

Intro to Neural Networks

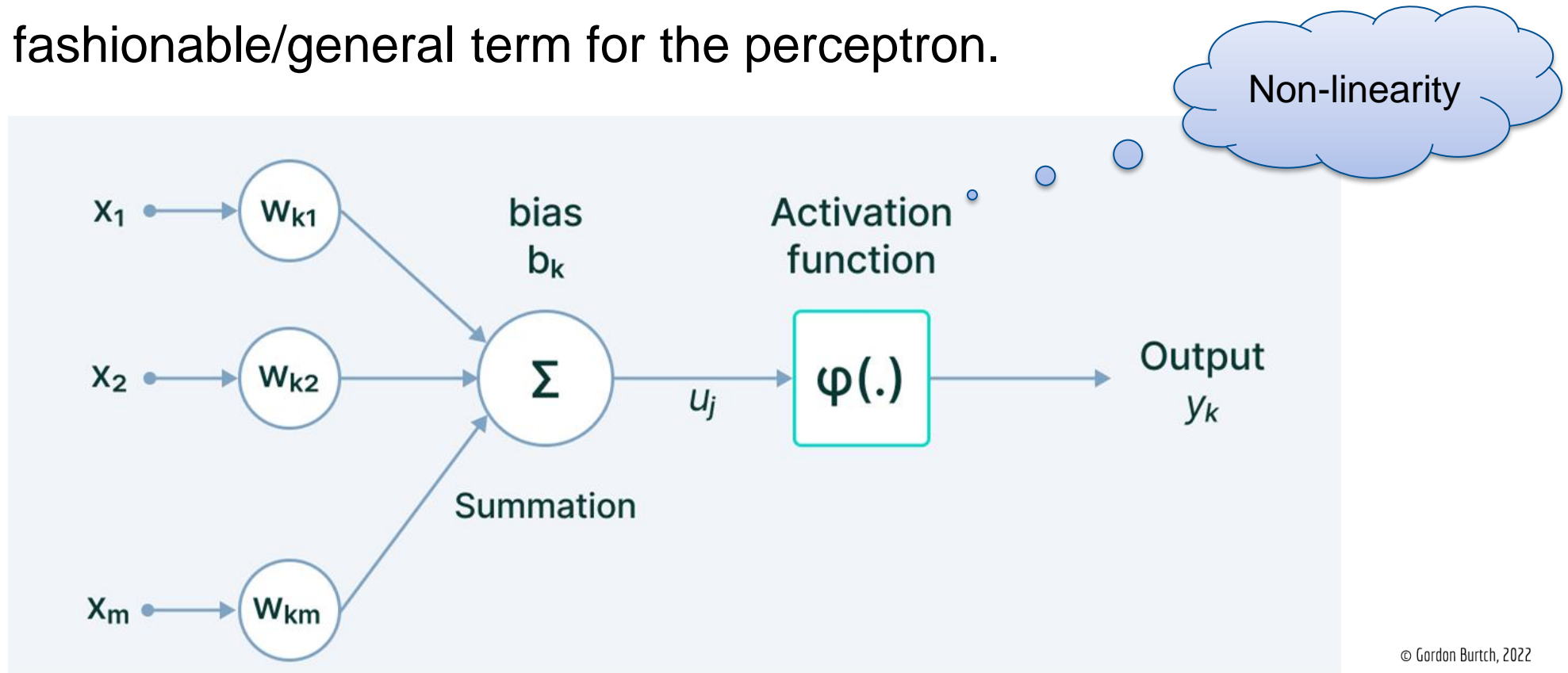
BA865 – Mohannad Elhamod

MLPs

The Multi-Layer Perceptron

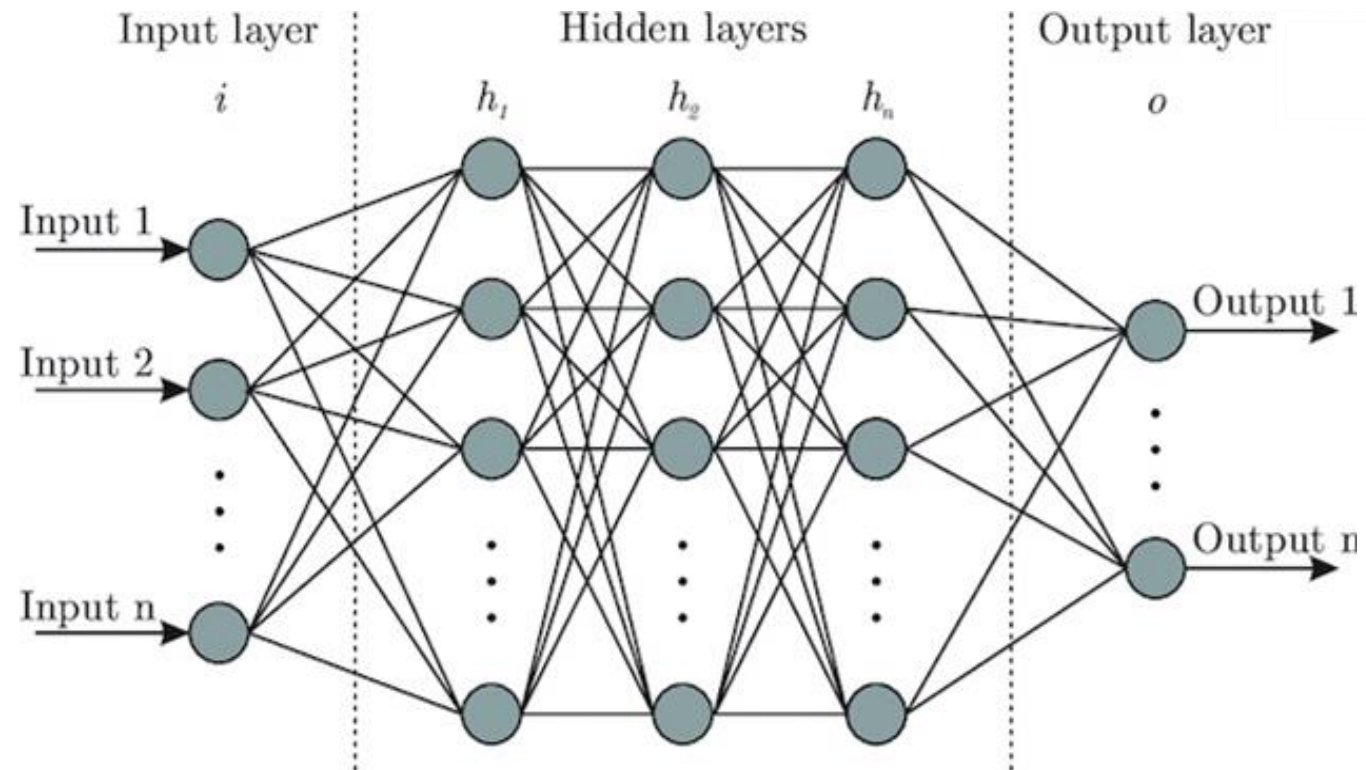
A Neuron

- A more fashionable/general term for the perceptron.



© Gordon Burtch, 2022

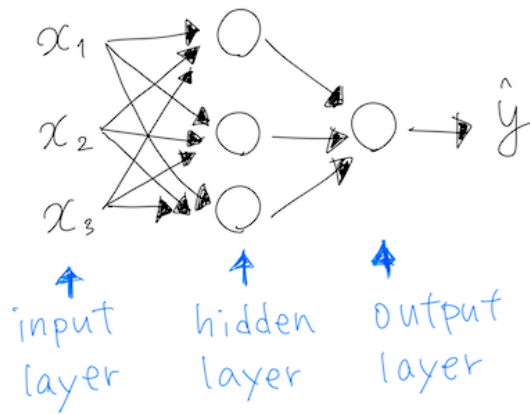
