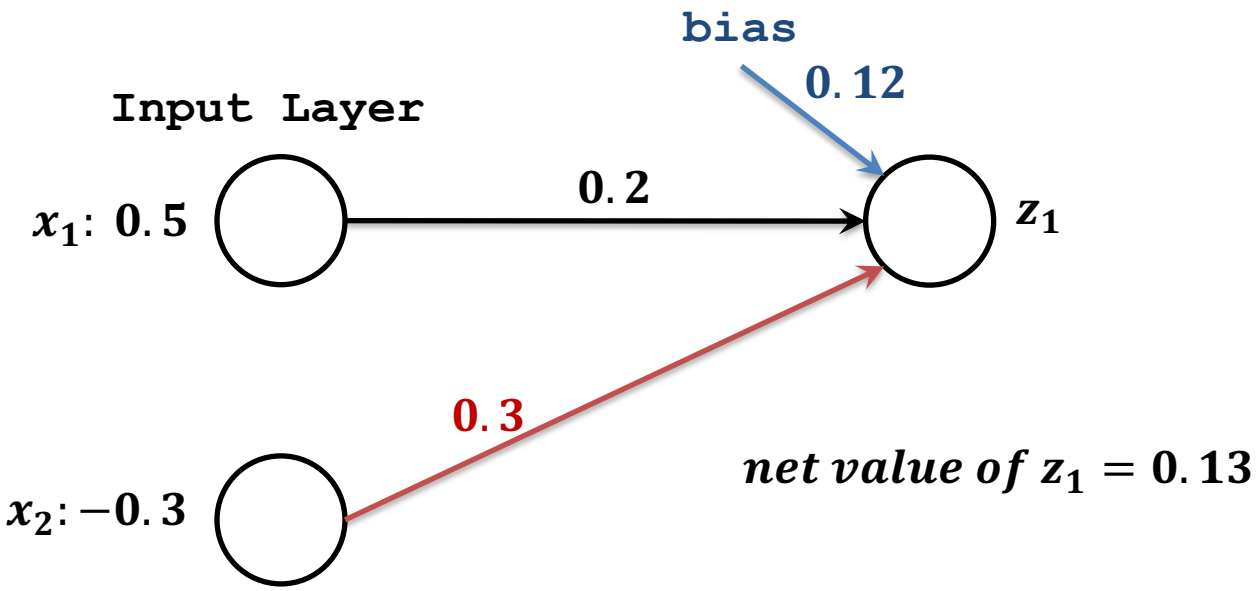
	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1

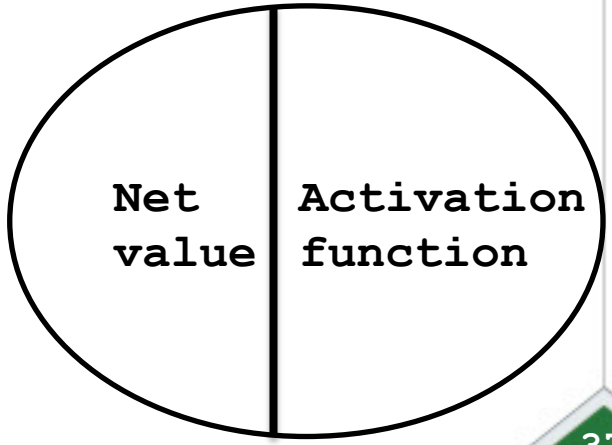




Forward Propagation

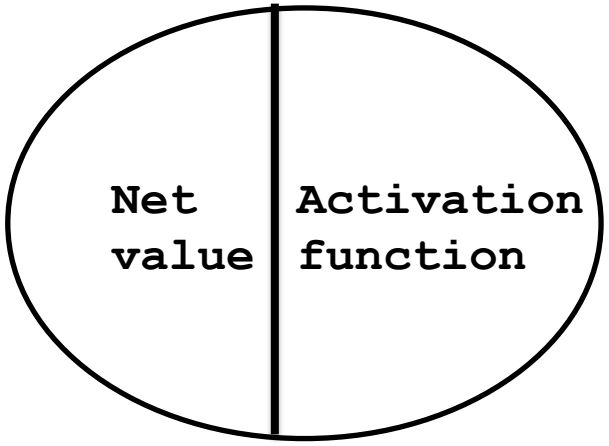
	Feature 1	Feature 2	Output 1	Output 2
Samples	0.5	-0.3	0.9	0.1
	0.4	0.1	0.9	0.8
	0.6	0.8	0.1	0.1



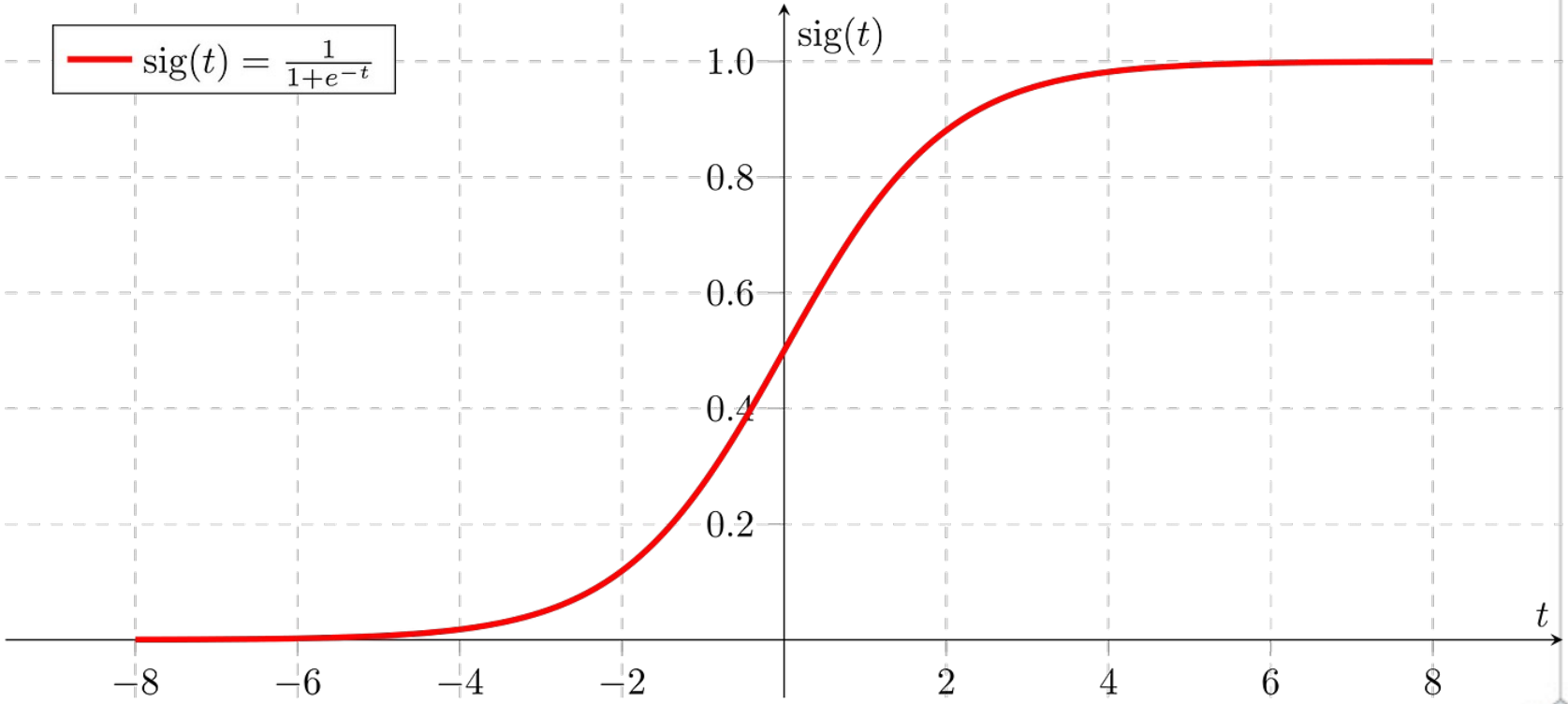


Activation function fires the net value, e.g. scaling it to range $[0,1]$.

Sigmoid: A Famous Activation Function

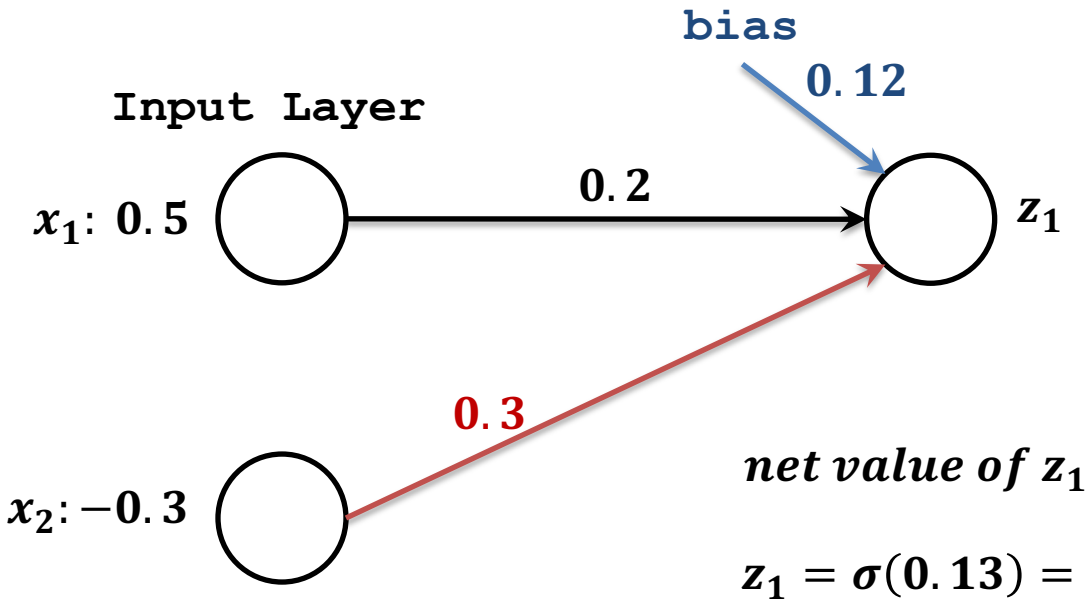


Activation function fires the net value, e.g. scaling it to range $[0, 1]$.



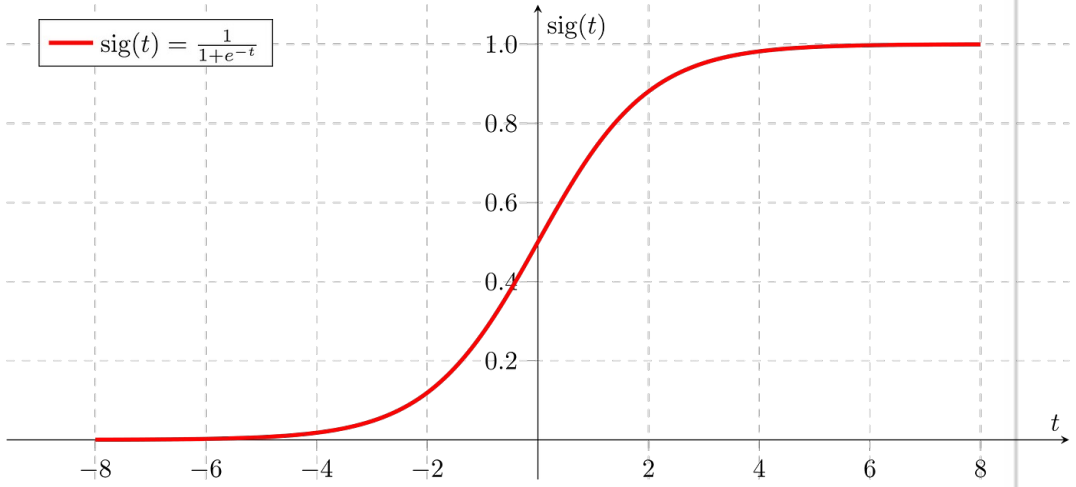
https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

Sigmoid: A Famous Activation Function

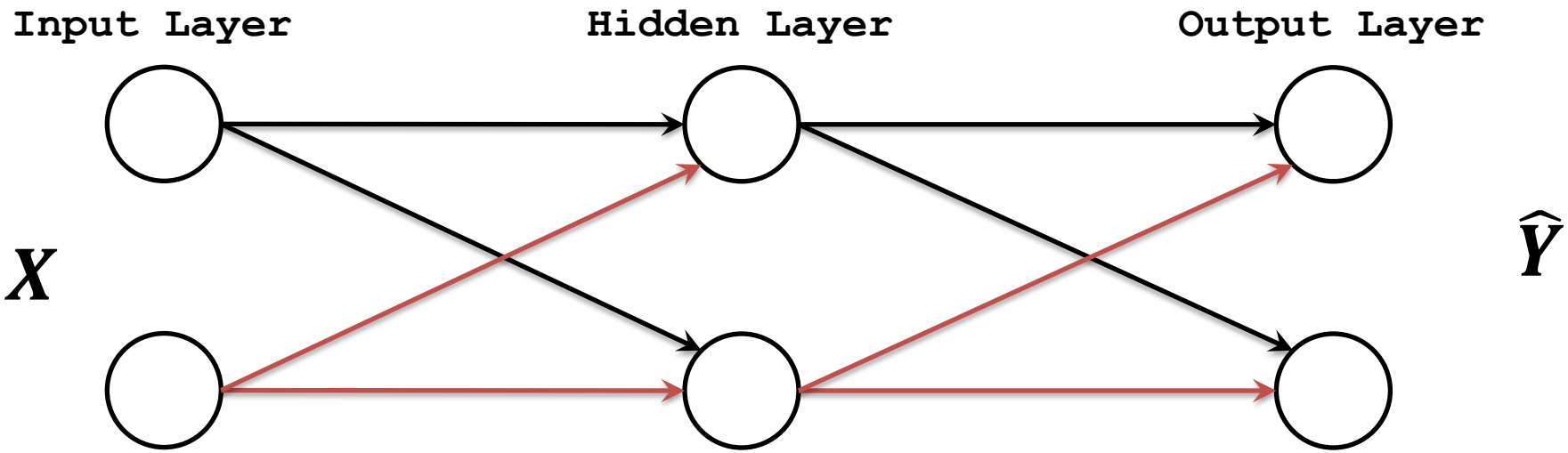


$$net\ value\ of\ z_1 = 0.13$$

$$z_1 = \sigma(0.13) = 0.53$$

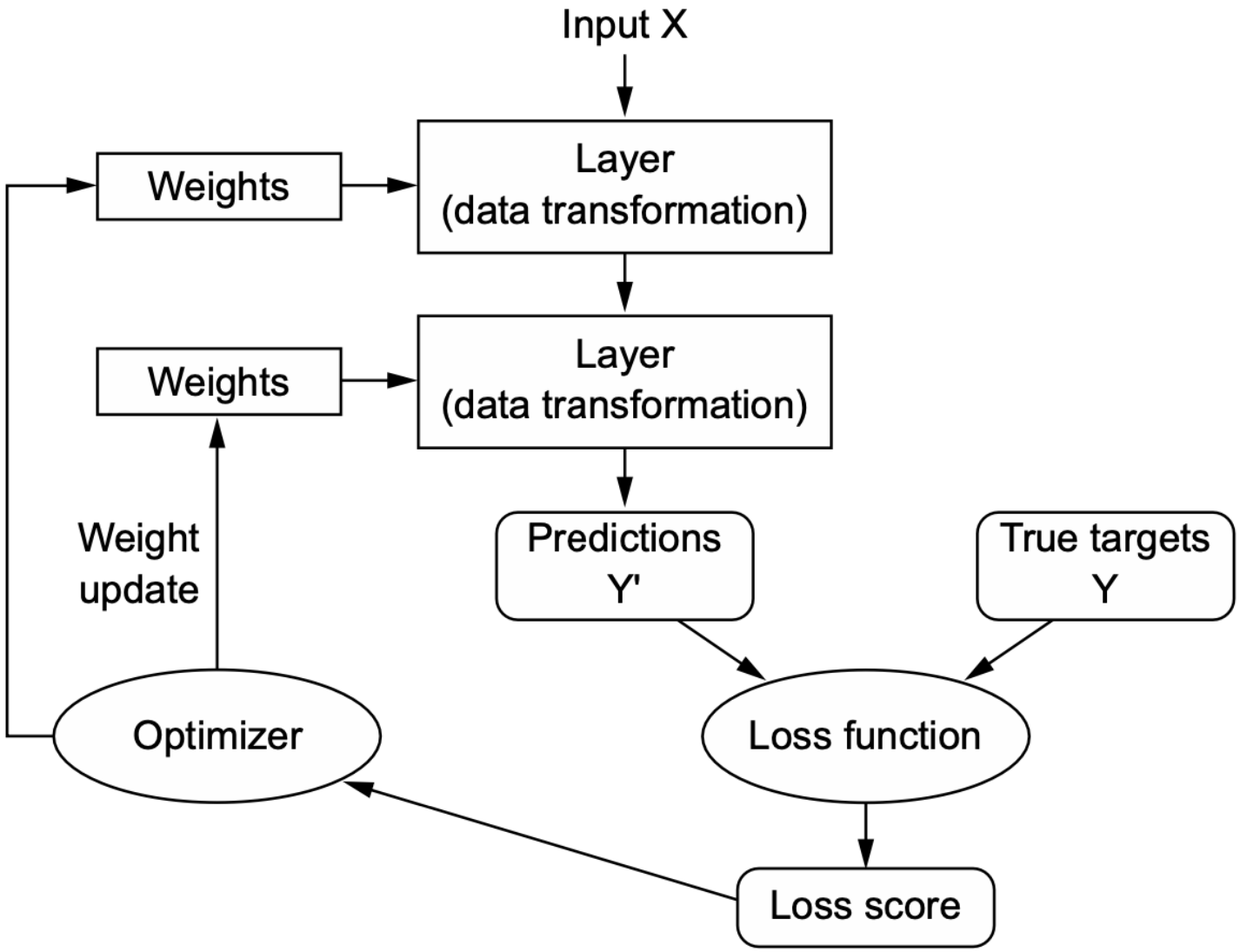


Forward Propagation



Error ----> Loss Function

A Neural Network

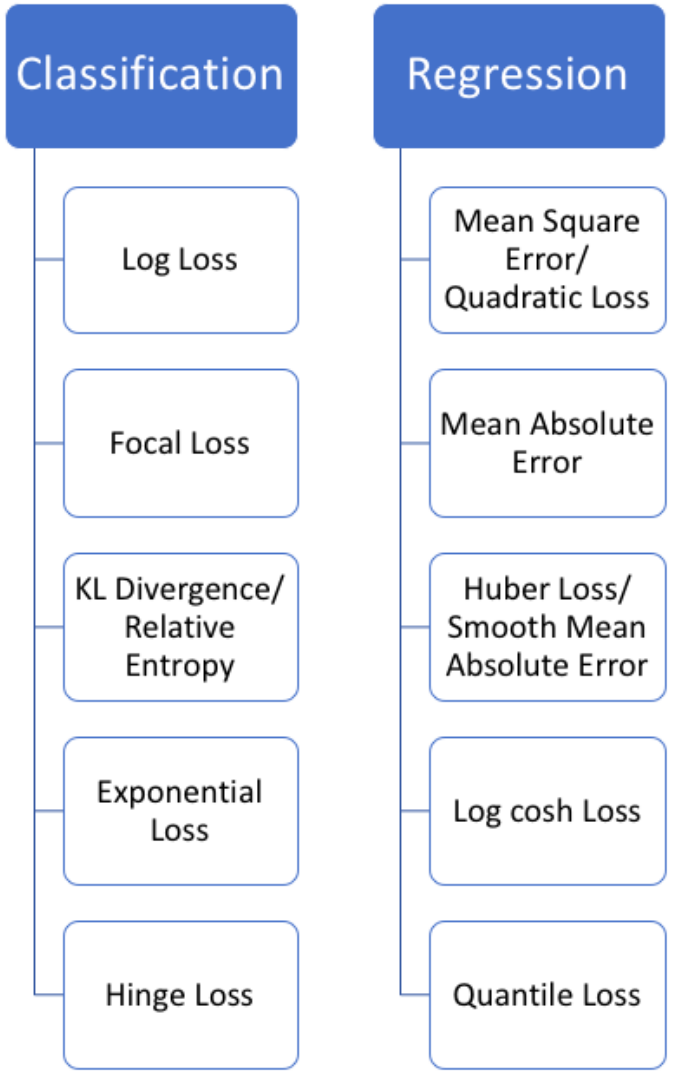


Neural Network ingredient

- Inputs
- Weights
- Some layers
- Some neurons in each layer
- Activation function for each neuron (better: each layer)
- Loss function
- Algorithms for minimizing loss function by updating weights (Optimizer)

