Intro to Neural Networks

BA865 – Mohannad Elhamod

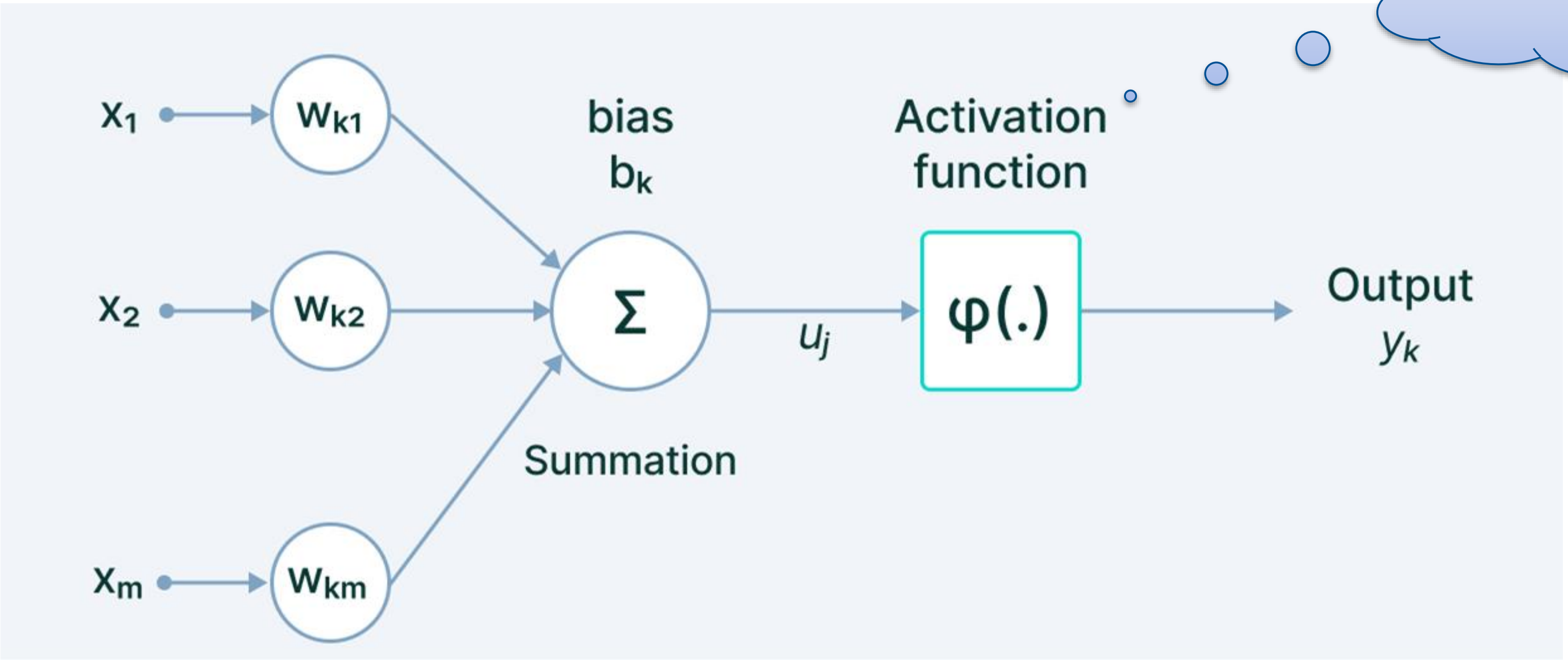
MLPs

The Multi-Layer Perceptron

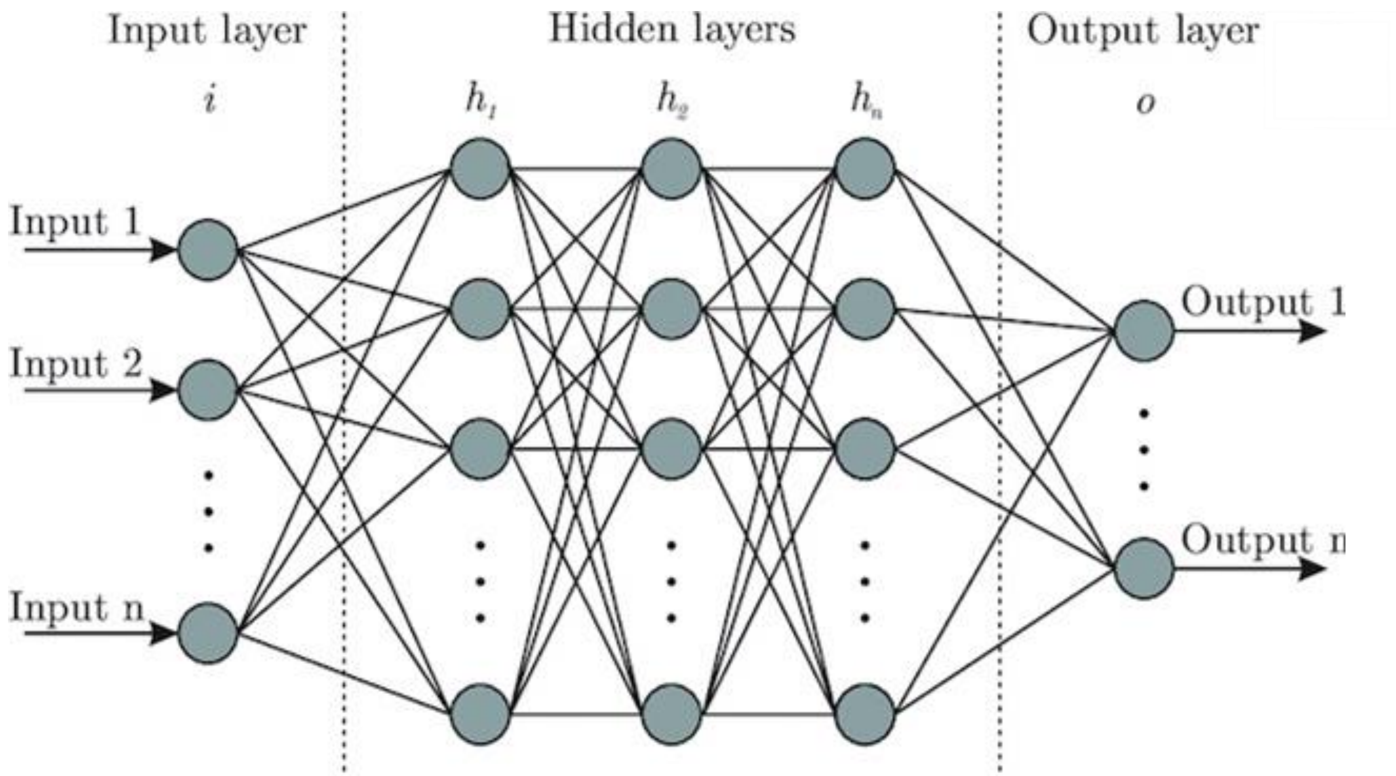
A Neuron

- A more fashionable/general term for the perceptron.