Neural Networks



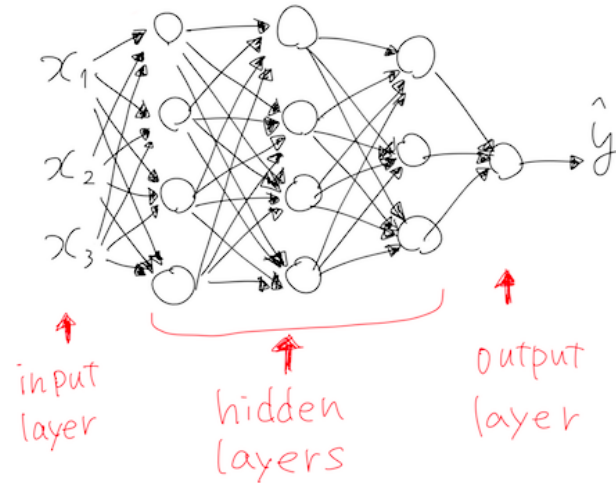
Deep Networks

- Deep = More and more layers...
- leading to more complexity and better capacity for capturing complex phenomena.

Shallow Neural Network

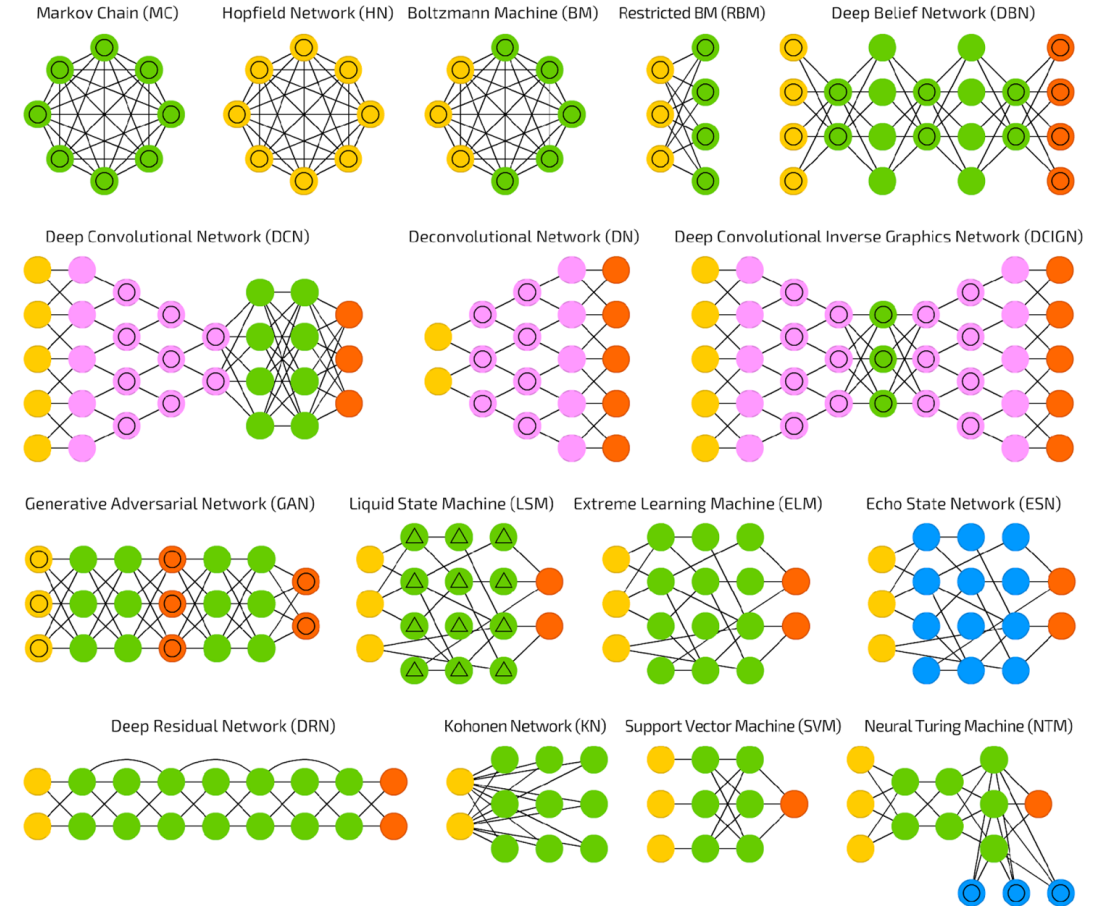
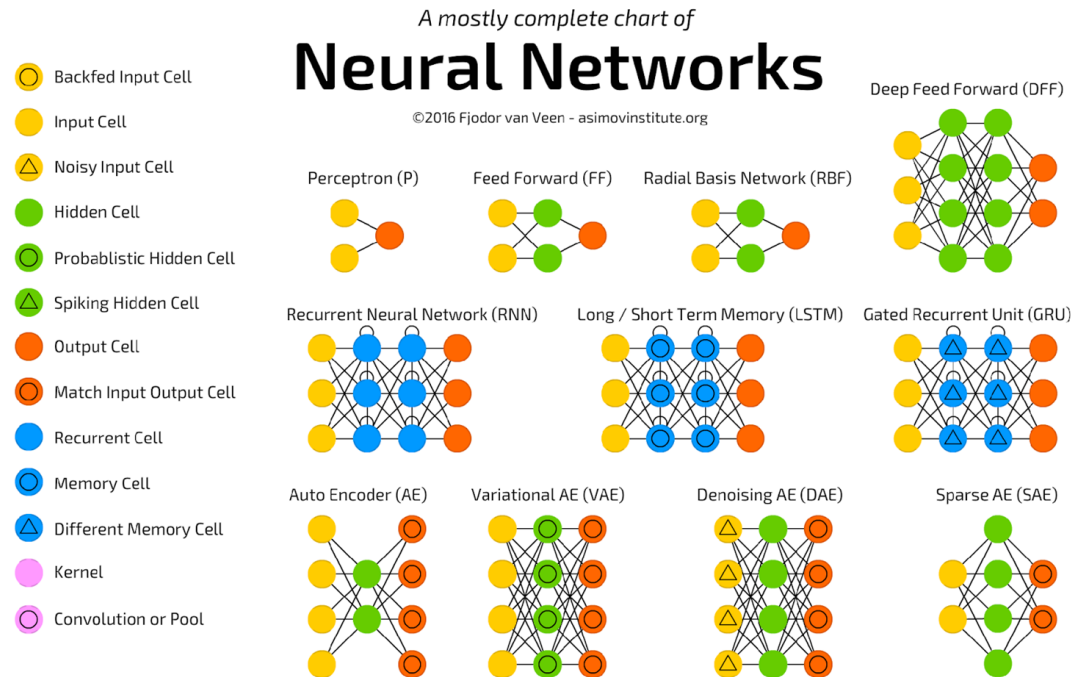


