Beyond Gradient Descent

Unit 4.1

MDS 655

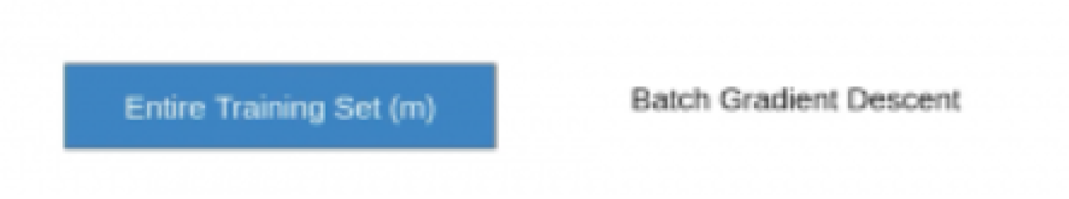
The Challenges with Gradient Descent

- Local minima
- Vanishing and exploding gradient
- Requires massive labeled datasets (ImageNet, CIFAR…)
- Requires better hardware (GPU)
- Several algorithms needs to be designed

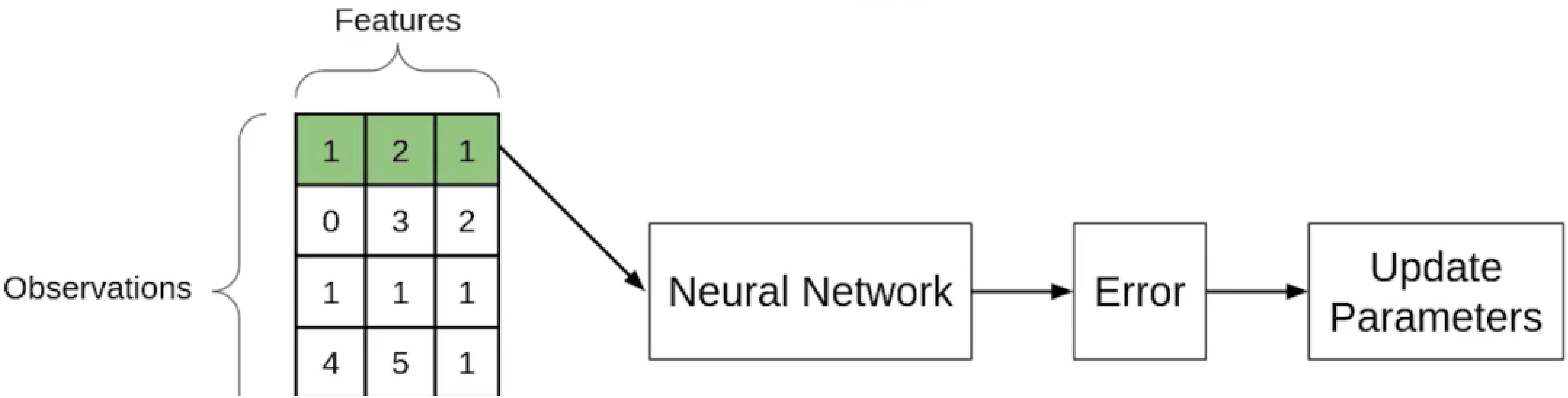
Breakthroughs to tackle the challenges

- Layer-wise greedy pre-training
- mini-batch gradient descent
- Training models in an end-to-end fashion
- Nonconvex optimizers

