Note: This is the summary note from Udacity Introduction to Deep Learning with PyTorch

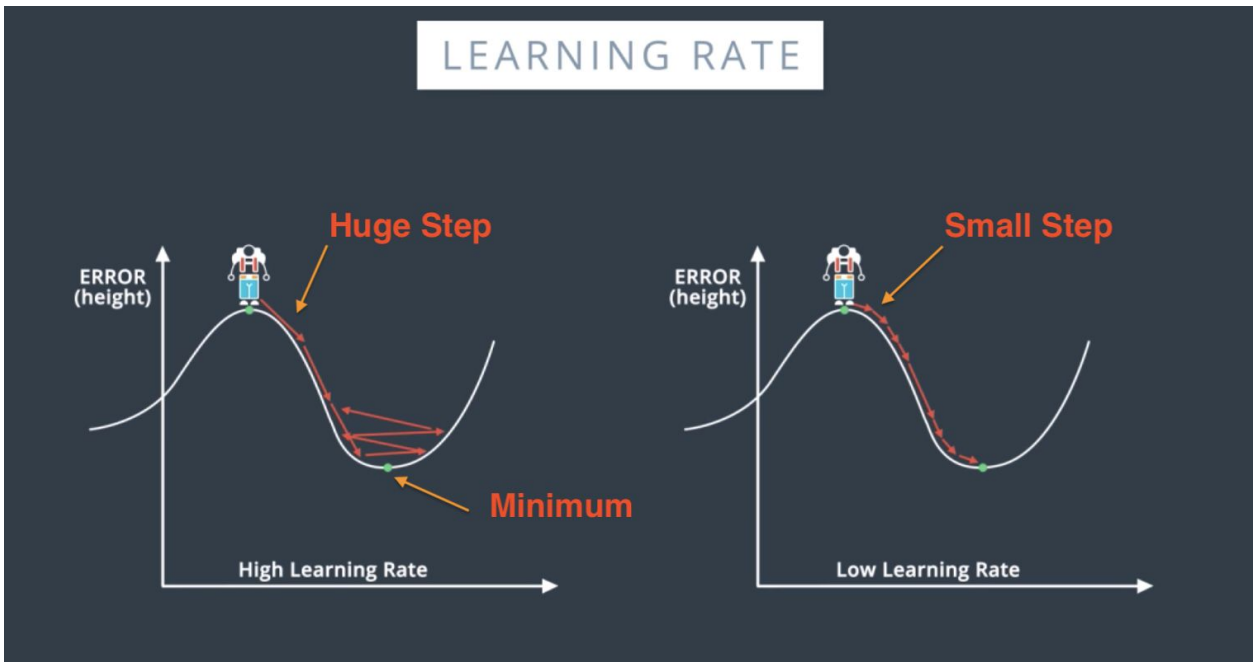
Batch vs Stochastic Gradient Descent

- Problem Statement
  - With the normal gradient descent, each step (epoch):
    - We take input (all of our data), run through entire neural network
    - Find prediction
    - Calculate error (how far they are from actual label)
    - Back-propagate this error to update the weight in the neural network
  - The step above is done for all data => it can be a huge matrix computations in case we have many data points
- Solutions
  - Stochastic GD:
    - take small subsets of data (batch)
    - Run through neural network
    - Calculate gradient of error function based on those points
    - Back-propagate
    - Move one step in that direction

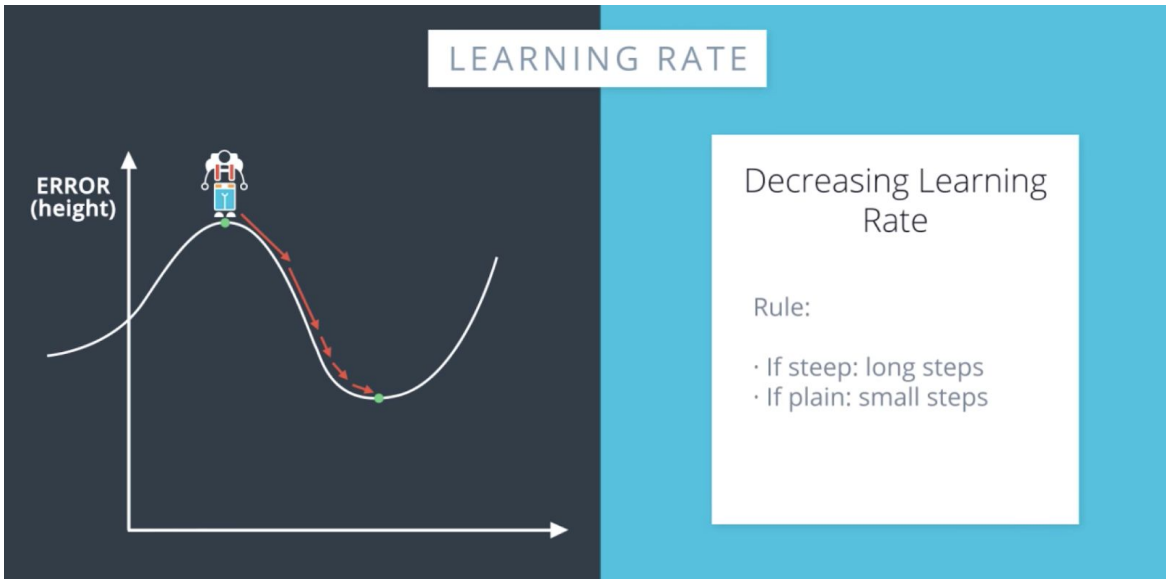
Normal GD	Stochastic GD
Take one step at a time	Take x step at a time (x is batch size)
More accurate	Less accurate but better in practices with many data
In case many data, it consume a lot of memories and storages	Less resources needed

Learning Rate Decay

- Problem: what is the learning rate to use ?
- General rule:
    - if the model is not working, decrease the learning rate, why so ?
    - Best learning rates are those which decrease as the model is getting closer to a solution



- Huge step can be faster, but can miss out minimum
- Small step slow model, but a good chance to find the minimum



Momentum

- Problem: How to find a local minimum?
- It is included in 08_Neural_Network, https://docs.google.com/document/d/1SUzhGwHmiZeJWDCvsYNIezDfwVcoD-nbzyDS-VdtqYE/edit

Error Functions Around the world

- There are multiple error functions that can be used
- What we know so far:
    - Log-Loss function
    - Cross entropy
    - Maximum likelihood

==========

Mini Summary

- Why we need Stochastic Gradient Descent ?
- How do we know if this is the right learning rate ?
- There are a lot other error functions