Deep Neural Network



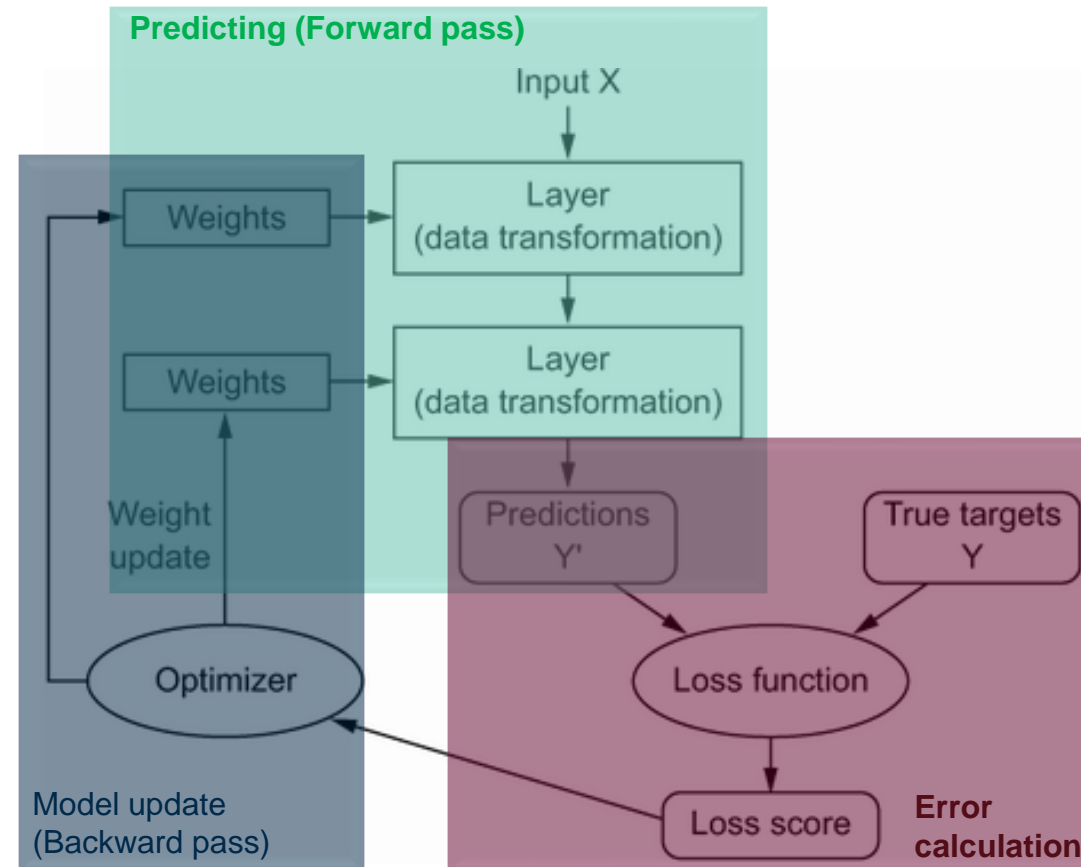
MLPs, One of Many Types...

- MLP = FF (Feed Forward) network = FC (Fully-Connected) layers.