Batch Gradient

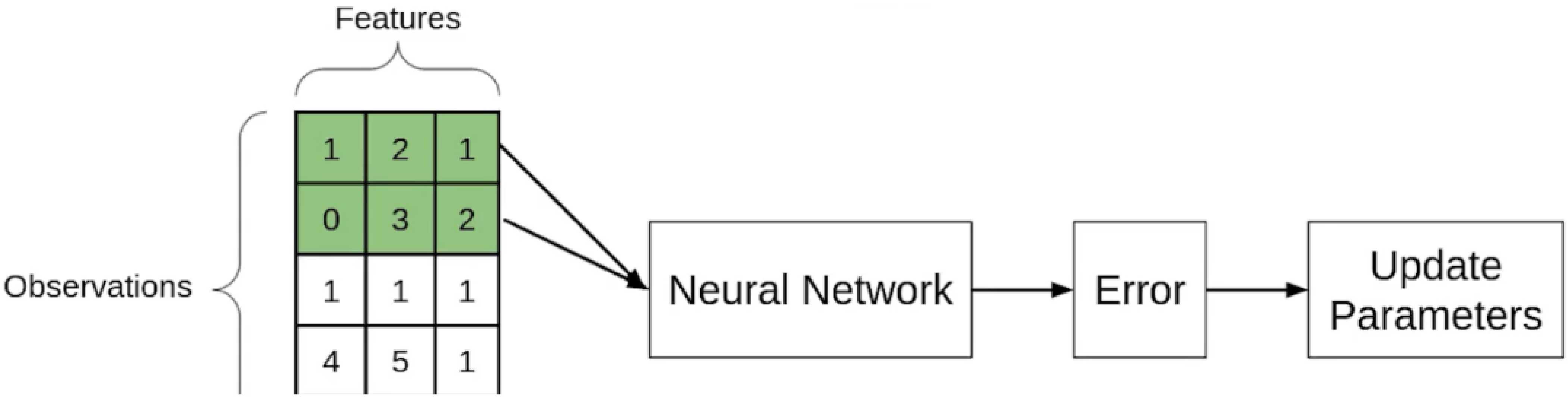


SGD

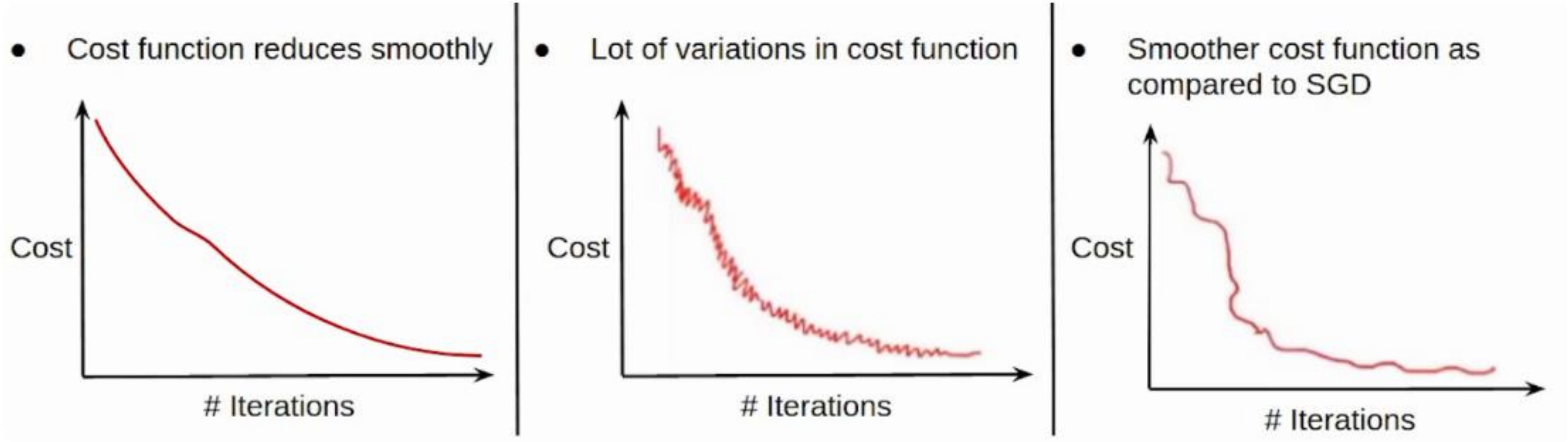


Features
Observations
1 2 1
0 3 2
1 1 1
4 5 1
Neural Network
Error
Update Parameters