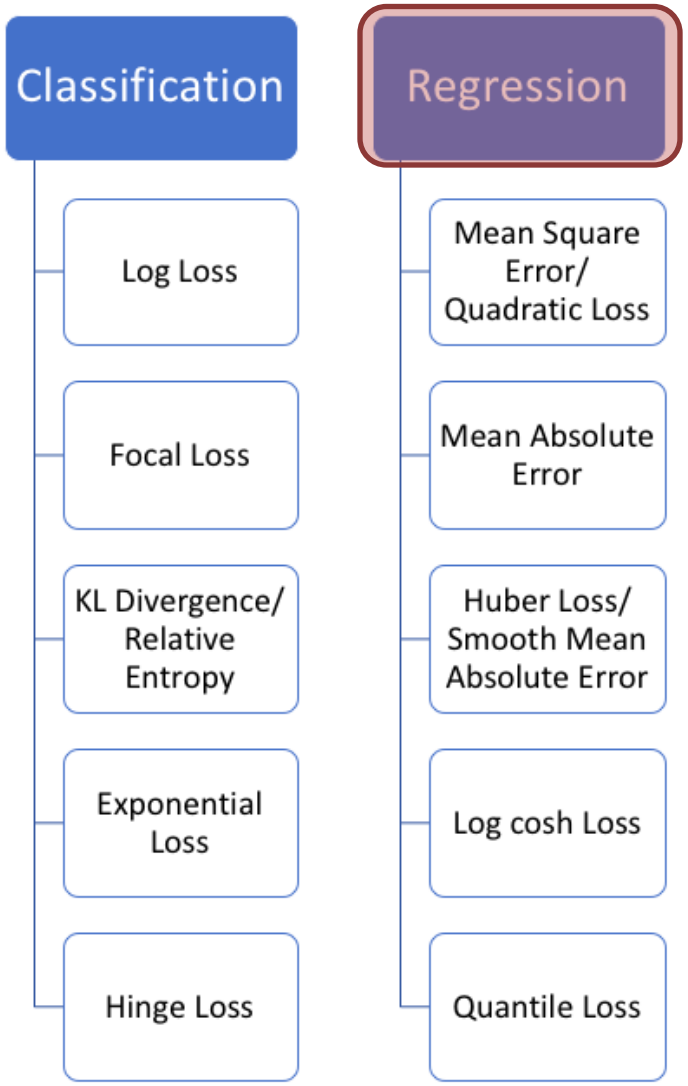
Loss Functions

Loss Functions



https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f

Loss Functions



https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f

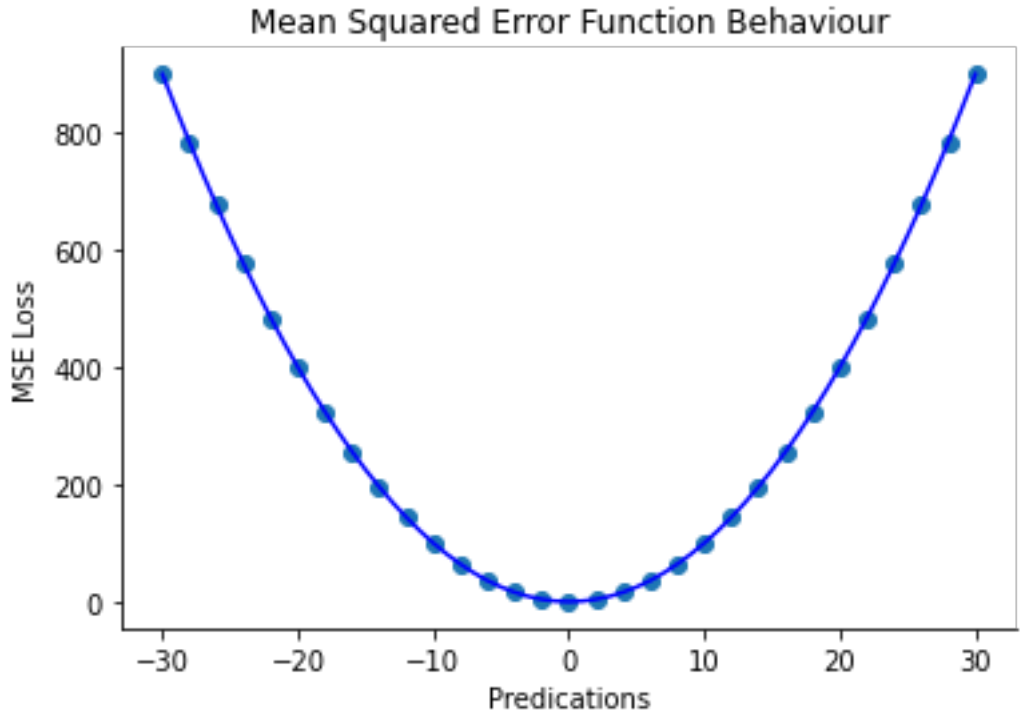
Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

n: *the number of samples*

Y_i: *target (true) data*

$\widehat{Y}_i$: *predicted data*



https://datamonje.com/regression-loss-functions/

Mean Squared Error (MSE)

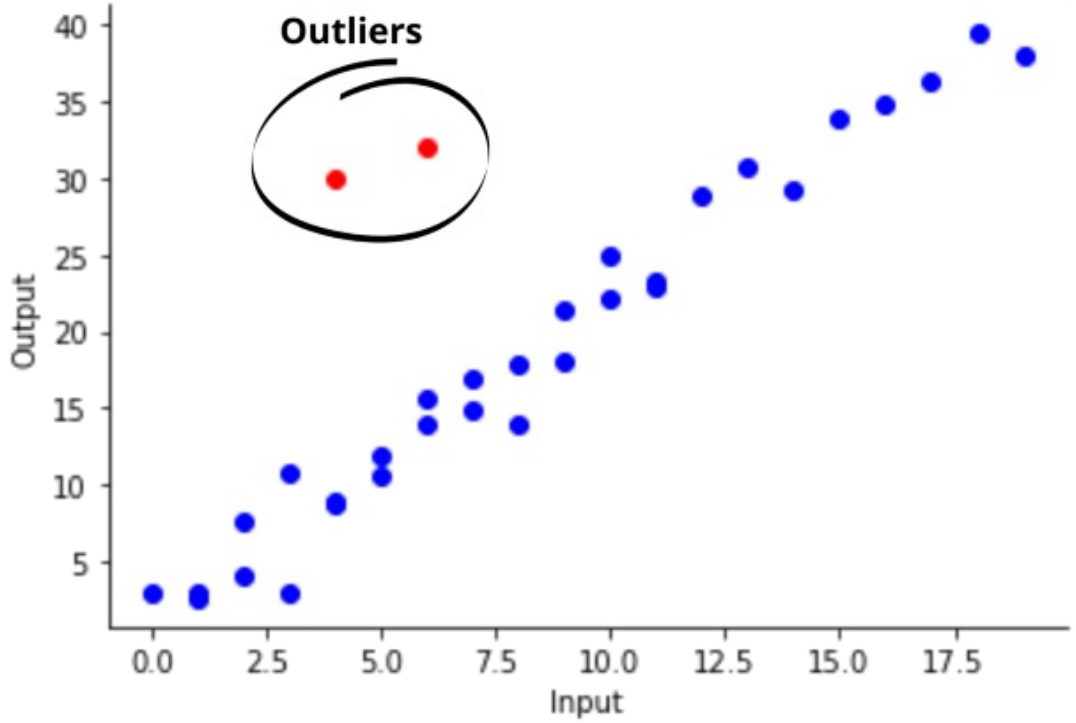
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

n: *the number of samples*

Y_i: *target (true) data*

$\hat{Y}_i$: *predicted data*

- Very sensitive to outliers



https://datamonje.com/regression-loss-functions/

Mean Squared Error (MSE)

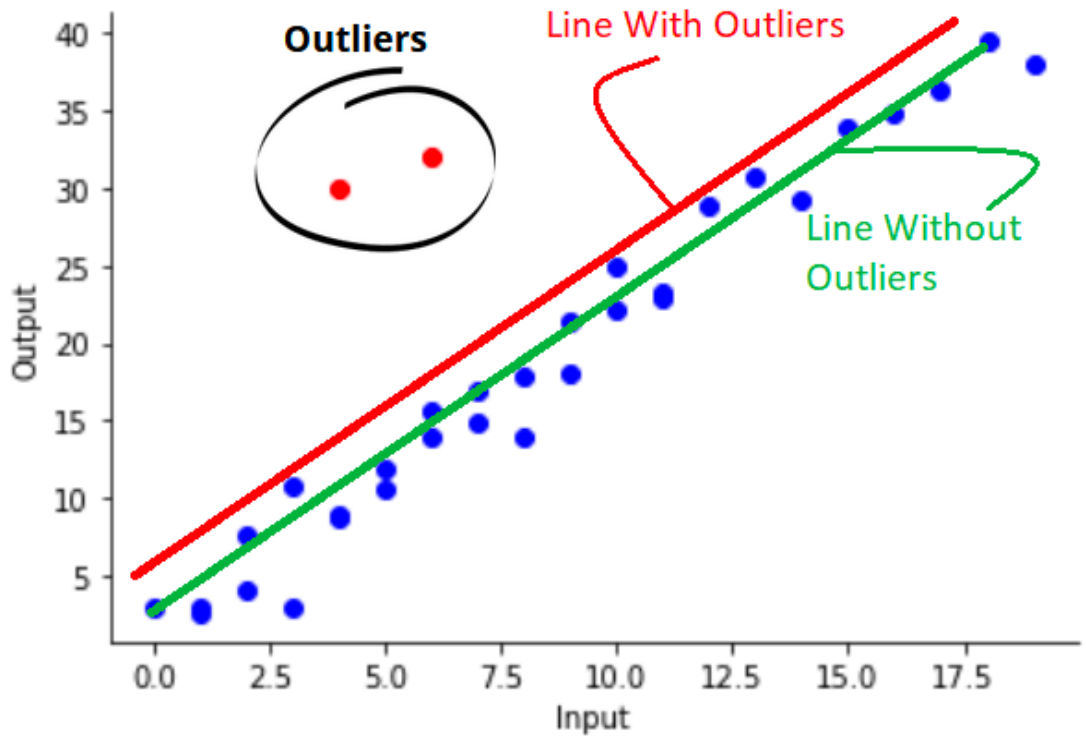
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

n: *the number of samples*

Y_i: *target (true) data*

$\hat{Y}_i$: *predicted data*

- Very sensitive to outliers



https://datamonje.com/regression-loss-functions/

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2}$$

n: *the number of samples*

Y_i: *target (true) data*

$\widehat{Y}_i$: *predicted data*

Root Mean Squared Error (RMSE)

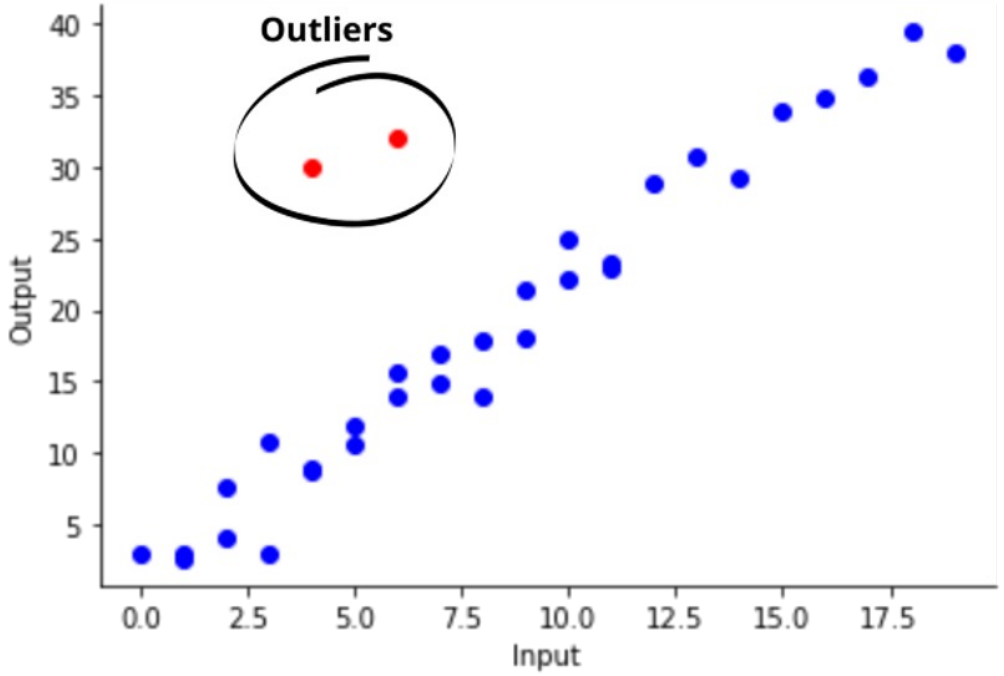
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}$$

n: *the number of samples*

Y_i: *target (true) data*

$\hat{Y}_i$: *predicted data*

- Very sensitive to outliers



https://datamonje.com/regression-loss-functions/

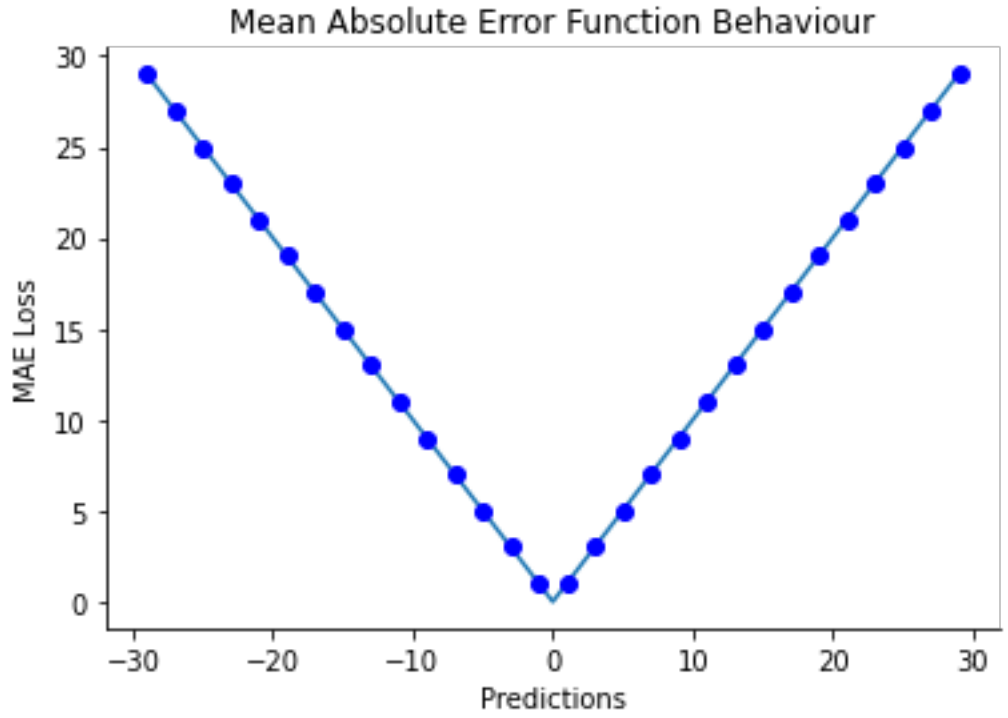
Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \widehat{Y}_i\right|$$

n: *the number of samples*

Y_i: *target (true) data*

$\widehat{Y}_i$: *predicted data*



https://datamonje.com/regression-loss-functions/

Mean Absolute Error (MAE)

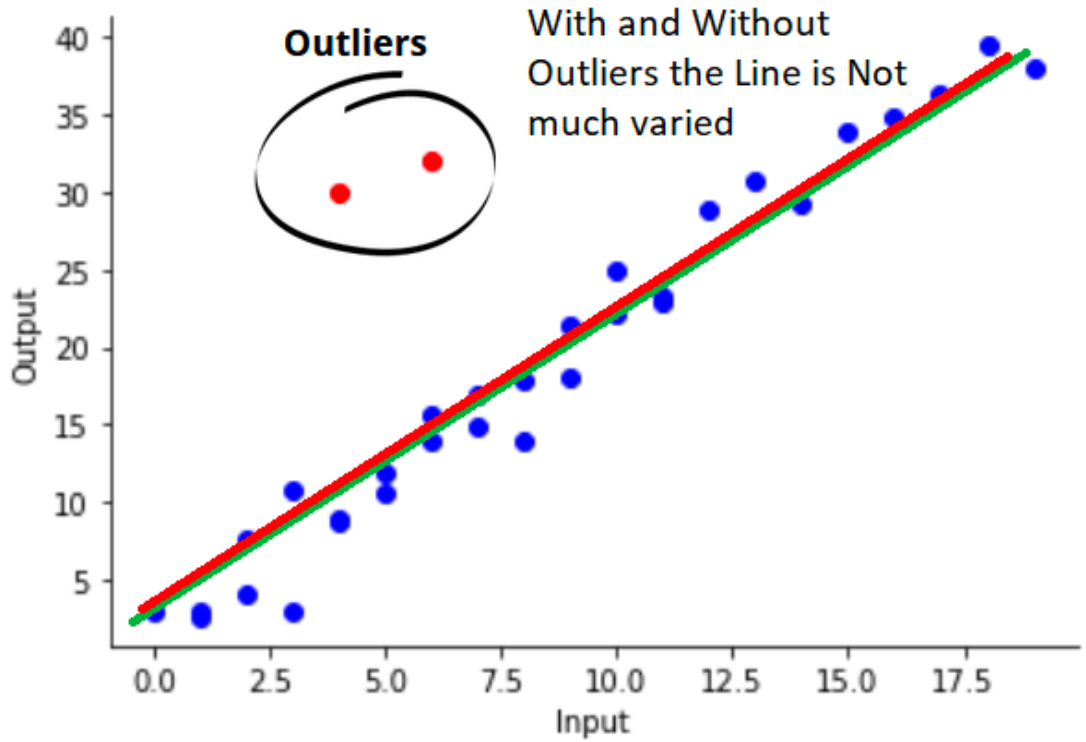
$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

n: *the number of samples*

Y_i: *target (true) data*

$\hat{Y}_i$: *predicted data*

- The Optimization is a little bit complex compared to MSE

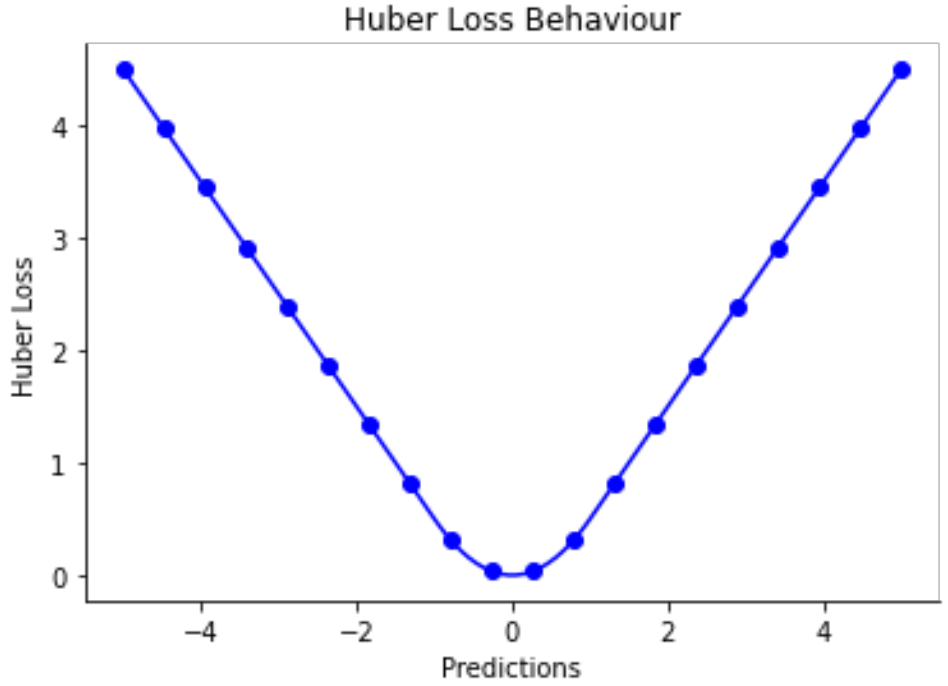


https://datamonje.com/regression-loss-functions/

Huber Loss

$$L_\delta(Y_i, \widehat{Y_i}) = \begin{cases} \frac{1}{2}(Y_i - \widehat{Y_i})^2, & |Y_i - \widehat{Y_i}| \le \delta \\ \delta\left(|Y_i - \widehat{Y_i}| - \frac{1}{2}\delta\right), & |Y_i - \widehat{Y_i}| > \delta \end{cases}$$

$$HuberLoss = \frac{1}{n}\sum_{i=1}^{n} L_\delta(Y_i, \widehat{Y_i})$$



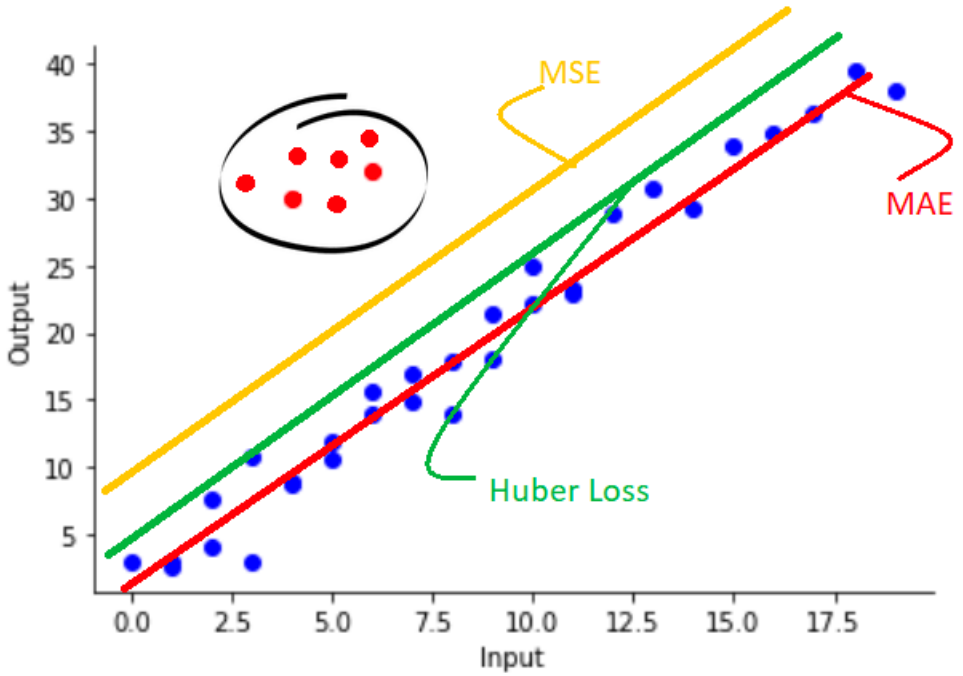
https://datamonje.com/regression-loss-functions/

Huber Loss

$$L_\delta(Y_i, \hat{Y}_i) = \begin{cases} \frac{1}{2}(Y_i - \hat{Y}_i)^2, & |Y_i - \hat{Y}_i| \le \delta \\ \delta\left(|Y_i - \hat{Y}_i| - \frac{1}{2}\delta\right), & |Y_i - \hat{Y}_i| > \delta \end{cases}$$