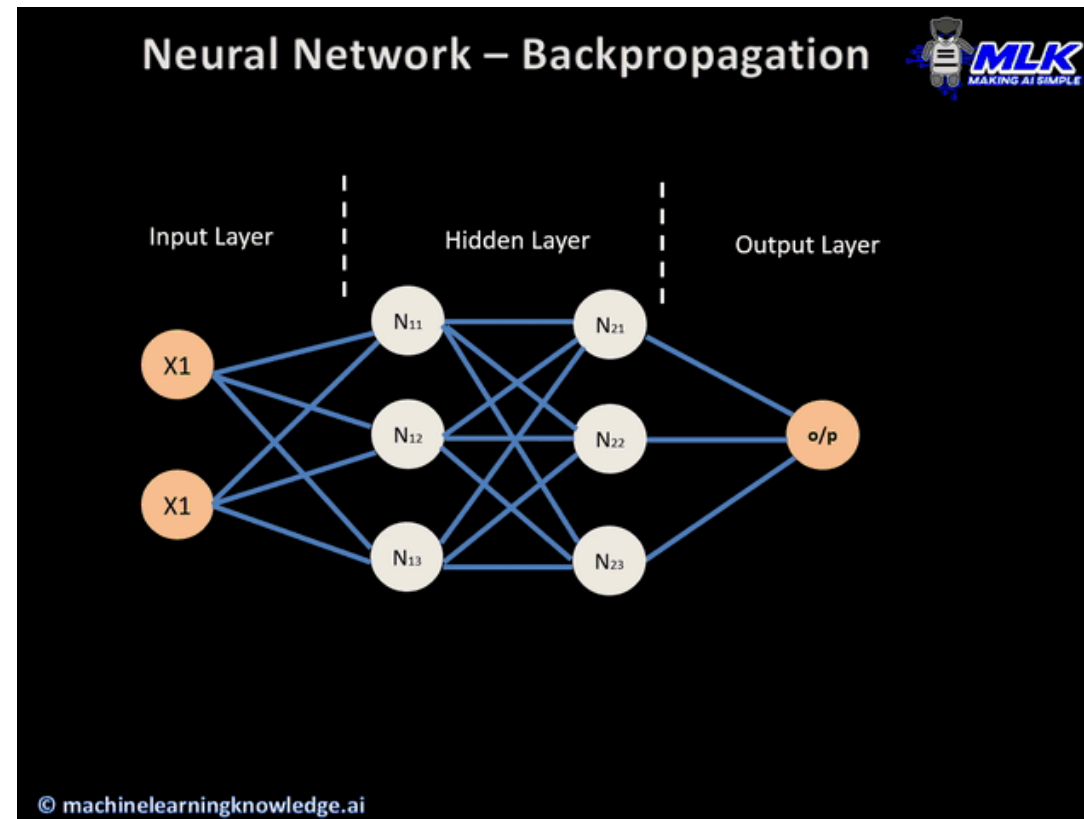


Disecting The Neural Network

The Framework

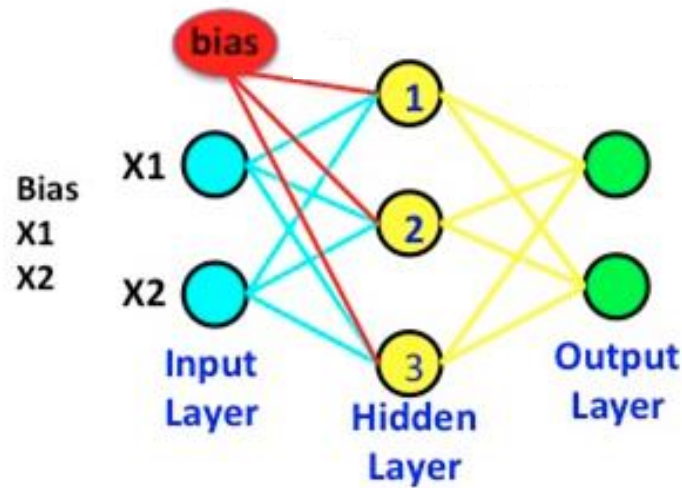


The Framework



Predicting

- What is a layer actually doing?
- Each layer is a matrix multiplication followed by a non-linearity!
 - Why bother with the non-linearity?!



Input Layer

bias	X1	X2
1	0	1
1	0	0
1	0	0
1	1	0

4 x 3

Weights w^T (transposed)

.5	.5	.5
.5	.5	.5
.5	.5	.5

3 x 3

Go to Hidden Nodes

1 2 3

Hidden Layer

1	1	1
.5	.5	.5
.5	.5	.5
.5	.5	.5
1	1	1

4 x 3

Sigmoid Function

Weights

.2	.1
.4	.1
.4	.1

3 x 2

Output Layer

1	.3
.5	.15
.5	.15
1	.3

4 x 2

Bias Node 1 Node 2 Node 3

Equation: $\frac{1}{1 + e^{-(wx+b)}}$

Figure (modified) courtesy of Baber Zaimen

Predicting

- [Demo](#)
- The non-linearities allow the neural net to “warp” a non-linear problem into a linear one!

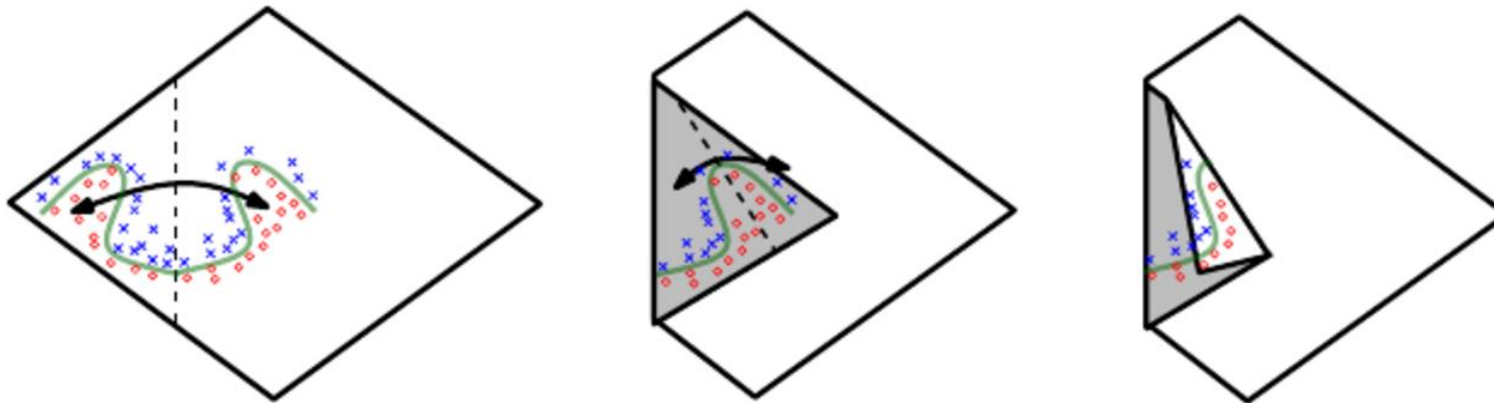


Figure courtesy of [Deep Learning Book](#)

Optimization

- Using Gradient Descent (or some other optimizer) to “update the network.”
 - What do we exactly mean by “updating the network”?