Mini Batch: Batch size=2



Batch/SGD/Mini Batch

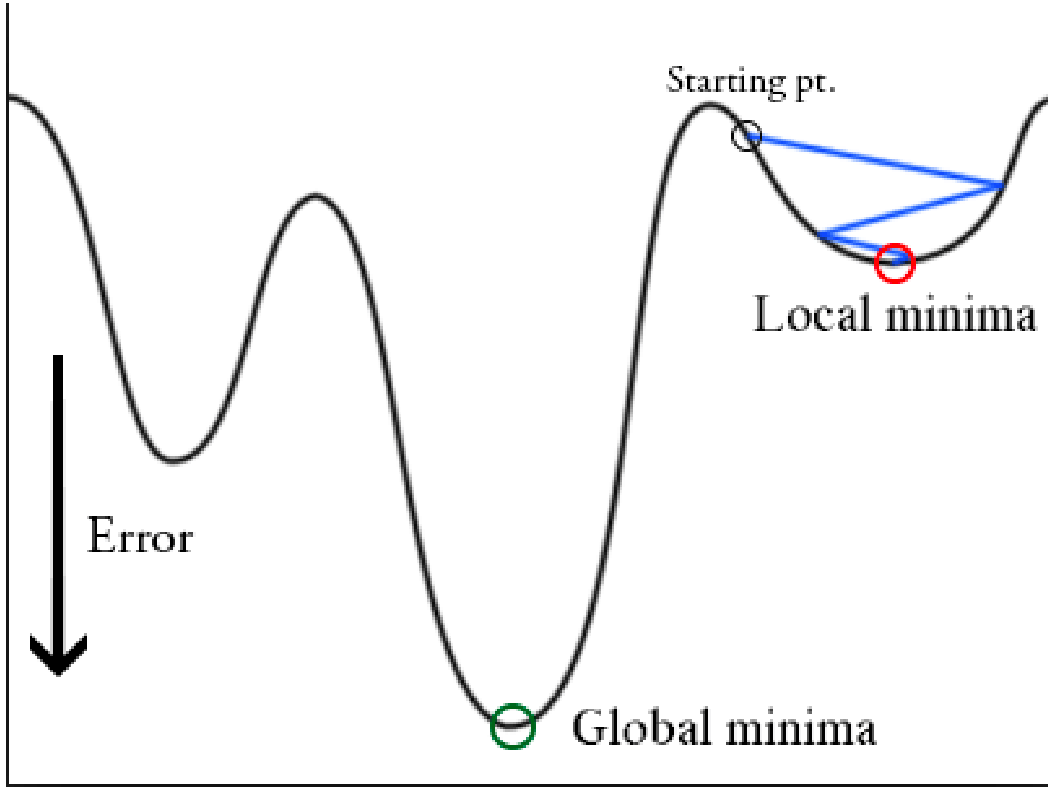


Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
• Entire dataset for updation	• Single observation for updation	• Subset of data for updation
• Cost function reduces smoothly	• Lot of variations in cost function	• Smoother cost function as compared to SGD
• Computation cost is very high	• Computation time is more	• Computation time is lesser than SGD
		• Computation cost is lesser than Batch Gradient Descent

Local Minima in the Error Surfaces of Deep Networks

- With minimal local information inferring global structure of the error surface is a challenge
- Correspondence between local and global structure is very little

