Gradient Descent

An optimization algorithm

Instructor, Nero Chan Zhen Yu

Forward School

Optimization

- Frequently when doing data science, we'll be trying to the find the best model for a certain situation. And usually "best" will mean something like "minimizes the error of its predictions" or "maximizes the likelihood of the data." In other words, it will represent the solution to some sort of optimization problem.
- This means we'll need to solve a number of optimization problems. And in particular, we'll need to solve them from scratch. Our approach will be a technique called *gradient descent*, which lends itself pretty well to a from-scratch treatment. You might not find it super-exciting in and of itself, but it will enable us to do exciting things throughout the book, so bear with me.

Forward School

The Idea Behind Gradient Descent

- Suppose we have some function f that takes as input a vector of real numbers and outputs a single real number. One simple such function is:

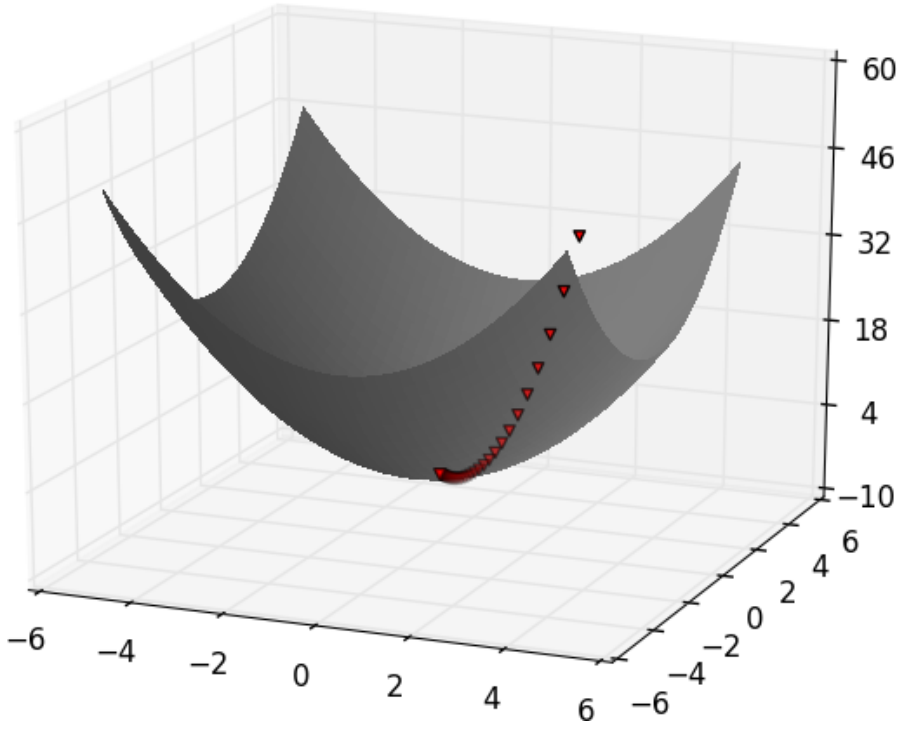
```python
from scratch.linear_algebra import Vector, dot

def sum_of_squares(v: Vector) -> float:
    """Computes the sum of squared elements in v"""
    return dot(v, v)
```

- We'll frequently need to maximize or minimize such functions. That is, we need to find the input v that produces the largest (or smallest) possible value.

Forward School

Gradient

- The gradient of such functions (if you remember your calculus, this is the vector of partial derivatives)
  - gives the input direction in which the function most quickly increases.
- Accordingly, one approach to maximizing a function is to pick a random starting point, compute the gradient, take a small step in the direction of the gradient (i.e., the direction that causes the function to increase the most), and repeat with the new starting point.
- Similarly, you can try to minimize a function by taking small steps in the opposite direction



Forward School

Estimating the Gradient

- If f is a function of one variable, its derivative at a point x measures how f(x) changes when we make a very small change to x. The derivative is defined as the limit of the difference quotients:

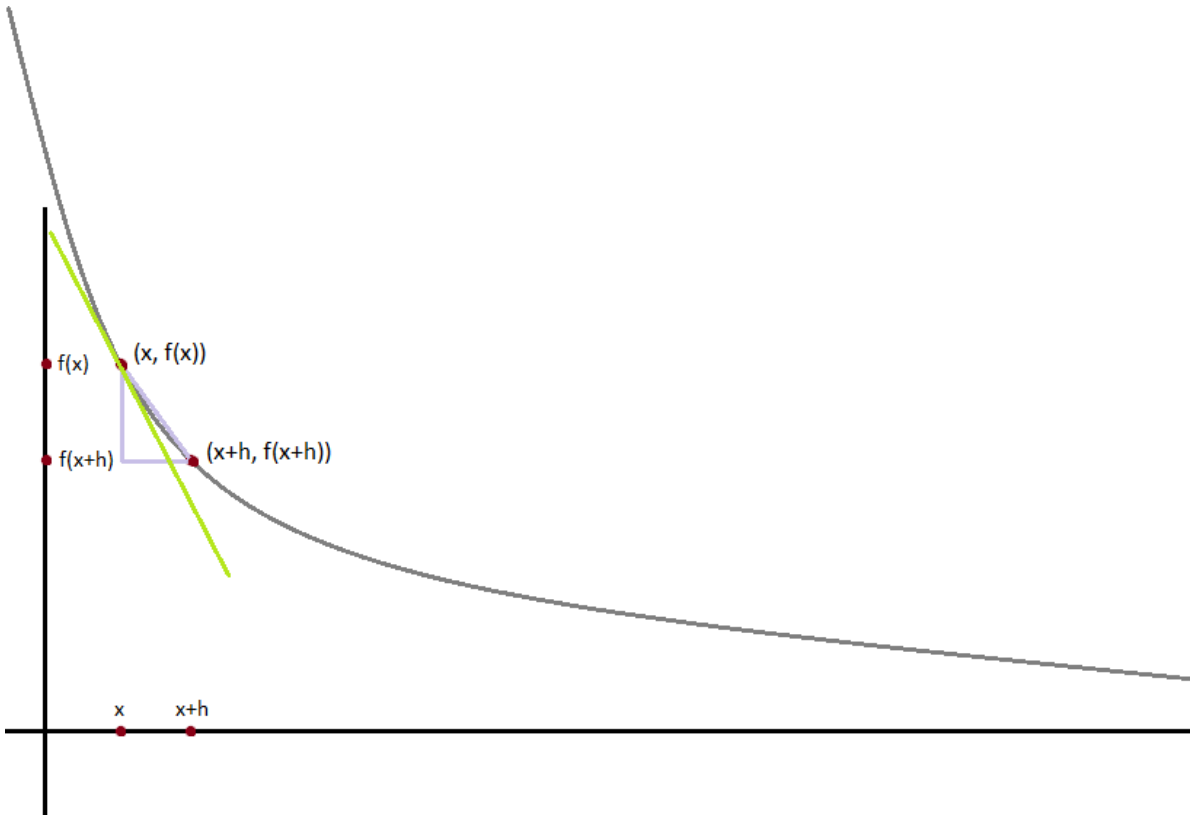
```
from typing import Callable

def difference_quotient(f: Callable[[float], float],
                        x: float,
                        h: float) -> float:
    return (f(x + h) - f(x)) / h
```

as h approaches zero.