Optimization

- Gradient descent is performed with respect to the weights/biases.
- [Behold...](#)

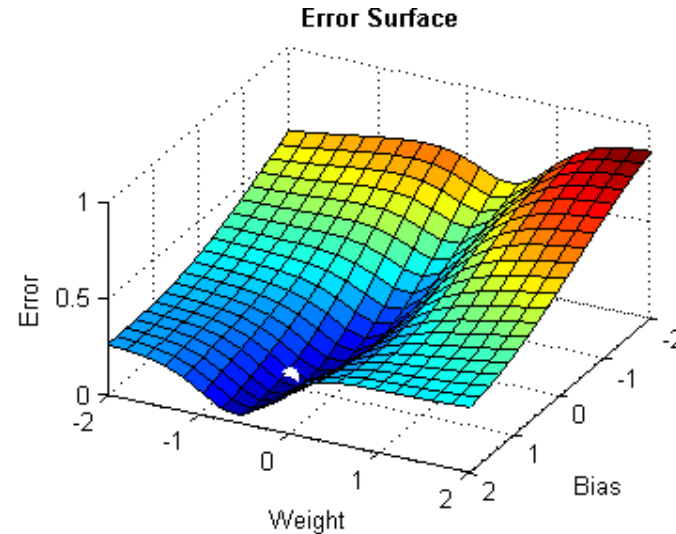
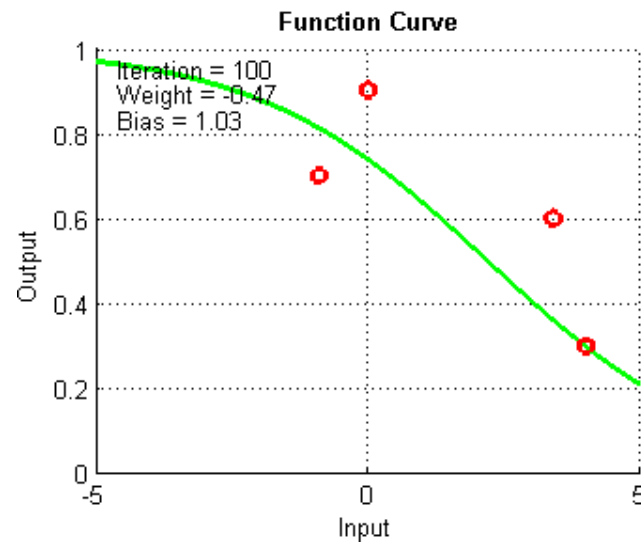
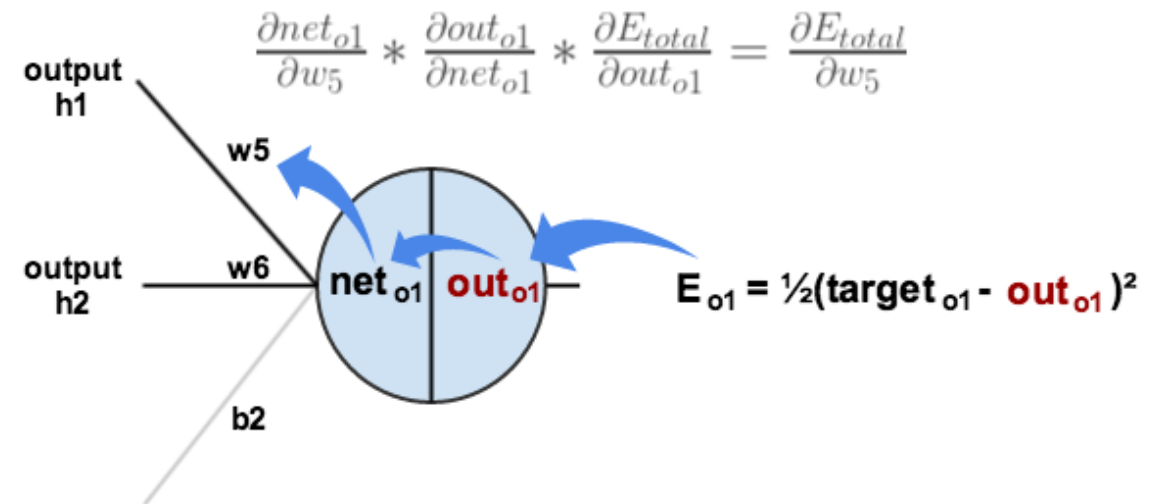


Figure courtesy of [Devin Soni](#)

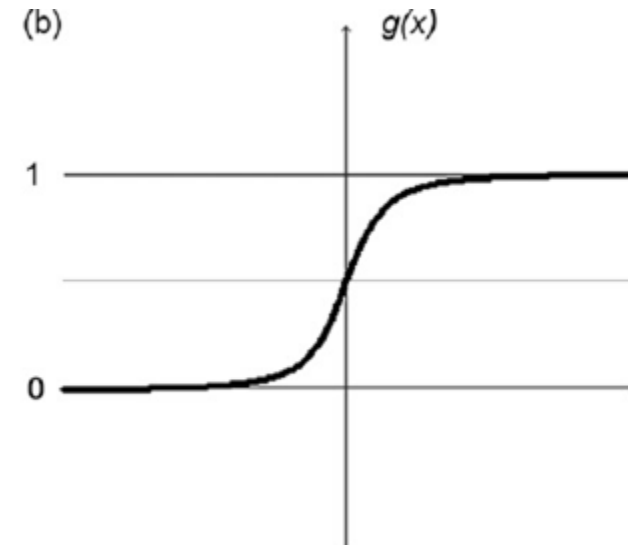
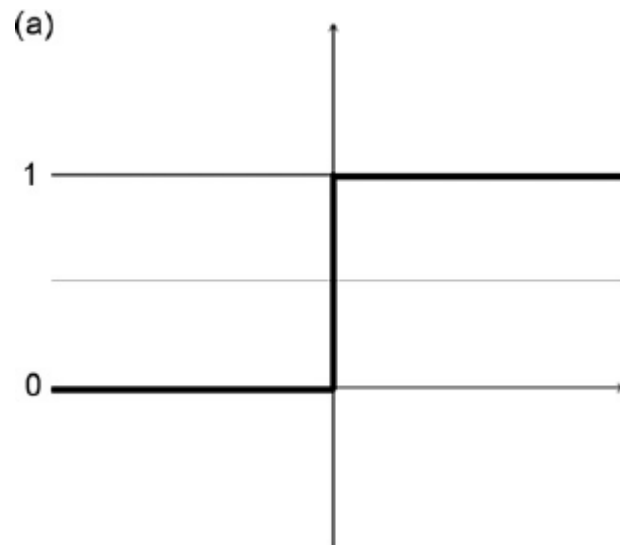
Optimization