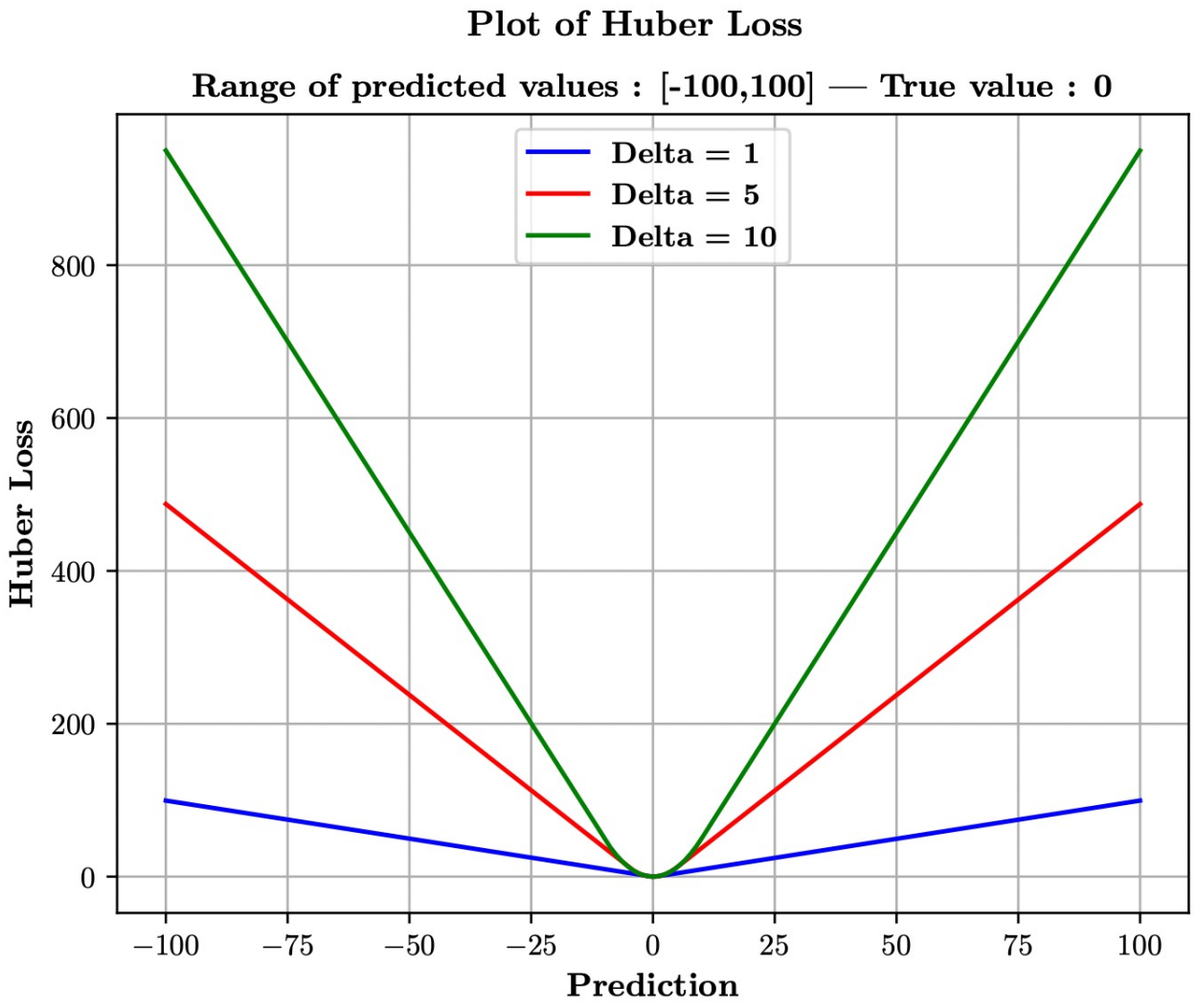
$$HuberLoss = \frac{1}{n}\sum_{i=1}^{n} L_\delta(Y_i, \hat{Y}_i)$$

- The equation is a bit complex
- we also need to adjust the δ based on our requirement



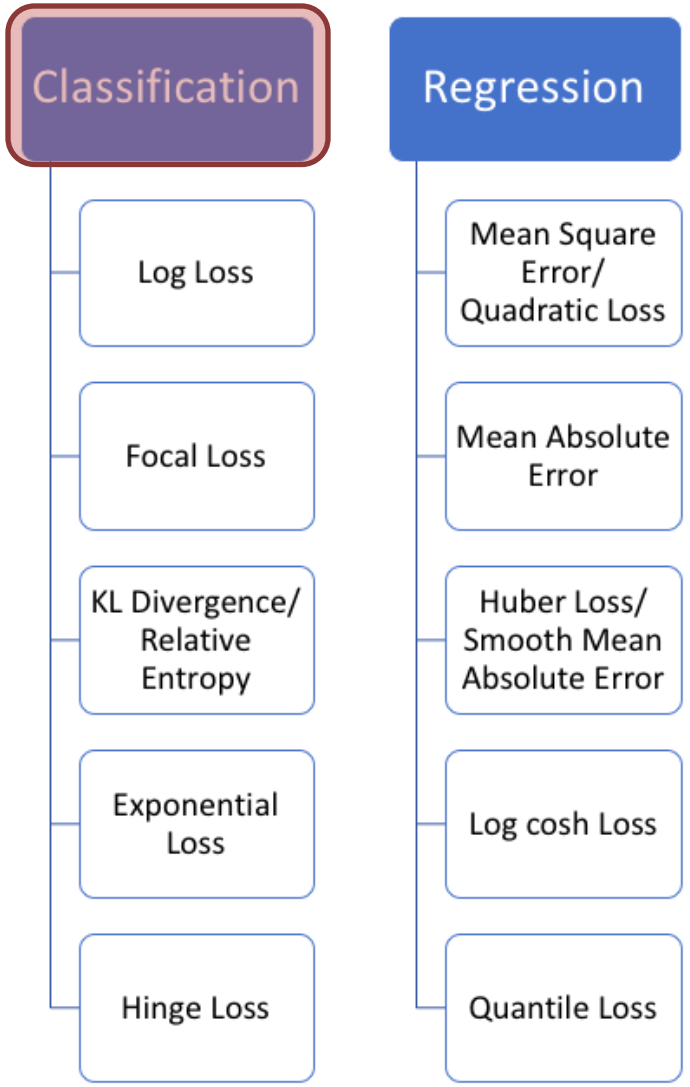
https://datamonje.com/regression-loss-functions/

Huber Loss



https://datasciencepreparation.com/blog/articles/what-loss-functions-can-be-used-for-regression-which-one-is-better-for-outliers/

Loss Functions

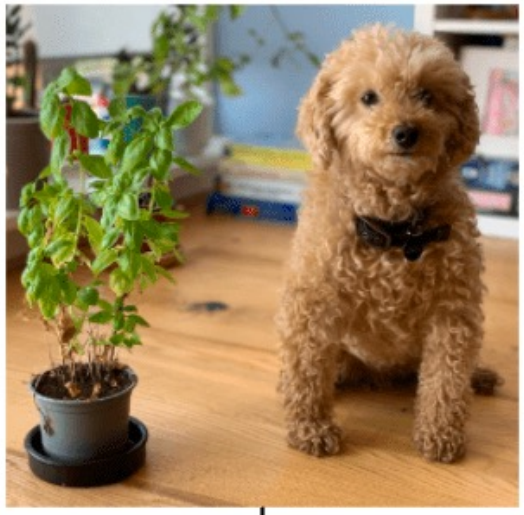


https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f

3 Types of Classification

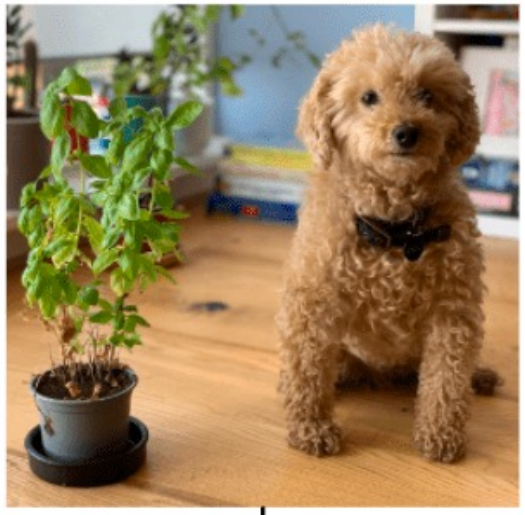
- The predicted value is a probability distribution.

Binary Classification



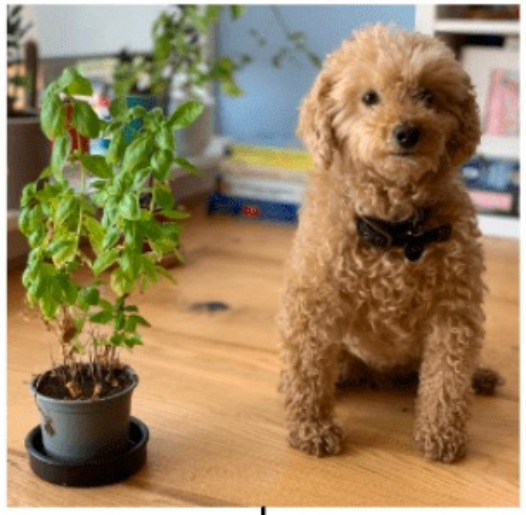
Dog 0.9 | Not Dog 0.1

Multiclass Classification



Dog 0.5 | Cat 0.09 | Bus 0.01 | Plant 0.4

Multilabel Classification



Dog 0.8 | Cat 0.2 | Bus 0.04 | Plant 0.7

https://www.mathworks.com/help/deeplearning/ug/multilabel-image-classification-using-deep-learning.html

Binary Cross Entropy

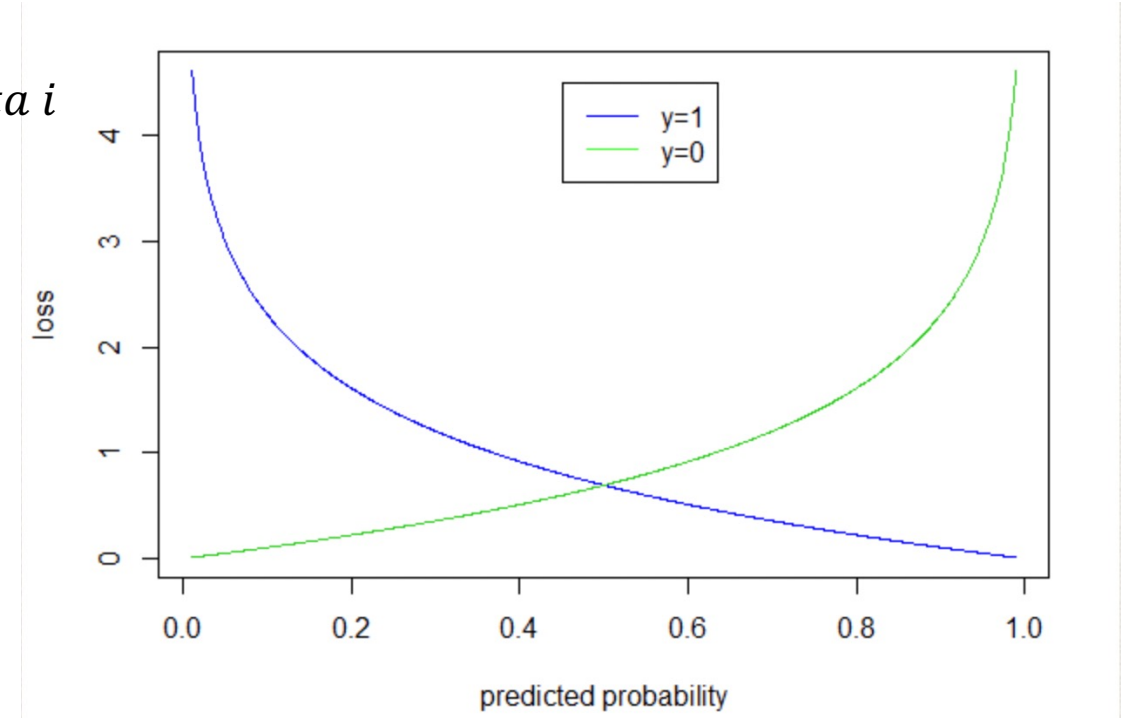
- Widely used for 2 classes

$$CrossEntropy = -\frac{1}{n}\left[\sum_{i=1}^{n} Y_i \log p_i + (1 - Y_i)\log(1 - p_i)\right]$$

n: *the number of samples*

Y_i: *target (true) class label* (0 *or* 1)

p_i: *predicted probability for the class of data i*



https://www.datasciencepreparation.com/blog/articles/what-is-cross-entropy-loss/

Cross Entropy for Multi-label Classification

$$CrossEntropy = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i + (1 - Y_i)\log(1 - p_i)\right]$$

c: *the number of classes*

n: *the number of samples*

Y_i: *target (true) class label* (0 *or* 1)

p_i: *predicted probability for the class of data i*

Cross Entropy for Multi-class Classification

$$CrossEntropy = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i\right]$$

c: *the number of classes*

n: *the number of samples*

Y_i: *probability of target class in* $[0,1]$

p_i: *predicted probability for the class of data i*

Cross Entropy for Multi-class Classification

$$CrossEntropy = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i\right]$$

c: *the number of classes*

n: *the number of samples*

Y_i: *probability of target class in* $[0,1]$

p_i: *predicted probability for the class of data i*

Example:
We have 4 different classes to classify:

The target label: $[0.3 \quad 0.5 \quad 0.2 \quad 0.\ ]$
The predicted probability: $[0.2. \quad 0.3. \quad 0.4. \quad 0.1]$

Cross Entropy loss: $-(0.3\times\log 0.2 + 0.5\times\log 0.3 + 0.2\times\log 0.4 + 0\times\log 0.1) = 1.83$

Categorical Cross Entropy for Multi-class Classification

$$CrossEntropy = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i\right]$$

c: *the number of classes*

n: *the number of samples*

Y_i: *target (true) class label* (0 *or* 1)

p_i: *predicted probability for the class of data i*

The target for multi-class classification is a one-hot vector, meaning it has 1 on a single position and 0's everywhere else.

Categorical Cross Entropy for Multi-class Classification

$$CrossEntropy = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i\right]$$

c: *the number of classes*

n: *the number of samples*

Y_i: *target (true) class label* $(0 \text{ or } 1)$

p_i: *predicted probability for the class of data i*

The target for multi-class classification is a one-hot vector, meaning it has 1 on a single position and 0's everywhere else.

Example:
We have 4 different classes to classify:

The target label: $[0. \quad 1. \quad 0 \quad 0.\ ]$
The predicted probability: $[0.2. \quad 0.3. \quad 0.4. \quad 0.1]$

Cross Entropy loss: $-(1 \times \log 0.3) = 1.74$

Dissimilarity of Two Probability Distribution

- Kullback-Leibler divergence (KL divergence) or relative entropy:

$$D_{KL}(Y_i||p) = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i - Y_i \log Y_i\right]$$

c: *the number of classes*

n: *the number of samples*

Y_i: *probability of target class in* [0,1]

p_i: *predicted probability for the class of data i*

Dissimilarity of Two Probability Distribution

$$D_{KL}(Y_i||p) = -\frac{1}{n}\sum_{j=1}^{n}\left[\sum_{i=1}^{c} Y_i \log p_i - Y_i \log Y_i\right]$$

Y_i: *probability of target class in* $[0,1]$

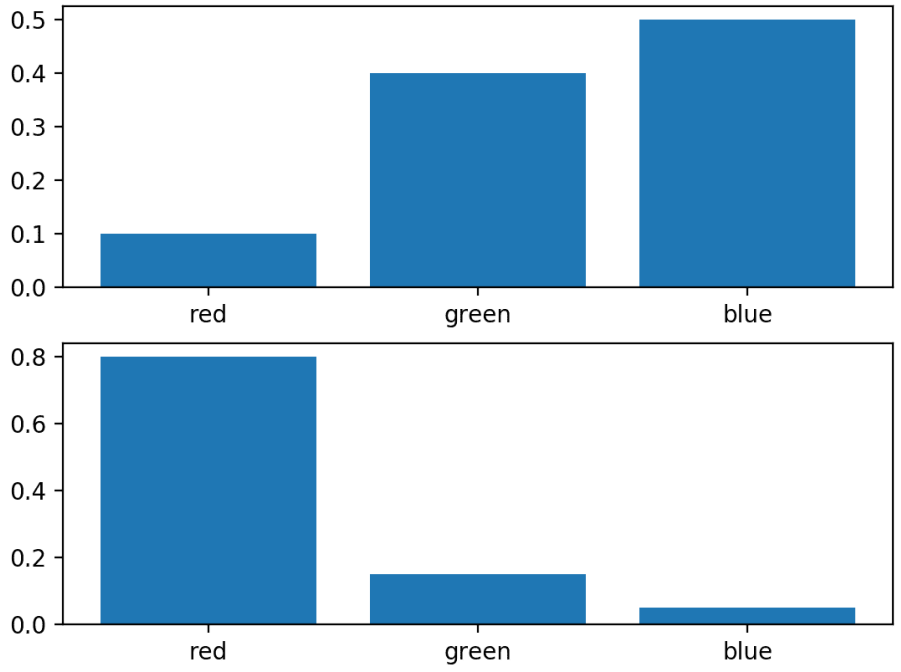
p_i: *predicted probability for the class of data i*

Example:

events = ['red', 'green', 'blue']

$Y = [0.10, \quad 0.40, \quad 0.50]$

$p = [0.80, \quad 0.15, \quad 0.05]$



$D_{KL}(Y \,||\, p)$: 1.93 *bits*

$D_{KL}(p \,||\, Y)$: 2.02 *bits*

https://machinelearningmastery.com/divergence-between-probability-distributions/

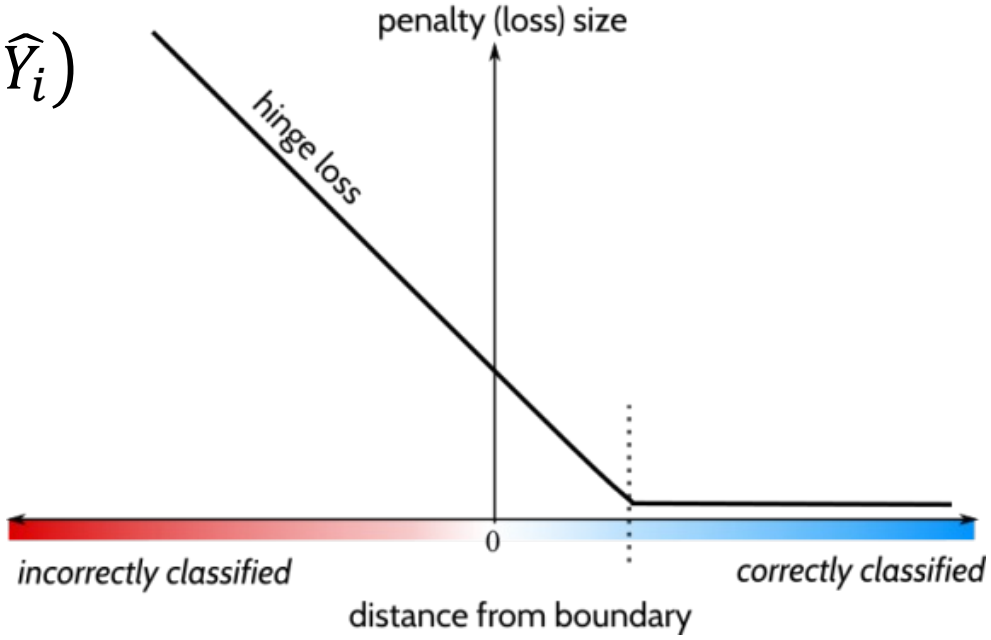
Hinge Loss

- Used for "maximum-margin" classification
  - most notably for support vector machines.
  - faster than cross entropy but accuracy s degraded
  - correcting predictions which are not confident enough

$$HingeLoss = \sum_{j=1}^{n} \max(0, 1 - Y_i \times \widehat{Y}_i)$$

Y_i: $target\ (true)\ class\ label\ (-1\ or\ 1)$

$\widehat{Y}_i$: $predicted\ data\ label\ (-1\ or\ 1)$



https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1

Other Loss Functions

- Contrastive Loss
  - considering margins
- Triplet Ranking Loss
  - used in image (e.g. CNN)

https://machinelearningmastery.com/divergence-between-probability-distributions/

Summary

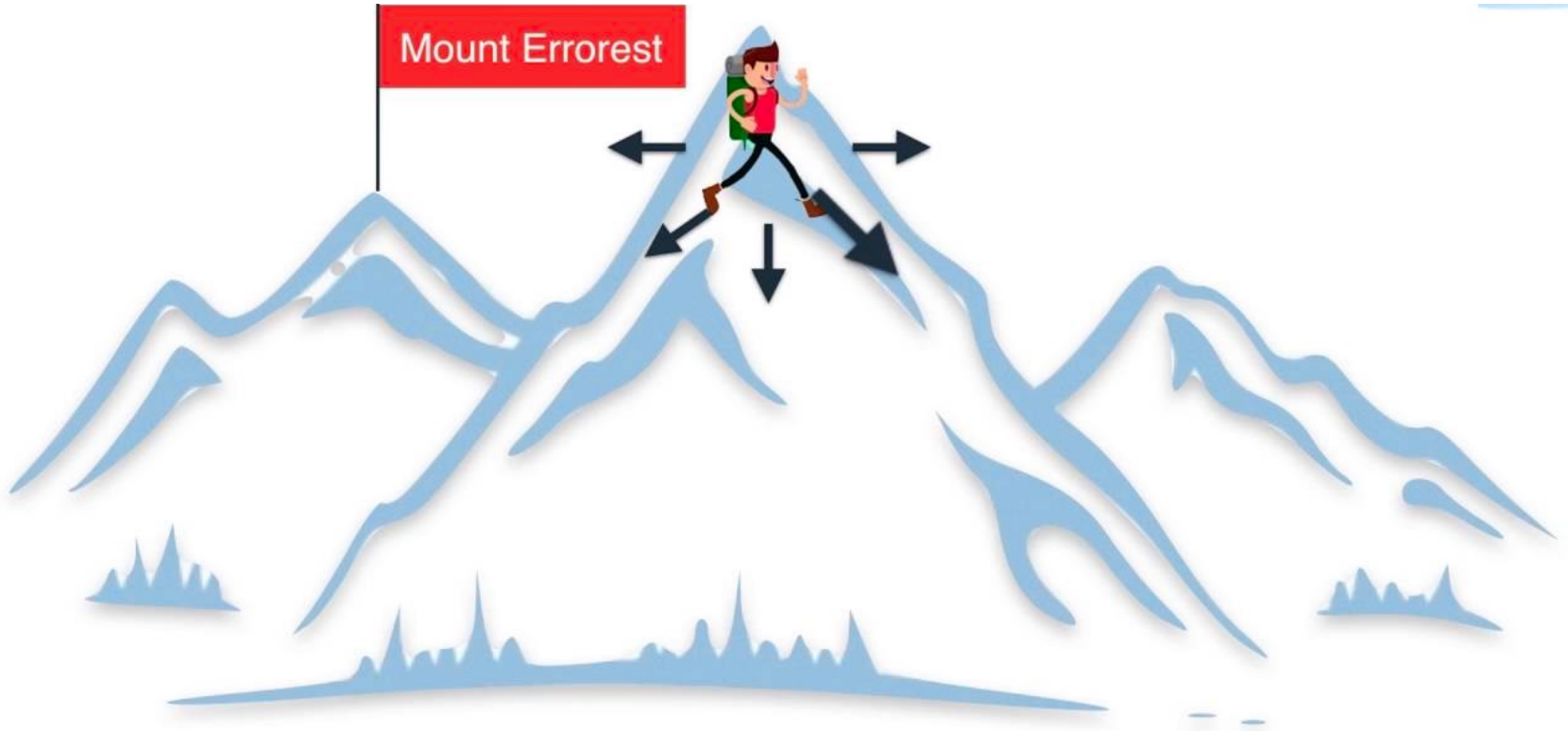
1	Loss functions in Regression based problem	a. Mean Square Error Loss
		b. Mean Absolute Error Loss
		c. Huber Loss
2	Loss functions in Binary classification-based problem	a. Binary Cross Entropy Loss
		b. Hinge Loss
3	Loss functions in Multiclass classification-based problem	a. Multiclass Cross Entropy Loss
		b. Spare Multiclass Cross Entropy Loss
		c. Kullback Leibler Divergence Loss

https://www.analyticsvidhya.com/blog/2022/11/advance-guide-on-interview-questions-of-deep-learning/

Optimization

How to minimize the loss function?