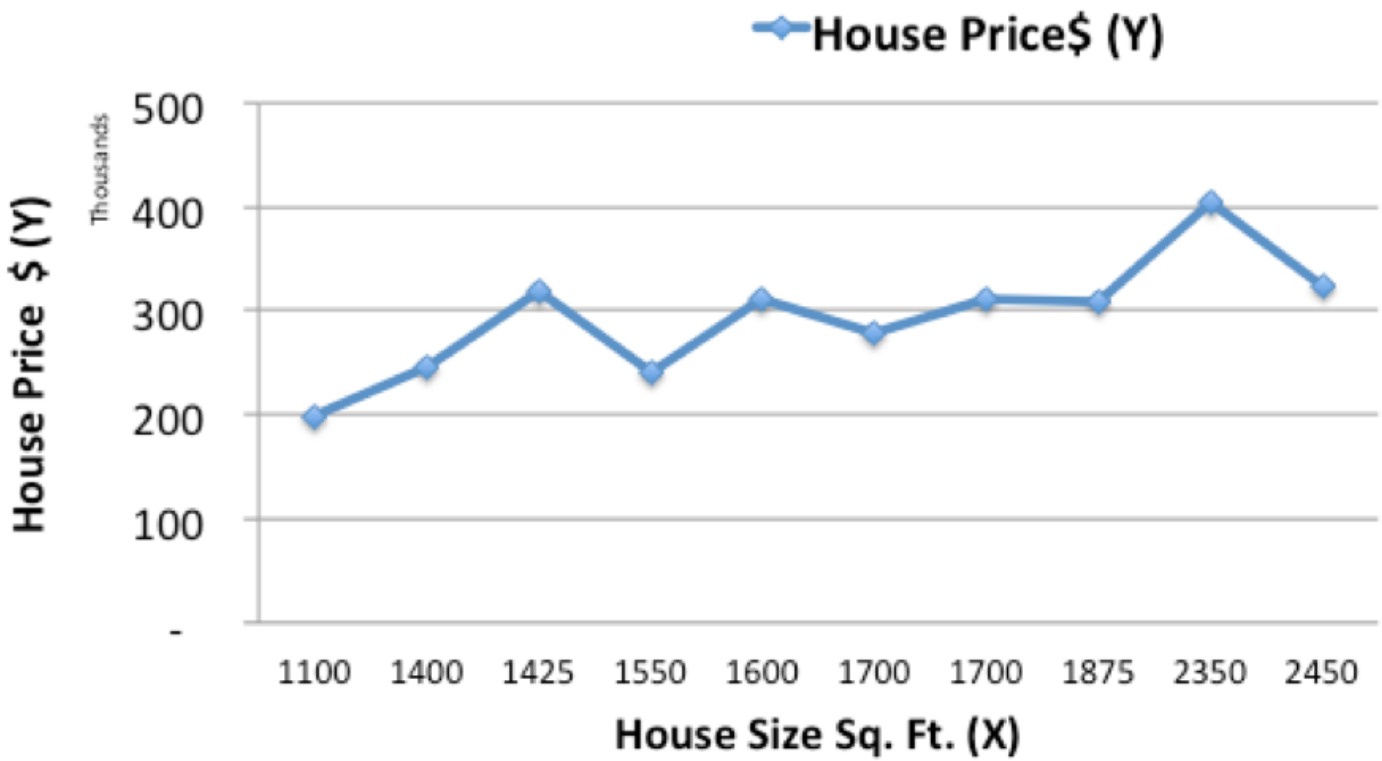
Forward School

In a more graphical way

- The derivative is the slope of the tangent line at (x,f(x)), while the difference quotient is the slope of the not-quite-tangent line that runs through (x+h,f(x+h)). As h gets smaller and smaller, the not-quite-tangent line gets closer and closer to the tangent line (Figure on the right).

