Machine Learning

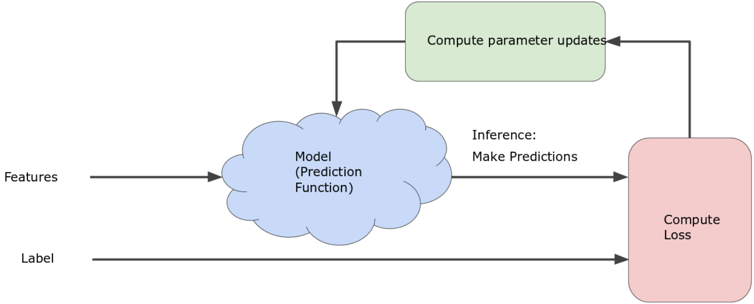
Pawel Wocjan

University of Central Florida

Fall 2020

An Iterative Approach

- The figure below depicts the iterative trial-and-error process that machine learning algorithms use to train a model:



Gradient Descent

- The iterative approach diagram contains a green box "compute parameter updates."
- Let us now discuss the gradient descent algorithm that is used to update the parameters.

Gradient Descent

- We consider the linear regression model

$$\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n b$$

- The loss function (mean squared error)

$$\mathcal{L} : \mathbb{R}^{n+1} \to \mathbb{R}$$

  depends on the parameters $n+1$ parameters $w_0, w_1, \ldots, w_n$, where w_0 is the bias term b.

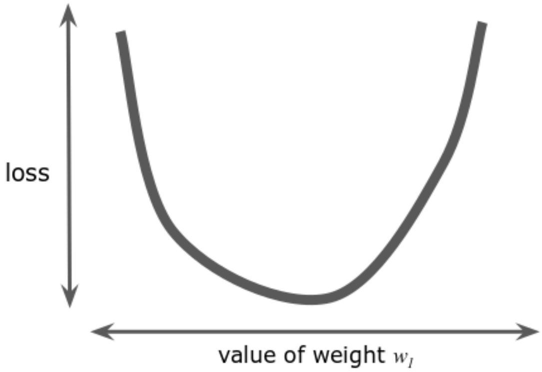
- Moreover, $\mathcal{L}$ defines a curve instead of a surface, which simplifies the presentation.

Gradient Decent

- For $n = 1$, the loss function $\mathcal{L}$ depends on two parameters – the bias term $b = w_0$ and the weight w_1 – and defines a surface.
- For $n > 1$, the loss function $\mathcal{L}$ defines a hypersurface and cannot be visualize so easily.
- To simplify the plots, we assume that $n = 1$ and the bias term $b = w_0$ is fixed to be 0. Then, the loss function $\mathcal{L}$ defines a curve.

Gradient Descent

- The resulting plot of the loss function $\mathcal{L}$ is be convex.
- Simply speaking, it means that it is bowl-shaped like this:

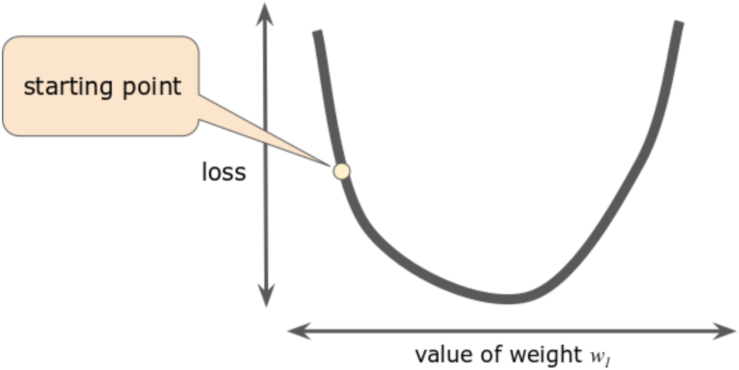


Gradient Descent

- It turns out that even in the general case the loss function $\mathcal{L}$ is convex.
- This is important because problems have only one minimum.
- Calculating the loss function for all parameter values $w_0, \ldots, w_n \in \mathbb{R}^{n+1}$ set would be an inefficient way of finding the minimum.
- Let's examine a better mechanism – very popular in machine learning – called **gradient descent**.