- How to minimize a functions?



https://towardsdatascience.com/gradient-descent-back-prop-and-derivatives-5b286bb8d8b5

How to minimize the loss function?

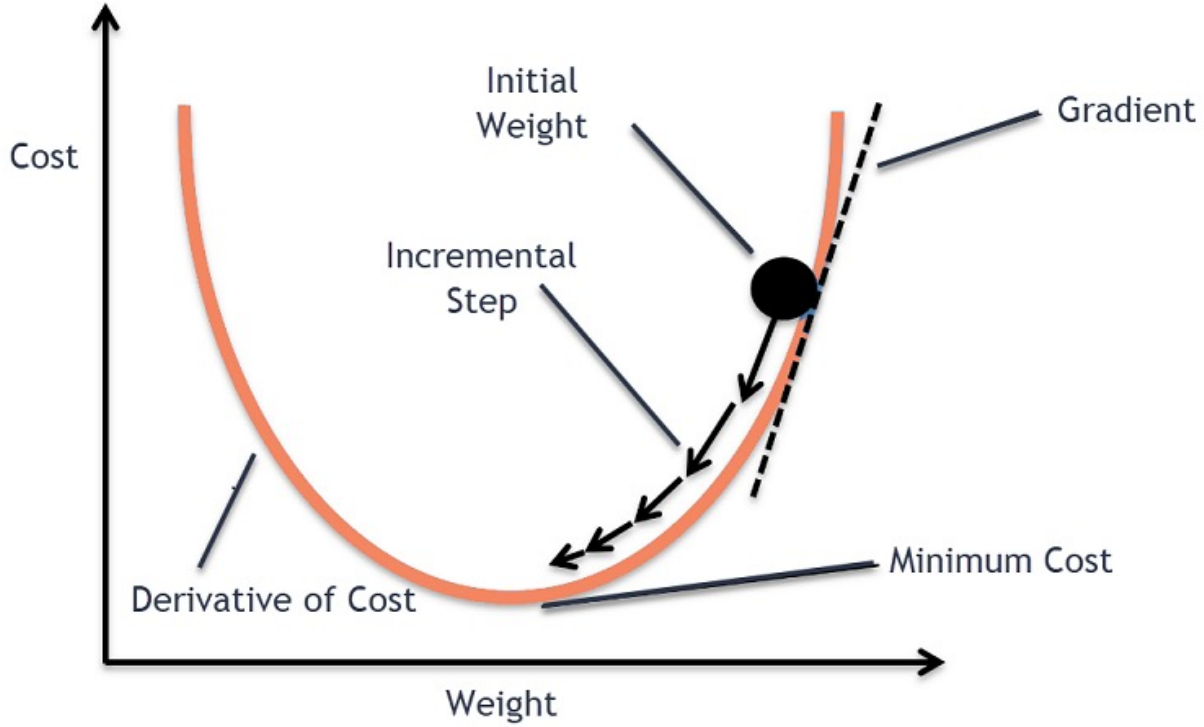
- How to minimize a functions?

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

How to minimize the loss function?

- How to minimize a functions?
  - The opposite side of the slope
  - calculating of gradient by differentiation of cost function
  - An epoch is a complete pass through all samples.
  - η is learning rate (the step size), i.e. how fast we update the weights.

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

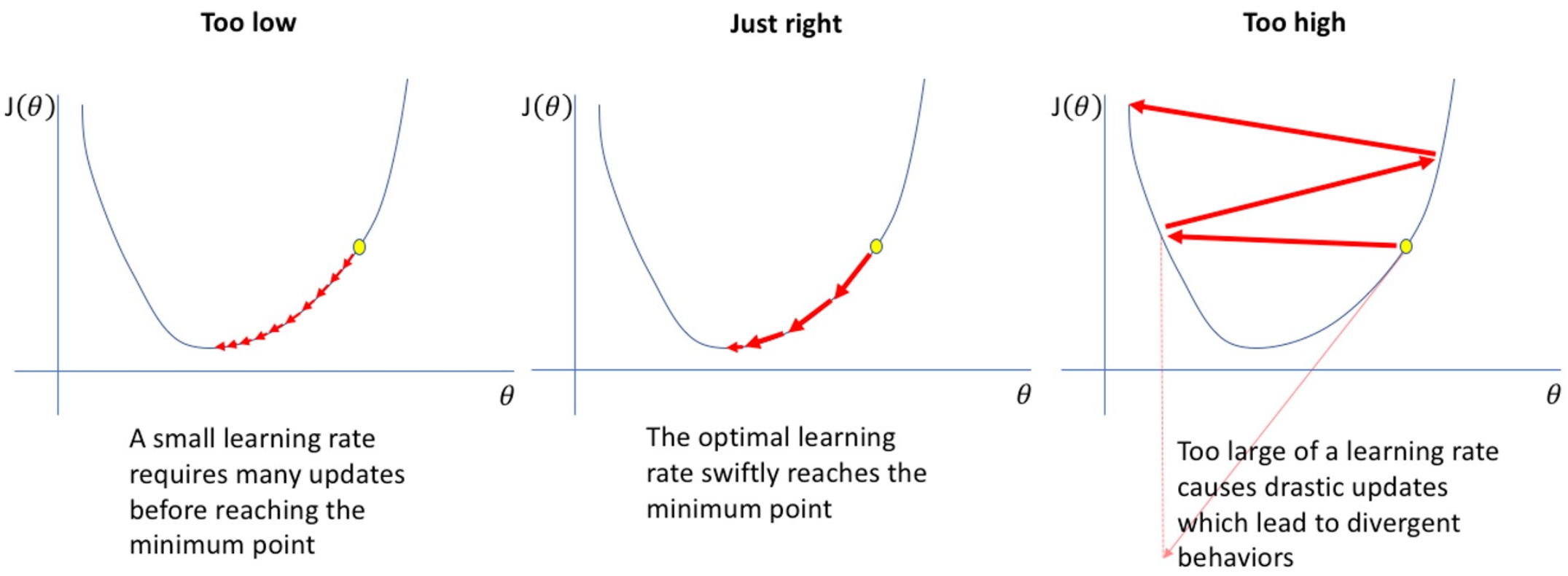


https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

The impact of learning rate

- How to minimize a functions?
  - η is learning rate (the step size), i.e. how fast we update the weights

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

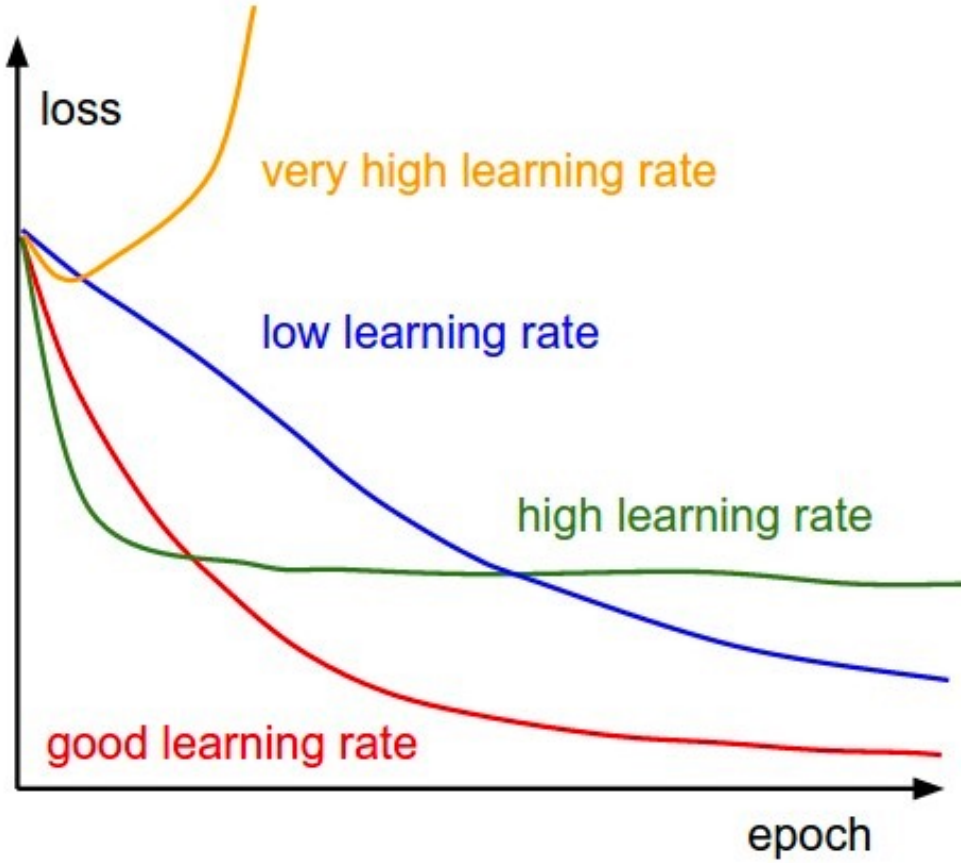


https://duchesnay.github.io/pystatsml/optimization/optim_gradient_descent.html

The impact of learning rate

- How to minimize a functions?
  - η is learning rate (the step size), i.e. how fast we update the weights

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

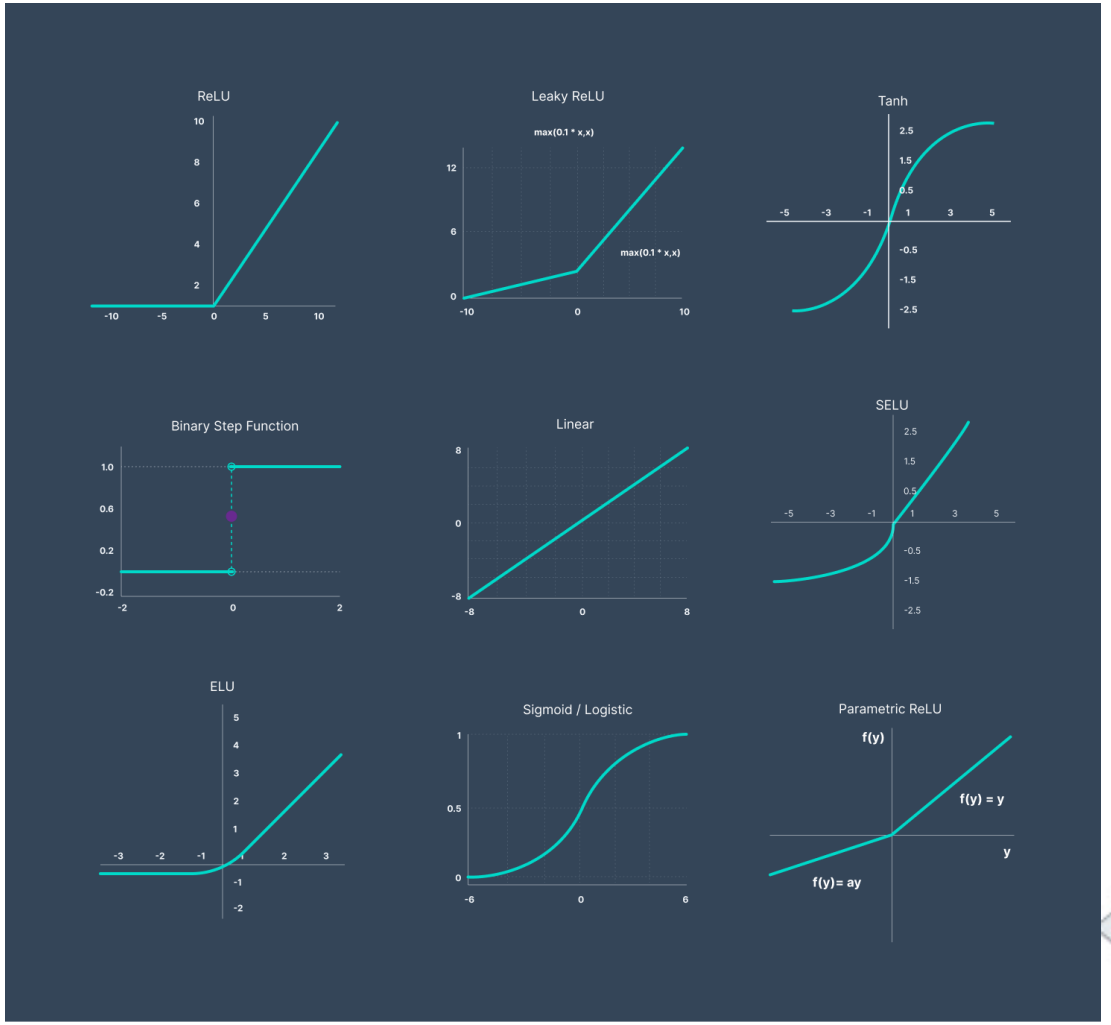


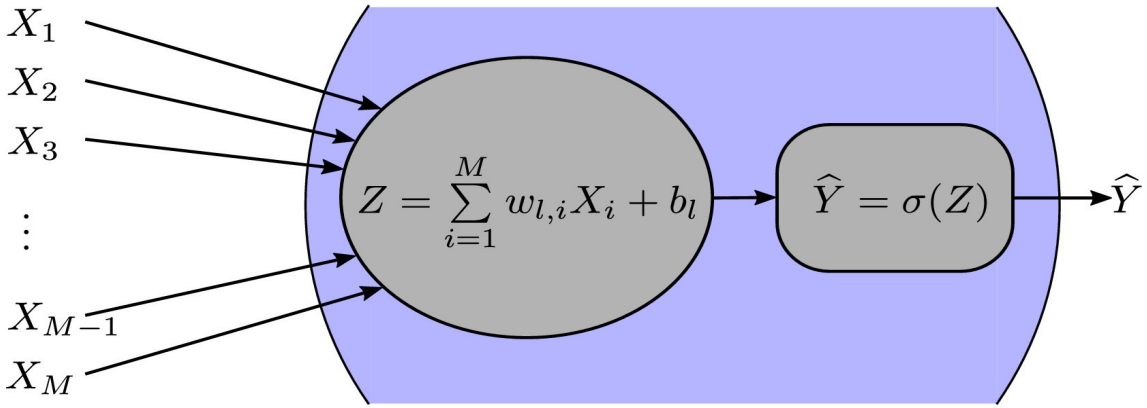
https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

Activation Functions

Activation Functions

- Being more powerful
- Learning complex and complicated data
- Representing non-linearity of data





(b) Representation of a neuron.

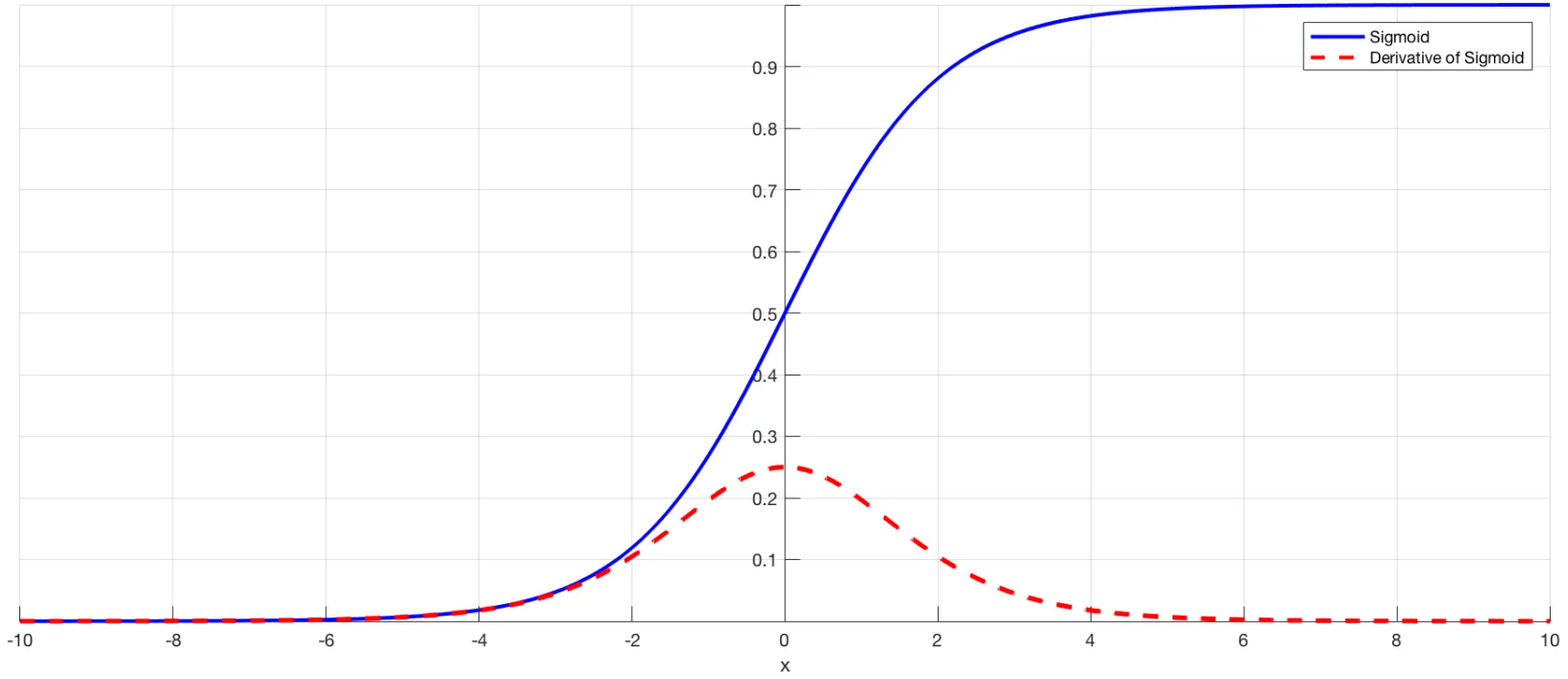
https://www.v7labs.com/blog/neural-networks-activation-functions

Sigmoid

- Any input will be scaled into [0,1]
- Suffering from vanishing gradient
- Used for output layer in binary classification

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

$$\sigma'(x) = \sigma(x) \times \big(1 - \sigma(x)\big)$$



https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

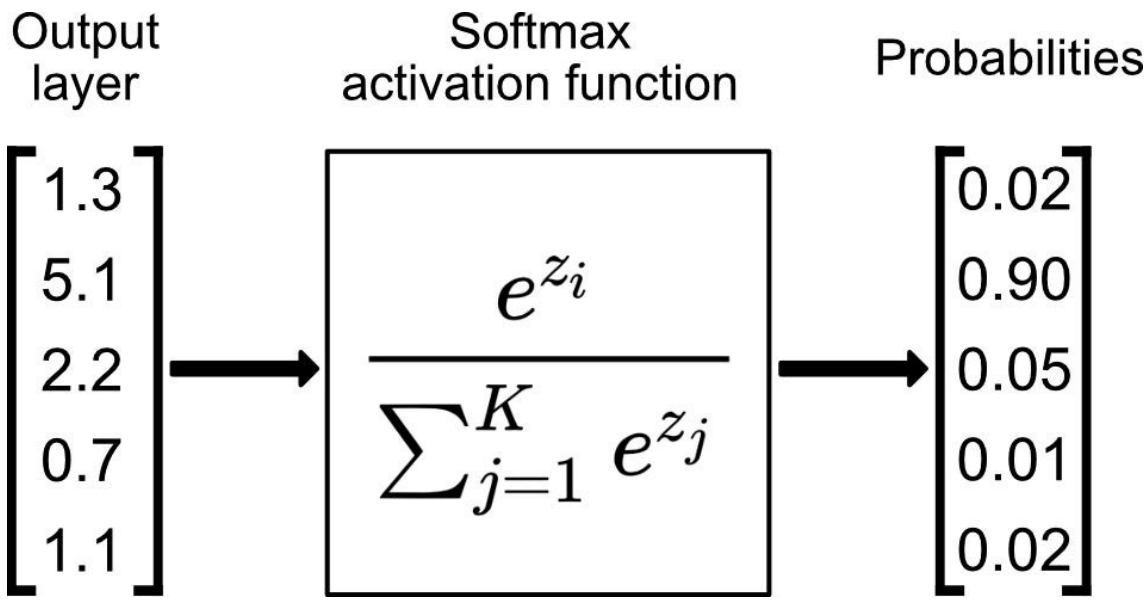
Softmax

- Softmax
  - How to change numbers to probability?

$$\sigma_j(z) = \frac{e^{z_j}}{\sum_{j=1}^{n} e^{z_j}}$$

$z: input$

$z_j: the\ j-th\ component\ of\ z$

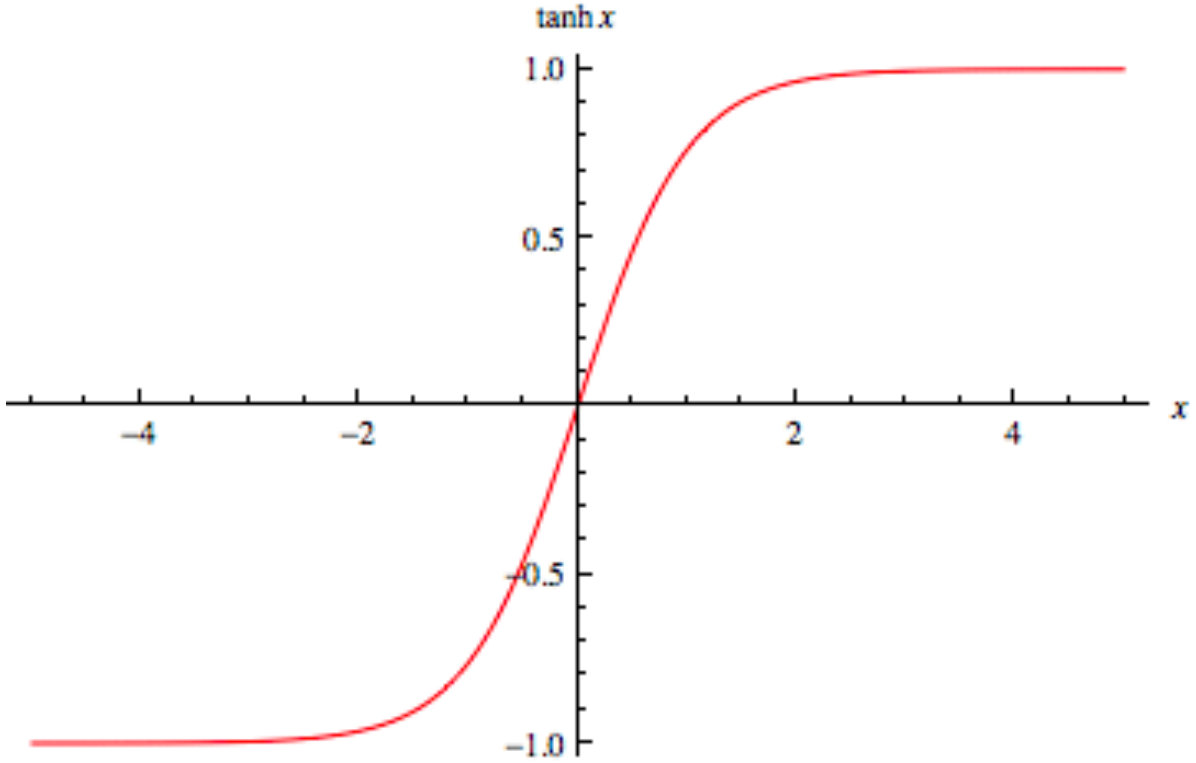


https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60

Hyperbolic Tangent (tanh)

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

z: $input$



https://mathworld.wolfram.com/HyperbolicTangent.html

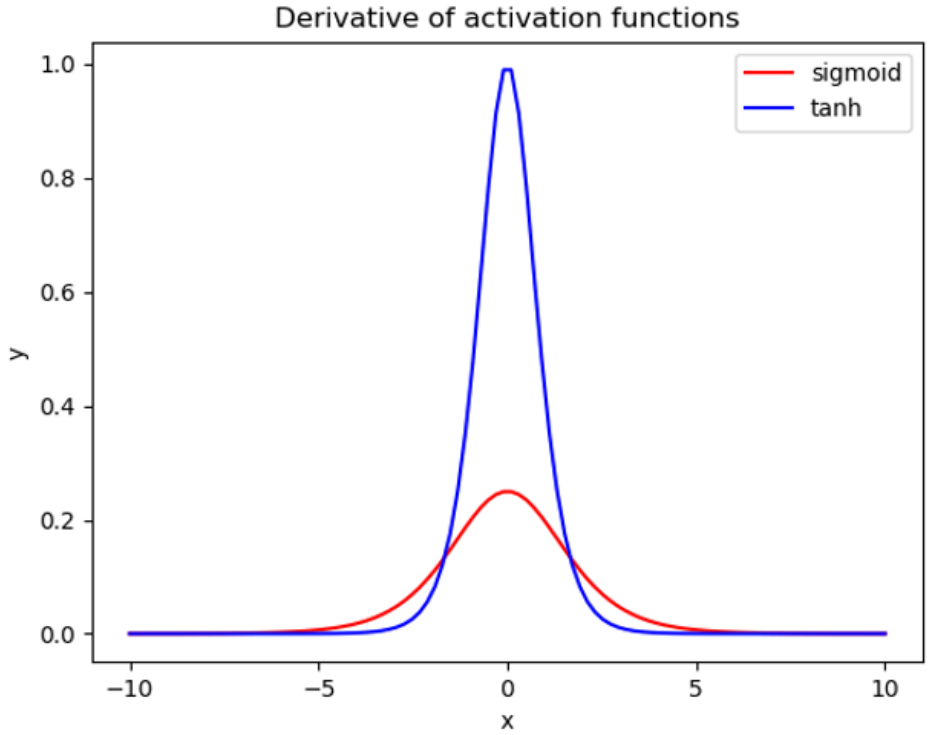
Hyperbolic Tangent (tanh)

- tanh vs. sigmoid
  - If we want strong gradients and big learning steps, we should use the tanh instead of sigmoid.
  - Another difference is that the output of tanh is symmetric around zero leading to faster convergence.
  - the problem of vanishing gradient is still stated.