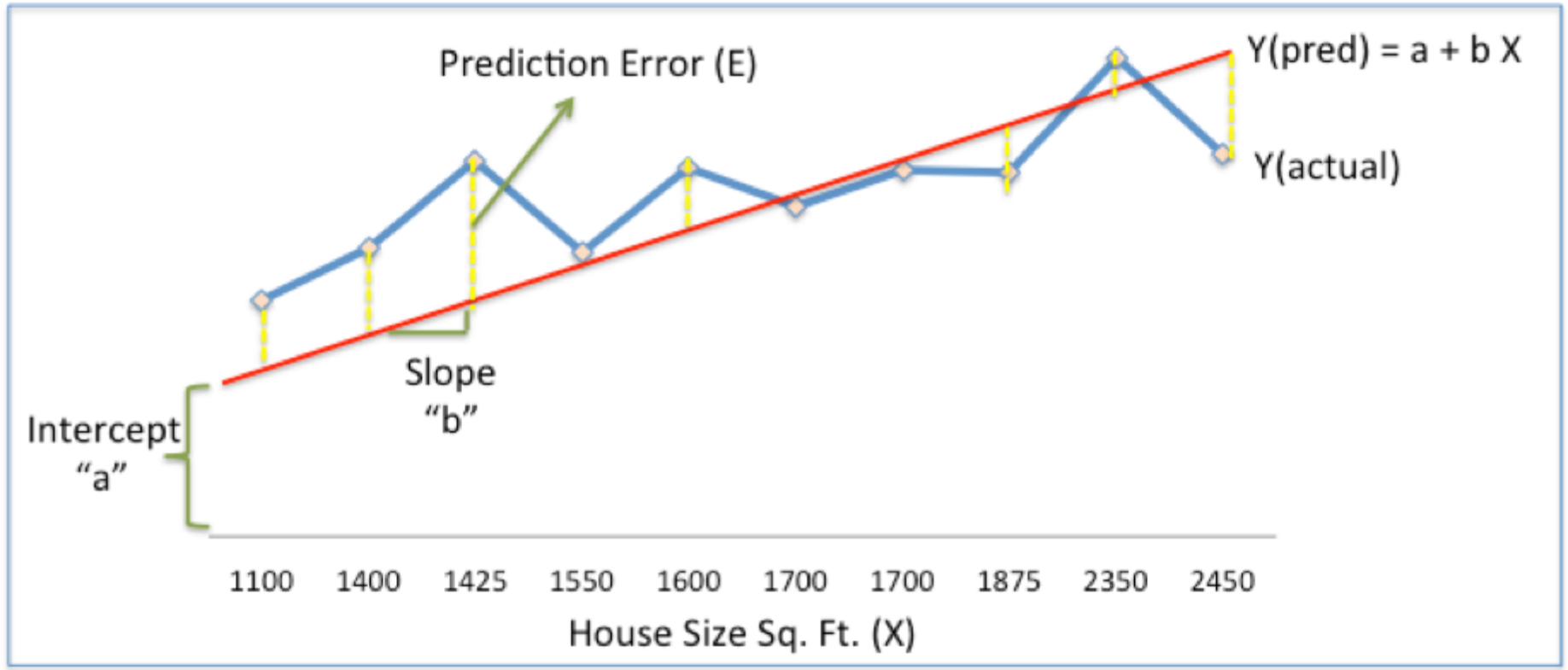


Forward School



https://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html