Non-linearity

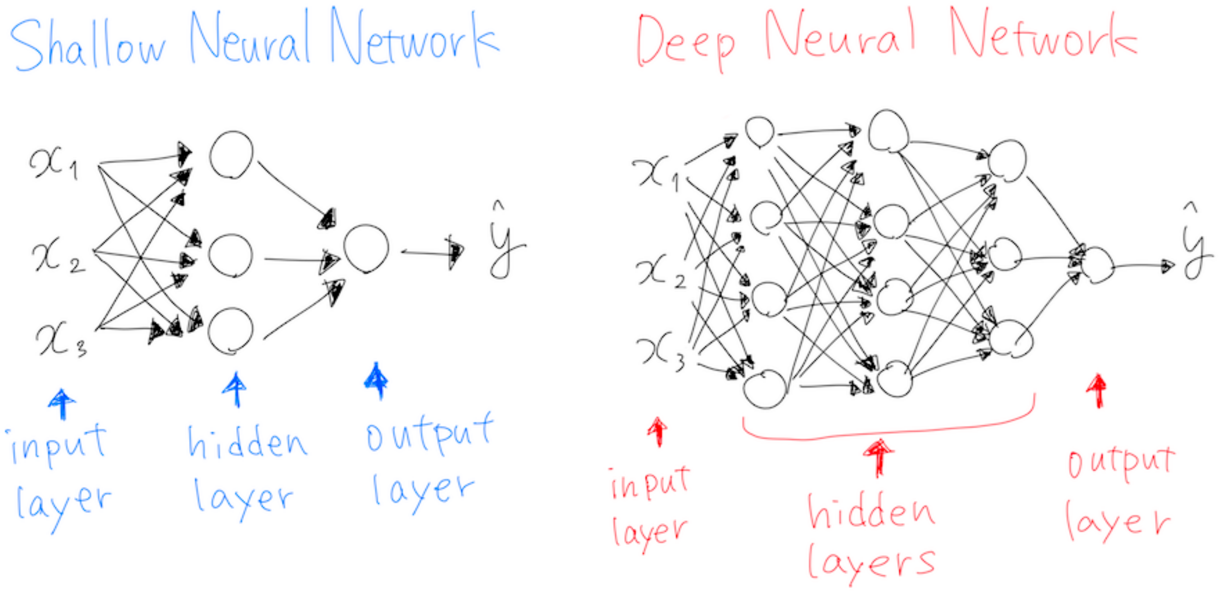


Neural Networks



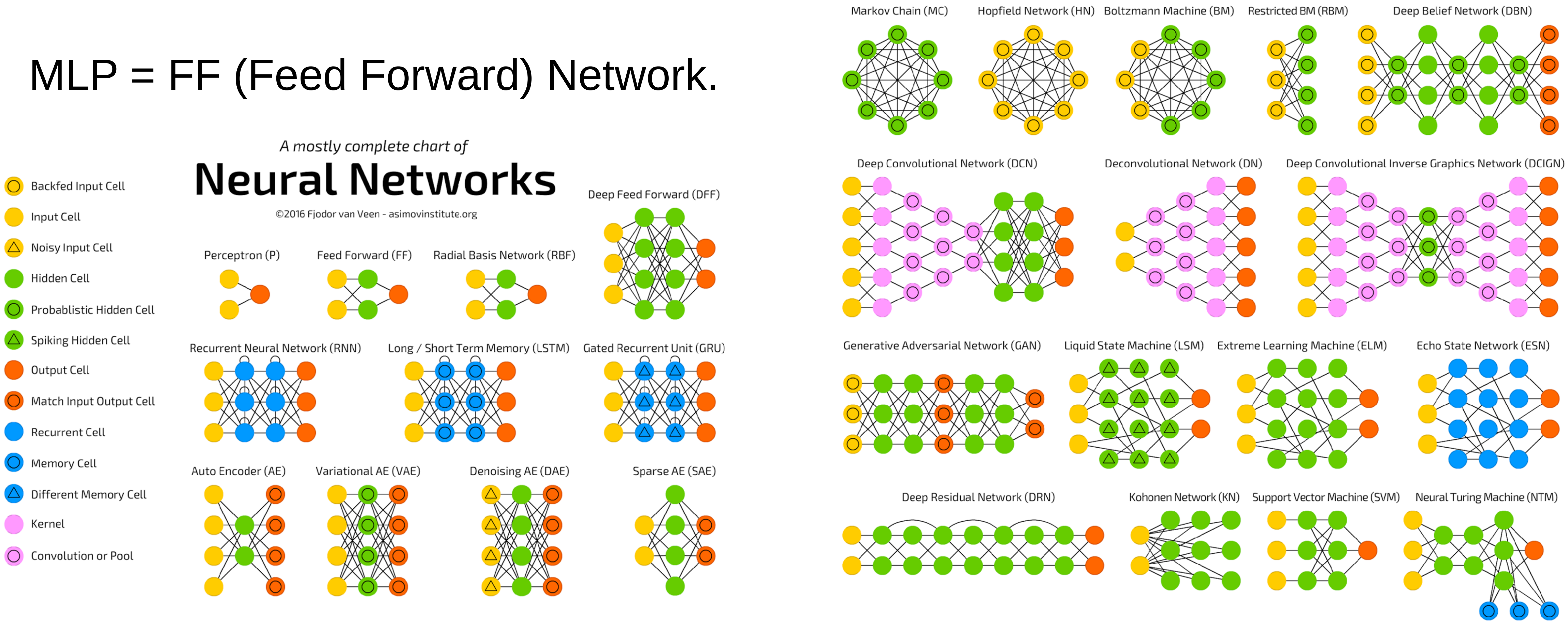
Deep Networks

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



MLPs, One of Many Types…

- MLP = FF (Feed Forward) Network.