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

$z: input$

$$tanh'(z) = 1 - tanh(z)^2$$



https://mathworld.wolfram.com/HyperbolicTangent.html

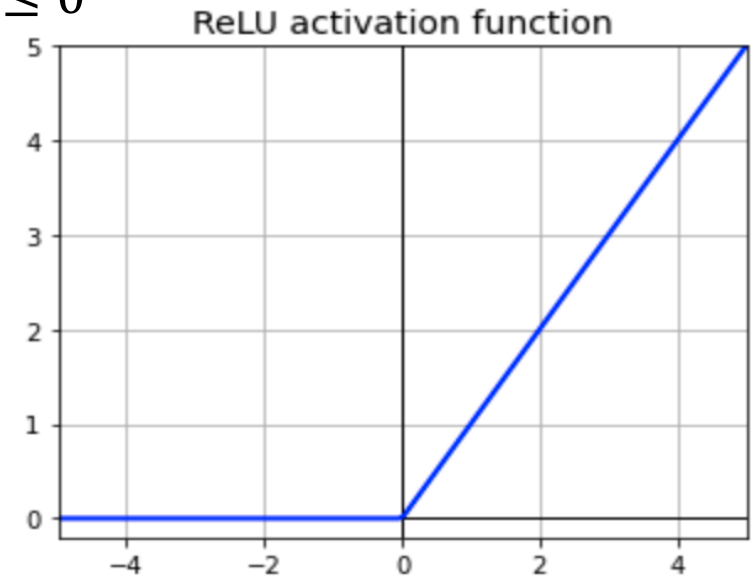
Rectified Linear Unit (ReLU)

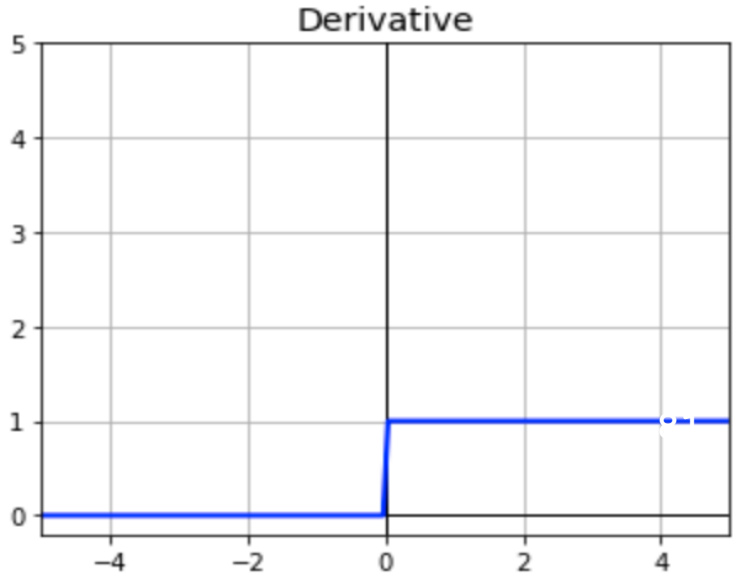
- ReLU:
  - When the input is negative, the output is 0 and the gradient will die (e.g. initializing weights with normal distribution).
  - continuous but not differentiable
  - very fast to compute (compare to sigmoid and tanh)

$$ReLU(z) = \max(0, z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

$z: input$

$$ReLU'(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$





https://towardsdatascience.com/why-rectified-linear-unit-relu-in-deep-learning-and-the-best-practice-to-use-it-with-tensorflow-e9880933b7ef

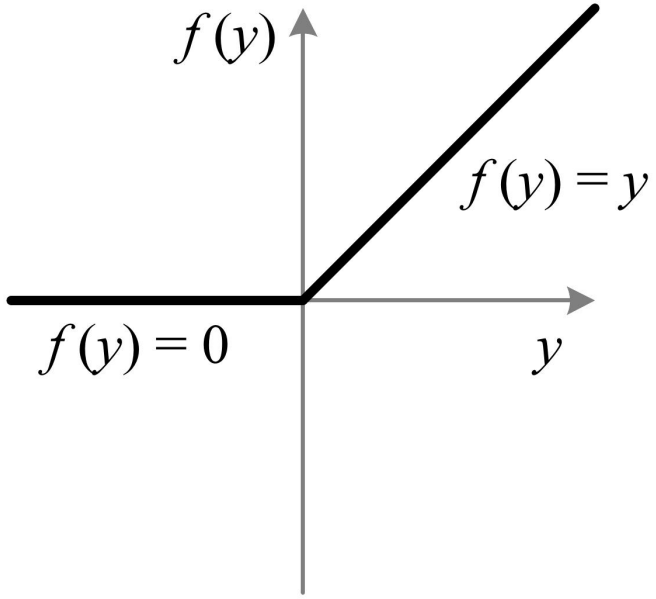
Parametric ReLU (PReLU)

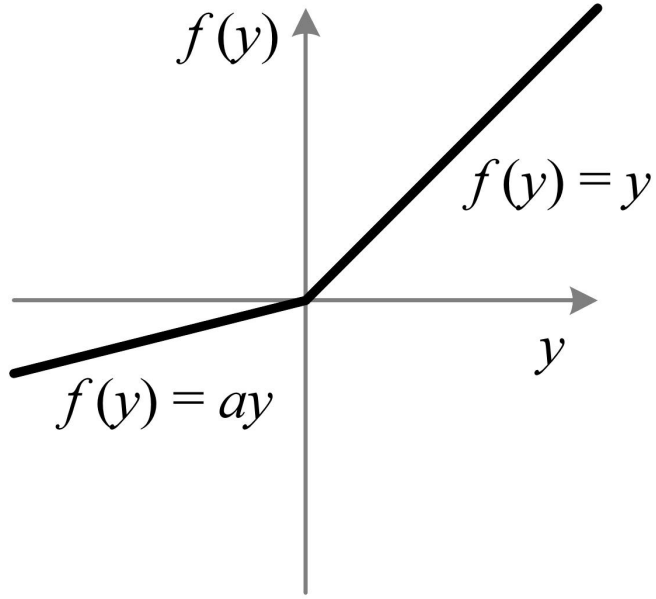
$$PReLU(z) = \begin{cases} z, & z > 0 \\ az, & z \leq 0 \end{cases}$$

z: *input*

a: *learnable parameter*

$$PReLU'(z) = \begin{cases} 1, & z > 0 \\ a, & z \leq 0 \end{cases}$$





https://paperswithcode.com/method/prelu

Leaky ReLU

$$Leaky\ ReLU(z) = \begin{cases} z, & z > 0 \\ 0.01z, & z \leq 0 \end{cases}$$

z: $input$

$$PReLU'(z) = \begin{cases} 1, & z > 0 \\ 0.01, & z \leq 0 \end{cases}$$

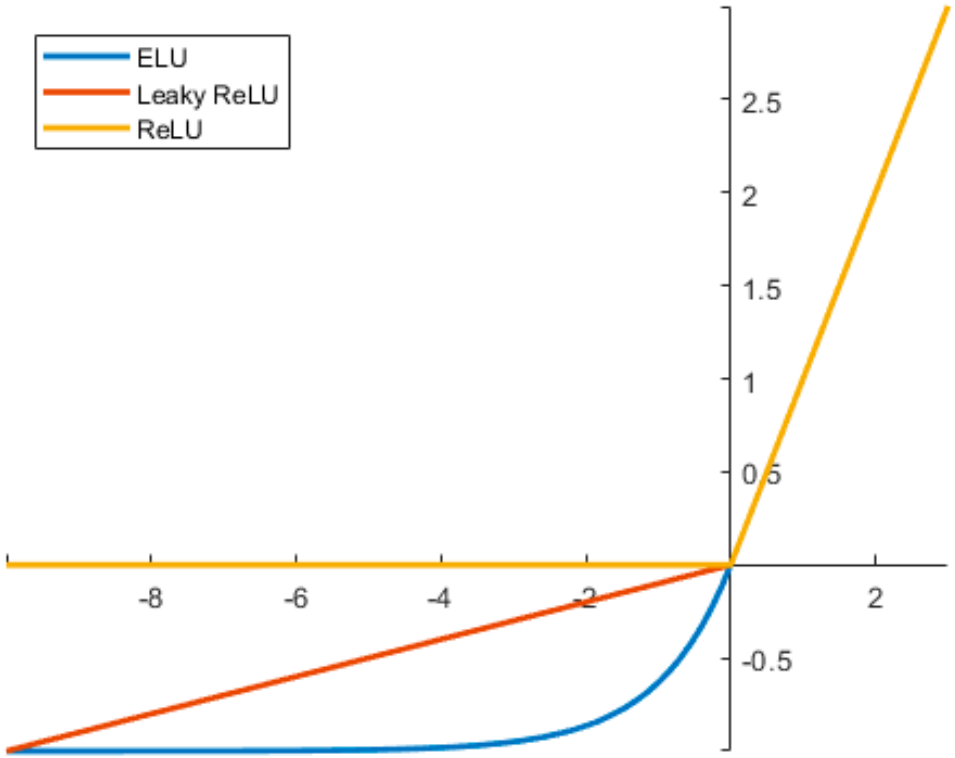
Exponentially Linear Unit (ELU)

$$ELU(z) = \begin{cases} z, & z > 0 \\ a(e^z - 1), & z \leq 0 \end{cases}$$

z: *input*

a: *learnable parameter*

$$ELU'(z) = \begin{cases} 1, & z > 0 \\ ELU(z) + a, & z \leq 0 \end{cases}$$

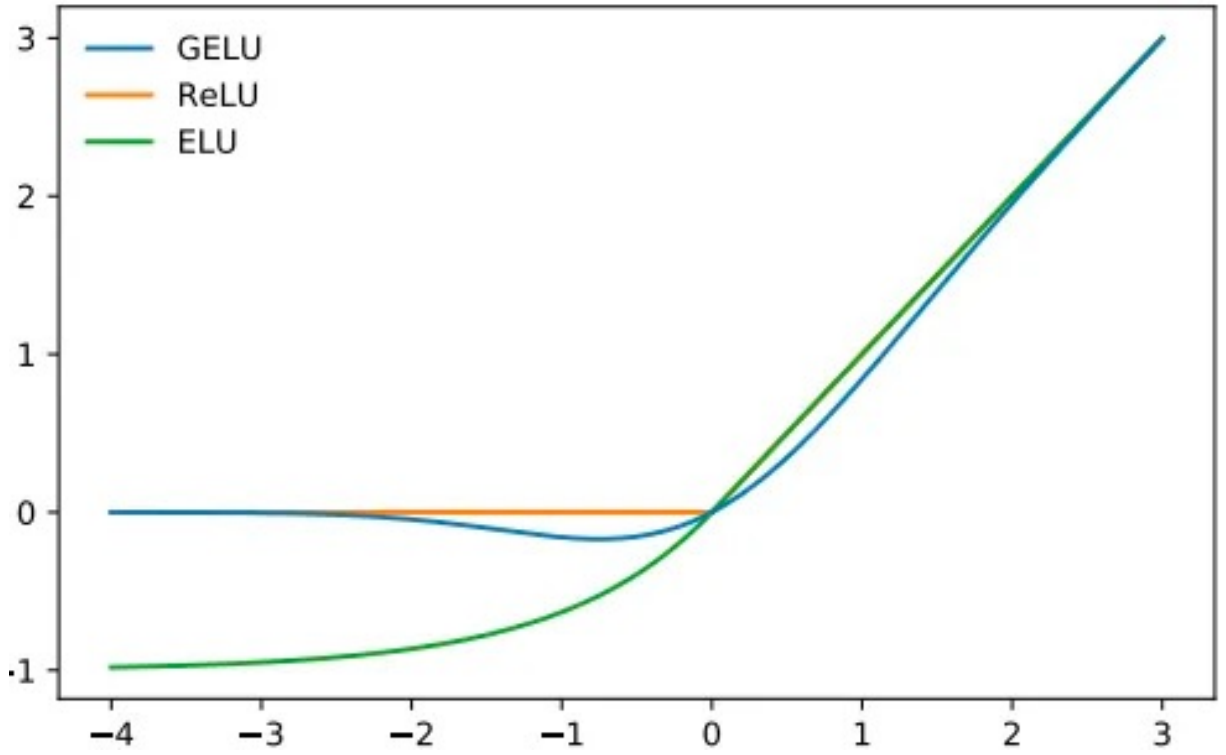


https://blogs.mathworks.com/deep-learning/2018/01/05/defining-your-own-network-layer/?doing_wp_cron=1685986674.8015179634094238281250

Gaussian Error Linear (GELU)

$$GELU(z) = z.P(Z \leq z) \approx 0.5\, z \left(1 + \tanh(\sqrt{(2/\pi)}(z + 0.044715z^3)\right)$$

z: $input$

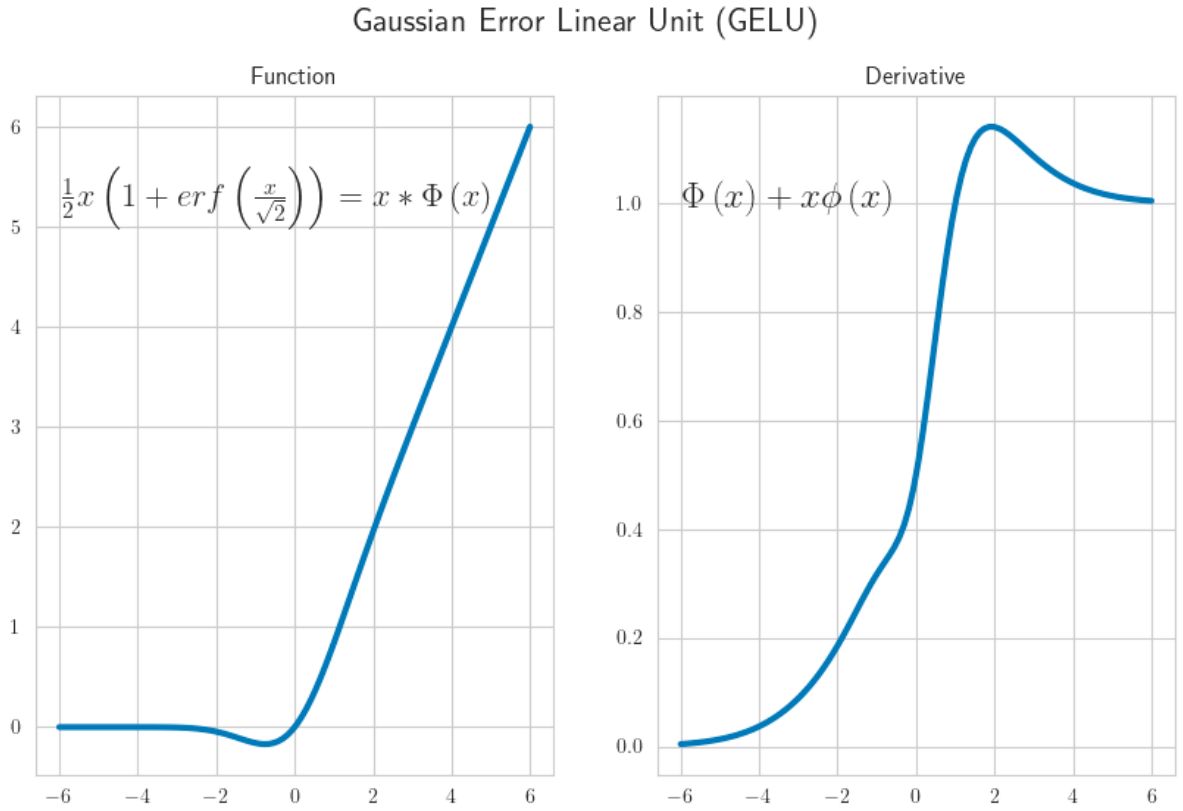


https://medium.com/@shauryagoel/gelu-gaussian-error-linear-unit-4ec59fb2e47c

Gaussian Error Linear (GELU)

$$GELU(z) = z.P(Z \leq z) \approx 0.5\, z \left(1 + \tanh(\sqrt{(2/\pi)}(z + 0.044715z^3)\right)$$

z: *input*

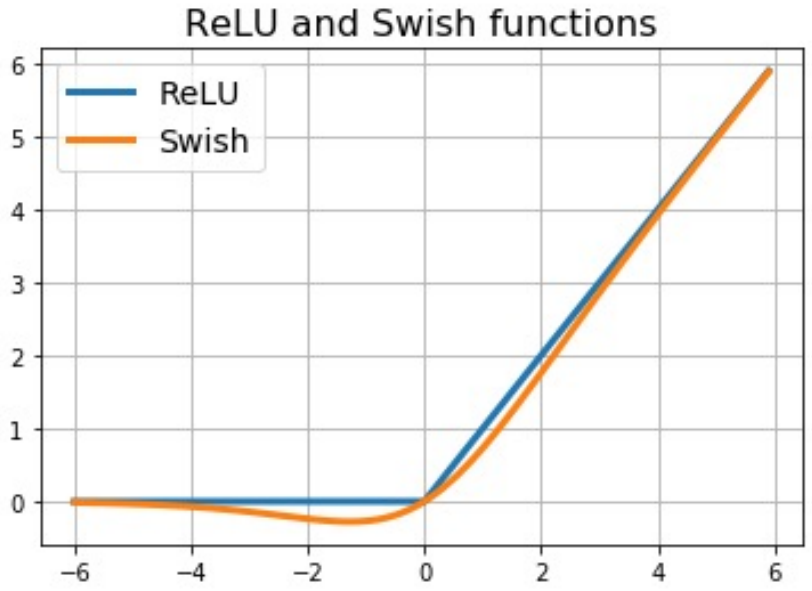


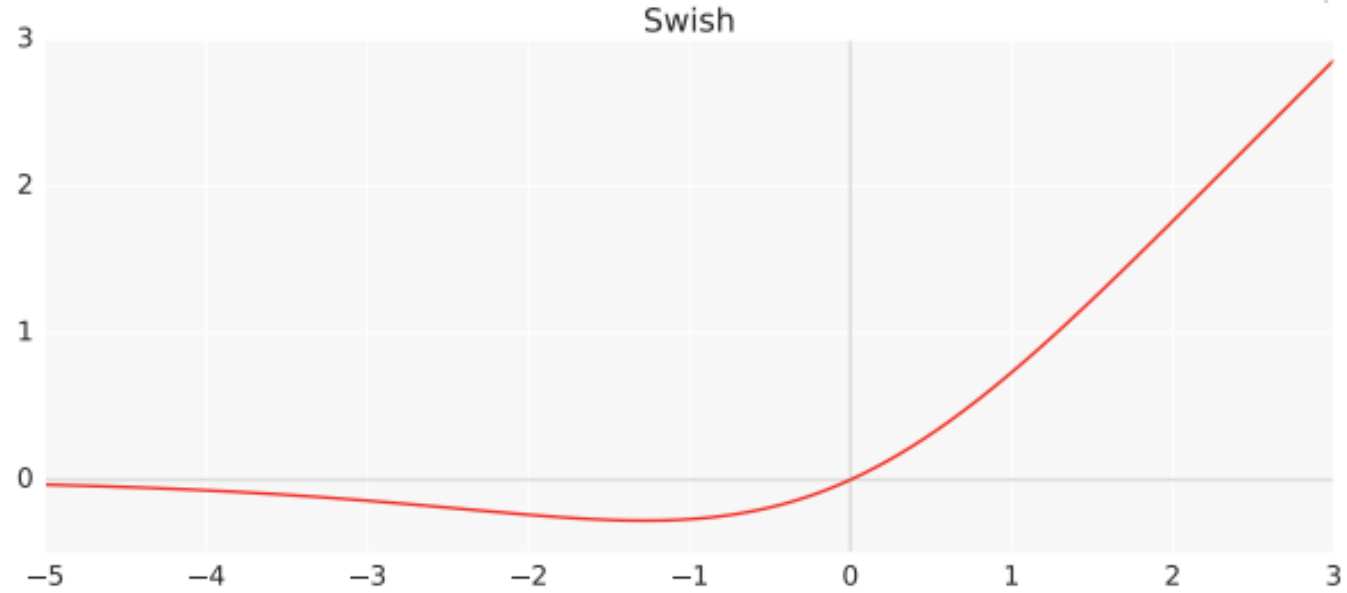
https://ml-explained.com/blog/activation-functions-explained