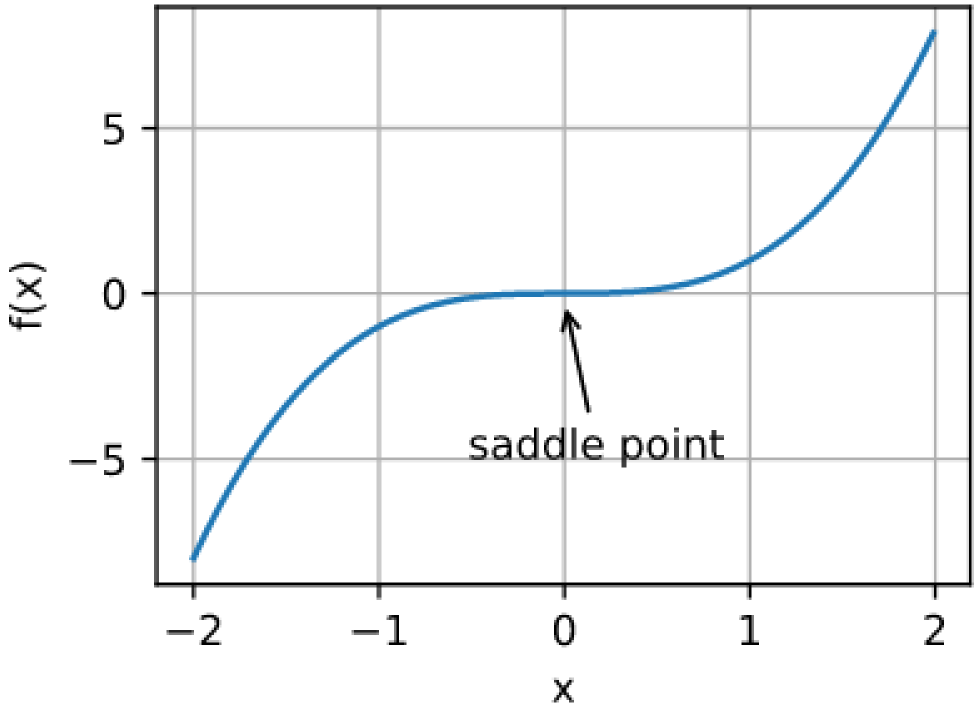
- Finding global minima is easy
    - Convex error surface
- Finding global minima is difficult
    - Non-Convex error surface