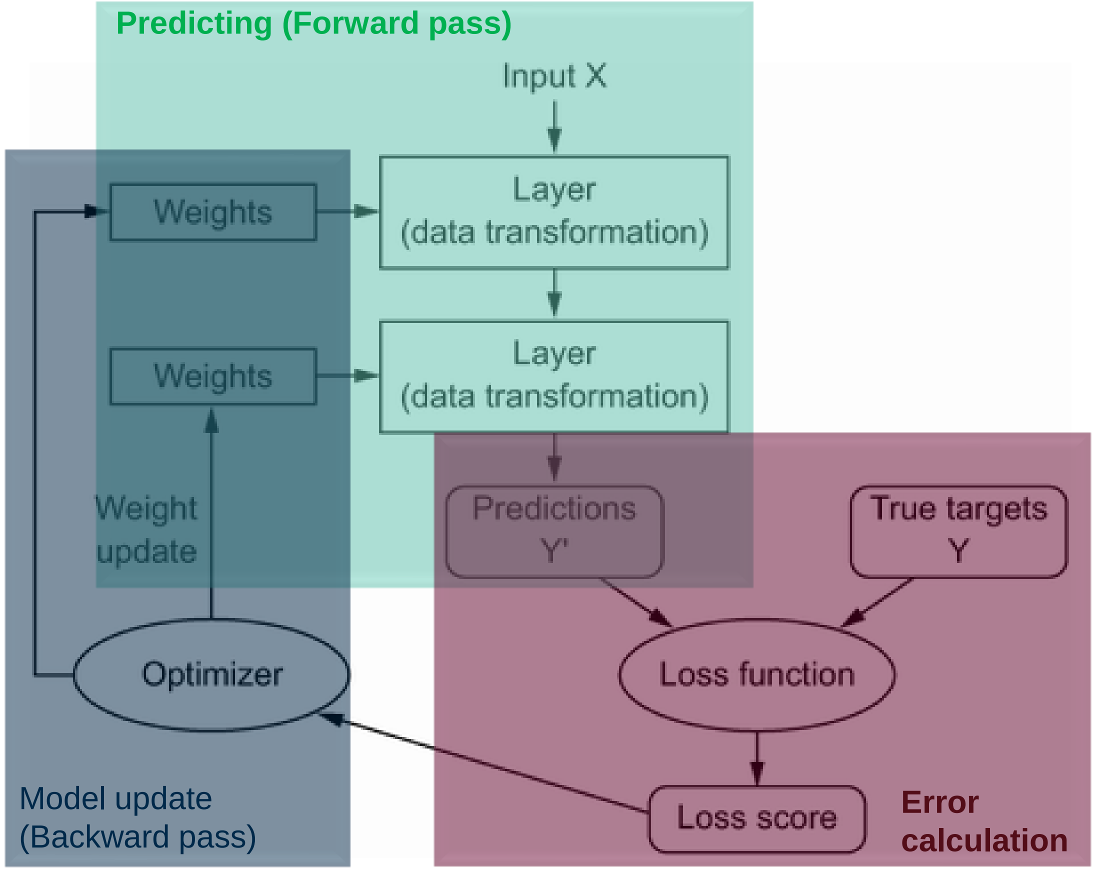


Disecting The Neural Network

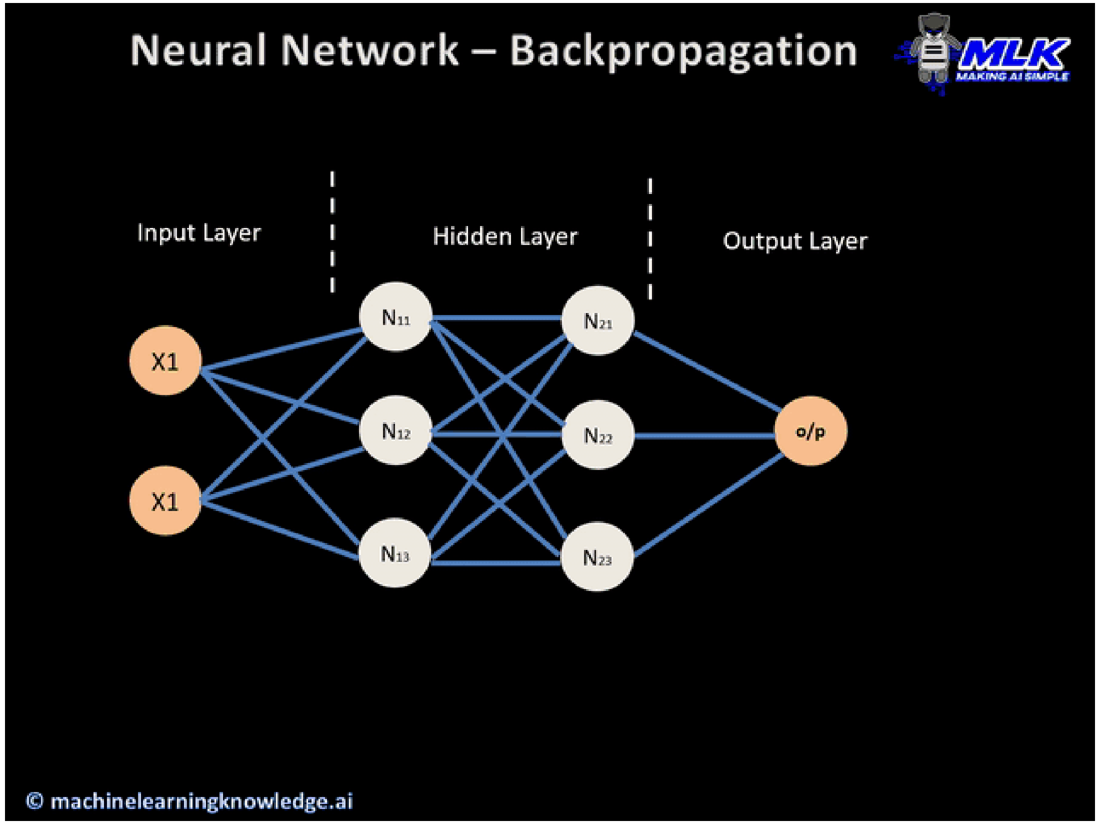
BOSTON UNIVERSITY

Boston University Questrom School of Business

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?!