Swish

$$swish(z) = z.sig(\beta z) = \frac{z}{1 + e^{-\beta z}}$$

z: $input$





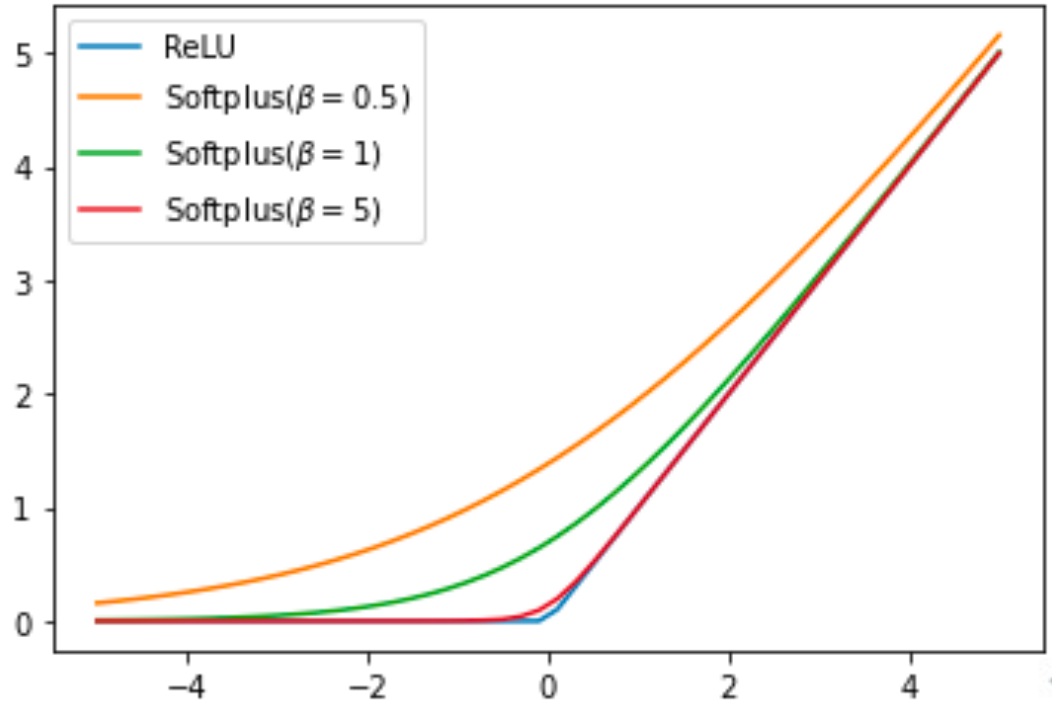
https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820

Softplus

$$Softplus_{\beta}(z) = \frac{1}{\beta}\log\left(1 + e^{\beta z}\right)$$

z: $input$

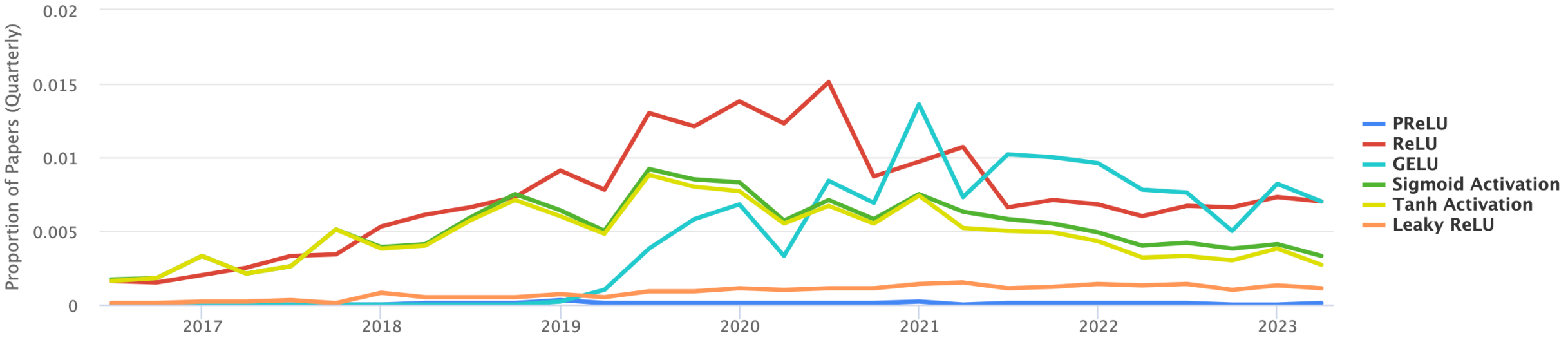
$$\beta = 1 \Rightarrow Softplus'(z) = sig(z)$$



https://pat.chormai.org/blog/2020-relu-softplus

Comparison

Usage Over Time



This feature is experimental; we are continuously improving our matching algorithm.

https://paperswithcode.com/method/prelu

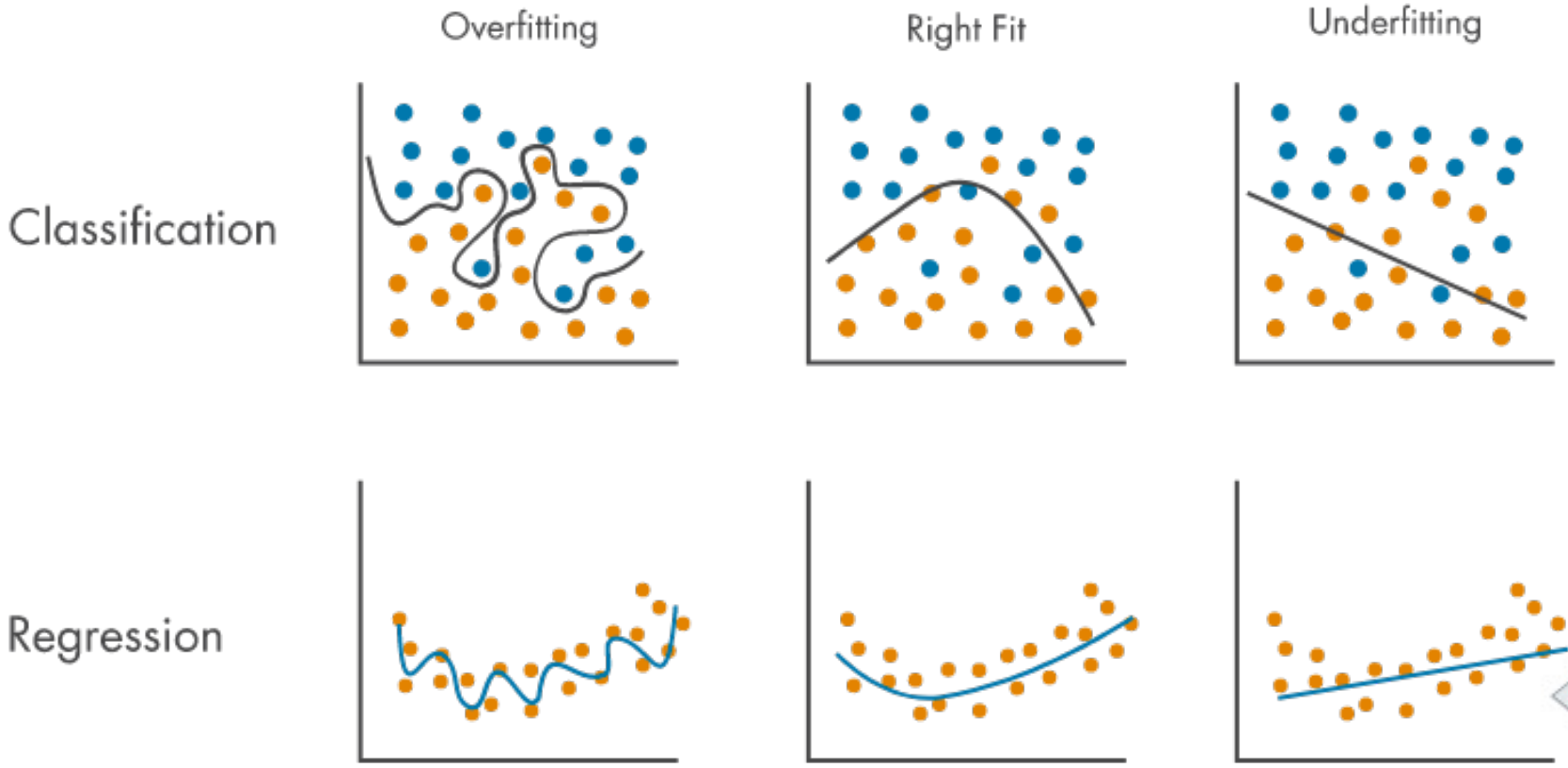
Other variants of ReLU

- Gated Linear Unit (GLU)
  - Dauphin, Yann N., Angela Fan, Michael Auli, and David Grangier. "Language modeling with gated convolutional networks." In International conference on machine learning, pp. 933-941. PMLR, 2017.
- Mish
  - modified of Softplus

Overfitting & Regularization

Overfitting and Underfitting

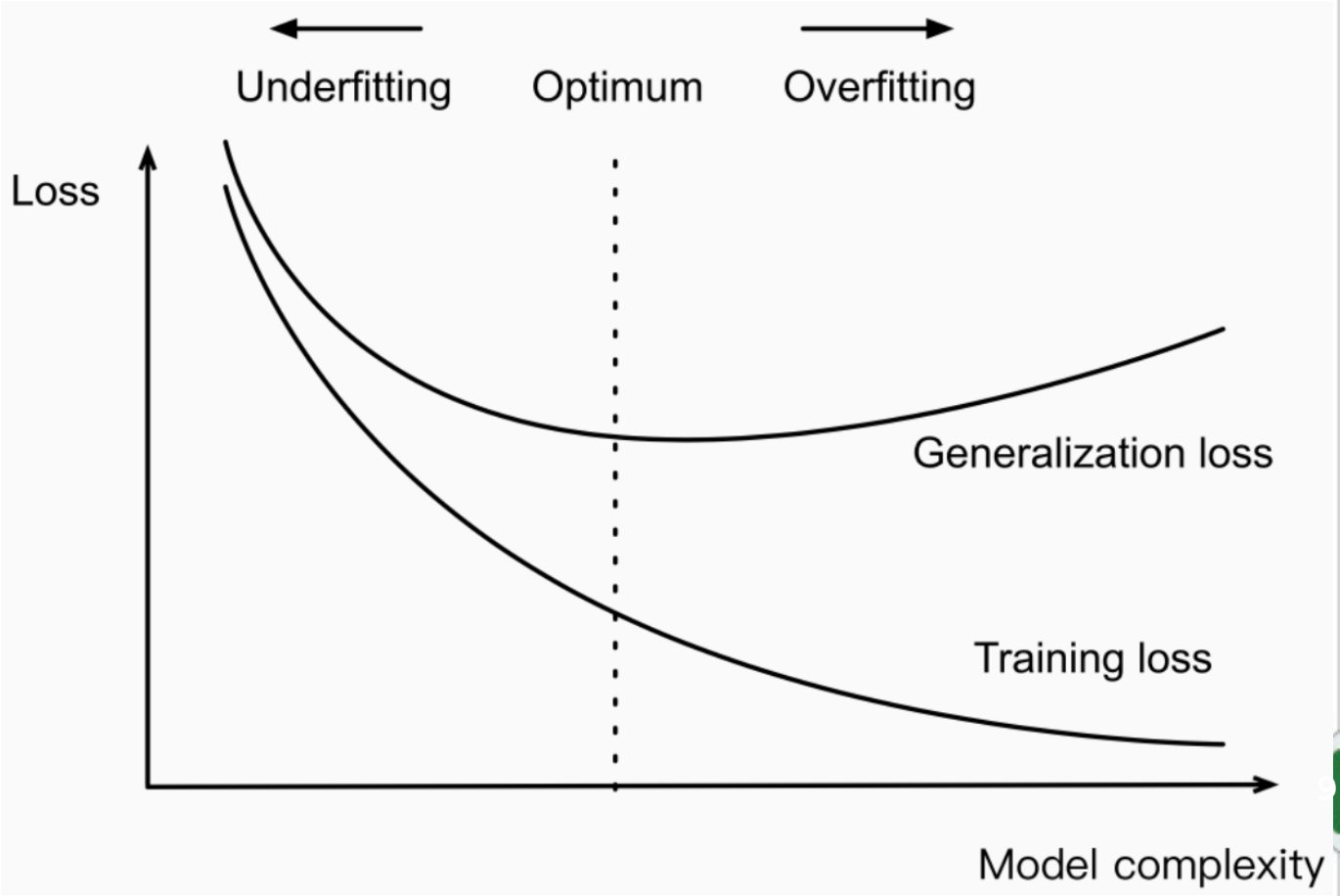
- Big problem
  - overfitting (high accuracy on training data, low accuracy on test data)



https://www.mathworks.com/discovery/overfitting.html

How to reduce Overfitting?

- Train on more data
- Use data augmentation
- Use early stopping
  - When errors start increasing
- Regularization
  - try to calibrate machine learning models in order to minimize the adjusted loss.
  - try to close irrelevant features to 0



https://vitalflux.com/overfitting-underfitting-concepts-interview-questions/

L1 and L2 Regularization

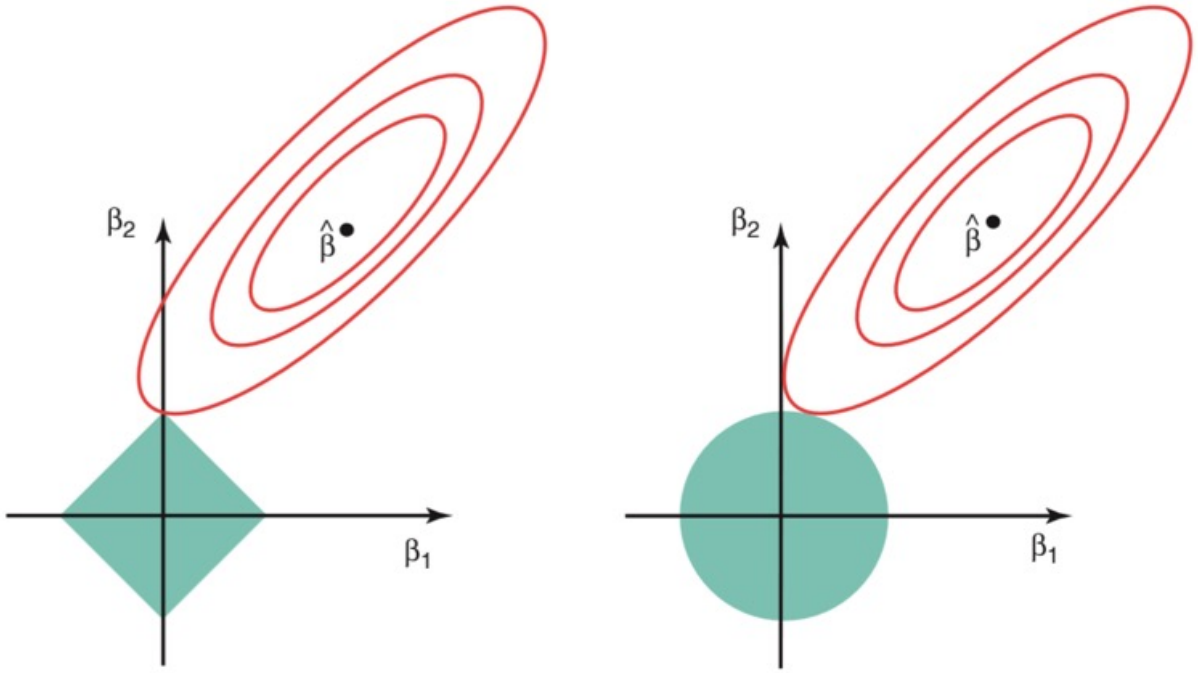
- L1 Regularization
  - Lasso Regression
    - yielding sparse models
    - Good for feature selection

$$new\ loss\ function = previous\ loss\ function + \lambda \sum_{j=1}^{p} |w_j|$$

- L2 Regularization
  - Ridge Regression
    - used when we have more parameters than samples
    - used in multicollinearity
    - not good for feature selection, because decreasing the model complexity but does not reduce the number of irrelevant features

$$new\ loss\ function = previous\ loss\ function + \lambda \sum_{j=1}^{p} w_j^2$$

L1 and L2 Regularization

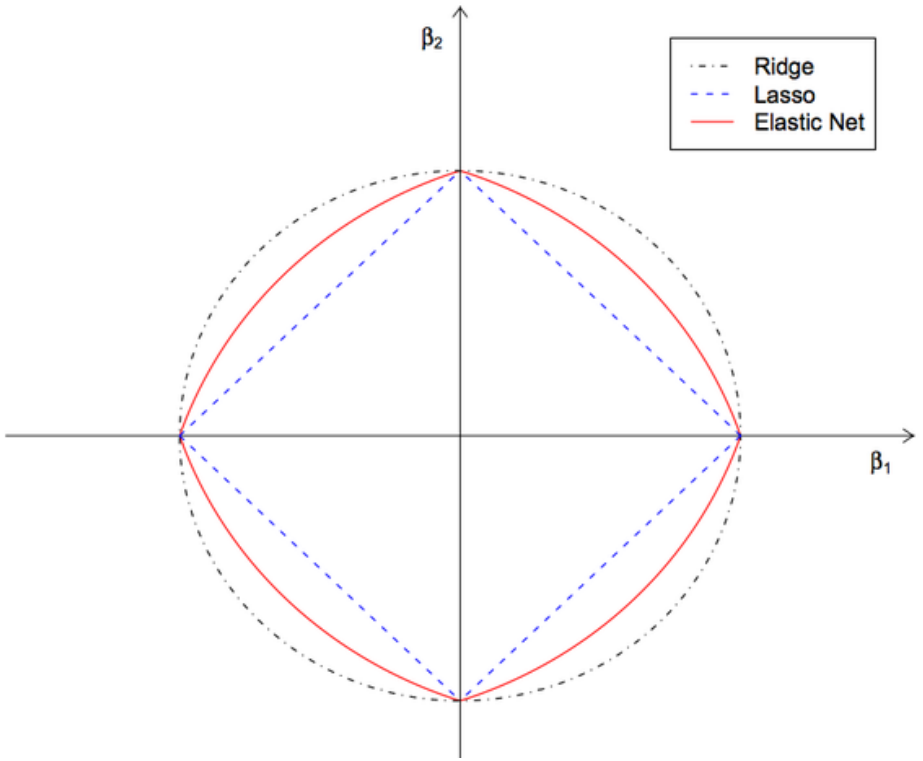


Lasso Regression Ridge Regression

https://www.analyticsvidhya.com/blog/2021/05/complete-guide-to-regularization-techniques-in-machine-learning/

Elastic net Regularization

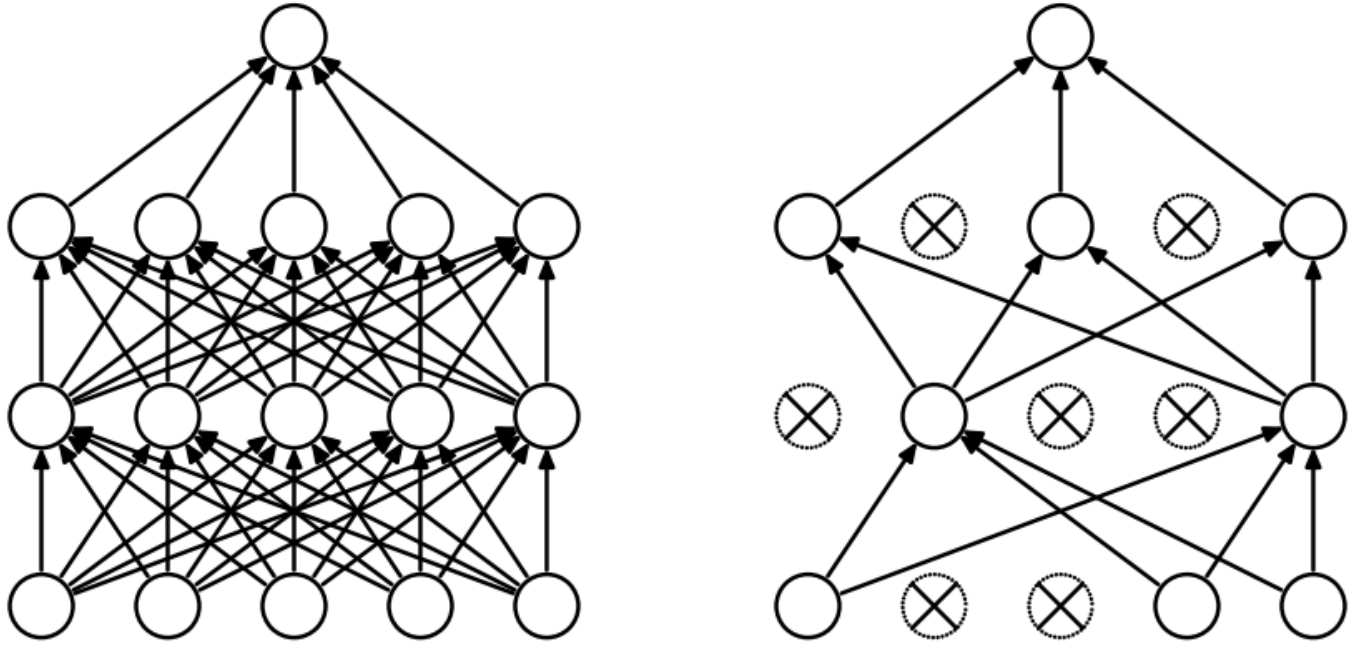
$$new\ loss\ function = previous\ loss\ function + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{p} w_j^2 + \alpha\sum_{j=1}^{p}|w_j|\right)$$



https://medium.com/mlearning-ai/elasticnet-regression-fundamentals-and-modeling-in-python-8668f3c2e39e

Dropout Regularization

- In a fully connected neural network, neurons are heavily dependent on each other, hence the power of each neurons to capture the essentials of the data may be lost.
  - By dropout, neurons are forced to capture the information needed from features