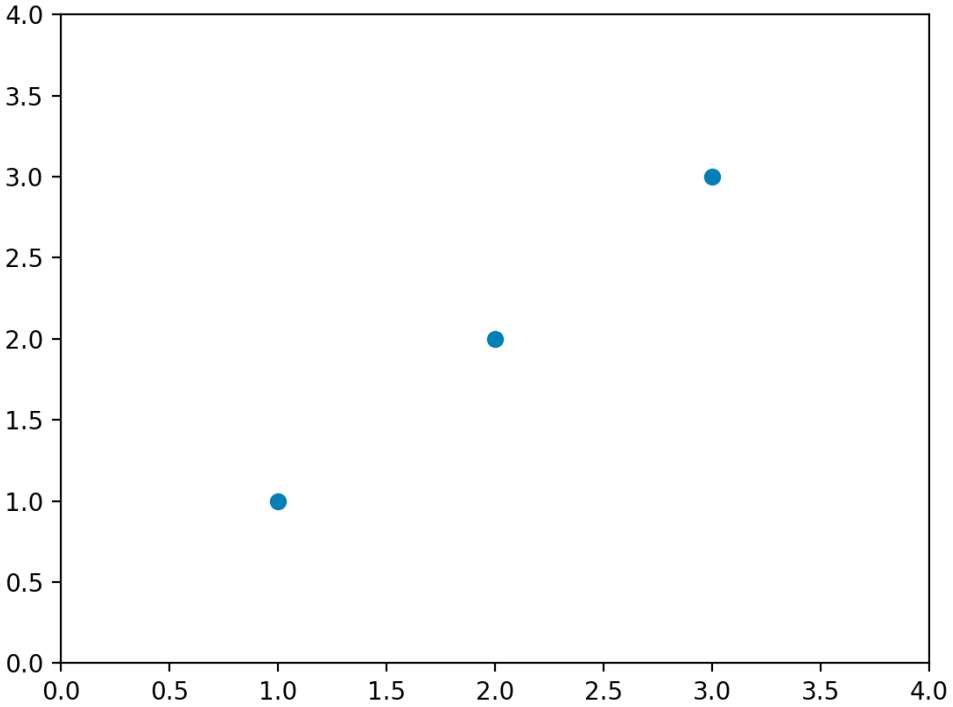
Forward School

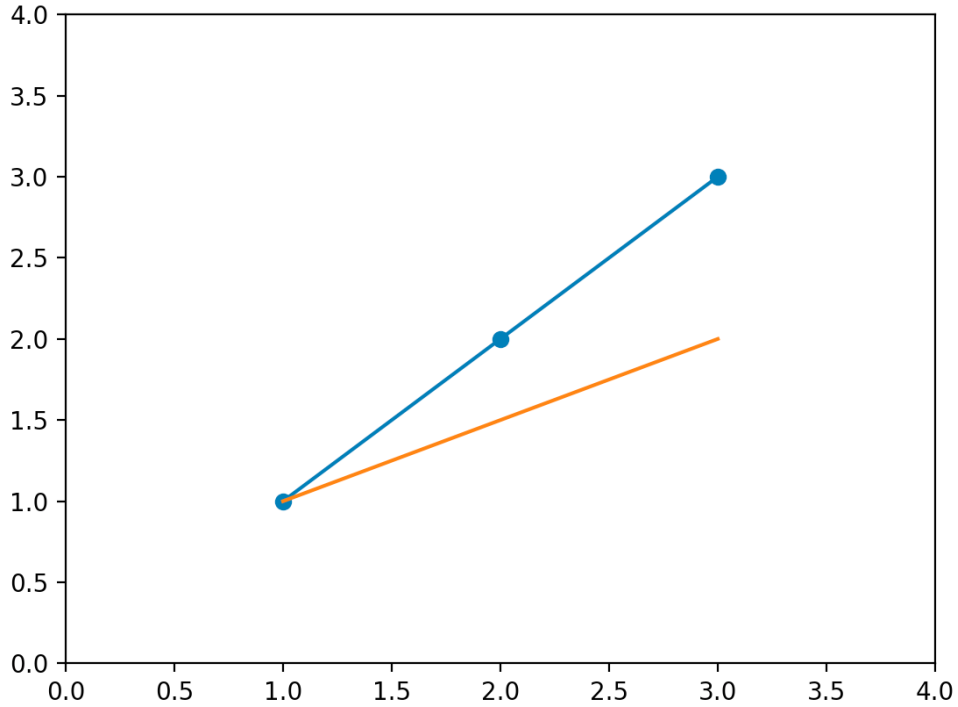


Forward School

Another (similar example)

