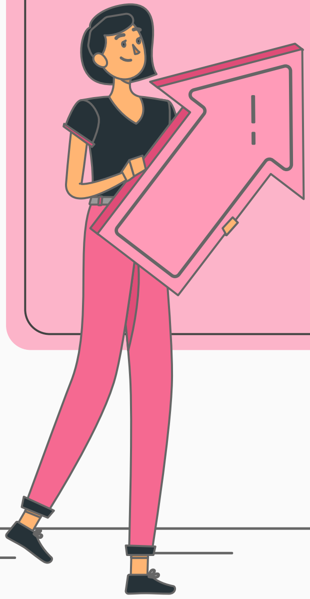
OPTIMIZATION



SureStart

MODEL OPTIMIZATION

Our goal is to minimize the loss and improve the model's performance. In order to decrease the loss value, we need *to adjust (increase or decrease) the weight and other parameters*, such as the learning rate. Algorithms or methods used to **change these attributes** of neural network in order to minimize the loss are optimizers.

Succinctly, optimizers influence model weights to find most accurate possible form by adjusting the weights.

Various optimizers to minimize the loss function:

- Gradient Descent
  - Batch Gradient Descent
  - Stochastic Gradient Descent (SGD)
  - Mini Batch Stochastic Gradient Descent (MB-SGD)
- SGD with momentum
- Nesterov Accelerated Gradient (NAG)
- Adaptive Optimization
  - Adaptive Gradient (AdaGrad)
  - AdaDelta
  - RMSprop
  - Adam

SureStart

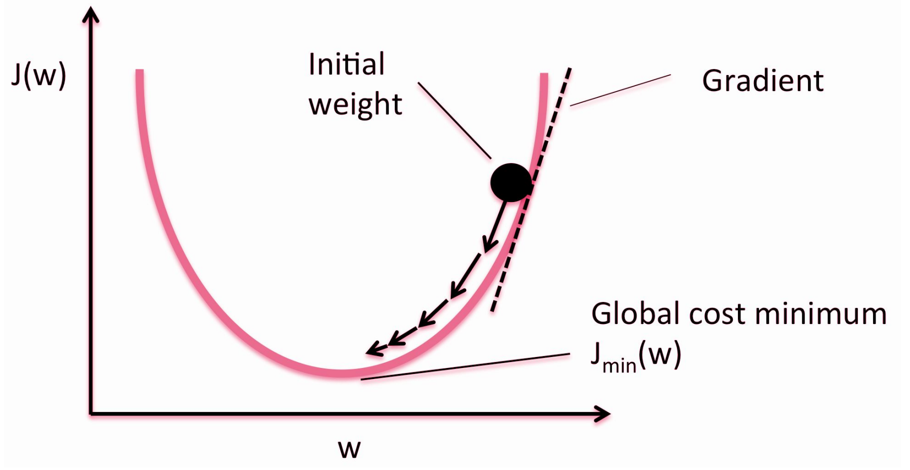
GRADIENT DESCENT

The main purpose of gradient descent is to minimize the cost function.

Gradient descent is the most basic but most used optimization algorithm. It's used heavily in linear regression and classification algorithms, also in the backpropagation in neural networks.

What is Gradient Descent?

- Gradient Descent is an iterative process that finds the parameters or coefficients of a function where the function has a minimum value.
- The algorithm seeks to change the weights so that the next evaluation reduces the error (navigating down the gradient/ slope of error) using backpropagation.