- But how does the update get carried all the way back?
 - Chain rule!
 - This is called back-propagation.
- Luckily, these calculations can be automated with automatic differentiation.



Optimization

- We need to make sure the gradient is non-zero...
 - Otherwise, the gradient can't “flow”!
- Replace the step function with a continuous one!



Hyper-Parameters

Learning Rate

- Generally, the most important hyperparameter of them all!
 - Too low: Really slow convergence.
 - Too high: No convergence.

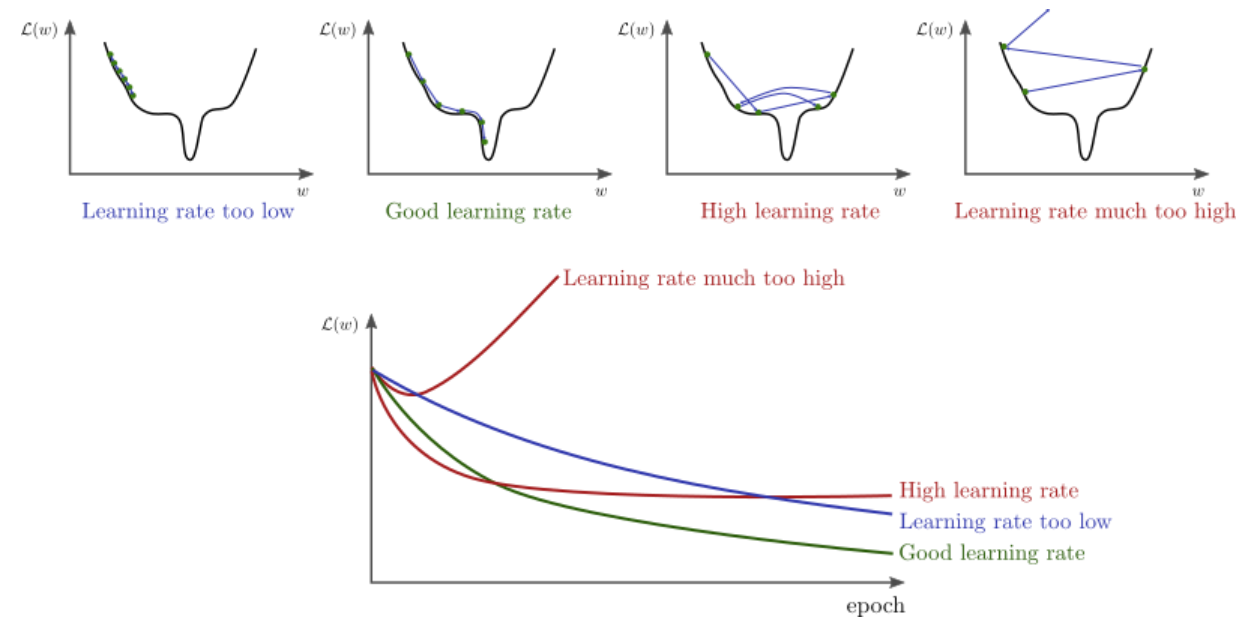


Figure courtesy of Stanford CS class CS231n

Learning Rate: Early Stopping

- The number of epochs impacts the model's fitness.
- Training needs to stop at the “right” epoch.
- How do we achieve that?
 - Better to stop when validation error stops decreasing for a certain number (n) of epochs.
 - Setting n too small or too large will impact convergence.

