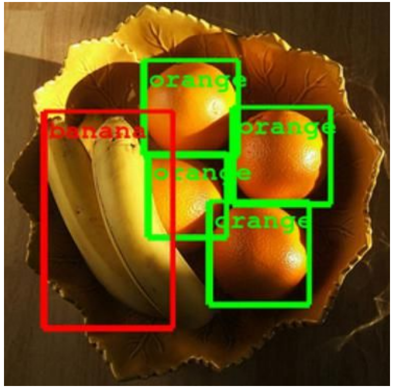
Stochastic Gradient Descent

Victor Zhou, Thant Zin Oo (Andy)

Machine Learning

- Machine learning, neural networks, and AI
- Real life applications:
    - Image recognition
    - Self driving cars
    - Financial services



Optimization

Gradient descent is the most commonly used optimization method for machine learning and neural networks.

Optimization

Score Function - maps data to scores that describe fit

- Percent match, predicted category, etc.

Loss Function - measures quality of parameters

- Compares learning data with training data

Optimization

The goal of optimization is to find a set of parameters to **minimize** the loss function.

Gradient Descent

Blindfolded Hiker Analogy - get to the bottom of a hill

Gradient Descent

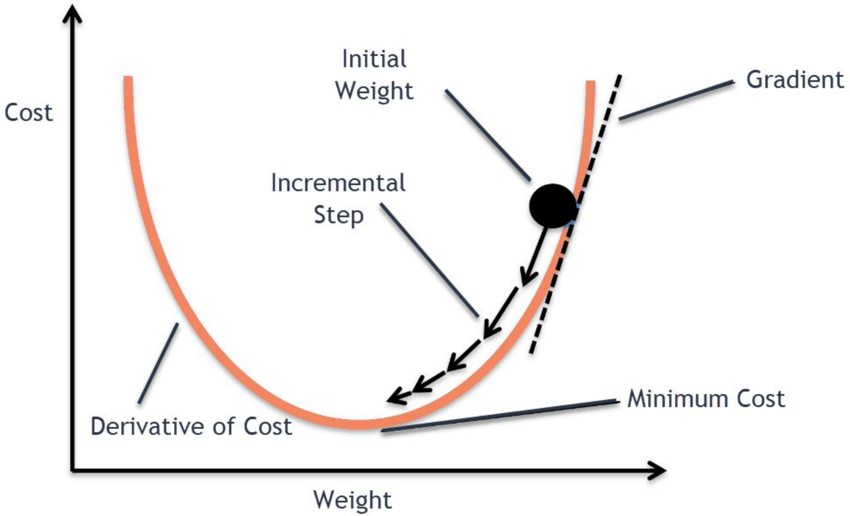
The partial derivative is the slope of a function parallel to a particular axis.

The gradient is a **vector** that points in the direction of greatest change for a function.

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k},$$

Gradient Descent

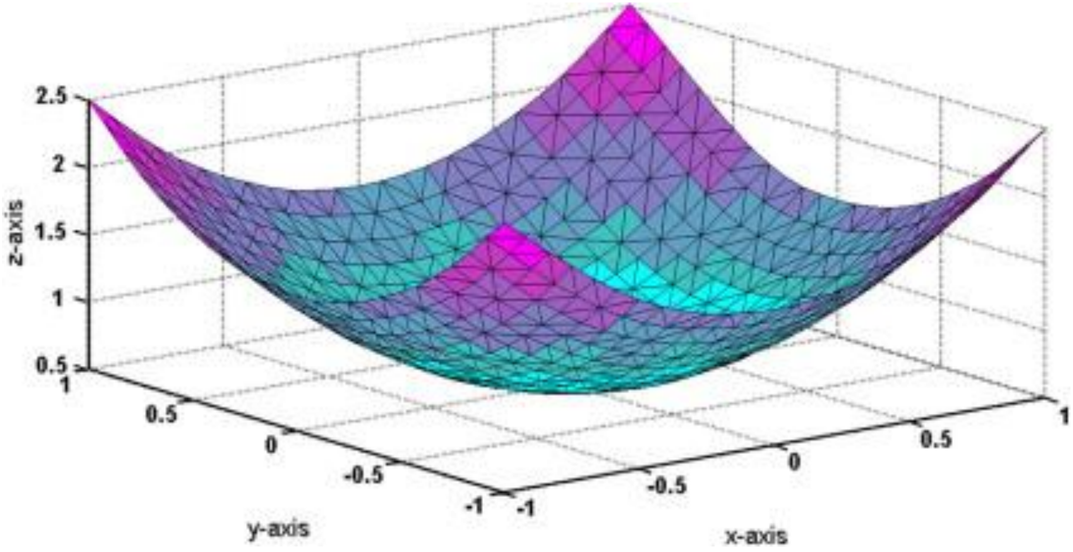
We should take **bigger** steps if we are far from the solution and **smaller** steps if we are close to the optimal solution.