SureStart

GRADIENT DESCENT

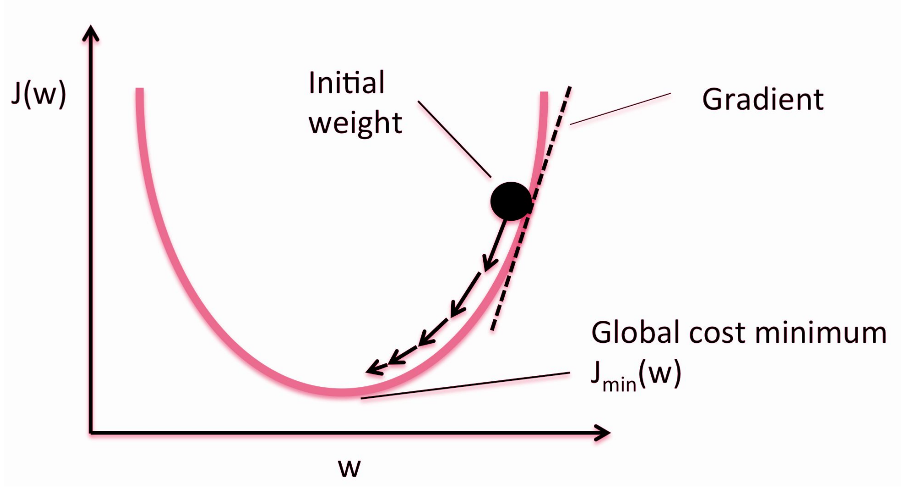
The main purpose of gradient descent is to minimize the cost function.

Gradient descent is the most basic but most used optimization algorithm. It's used heavily in linear regression and classification algorithms, also in the backpropagation in neural networks.

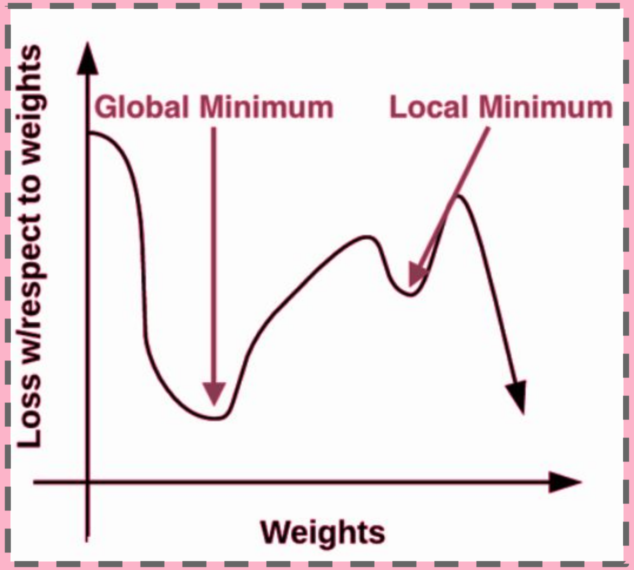
The steps are as follows:

1. Calculate what a small change in each individual weight would do to the loss function.
2. Adjust each individual weight based on its gradient.
3. Keep doing steps #1 and #2 until the loss function gets as low as possible.

- And the sequence of steps will stop once we hit convergence (when the cost function is at the minima).



SureStart



In an ideal world, the goal is to find global minimum by ensuring the most optimal parameter values.

LOCAL MINIMA AND GLOBAL MINIMA

As we can see, a loss landscape has many peaks and valleys based on which values our parameters take on. Each peak is a local maximum that represents very high regions of loss.

The local maximum with the largest loss across the entire loss landscape is the **global maximum**.

Similarly, **local minimum** represents many small regions of loss.

SureStart

TYPES OF GRADIENT DESCENT

BATCH GRADIENT DESCENT

Parameters are updated after computing the gradient of error with respect to the entire training set.

It takes a lot of time for making a single update.

It makes smooth updates in the model parameters.

STOCHASTIC GRADIENT DESCENT

Parameters are updated after computing the gradient of error with respect to a single training example.

It makes very noisy updates in the parameters.

MINI-BATCH GRADIENT DESCENT

Parameters are updated after computing the gradient of error with respect to a subset of the training set.

It can make quick updates in the model parameters.