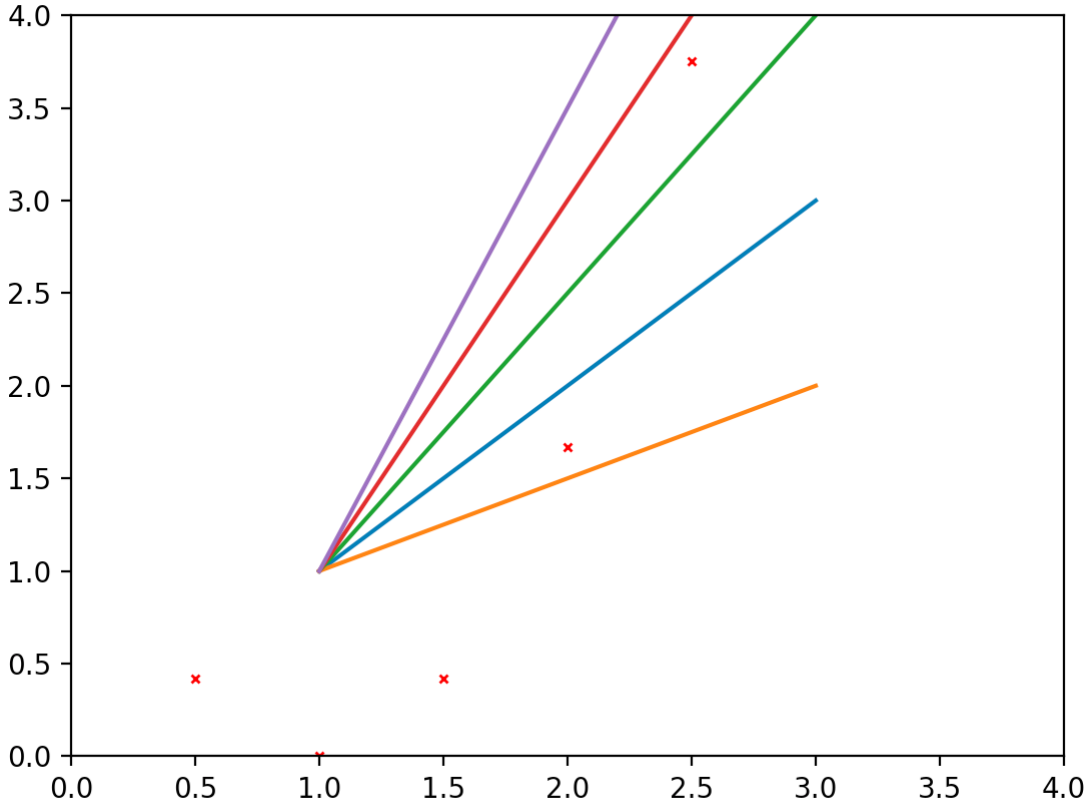
- https://medium.com/@lachlanmiller_52885/machine-learning-week-1-cost-function-gradient-descent-and-univariate-linear-regression-8f5fe69815fd