Convex surfaces

Def: the line segment between any two points lies either above or on the graph

Subtract alpha value because gradient points in direction of steepest ascent



Learning Rate

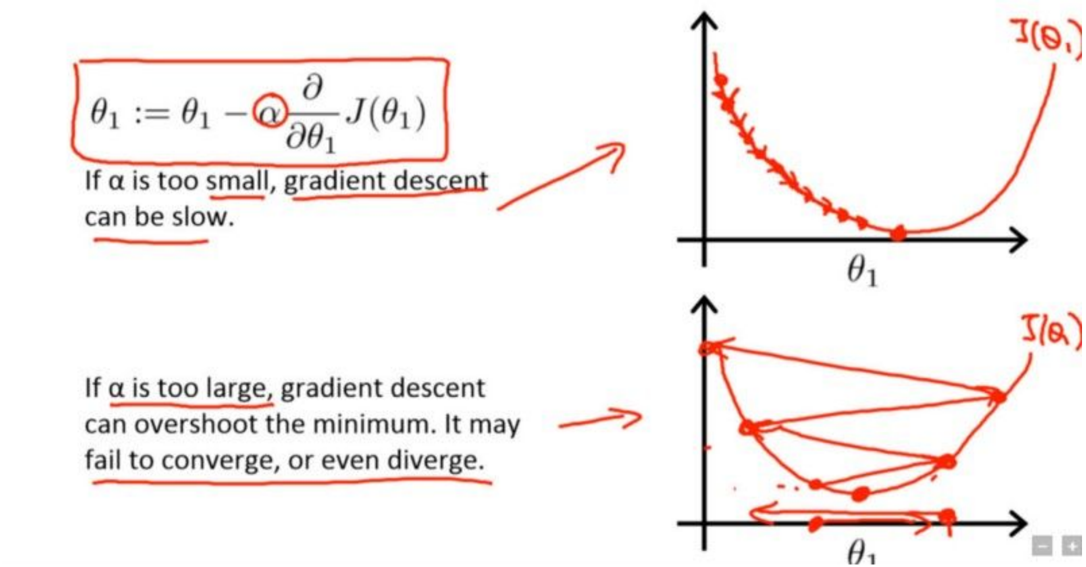
The step size **varies** based on the derivative of the loss function.