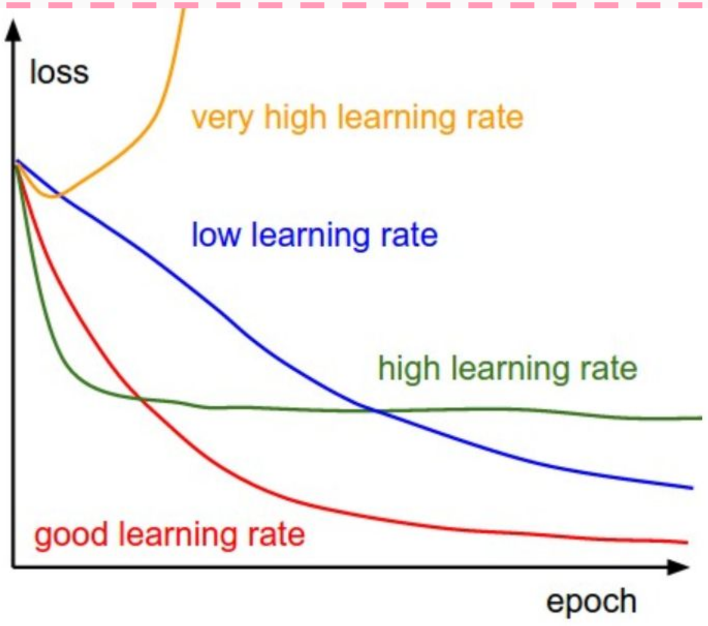
Depending upon the batch size, the updates can be made less noisy (greater the batch size, the update is less noisy).

If you want to use gradient descent algorithm for CNN, then mini-batch gradient descent is the best option.

SureStart

LEARNING RATE

- Stochastic gradient descent algorithm estimates the error gradient for the current state of the model and then updates the weights of the model using backpropagation. The amount that the weights are updated during training is referred to as the step size or the "learning rate."
- The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.
- The learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.
- The learning rate influences the optimizer's convergence.



Effect of various learning rates on convergence (Img Credit: cs231n)

SureStart

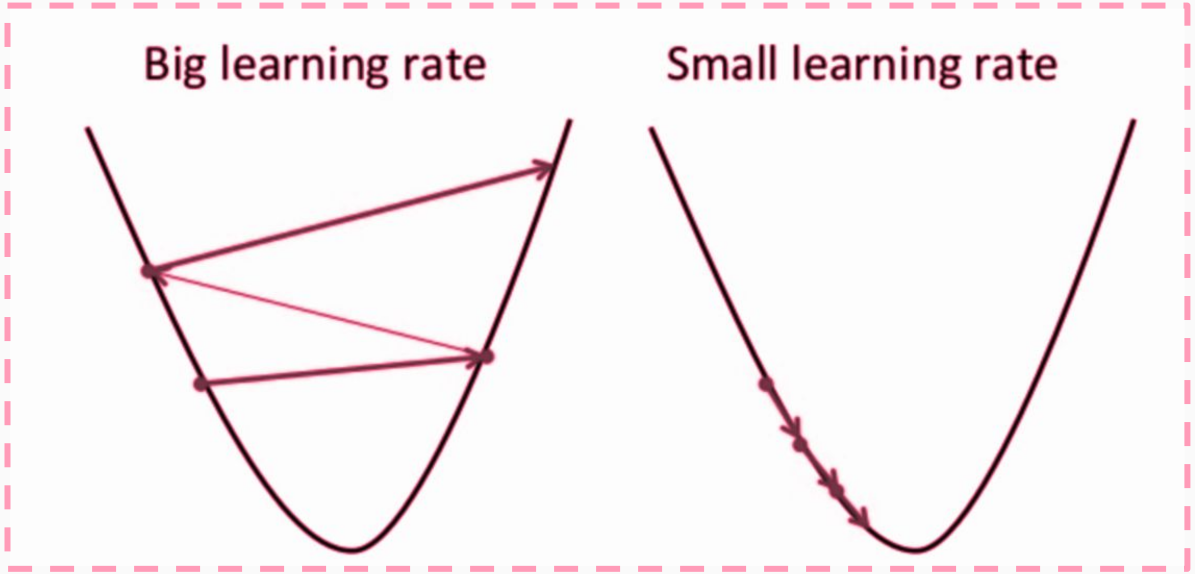
LEARNING RATE

The learning rate influences the optimizer's convergence. For gradient descent to reach the local minimum, the learning rate must set to an appropriate value, which is neither too low nor too high.

