Intro to Gradient Descent

Introduction

- We have a loss function associated with how well our function does
  - One such example is MSE: $\mathrm{J}(x) = \sum_{i=1}^{m}(f(x_i) - y_i)^2$
- How do we minimize the loss?
  - Before, we were able to numerically solve.
  - Can you do this with everything?
  - How can we reliably minimize our loss function?

Introduction

- Observation:
  - If our function is differentiable, we can find the direction that decreases our function the fastest at any given point
  - If we do this over and over again, we can approach the global minimum

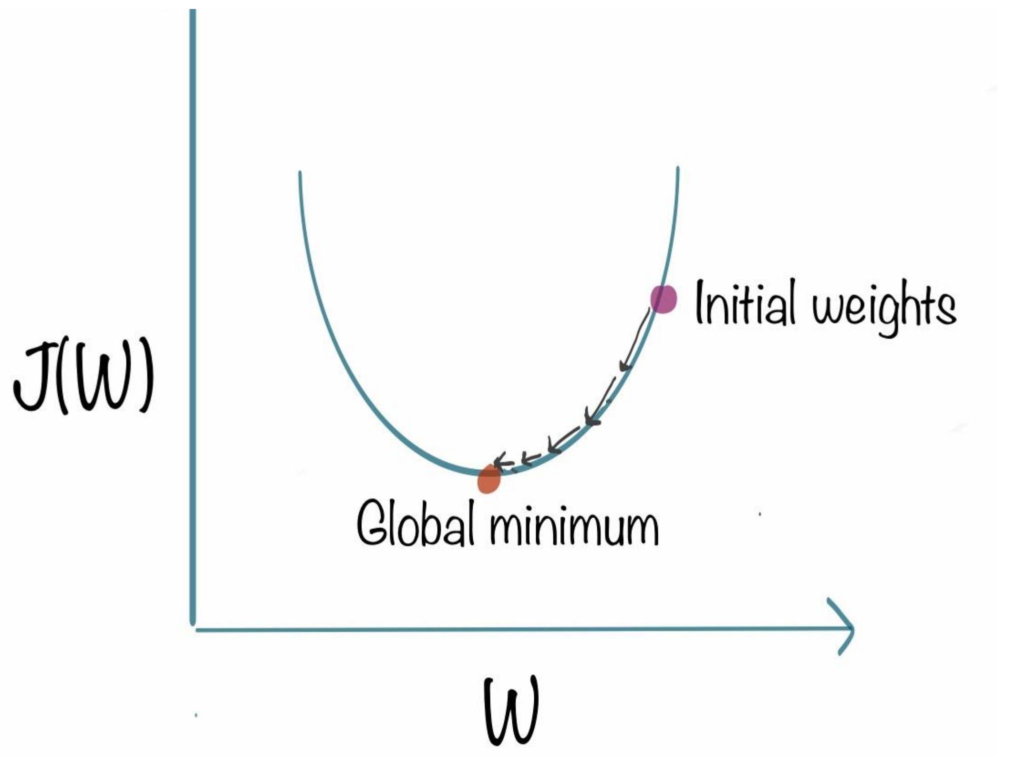
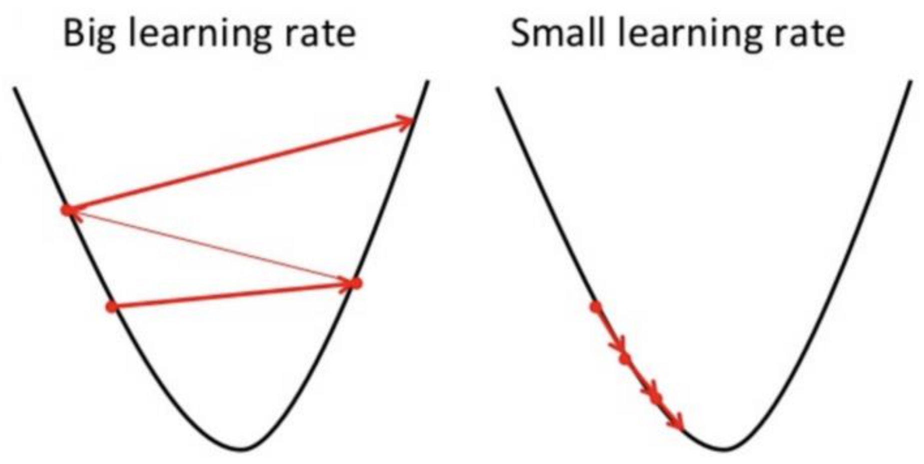


Image credit to KDNuggets

Gradient Descent Rule

$$\vec{w}_{t+1} = \vec{w}_t - \alpha \nabla J$$

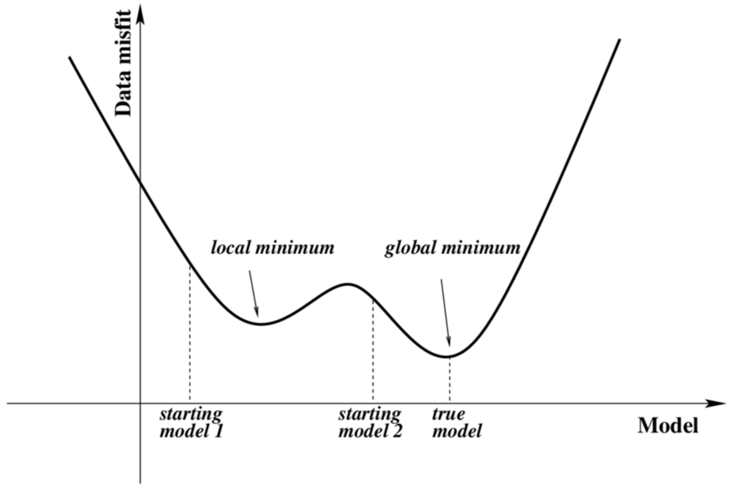
- *w* is our parameters. In a linear classifier, it is the coefficients.
- With each iteration, *J* decreases.
- α is called the **learning rate,** which dictates how much the algorithm moves at each iteration



Credit to Towards Data Science for the image

Problems

- Doesn't this only find the local minimum? Won't you get stuck?
  - Yes and yes. This is why people like to use convex loss functions, so the local minimum is the global minimum
  - There are also saddle points, where the gradient is 0, but the point is not a local minimum
    - E.g: $y = {x_1}^2 - {x_2}^2$
  - Techniques to escape from these are outside the scope of the course



Variations

- Batch gradient descent
- Stochastic gradient descent

Stochastic gradient descent

- Classical gradient descent works by generating a gradient for each data point in the dataset, then applying the gradients all at once for each iteration through the dataset
- Stochastic gradient descent instead works by applying the gradient after each individual point, so that the calculation of the loss function uses the updated parameters for the next data point
- In practice, most people use stochastic gradient descent, or a variation called mini-batching, as it seems to converge faster.

Momentum

- See slide 48 of Cottrell slides