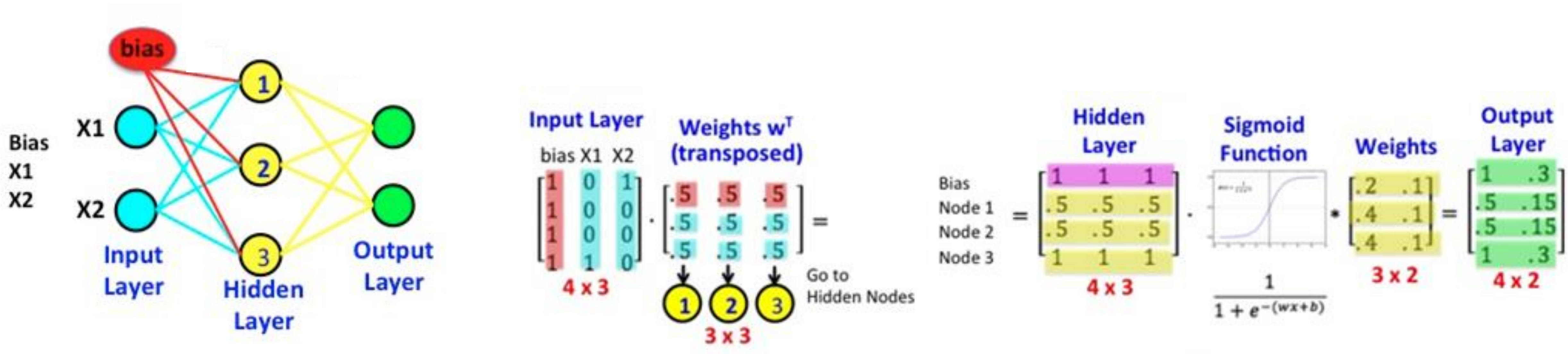


Figure (modified) courtesy of Rubens Zimbres

Predicting

- Demo
- The non-linearities allow the neural net to “warp” a non-linear problem into a linear one!