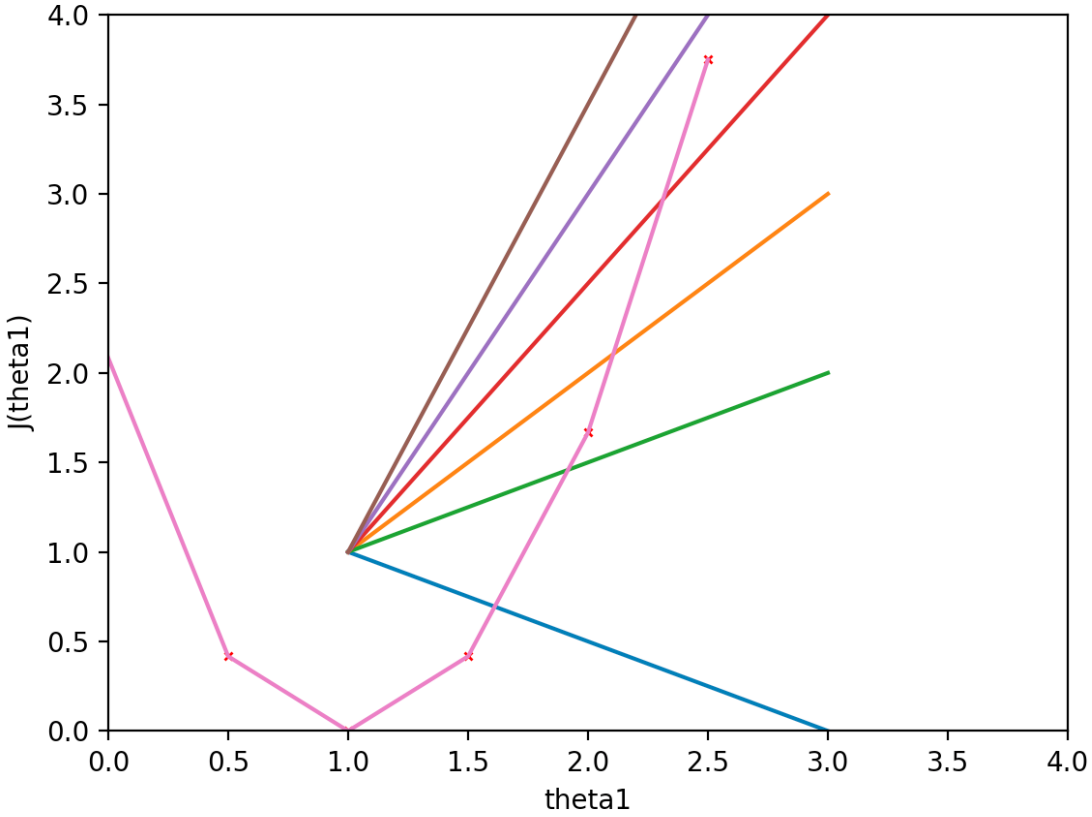




Forward School

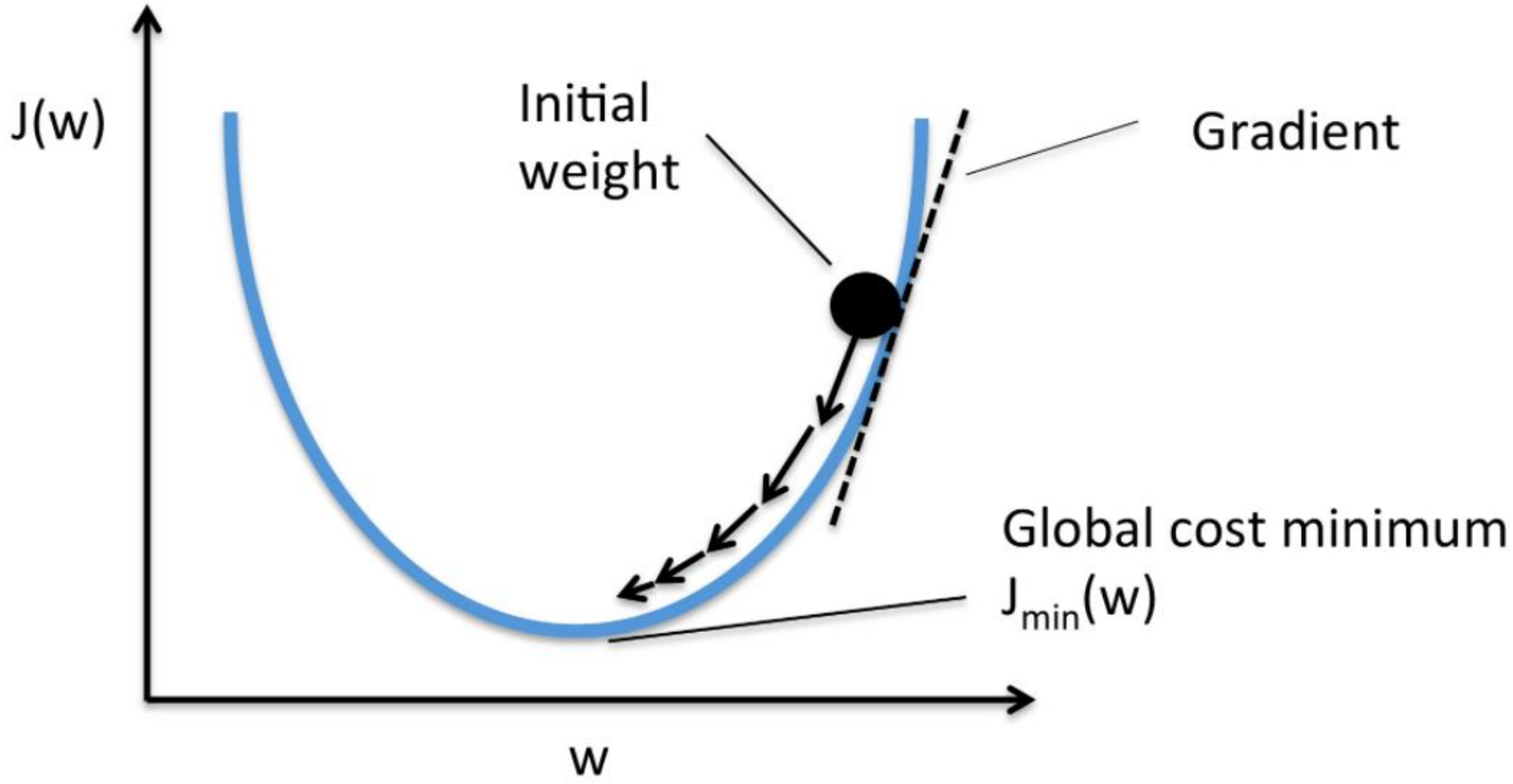
If you continue with more variations….