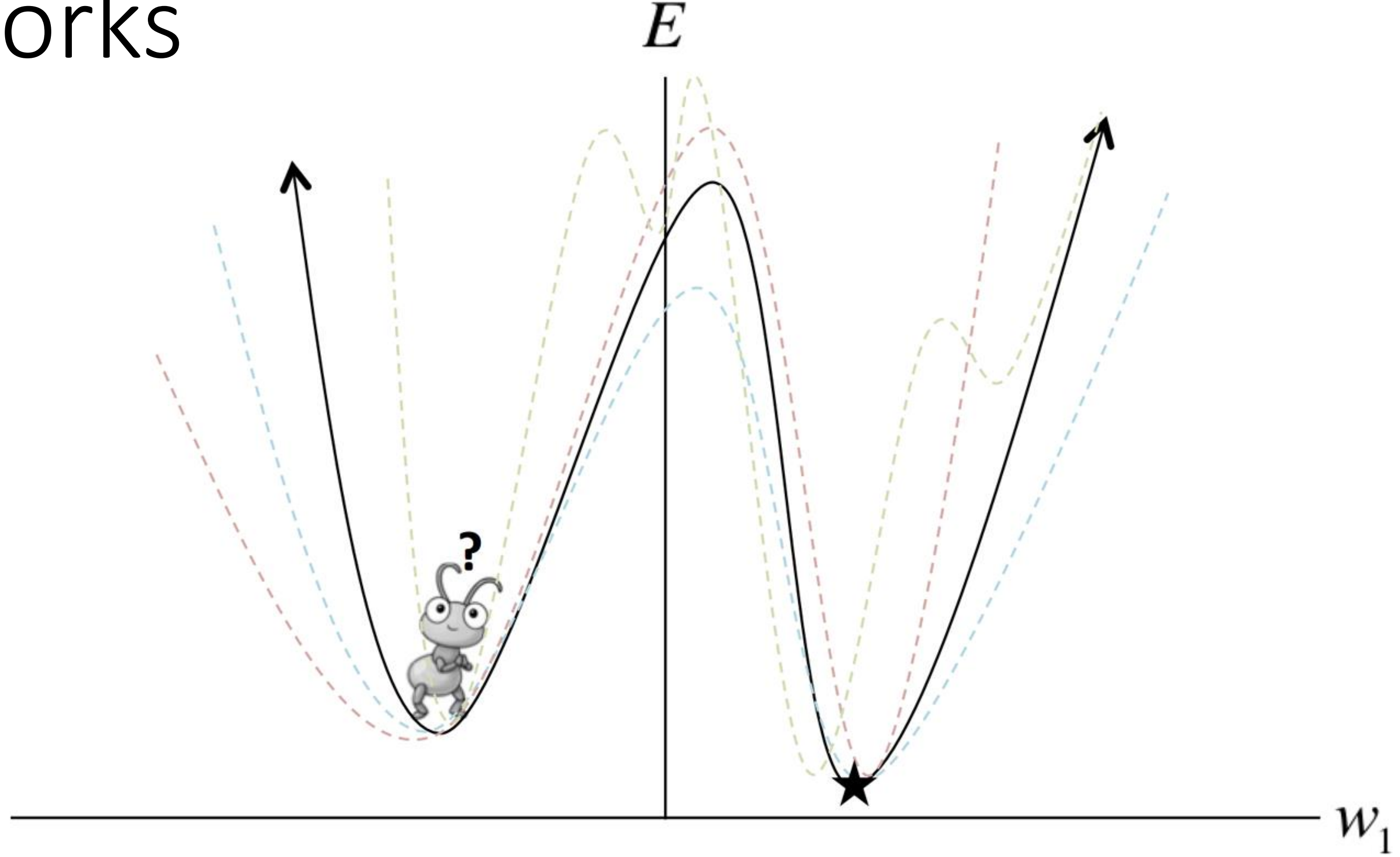
- Convergence point of gradient descent depends on the starting point and learning rate
- Slope of cost functions is very close to zero at local minima and thus model stops learning

Saddle point

- points where the function attains neither a local maximum value nor a local minimum value
- Slope at this point is also very near to zero and thus model stops learning



Local Minima in the Error Surfaces of Deep Networks



Mini-batch gradient descent may aid in escaping shallow local minima
but
often fails when dealing with deep local minima

Vanishing and exploding gradient

- If Partial derivative are large
  - Gradient will increase exponentially
  - Back propagation with lots of epochs will cause problem of exploding gradient
  - Eventually will overshoot the minima
- If Partial derivative are small
  - Gradient will decrease exponentially
  - Back propagation with lots of epochs will cause problem of vanishing gradient
  - Eventually causes the weights update unchanged