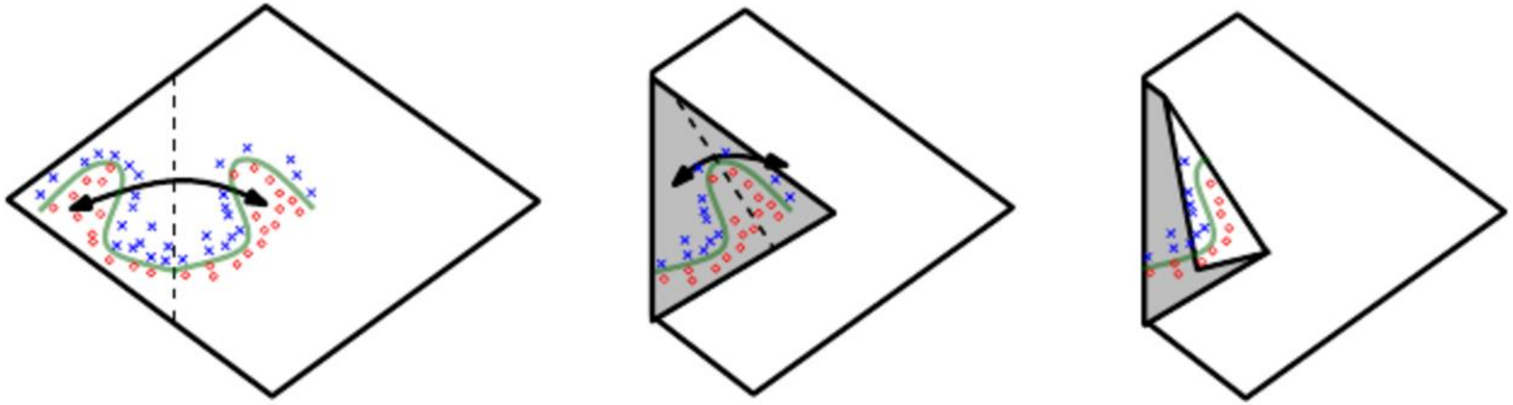


Figure courtesy of Deep Learning Book

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…

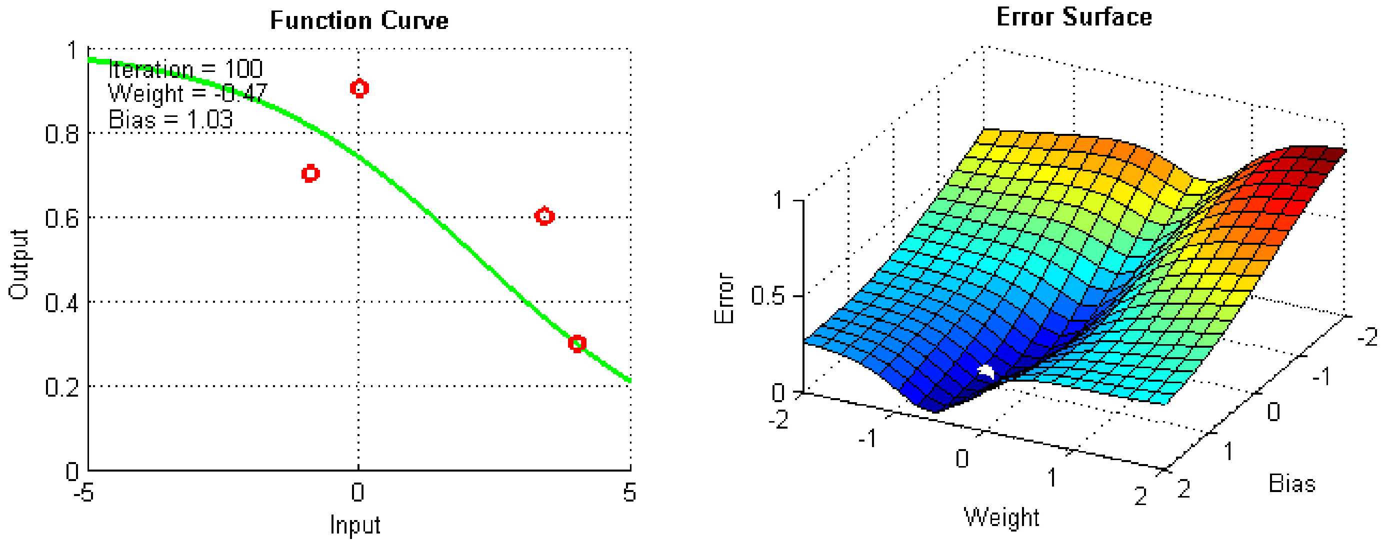
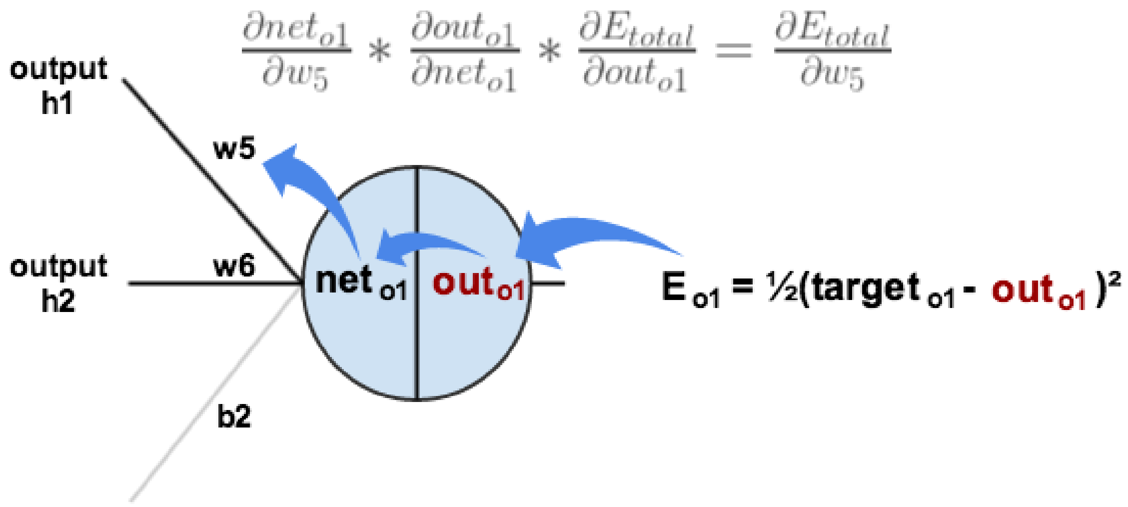