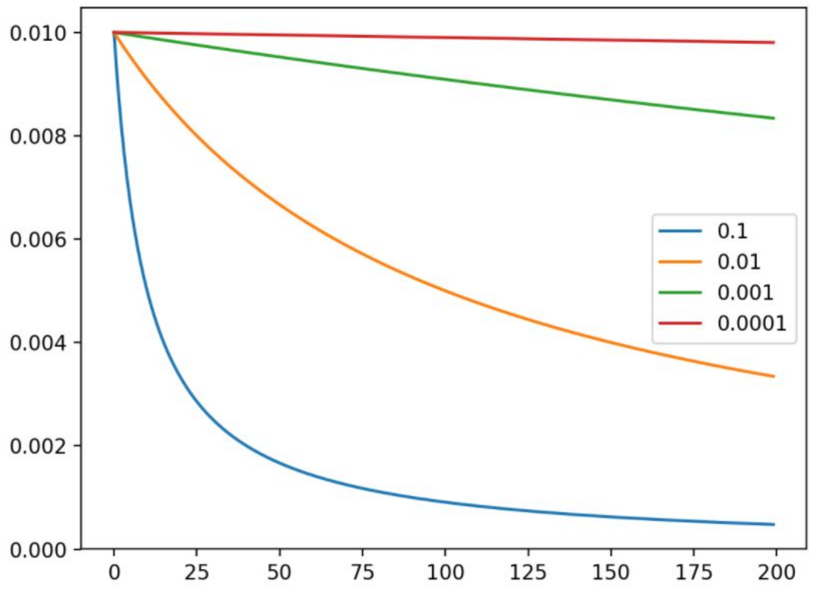
- If the learning rate is too small, updates are small, and optimization is slow. Also, the model is likely to get stuck into a local minimum or plateau.

- If the learning rate is too large, updates will be large, and the optimization is likely to diverge.

- If the learning rate is chosen well, updates are appropriate, and the optimization should converge to a good set of parameters.



LEARNING RATE



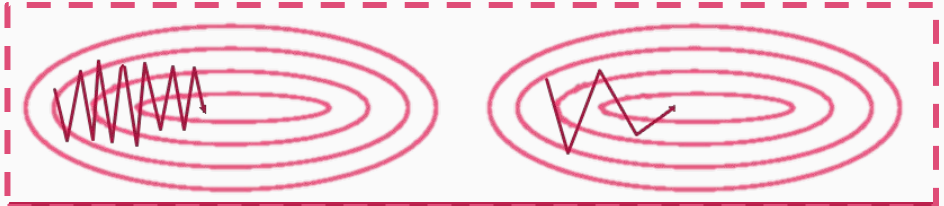
Line Plot of the Effect of Decay on Learning Rate Over Multiple Weight Updates

SureStart

BACK TO MODEL OPTIMIZATION

Momentum helps to accelerate the gradient descent (GD) and move faster towards convergence.

The weight update of mini-batch gradient descent is very noisy. SGD with momentum helps to reduce the fluctuation of gradients.



(Left) Vanilla SGD, (right) SGD with momentum. Goodfellow et al. (2016)

Momentum symbolized by ‘γ’ which is usually set to 0.9 or a similar value.

ADAPTIVE OPTIMIZATION

In optimization algorithms, the parameters like learning rate and momentum coefficient **will not stay constant** throughout the training process. Instead, **these values will constantly adapt** for each and every weight in the network and hence will also change along with the weights.

Some adaptive optimization algorithms are - *Adaptive gradient (AdaGrad), AdaDelta, RMSprop, Adam*.

Adam optimizer is one of the most popular gradient descent optimization algorithms.

Adam is computationally efficient and has very little memory requirement.

Adam implements the exponential moving average of the gradients to scale the learning rate. It keeps an exponentially decaying average of past gradients.

SureStart