Gradient Descent

- The first stage in gradient descent is to pick a starting value.
- The starting point doesn't matter much; therefore, many algorithms simply set $w_i = 0$ or set the w_i to random values.



Gradient Descent

- The gradient descent algorithm then calculates the gradient of the loss function $\mathcal{L}$ at the starting point.
- The gradient

$$\nabla \mathcal{L} \in \mathbb{R}^{n+1}$$

is a vector whose entries

$$(\nabla \mathcal{L})_i$$

are given by the partial derivatives

$$\partial \mathcal{L} / \partial w_i$$

of the loss function $\mathcal{L}$ with respect to the weights w_i.

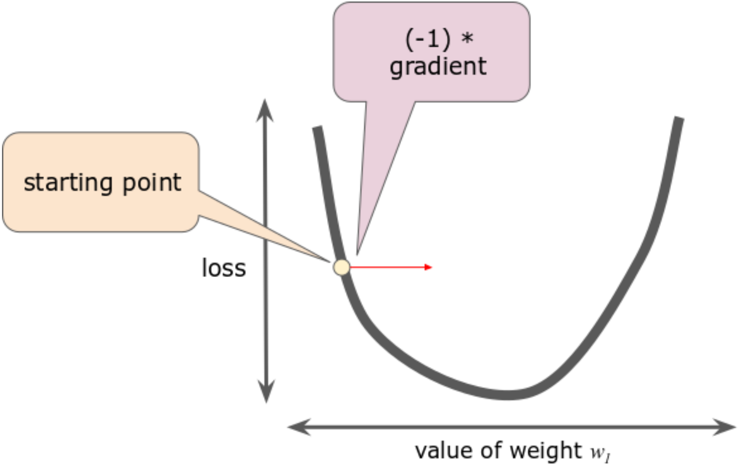
Gradient Descent

- The $\nabla\mathcal{L}$ gradient has both a direction and a magnitude.
- The gradient points which way is "warmer" or "colder."
- The gradient always points in the direction of steepest increase in the loss function.

  For the case $n = 1$ and the bias $w_0 = b$ is fixed to be 0, the gradient of the loss function $\mathcal{L}$ is simply the slope of the curve $\mathcal{L}$, that is, the derivative with respect to w_1.

Gradient Descent

- The gradient descent algorithm takes a step in the direction of the negative gradient $-\nabla\mathcal{L}$ to reduce the loss.

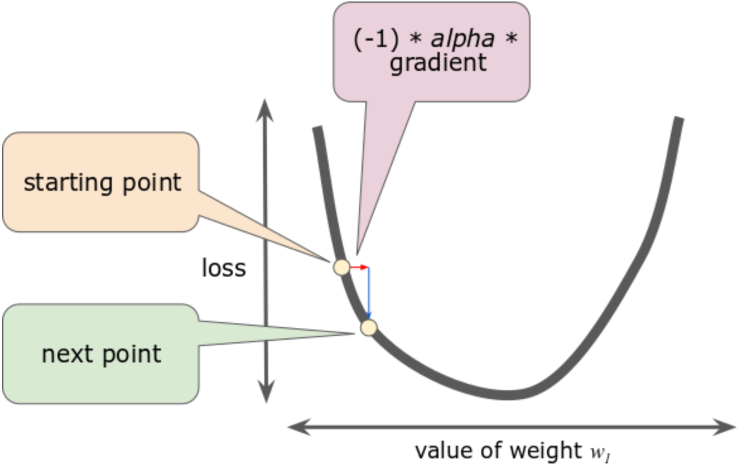


Gradient Descent

- More precisely, the gradient descent algorithm updates the starting point as follows:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla \mathcal{L}$$

where α is the learning rate.