Figure courtesy of Devin Soni

Optimization

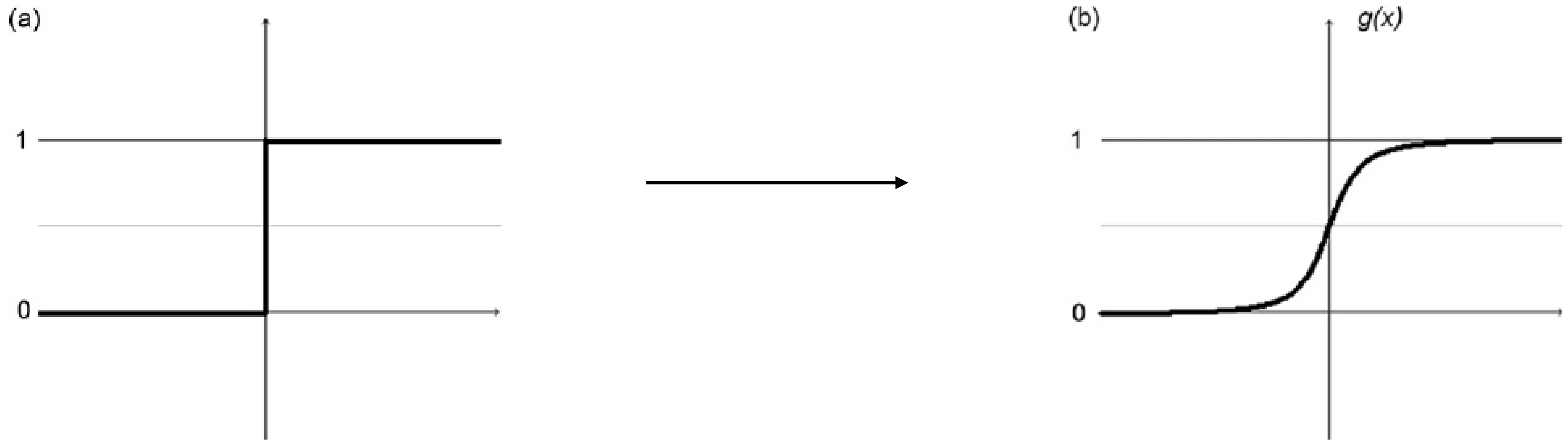
- But how does the update get carried all the way back?
  - Chain rule!
  - This is called back-propagation.

- 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.

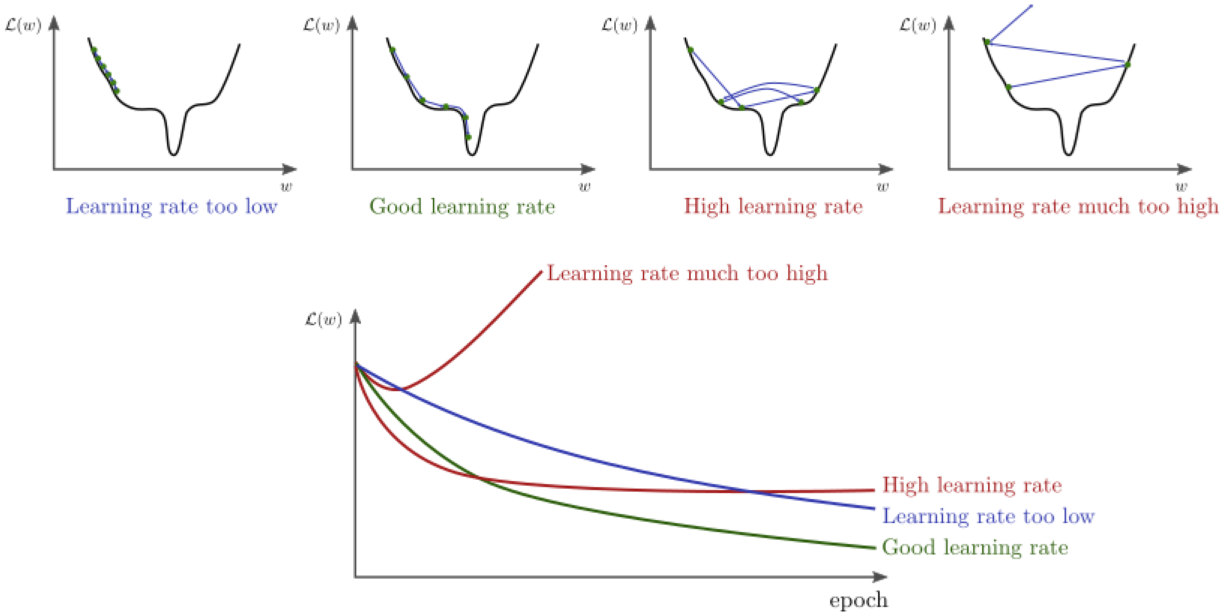


Figure courtesy of Stanford CS class CS231n

Learning Rate: Schedulers

- To get the best of both worlds, you could adjust the learning rate in phases.
  - This way, you still converge but faster.

- Using a scheduler is a common practice.