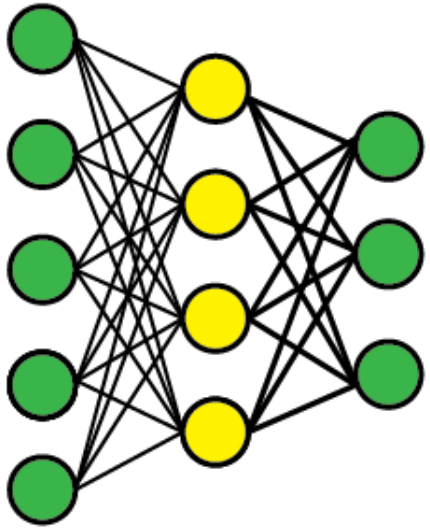


(a) Standard Neural Net

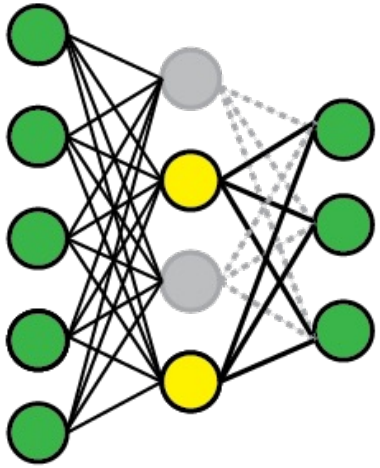
(b) After applying dropout.

https://paperswithcode.com/method/dropout

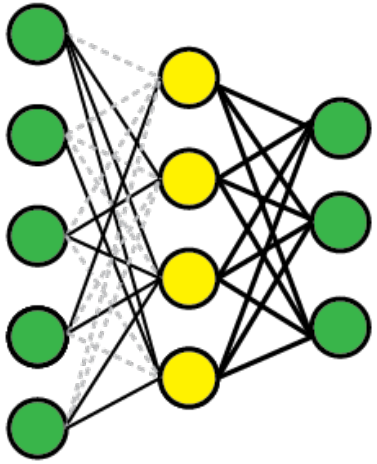
Drop Connect (weight drop) Regularization

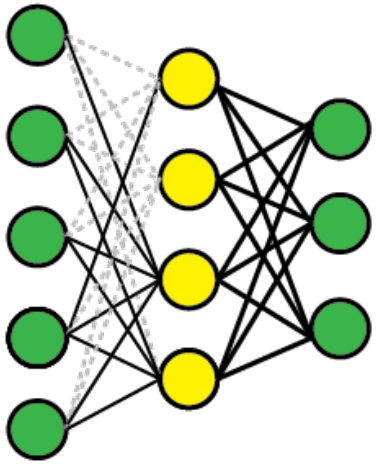


A neural network



Dropout



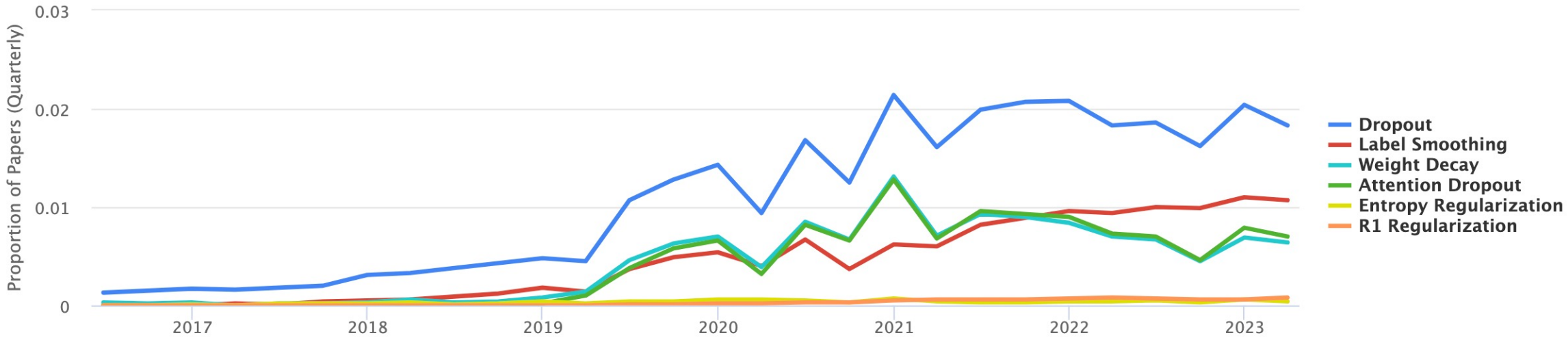


Drop Connect

https://stats.stackexchange.com/questions/201569/what-is-the-difference-between-dropout-and-drop-connect

Comparison

Usage Over Time



This feature is experimental; we are continuously improving our matching algorithm.

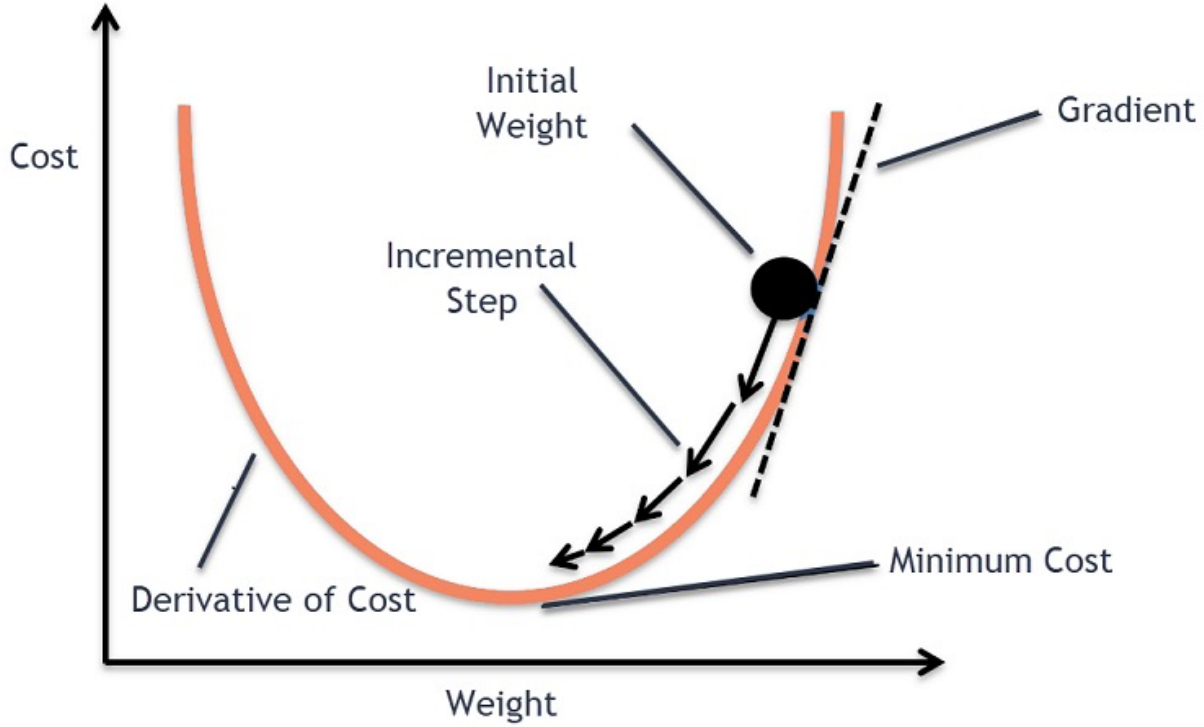
https://paperswithcode.com/method/dropout

Optimization Algorithms

How to minimize the loss function?

- How to minimize a functions?
  - The opposite side of the slope
  - calculating of gradient by differentiation of cost function
  - An epoch is a complete pass through all samples.
  - η is learning rate (the step size), i.e. how fast we update the weights.

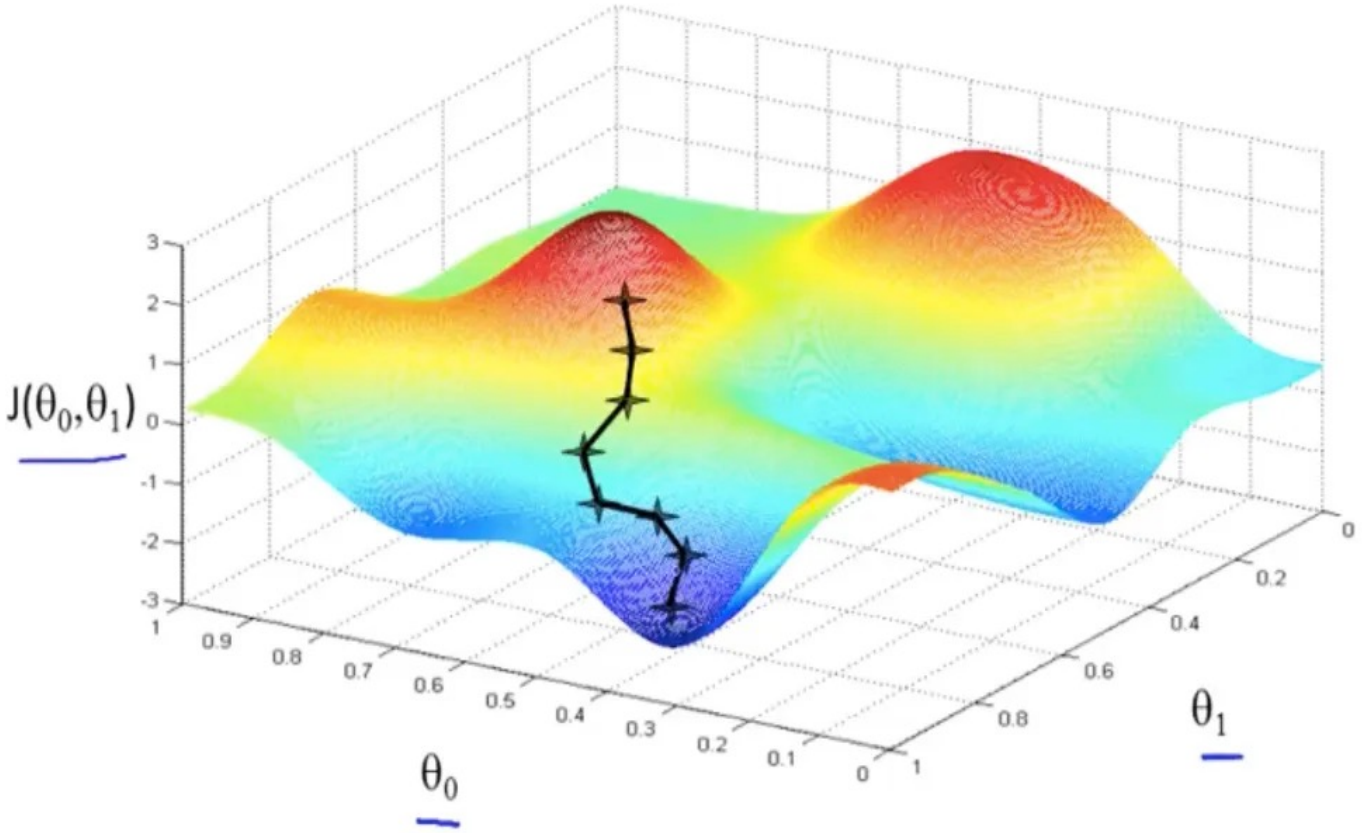
$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$



https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

Gradient Descent

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

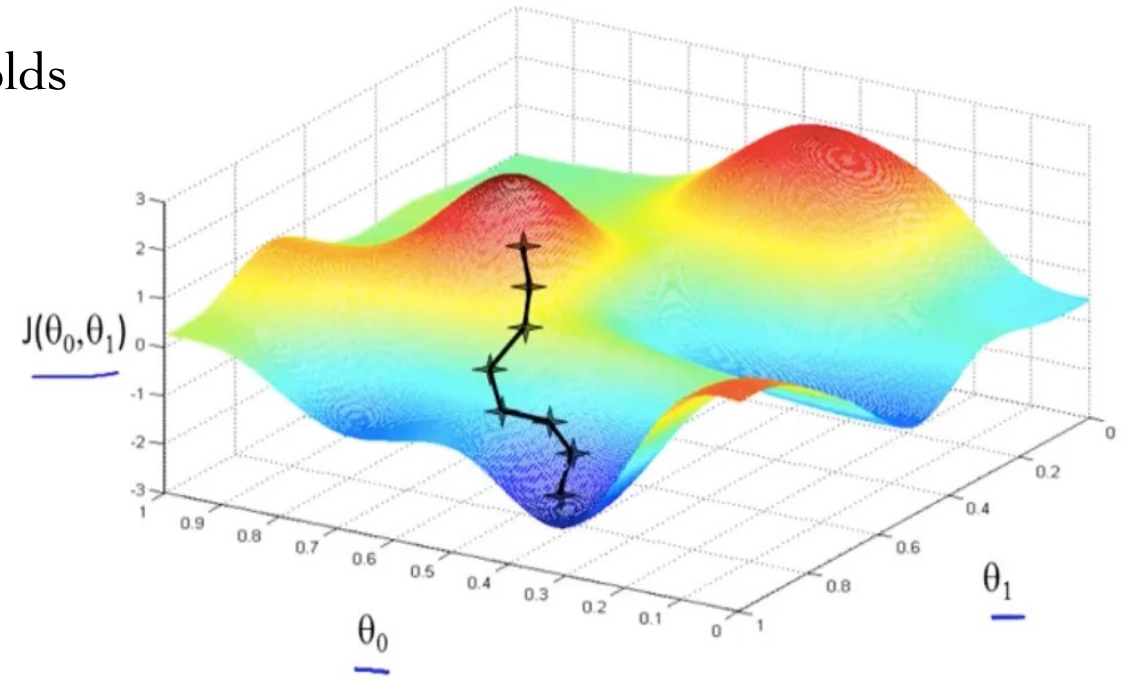


Batch Gradient Descent

- Procedure in just one epoch:
  - Considering all the training data
  - Taking the average of the gradients
  - Using the mean gradient to update the weights

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

- Batch Gradient Descent is great for:
  - convex or relatively smooth error manifolds
  - Stable convergence
  - Very slow for big datasets



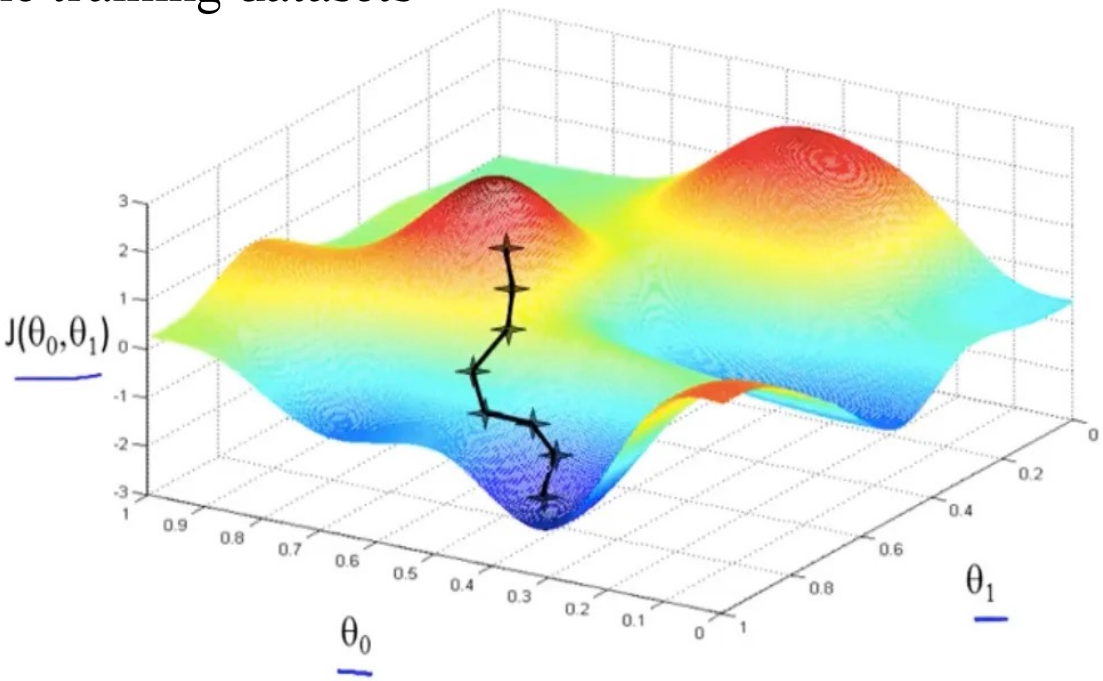
https://www.coursera.org/learn/machine-learning

Stochastic Gradient Descent (SGD)

- Procedure in one epoch:
  - Taking an example
  - Feeding it to neural network
  - Computing the gradient
  - Updating the weights
  - Repeating the above for all examples in the training datasets

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

- Stochastic Gradient Descent:
  - Great for huge datasets
  - Faster than batch one
  - Faster convergence but not necessarily
  - Fluctuating the cost



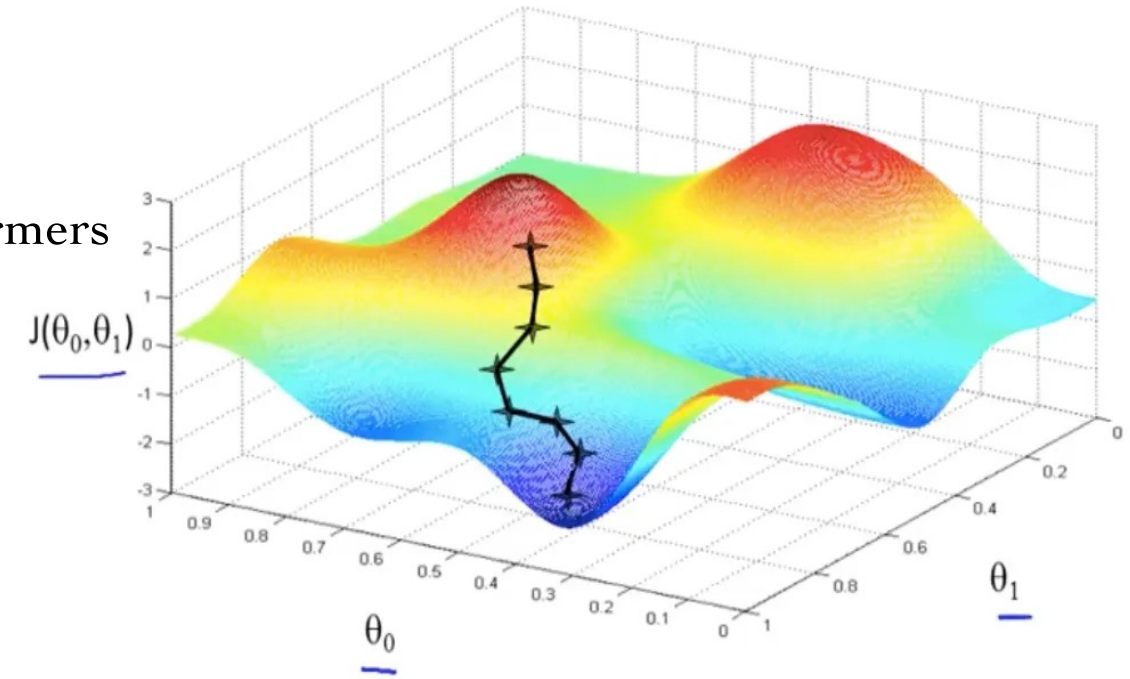
https://www.coursera.org/learn/machine-learning

Mini-Batch Gradient Descent

- Procedure in one epoch:
  - Picking a mini-batch
  - Feeding it to neural network
  - Calculating the mean gradient of the mini-batch
  - Using the mean gradient we calculated to update the weights
  - Repeating the above for the mini-batches we created

