Linear Regression

Linear regression is a statistical technique used to find the relationship between variables. In an ML context, linear regression finds the relationship between features and a label.

In ML, we write the equation for a linear regression model as follows:

$$y' = b + w_1 x_1$$

where:

- y' is the predicted label—the output.
- b is the **bias** of the model. Bias is the same concept as the y-intercept in the algebraic equation for a line. In ML, bias is sometimes referred to as . Bias is a **parameter** of the model and is calculated during training.
- w_1 is the **weight** of the feature. Weight is the same concept as the slope in the algebraic equation for a line. Weight is a **parameter** of the model and is calculated during training.
- x_1 is a **feature**—the input.

During training, the model calculates the weight and bias that produce the best model.

What parts of the linear regression equation are updated during training?

The bias and weights

Loss

Loss is a numerical metric that describes how wrong a model's **predictions** are. Loss measures the distance between the model's predictions and the actual labels. The goal of training a model is to minimize the loss, reducing it to its lowest possible value.

Distance of Loss

In statistics and machine learning, loss measures the difference between the predicted and actual values. Loss focuses on the distance between the values, not the direction. For example, if a model predicts 2, but the actual value is 5, we don't care that the loss is negative -3 (2 - 5 = -3). Instead, we care that the distance between the values is. Thus, all methods for calculating loss remove the sign.

The two most common methods to remove the sign are the following:

- Take the absolute value of the difference between the actual value and the prediction.

- Square the difference between the actual value and the prediction.

Types of Loss

Loss type	Definition	Equation
L_1 loss	The sum of the absolute values of the difference between the predicted values and the actual values.	$\sum \lvert actual\ value - predicted\ value \rvert$
Mean absolute error (MAE)	The average of L_1 losses across a set of examples.	$\frac{1}{N} \sum \lvert actual\ value - predicted\ value \rvert$
L_2 loss	The sum of the squared difference between the predicted values and the actual values.	$\sum (actual\ value - predicted\ value)^2$
Mean squared error (MSE)	The average of L_2 losses across a set of examples.	$\frac{1}{N} \sum (actual\ value - predicted\ value)^2$

Gradient Descent

Gradient descent is a mathematical technique that iteratively finds the weights and bias that produce the model with the lowest loss. Gradient descent finds the best weight and bias by repeating the following process for a number of user-defined iterations.

The model begins training with randomized weights and biases near zero, and then repeats the following steps:

1. Calculate the loss with the current weight and bias.
2. Determine the direction to move the weights and bias that reduce loss.
3. Move the weight and bias values a small amount in the direction that reduces loss.
4. Return to step one and repeat the process until the model can't reduce the loss any further.

A model trains until it **converges**. When a model converges, additional iterations don't reduce loss more because gradient descent has found the weights and bias that nearly minimize the loss.

If the model continues to train past convergence, loss begins to fluctuate in small amounts as the model continually updates the parameters around their lowest values. This can make it hard to verify that the model has actually converged. To confirm the model has converged, you'll want to continue training until the loss has stabilized

Hyperparameters

Hyperparameters are variables that control different aspects of training. Three common hyperparameters are:

- **Learning rate**

- **Batch size**
- **Epochs**

In contrast, **parameters** are the variables, like the weights and bias, that are part of the model itself. In other words, hyperparameters are values that you control; parameters are values that the model calculates during training.

Learning Rate

Learning rate is a floating point number you set that influences how quickly the model converges. If the learning rate is too low, the model can take a long time to converge. However, if the learning rate is too high, the model never converges, but instead bounces around the weights and bias that minimize the loss. The goal is to pick a learning rate that's not too high nor too low so that the model converges quickly.

Batch Size

Batch size is a hyperparameter that refers to the number of **examples** the model processes before updating its weights and bias.

Two common techniques to get the right gradient on *average* without needing to look at every example in the dataset before updating the weights and bias are **stochastic gradient descent** and **mini-batch stochastic gradient descent**:

- **Stochastic gradient descent (SGD)**: Stochastic gradient descent uses only a single example (a batch size of one) per iteration. Given enough iterations, SGD works but is very noisy. "Noise" refers to variations during training that cause the loss to increase rather than decrease during an iteration. The term "stochastic" indicates that the one example comprising each batch is chosen at random.
- **Mini-batch stochastic gradient descent (mini-batch SGD)**: Mini-batch stochastic gradient descent is a compromise between full-batch and SGD. For N number of data points, the batch size can be any number greater than 1 and less than N. The model chooses the examples included in each batch at random, averages their gradients, and then updates the weights and bias once per iteration.

Epoch

During training, an **epoch** means that the model has processed every example in the training set *once*. For example, given a training set with 1,000 examples and a mini-batch size of 100 examples, it will take the model 10 **iterations** to complete one epoch.

Training typically requires many epochs. That is, the system needs to process every example in the training set multiple times.

The number of epochs is a hyperparameter you set before the model begins training. In many cases, you'll need to experiment with how many epochs it takes for the model to converge. In general, more epochs produces a better model, but also takes more time to train.

Programming Exercise

Completed this programming exercise by google - https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/linear_regression_taxi.ipynb?utm_source=mlcc&utm_campaign=colab-external&utm_medium=referral&utm_content=linear_regression#scrollTo=_yW7nVxlO1WY

Completion

EARNED APR 8, 2025

Machine Learning Crash Course: Linear regression

Completed the Machine Learning Crash Course linear regression module.

Share