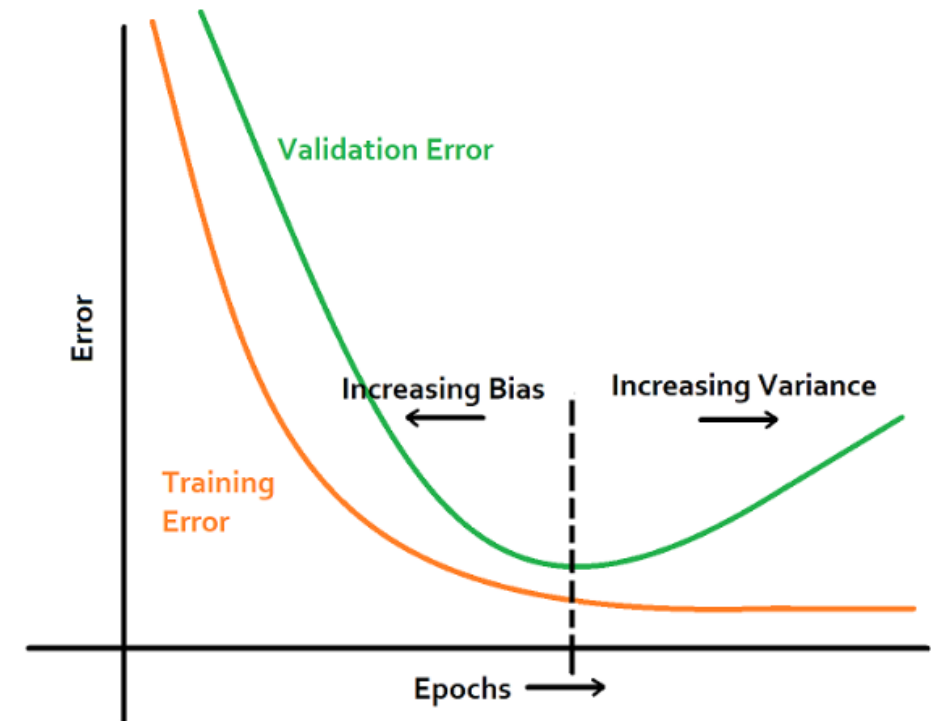


Figure courtesy of RAHUL JAIN

Optimization: Batches

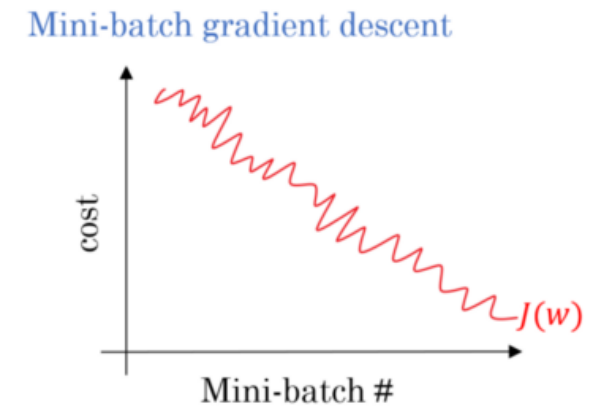
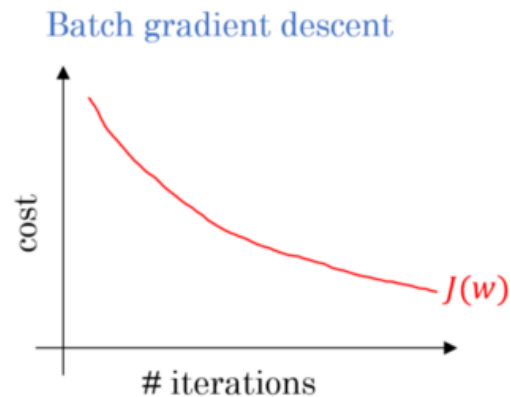
- Datasets are usually huge and won't fit in GPU memory in its entirety.
- So, we split the dataset into batches.
 - This is also called SGD (Stochastic Gradient Descent) or mini-batch GD.
- What is the effect of using batches?
 - Speeds up convergence.



Figure courtesy of [Ashish Singhal](#)

Optimization: Batches

- Gradient descent will take the model to the closest minima, not necessarily the global minima.
- By taking batches, we introduce noisiness (randomness) to the loss surface, which may help us avoid local minima.



Optimization: Momentum

- Adding a momentum term (i.e., gradients from previous epochs), helps the convergence process.

$$\begin{aligned} z^{k+1} &= \beta z^k + \nabla f(w^k) \\ w^{k+1} &= w^k - \alpha z^{k+1} \end{aligned}$$

Momentum coefficient

Momentum

Gradient

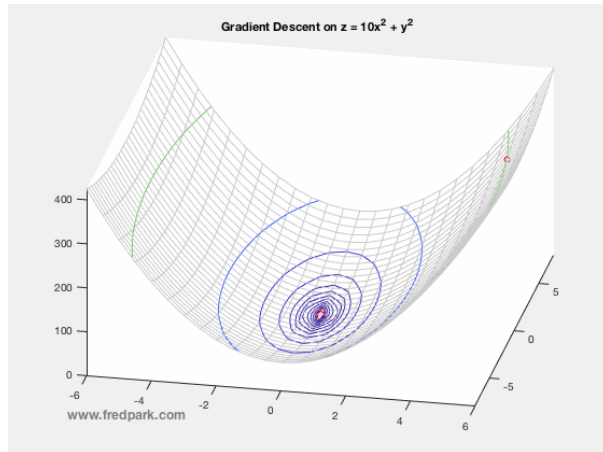


Figure courtesy of [Fred Park](#)

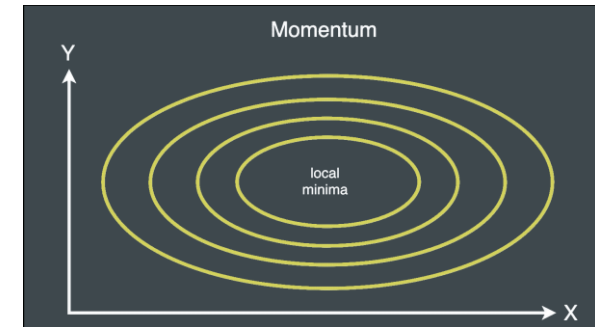
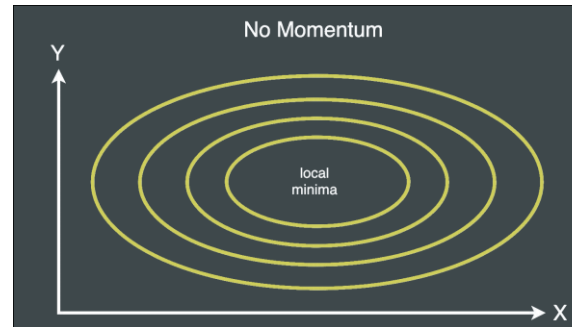
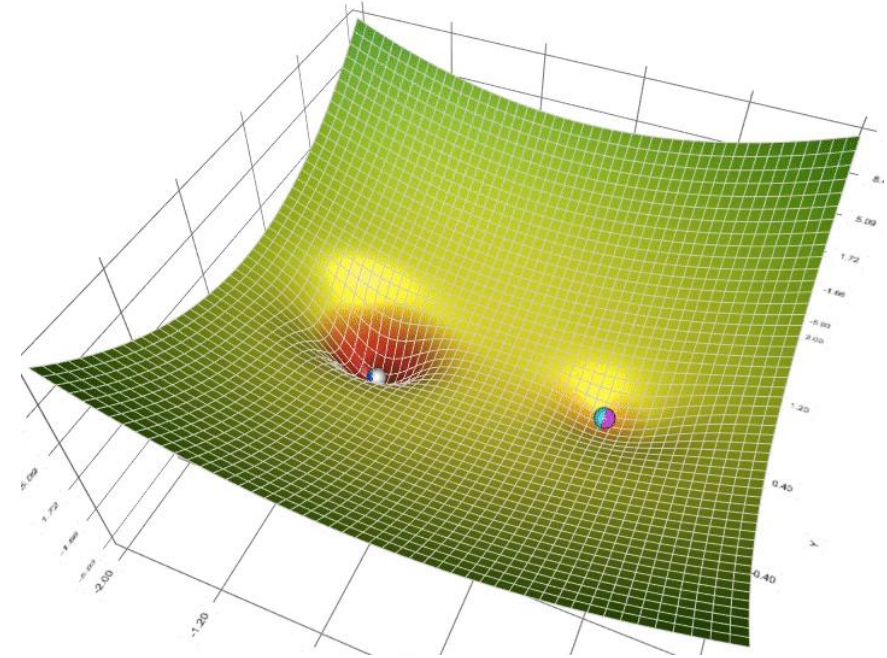