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

- Mini-Batch Gradient Descent:
  - achieve both the advantageous of the formers

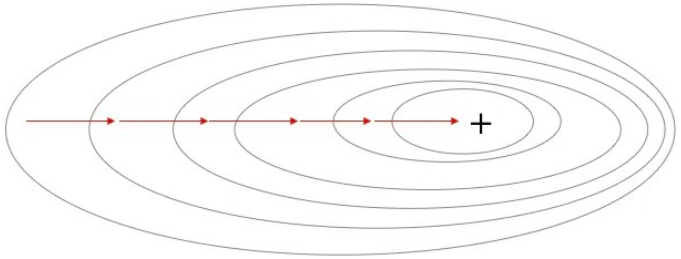


https://www.coursera.org/learn/machine-learning

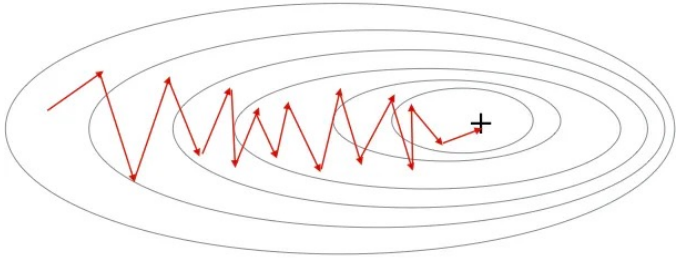
Comparison

$$w \leftarrow w - \eta \frac{\partial Error}{\partial w}$$

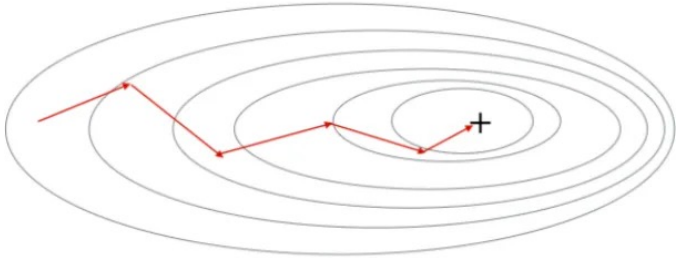
Gradient Descent



Stochastic Gradient Descent

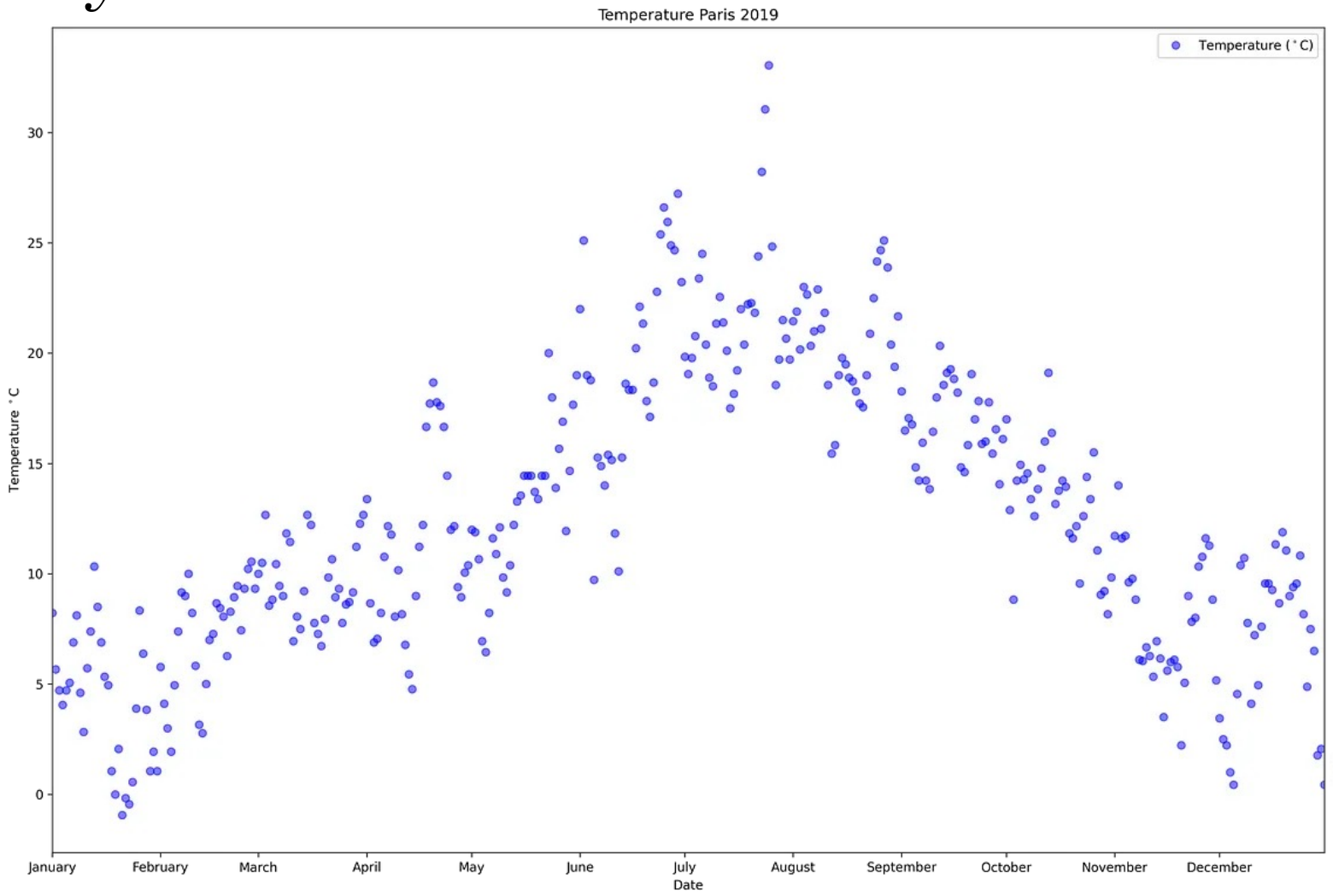


Mini-Batch Gradient Descent



Exponentially Weighted Average

- Considering the history

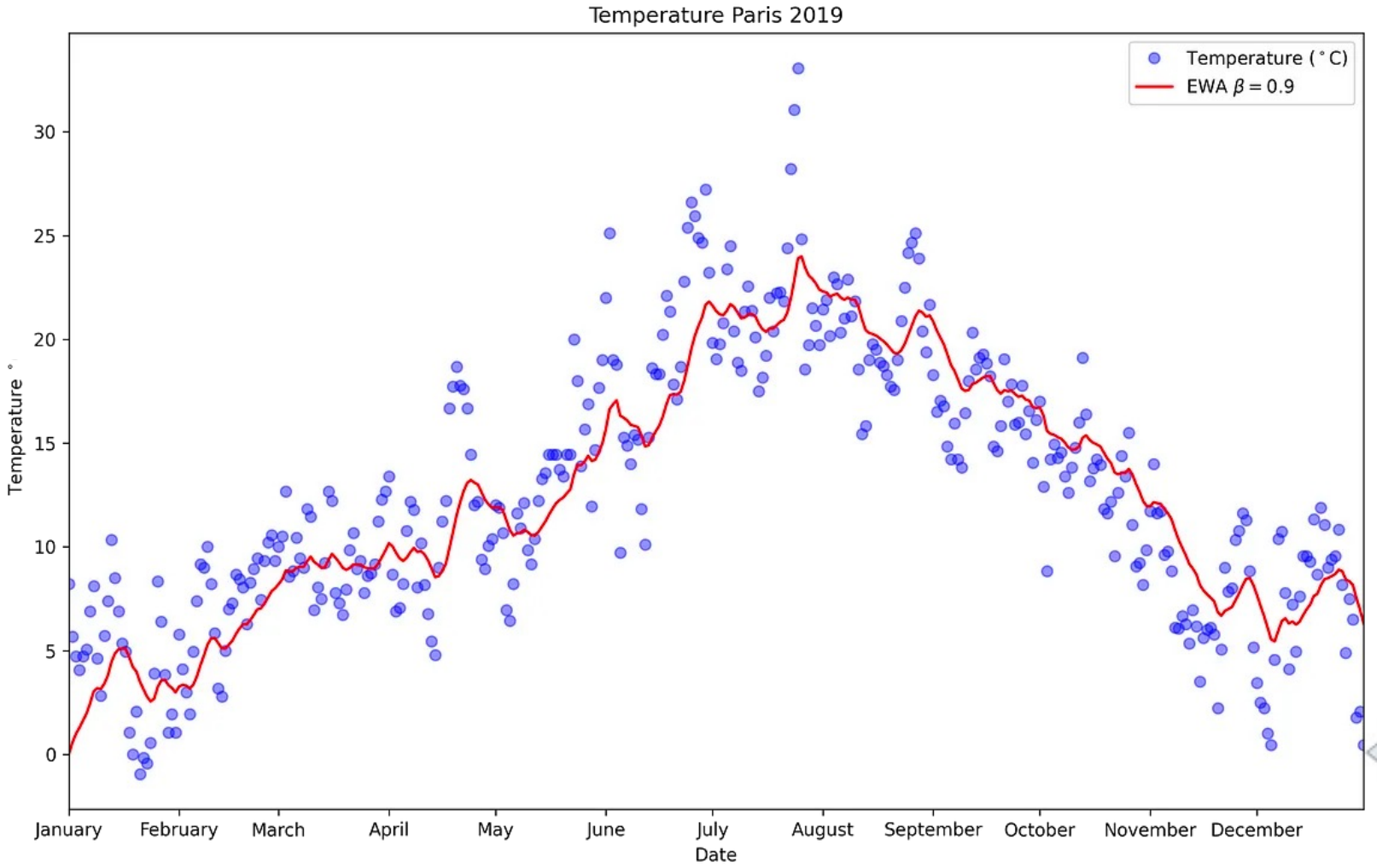


https://medium.com/mlearning-ai/exponentially-weighted-average-5eed00181a09

Exponentially Weighted Average

- Considering the history

$$W_t = \underbrace{\beta \cdot W_{t-1}}_{\text{trend}} + \overbrace{(1-\beta) \cdot \theta_t}^{\text{current value}}$$

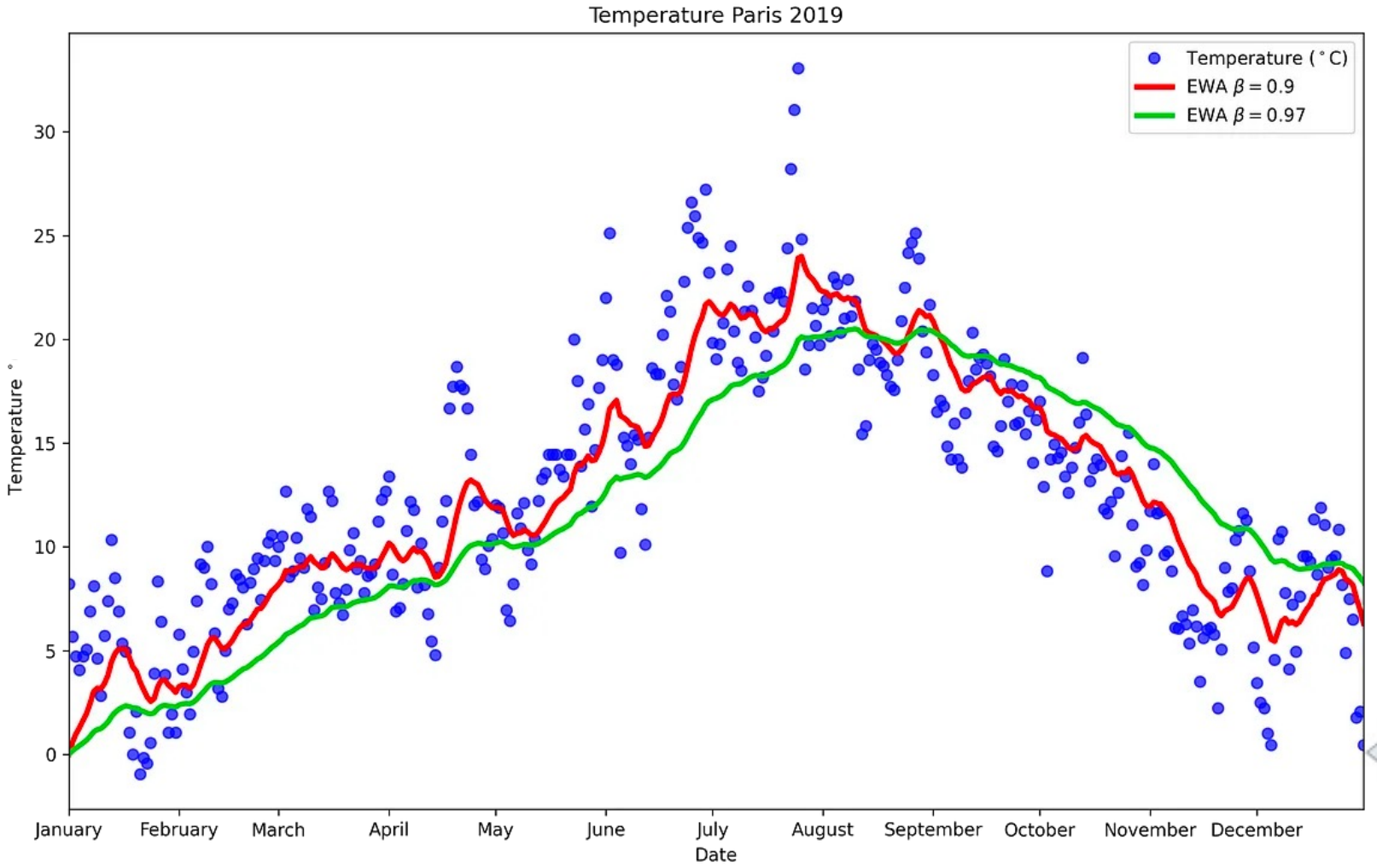


https://medium.com/mlearning-ai/exponentially-weighted-average-5eed00181a09

Exponentially Weighted Average

- Considering the history

$$W_t = \underbrace{\beta \cdot W_{t-1}}_{\text{trend}} + \overbrace{(1-\beta) \cdot \theta_t}^{\text{current value}}$$

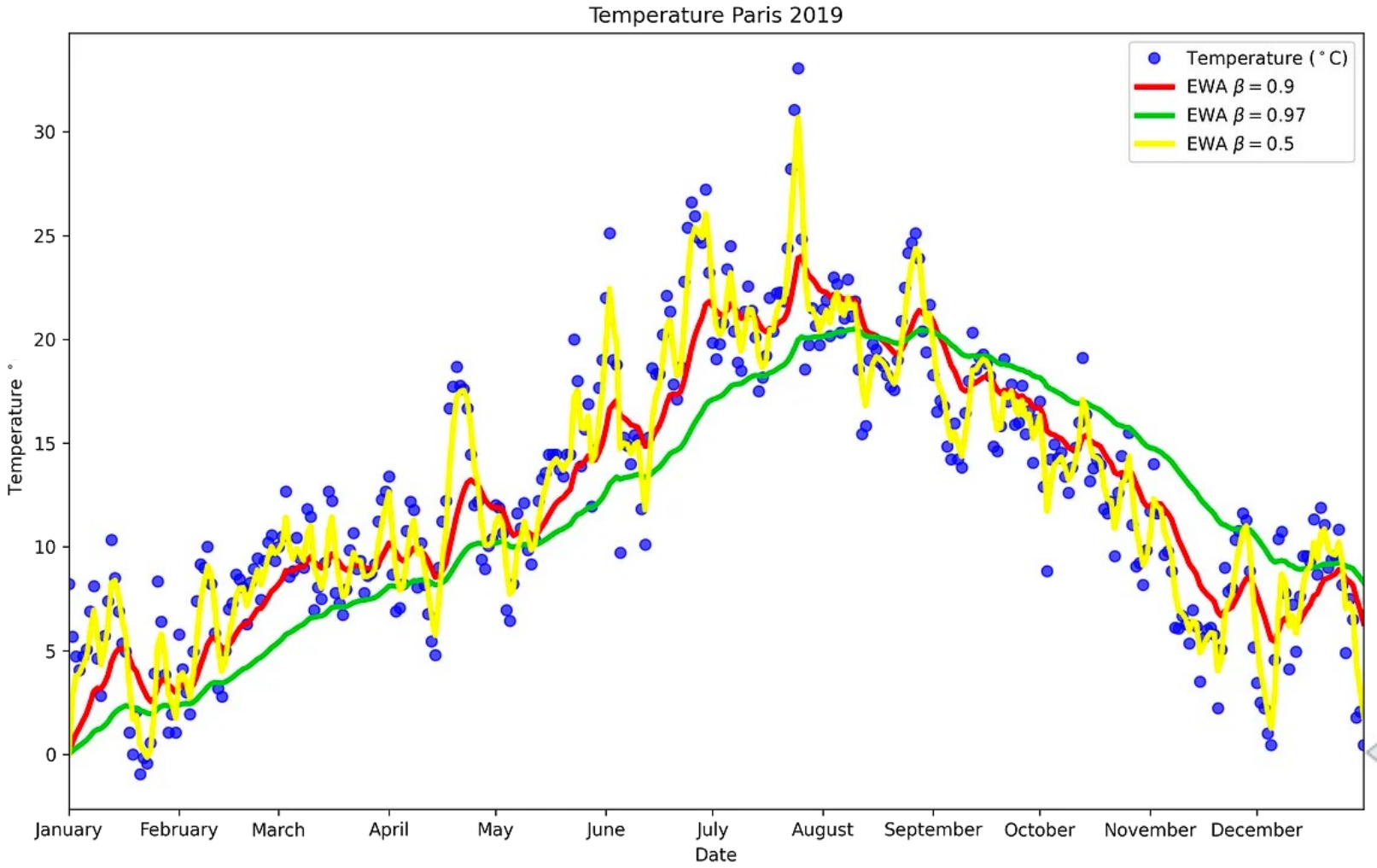


https://medium.com/mlearning-ai/exponentially-weighted-average-5eed00181a09

Exponentially Weighted Average

- Considering the history

$$W_t = \underbrace{\beta \cdot W_{t-1}}_{\text{trend}} + \overbrace{(1-\beta) \cdot \theta_t}^{\text{current value}}$$



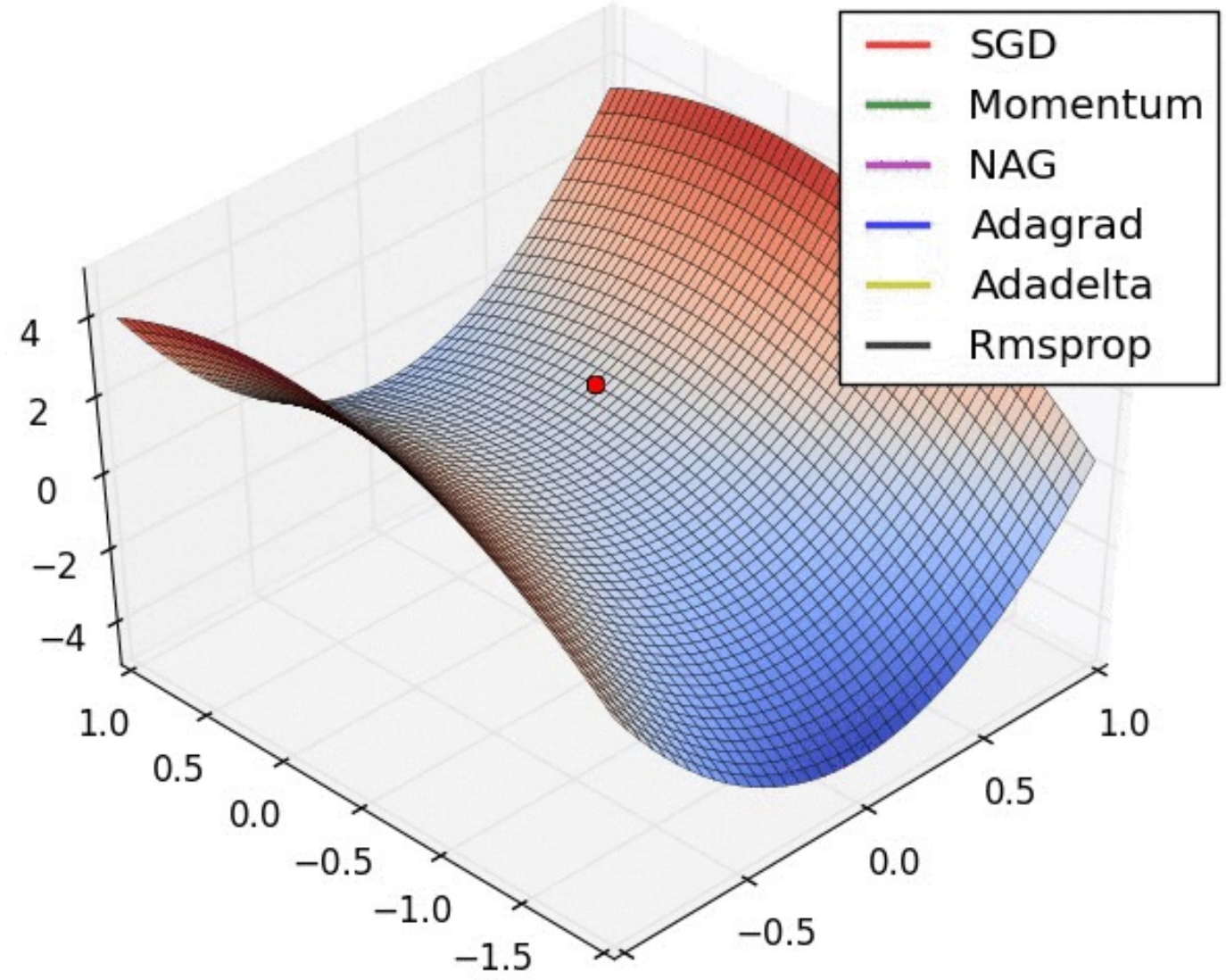
https://medium.com/mlearning-ai/exponentially-weighted-average-5eed00181a09

Other methods

- Gradient Descent with momentum
  - Here, we tweak the above algorithms in such a way that we pay heed to the prior step before taking the next step.
- ADAGRAD
  - ADAGRAD uses adaptive technique for learning rate updation. In this algorithm, on the basis of how the gradient has been changing for all the previous iterations we try to change the learning rate.
- ADAM
  - ADAM is one more adaptive technique which builds on ADAGRAD and further reduces it downside. In other words, you can consider this as momentum + ADAGRAD.
- RMSPROB

https://medium.com/mlearning-ai/exponentially-weighted-average-5eed00181a09

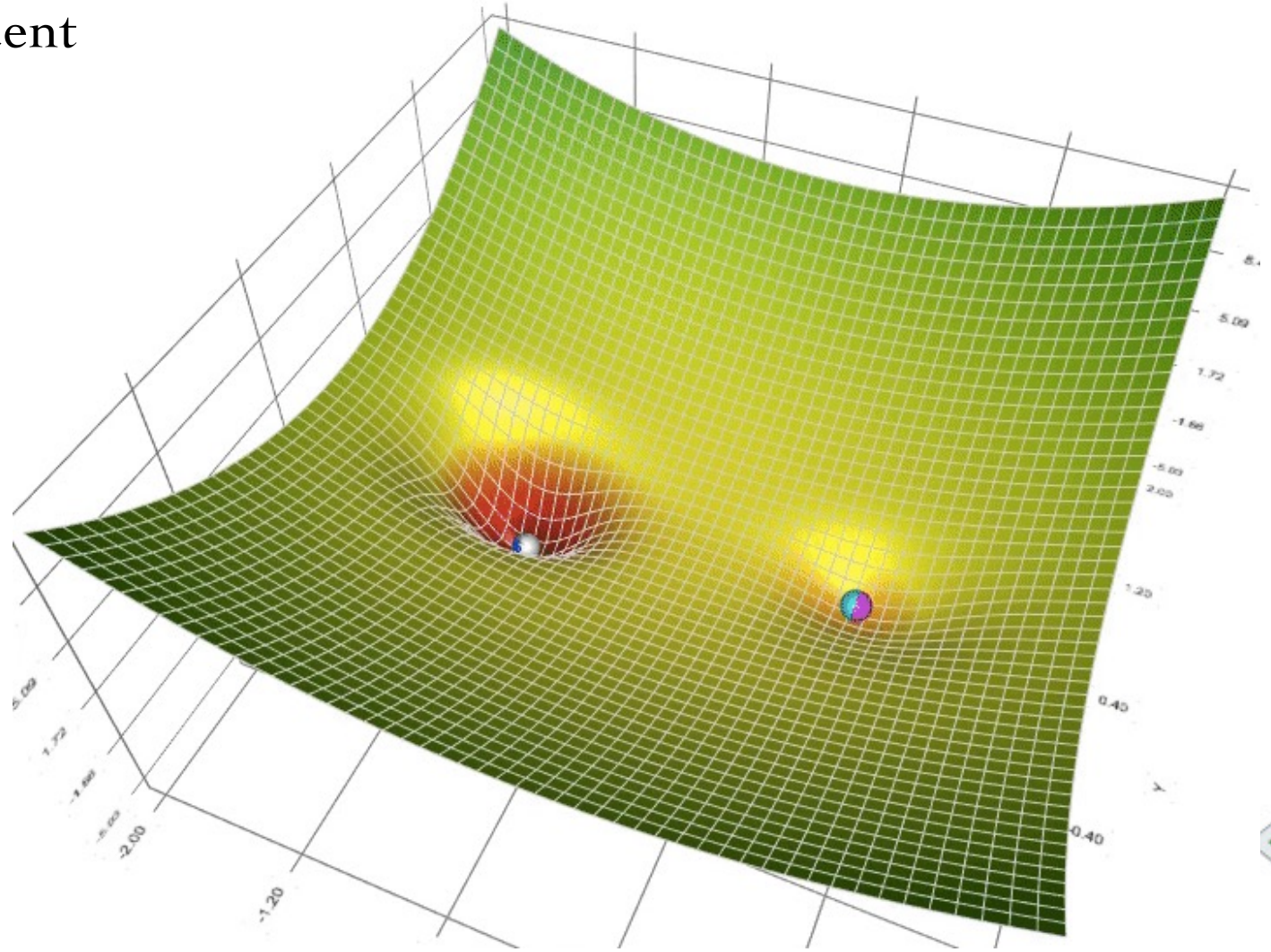
Other methods



https://medium.com/analytics-vidhya/optimization-in-neural-networks-d8d08ecf6659

Other methods

- Animation of 5 gradient descent
  - Cyan: Gradient Descent
  - Magenta: Momentum
  - White: AdaGrad
  - Green: RMSProp
  - Blue: Adam



https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c


THANK
YOU !