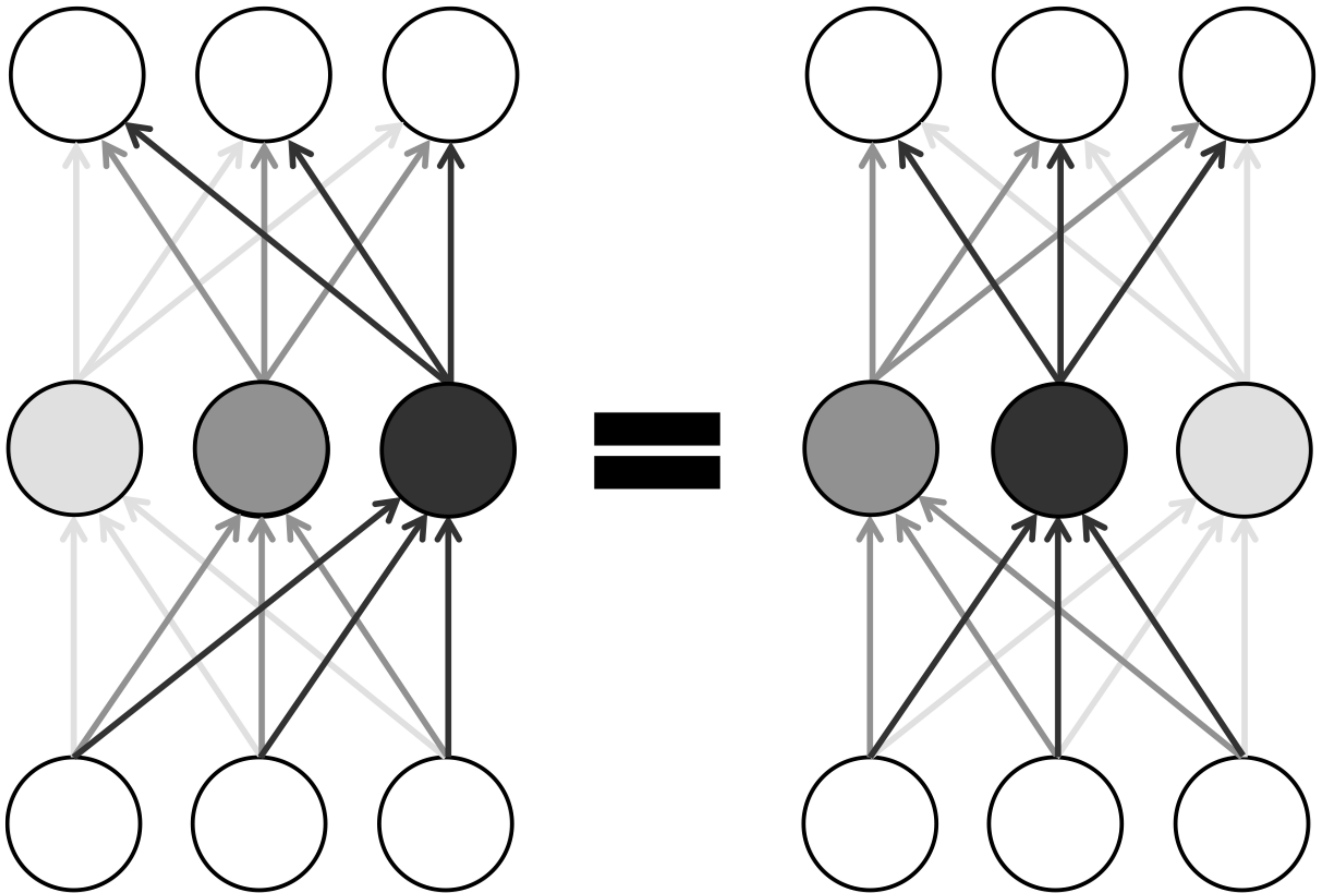
Model Indentifiability

Neural Networks are not identifiable

First source of local minima

- within a layer with *n* neurons
  - there are $n!$ ways to rearrange parameters
- for a deep network with l layers, each with *n* neurons
  - we have a total of $n!^l$ equivalent configurations

Rearranging neurons
In a layer of a neural
network results in
equivalent configurations
due to symmetry



Model Non-Indentifiability

- Second source of local minima
- For eg. ReLU neuron
  - ReLU uses a piecewise linear function
  - Infinite number of equivalent configurations

- No change in the behavior of the network
  - Multiplying all of the incoming weights by any nonzero constant *k*
  - *S*caling all of the outgoing weights by 1/*k*

Local minima are not problematic

- Caused due to non-identifiability characteristic of neural network
- As nonidentifiable configurations behave in an indistinguishable fashion no matter what input values are
- i.e. they will achieve the same error on the training, validation, and testing datasets
- All of these models will have learned equally from the training data
- will have identical behavior during generalization to unseen examples

Local minima are only problematic

- When they are spurious
- It corresponds to a configuration of weights in a neural network that incurs a higher error
- Gradient based optimization methods will be a failure

How Pesky Are Spurious Local Minima in Deep Networks?