Step size = Slope * **Learning rate**

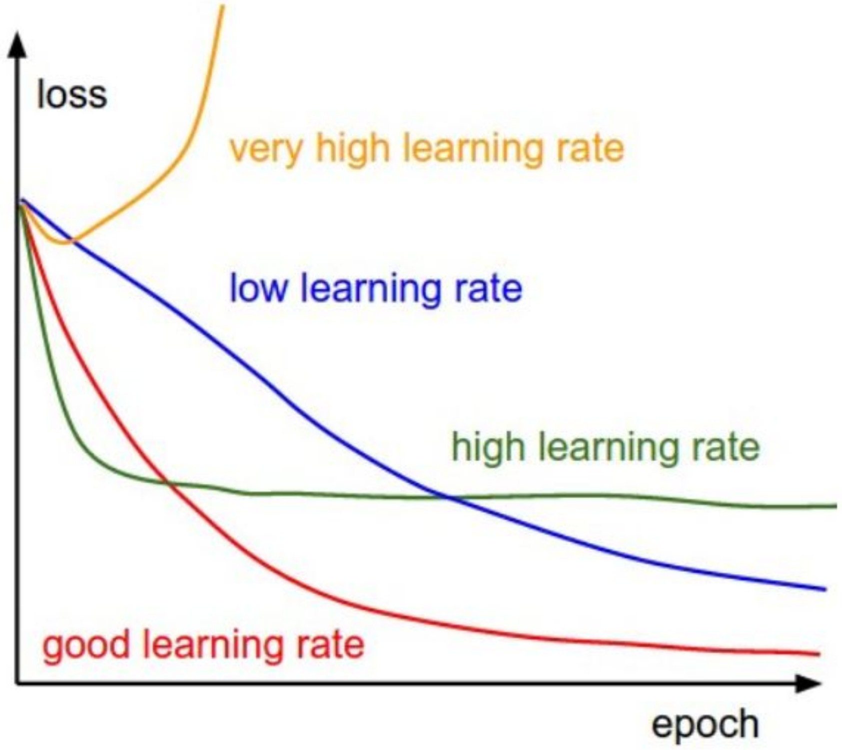
Learning rate sensitivity

Too **small** a value: slow convergence, computationally expensive

Too **large** a value: can overshoot, premature convergence



Learning rate sensitivity (visualization)



Example

Linear Regression:

Score Function: $r(x) = mx+b$

Loss Function: $\Sigma(f(x) - r(x))^2$

We want to minimize the loss function to get the line of best fit.

Example

Linear Regression:

Let us do a simple example with the slope fixed at 1, optimizing the *intercept*.

We will do a small sample with 3 data points at **(1,2)**, **(4,3)**, and **(5,6)**.

Example

Loss Function:

$\Sigma(f(x) - r(x))^2 = \Sigma(y - (x + b))^2$

$= (2-(1+b))^2 + (3-(4+b))^2 + (6-(5+b))^2$

$= 3b^2 + 2b + 3$

Example

Loss Function: $3b^2 + 2b + 3$

$r'(x) = 6b + 2$

Sample Learning Rate: 0.1

Step size = slope * learning rate

New intercept = Old intercept - Step size

Example

Derivative: $r'(x) = 6b - 3$

Starting at 0:

$0 - (-3 * 0.1) = 0.3$

$0.3 - (-1.2 * 0.1) = 0.42$

$0.42 - (-0.48 * 0.1) = 0.468$

$0 \rightarrow 0.3 \rightarrow 0.42 \rightarrow 0.468 \rightarrow 0.487 \rightarrow 0.495 \rightarrow 0.498 \rightarrow 0.499$

Multiple Variables

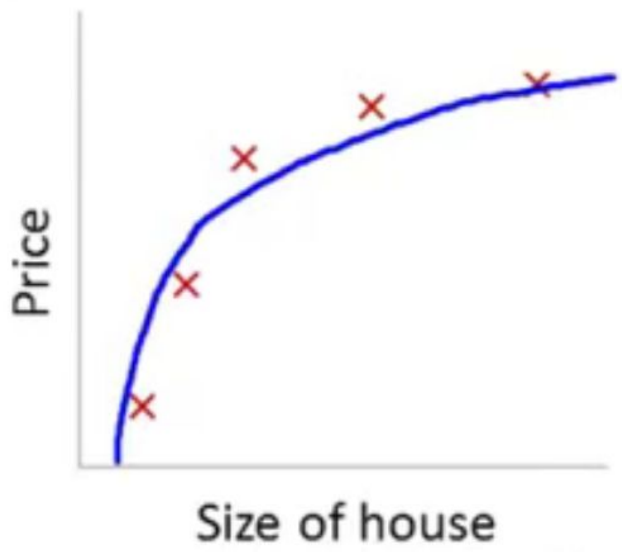
Recall a gradient is calculated from multiple partial derivatives.

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k},$$

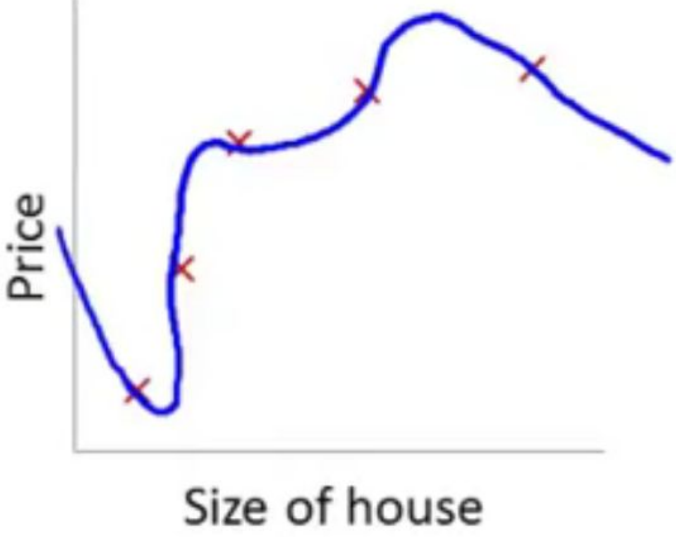
We can optimize functions with multiple parameters simultaneously by taking partial derivative with respect to each variable.