Forward School

Finding the local/global minimum is the key….



Forward School

To summarize

1. Find the gradient of the Loss Function – take the derivative of the Loss Function for each parameter in it
2. Pick random values for the parameters
3. Get the Gradient (put the random values from Step 2 into the derivatives)
4. Calculate the Step Sizes
5. Calculate the new parameters
6. Repeat Step 3 until Step size is very small (or when you reach the maximum numbers of steps)

Forward School

Minibatch and Stochastic Gradient Descent

- Gradient descent works well, when data is large but not too large
  - Because the number of steps until finding the global minimum will take longer
- Two techniques may help to speed things up
  - Minibatch Gradient Descent
    - To compute the gradients based on mini batches sampled from larger dataset
  - Stochastic Gradient Descent
    - Take gradient step based on one training example at a time

Based on https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

Forward School

Mini Batch Gradient Descent

- Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients.

- Implementations may choose to sum the gradient over the mini-batch which further reduces the variance of the gradient.

- Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

Forward School

Mini Batch Gradient Descent

- Upsides
  - The model update frequency is higher than batch gradient descent which allows for a more robust convergence, avoiding local minima.
  - The batched updates provide a computationally more efficient process than stochastic gradient descent.
  - The batching allows both the efficiency of not having all training data in memory and algorithm implementations.

Forward School

Mini Batch Gradient Descent

- Downsides
  - Mini-batch requires the configuration of an additional “mini-batch size” hyperparameter for the learning algorithm.
  - Error information must be accumulated across mini-batches of training examples like batch gradient descent.

Forward School

Stochastic Gradient Descent

- Stochastic gradient descent, often abbreviated SGD, is a variation of the gradient descent algorithm that calculates the error and updates the model for each example in the training dataset.

- The update of the model for each training example means that stochastic gradient descent is often called an online machine learning algorithm.

Forward School

Stochastic Gradient Descent

- Upsides
  - The frequent updates immediately give an insight into the performance of the model and the rate of improvement.
  - This variant of gradient descent may be the simplest to understand and implement, especially for beginners.
  - The increased model update frequency can result in faster learning on some problems.
  - The noisy update process can allow the model to avoid local minima (e.g. premature convergence).

Forward School

Stochastic Gradient Descent

- Downsides
  - Updating the model so frequently is more computationally expensive than other configurations of gradient descent, taking significantly longer to train models on large datasets.
  - The frequent updates can result in a noisy gradient signal, which may cause the model parameters and in turn the model error to jump around (have a higher variance over training epochs).
  - The noisy learning process down the error gradient can also make it hard for the algorithm to settle on an error minimum for the model.