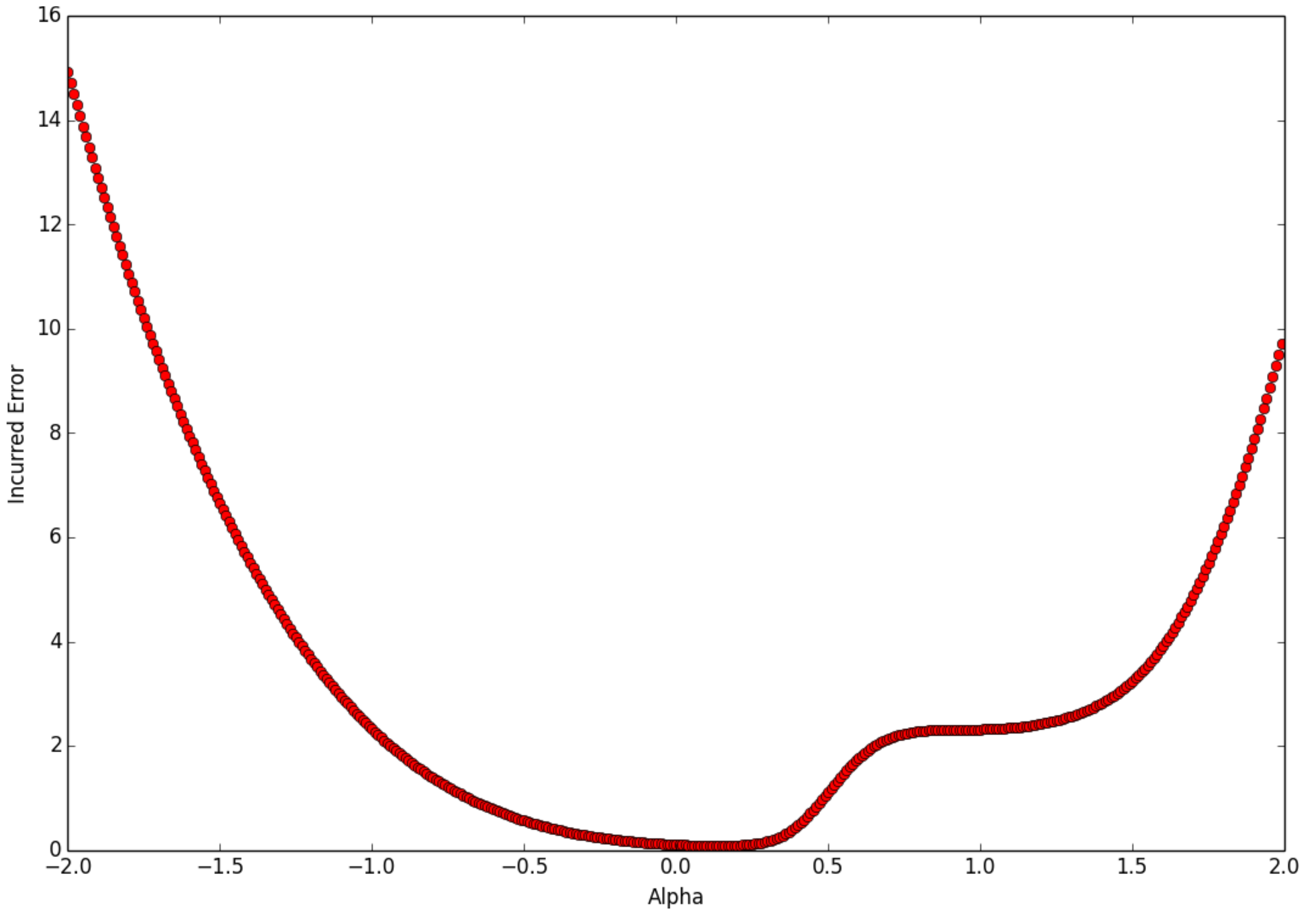
- Local minima have error rates and generalization characteristics that are very similar to global minima
- Plotting the error values does not give enough information about the error surface
- Goodfellow et al. (a team of researchers collaborating between Google and Stanford) published a paper in 2014