Regularization

Reduce overfitting of data points



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

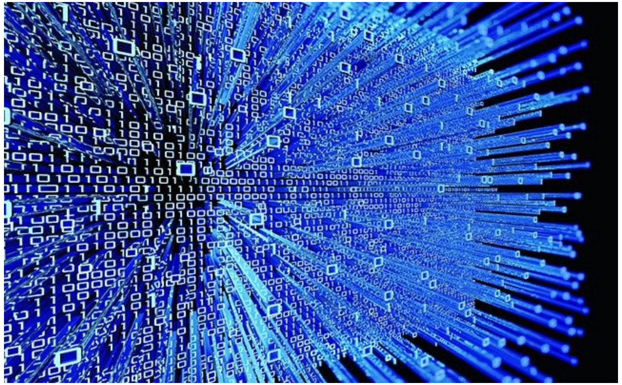


$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Drawbacks

In real life, data sets can be **very large**.

- Stocks
- Games
- Pictures



Drawbacks

The computation time required grows astronomically.

Example:

100 variables * 100,000 data points * 1,000 steps =

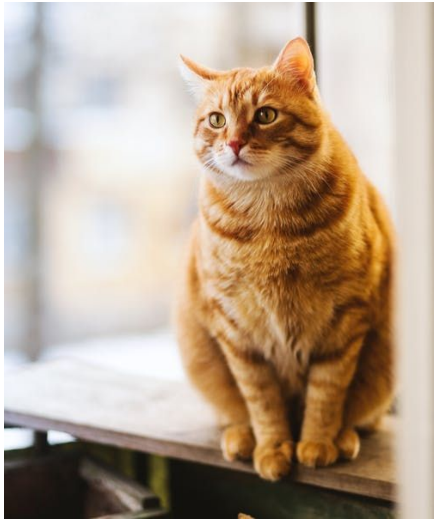
10 billion calculations

Reducing Computation

In real life, many data points are similar to each other.

Small sets of data represent the whole.





Adding Randomness

Mini-batch gradient descent - randomly choose small subset of data to test at each step.

- Much faster parameter updates
- Still very accurate

Stochastic Gradient Descent

sto·chas·tic

/stəˈkastik/

adjective TECHNICAL

randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

Stochastic gradient descent is when only 1 data point is used.

In practice, usually small batches are done for efficiency.

Batch / Stochastic / Mini-batch

Batch: iterate over all m training samples before updating loss function

Stochastic: update loss based on *1* randomly selected sample

Mini-batch: iterate over b training samples

Batch vs Stochastic

Batch

Pros:

- Accuracy
- Vectorization optimizations

Cons:

- Requires larger memory space to fit entire *m* set of data samples
- Infrequent loss updates

Stochastic

Pros:

- Better with memory limitations, or when data is through input stream
- Faster computationally (most of the time)
- updates loss more frequently

Cons:

- Individual variances in gradient
- Iterative process can be bottleneck during parallelization

Works Cited / Further Reading

- "CS231n Convolutional Neural Networks for Visual Recognition." http://cs231n.github.io/optimization-1/.
- Bottou, Léon. "Stochastic Gradient Descent Tricks." In *Neural Networks: Tricks of the Trade*, edited by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, 7700:421–36. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. https://doi.org/10.1007/978-3-642-35289-8_25.
- Cui, Xiaodong, Wei Zhang, Zoltán Tüske, and Michael Picheny. "Evolutionary Stochastic Gradient Descent for Optimization of Deep Neural Networks," n.d., 11. http://papers.nips.cc/paper/7844-evolutionary-stochastic-gradient-descent-for-optimization-of-deep-neural-networks.pdf
- "Optimization - Batch Gradient Descent versus Stochastic Gradient Descent." Accessed December 1, 2019. https://stats.stackexchange.com/questions/49528/batch-gradient-descent-versus-stochastic-gradient-descent.