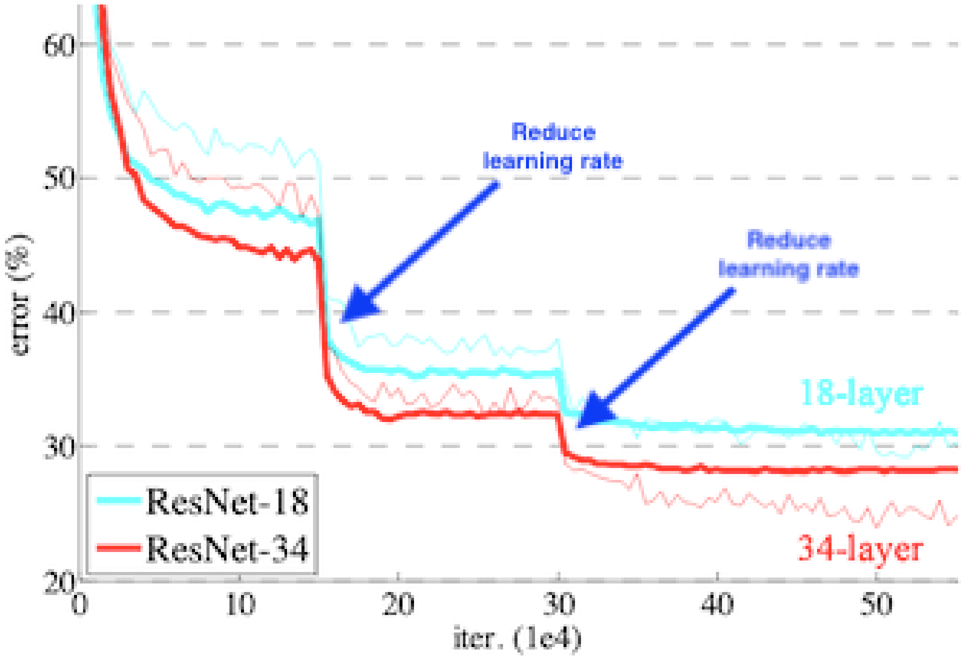


Figure courtesy of B. D. Hammel

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.

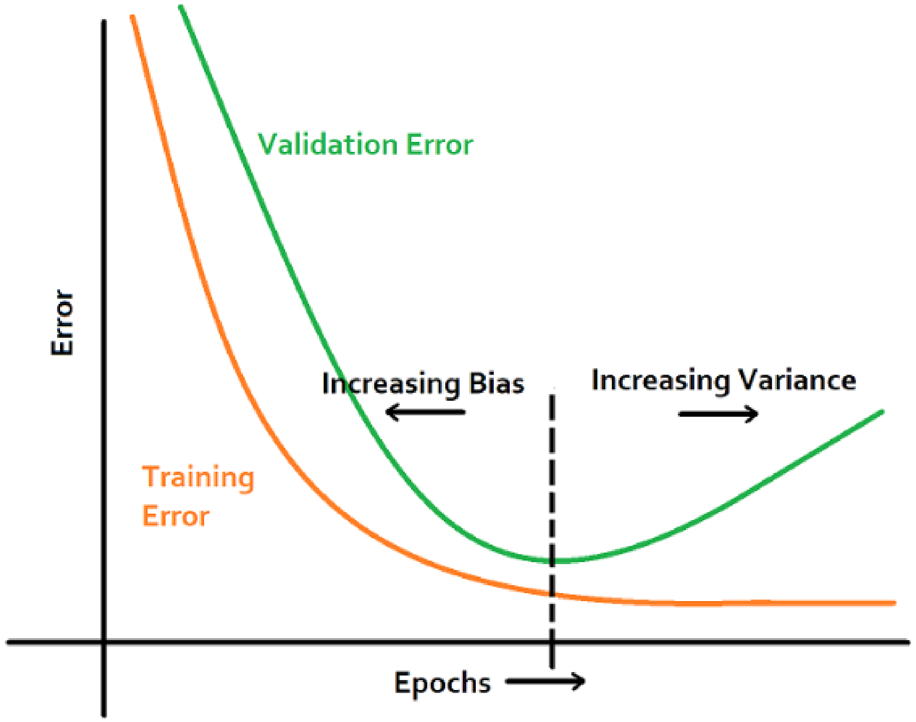


Figure courtesy of RAHUL JAIN

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.

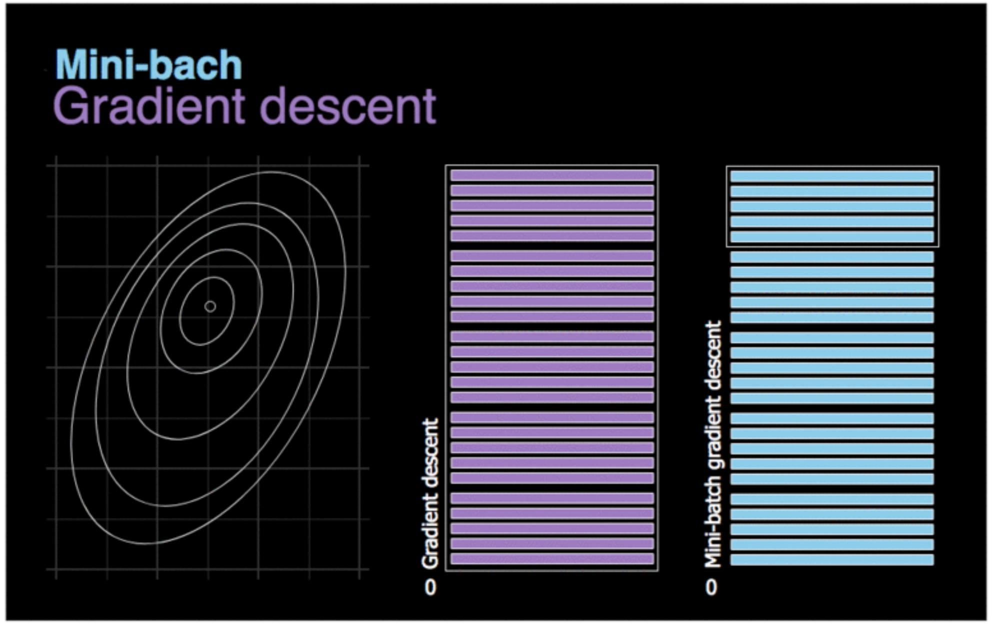
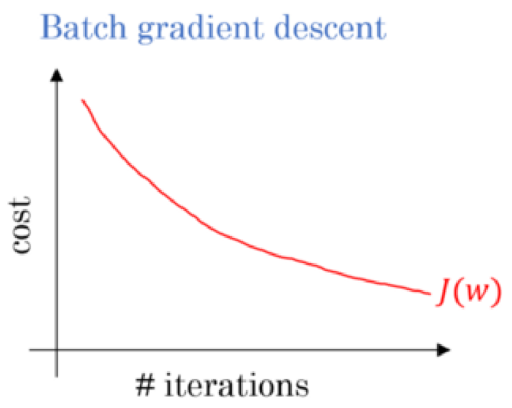
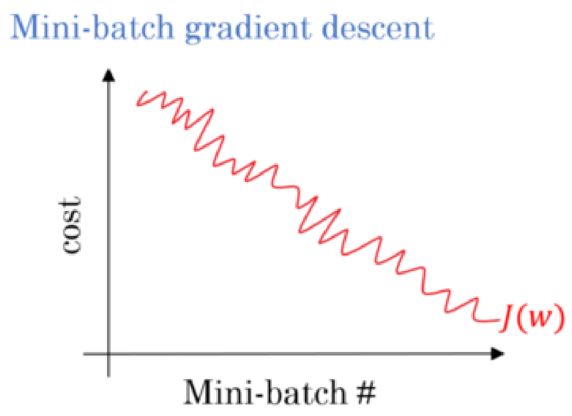