- Instead of analyzing the error function over time
  - investigated what happens on the error surface between a randomly initialized parameter vector and a successful final solution
  - used linear interpolation.

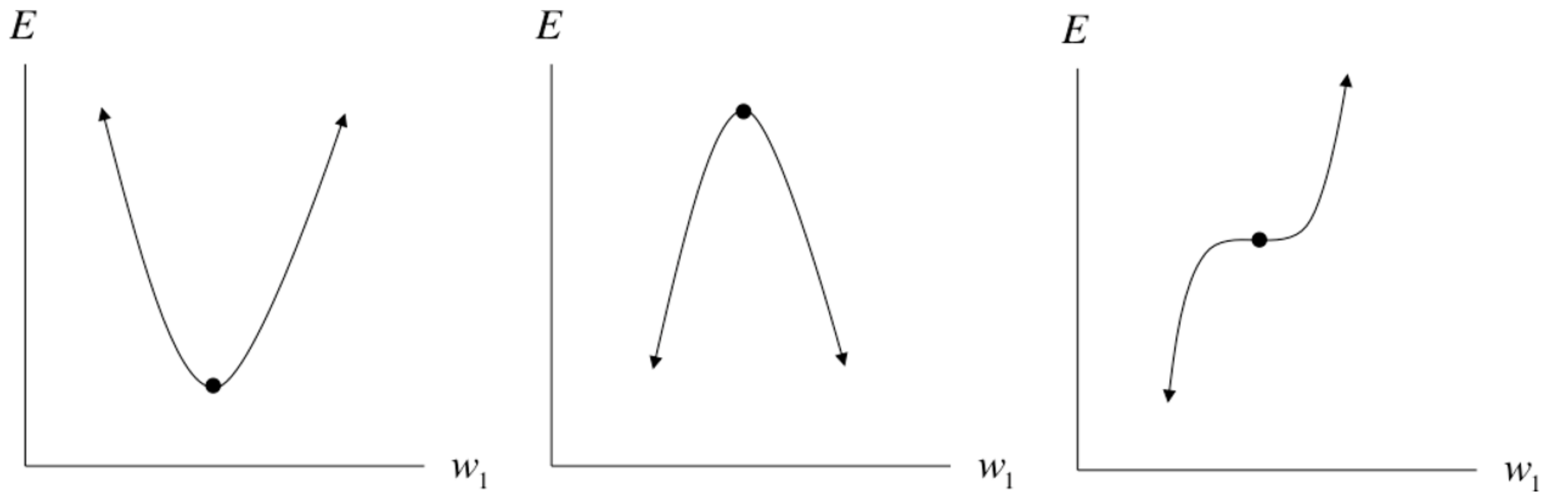
$$\theta_\alpha = \alpha \cdot \theta_f + (1 - \alpha) \cdot \theta_i.$$

Error function SGD randomly initialized parameter vector

- True struggle of gradient descent isn't the existence of troublesome local minima
- But instead is that we have a tough time finding the appropriate direction to move in



Flat regions in the error surface



- Assuming each of these three configurations is equally likely
- Given a random critical point in a random one-dimensional function
  - it has one-third probability of being a local minimum
  - if we have a total of k critical points, we can expect to have a total of $k/3$ local minima

- A random function with k critical points has an expected number of $k/3^d$ local minima
- In other words, as the dimensionality of our parameter space increases, local minima becomes exponentially more rare