Figure courtesy of [Casper Hansen](#)

Optimization: Optimizer

- Optimizers differ in how they scale the gradient differently over epochs and different weights.
- Different optimizers perform differently for different models and datasets.
- More mathematical info can be found [here](#).



Animation of 5 gradient descent methods on a surface: gradient descent (cyan), momentum (magenta), AdaGrad (white), RMSProp (green), Adam (blue). Left well is the global minimum; right well is a local minimum

Figure courtesy of Lili Jiang

Optimization: Regularization

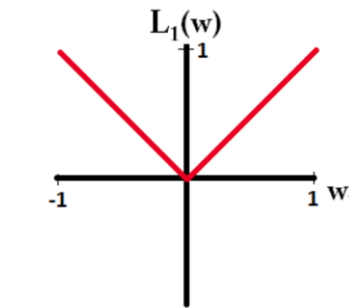
- Large models allow modeling more complex data.
 - However, if too large, they may overfit.
- We need a way to dynamically control the complexity.
- How about we add an additional loss term to it?!

$$\nabla_W (L_{Error}) \longrightarrow \nabla_W (L_{Error} + \beta L_{Regularization})$$

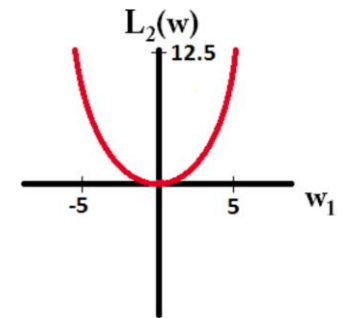
Trade-off coefficient (for control)

Optimization: Regularization

- How about we add a loss term for it?!
 - We want to construct a loss term such that, when minimized, the model becomes less complex.
 - We can do that by controlling the magnitudes of the model weights.
- This is also called *weight decay*.
- [Demo](#)



$$L_1(w) = \sum_i |w_i|$$



$$L_2(w) = \frac{1}{2} \sum_i w_i^2$$

Figure courtesy of Prashant Gupta

Weight Initialization

- Has a great impact on optimization.
 - Improper initialization (e.g., all zeros or [constants](#)) leads to gradient pathologies.
- [Demo](#)

More Hyper-Parameters Later...

MLPs for Regression

- Just like they create decision boundaries in classification problems, neurons can be used to create discrete approximations in regression problems.
- Demo

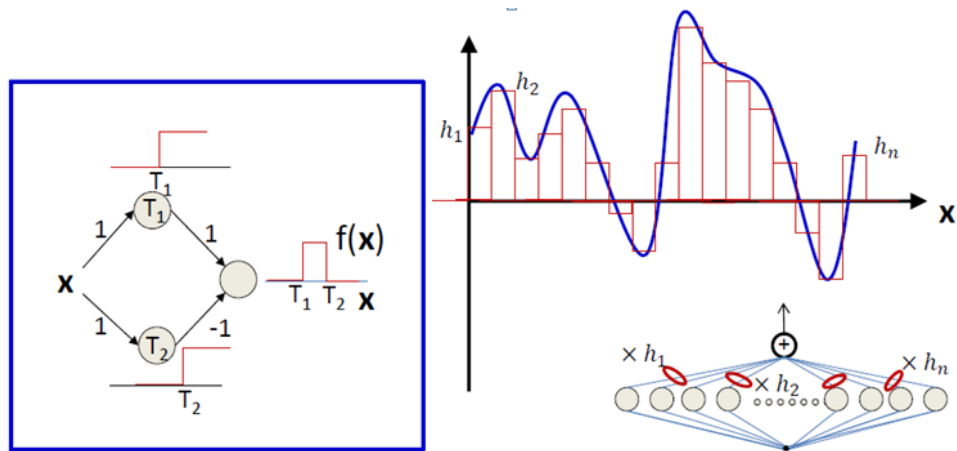


Figure courtesy of [Abhishek](#)

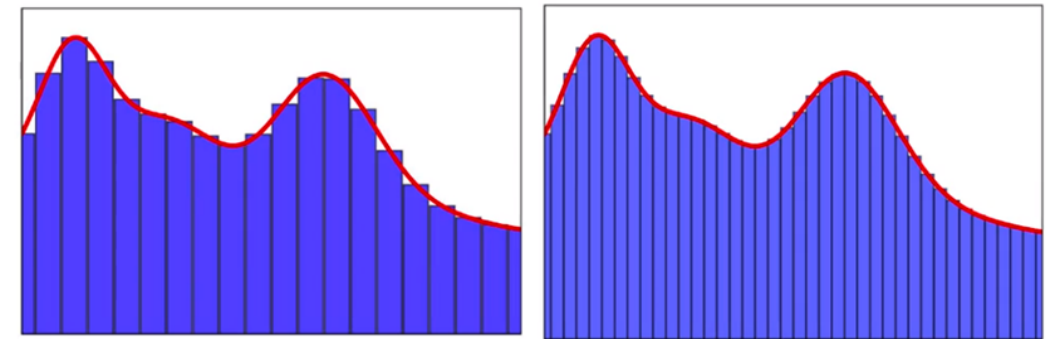


Figure courtesy of [Niranjan Kumar](#)