Key Terms

- gradient
- gradient descent
- step

Learning Rate

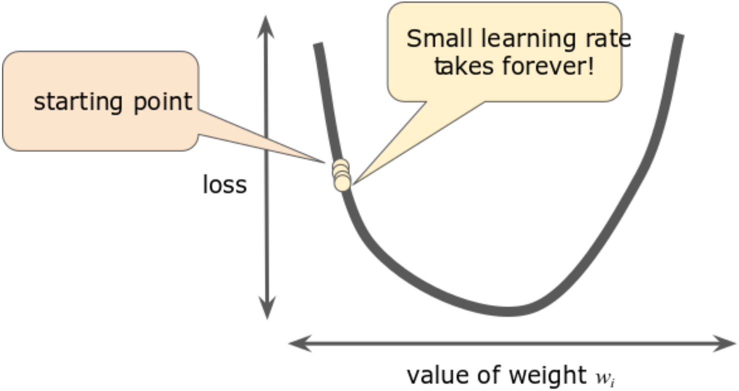
- The gradient vector has both a direction and a magnitude.
- The gradient descent algorithm multiplies the gradient by a scalar known as the learning rate (also sometimes called step size) to determine the next point.
- For example, if the gradient magnitude is 2.5 and the learning rate is 0.01, then the gradient descent algorithm will pick the next point 0.025 away from the previous point.

Learning Rate

- The learning rate is a so-called **hyperparameter**.
- A hyperparameter is a parameter that is external to the model.

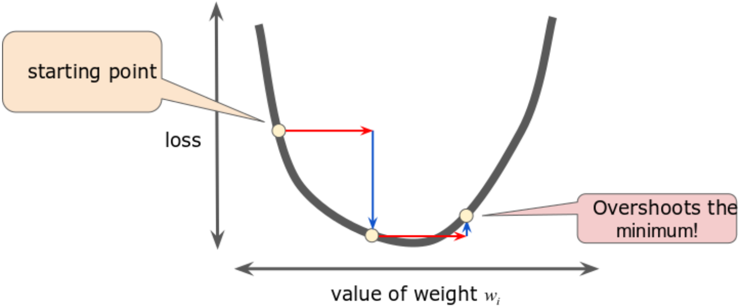
Learning Rate

- If the learning rate that is too small, learning will take too long:



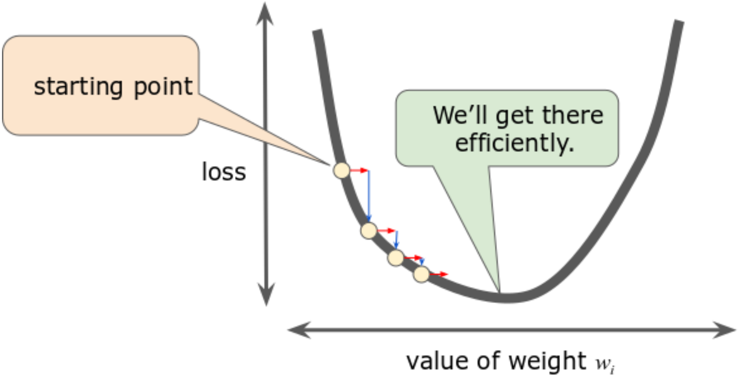
Learning Rate

- If the learning rate is too large, the next point will perpetually bounce haphazardly across the bottom of the well:



Learning Rate

- There's a Goldilocks learning rate for every linear regression problem.



Key Terms

- hyperparameter
- learning rate
- step size

Stochastic Gradient Descent

- In gradient descent, a **batch** is the total number of examples you use to calculate the gradient in a single iteration.
- So far, we've assumed that the batch has been the entire data set.
- But often data sets contain huge numbers of examples with huge numbers of features.
- Consequently, a batch can be enormous. A very large batch may cause even a single iteration to take a very long time to compute.
- A large data set with randomly sampled examples probably contains redundant data. In fact, redundancy becomes more likely as the batch size grows.
- Some redundancy can be useful to smooth out noisy gradients, but enormous batches tend not to carry much more predictive value than large batches.

Stochastic Gradient Descent

- What if we could get the right gradient on average for much less computation?
- By choosing examples at random from our data set, we could estimate (albeit, noisily) a big average from a much smaller one.
- **Stochastic gradient descent (SGD)** takes this idea to the extreme–it uses only a single example (a batch size of 1) per iteration.
- Given enough iterations, SGD works but is very noisy. The term "stochastic" indicates that the one example comprising each batch is chosen at random.

Reducing Loss

- **Mini-batch stochastic gradient descent (mini-batch SGD)** is a compromise between full-batch iteration and SGD. A mini-batch is typically between 10 and $1,000$ examples, chosen at random.
- Mini-batch SGD reduces the amount of noise in SGD but is still more efficient than full-batch.

Key Terms

- batch
- batch size
- mini-batch
- stochastic gradient descent (SGD)