Figure courtesy of Ashish Singhal

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.

$$z^{k+1} = \overbrace{\underbrace{\beta}_{\text{Momentum coefficient}} z^k}^{\text{Momentum}} + \underbrace{\nabla f(w^k)}_{\text{Gradient}}$$

$$w^{k+1} = w^k - \alpha z^{k+1}$$

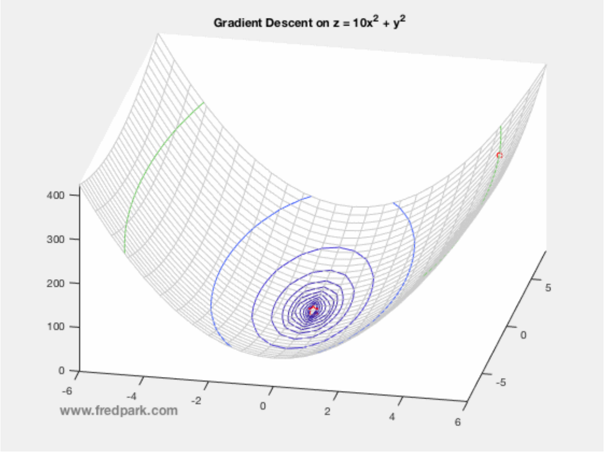
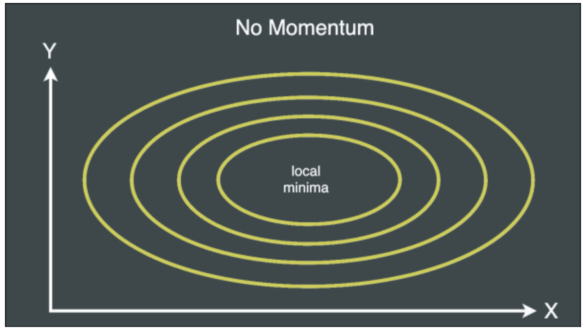


Figure courtesy of Fred Park



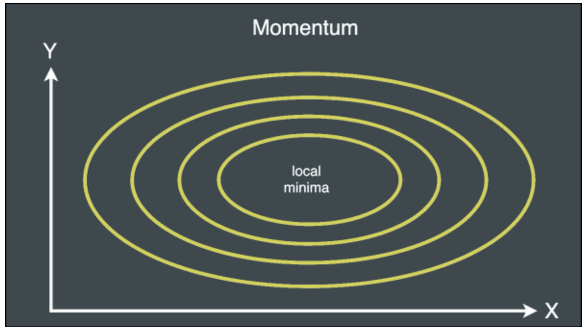
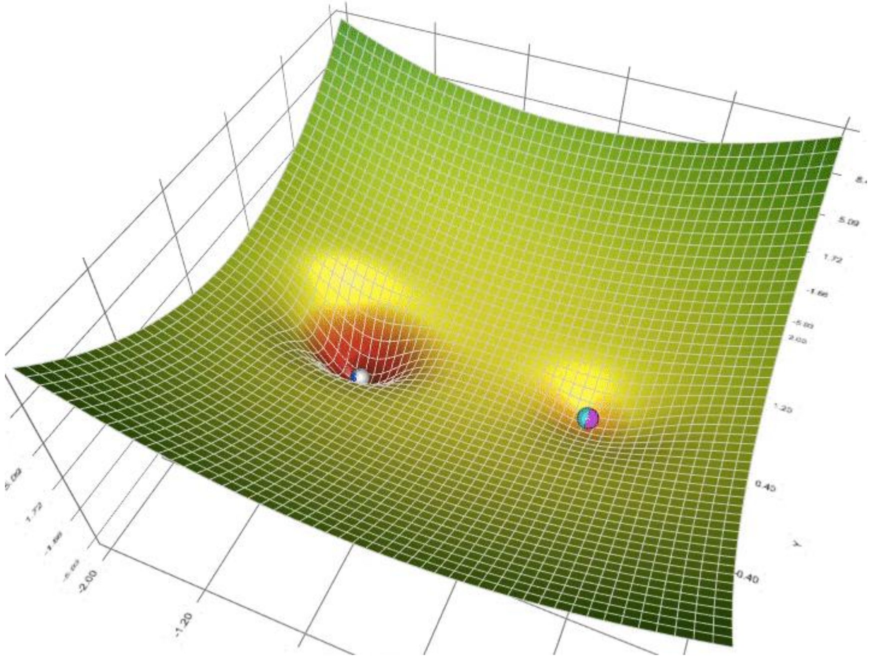


Figure courtesy of Casper Hansen

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

More Hyper-Parameters Later...