IFA-305 Sistem Cerdas (Intelligent System)
Lecture 13-14

Stochastic Gradient Descent

Nur Uddin, PhD.

Program Studi Informatika

Universitas Pembangunan Jaya

Tangerang Selatan

Note:

Materi ini diambil dari Kaagle course berjudul “Introduction to Deep Learning” pada topik ketiga “Stochastic Gradient Descent”.

Introduction-1

- In the first two lessons, we learned how to build fully-connected networks out of stacks of dense layers.
- When first created, all of the network's weights are set randomly -- the network doesn't "know" anything yet.
- In this lesson we're going to see how to train a neural network; we're going to see how neural networks learn.

Introduction-2

- As with all machine learning tasks, we begin with a set of training data.
- Each example in the training data consists of some features (the inputs) together with an expected target (the output).
- Training the network means adjusting its weights in such a way that it can transform the features into the target.
- In the 80 Cereals dataset, for instance, we want a network that can take each cereal's 'sugar', 'fiber', and 'protein' content and produce a prediction for that cereal's 'calories’.
- If we can successfully train a network to do that, its weights must represent in some way the relationship between those features and that target as expressed in the training data

Introduction-3

In addition to the training data, we need two more things:

- A "loss function" that measures how good the network's predictions are.
- An "optimizer" that can tell the network how to change its weights.

The Lost Function-1

- We've seen how to design an architecture for a network, but we haven't seen how to tell a network what problem to solve. This is the job of the loss function.

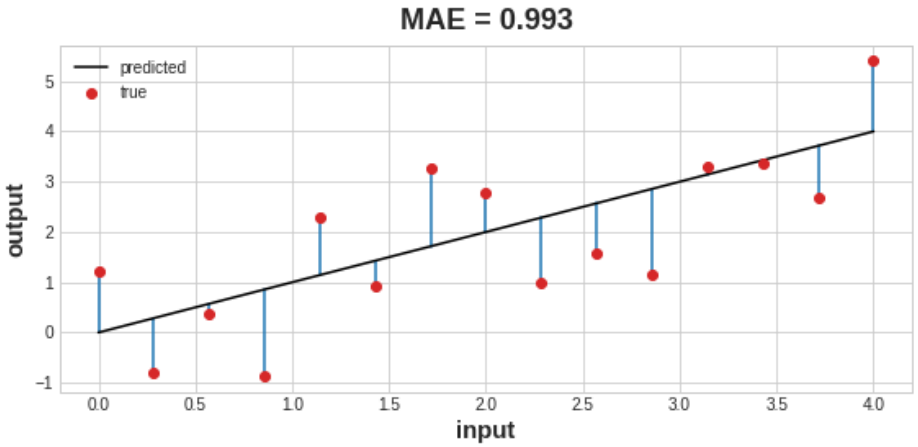
- The loss function measures the disparity between the the target's true value and the value the model predicts.

- Different problems call for different loss functions. We have been looking at regression problems, where the task is to predict some numerical value -- calories in 80 Cereals, rating in Red Wine Quality. Other regression tasks might be predicting the price of a house or the fuel efficiency of a car.

The Lost Function-2: MAE

- A common loss function for regression problems is the mean absolute error or MAE.
- For each prediction y_pred, MAE measures the disparity from the true target y_true by an absolute difference abs(y_true - y_pred).

- The total MAE loss on a dataset is the mean of all these absolute differences.



The Lost Function-3

- Besides MAE, other loss functions you might see for regression problems are the mean-squared error (MSE) or the Huber loss (both available in Keras).

- During training, the model will use the loss function as a guide for finding the correct values of its weights (lower loss is better). In other words, the loss function tells the network its objective.

The Optimizer – Stochastic Gradient Descent (1)

- We've described the problem we want the network to solve, but now we need to say how to solve it. This is the job of the optimizer. The optimizer is an algorithm that adjusts the weights to minimize the loss.

- Virtually all of the optimization algorithms used in deep learning belong to a family called stochastic gradient descent. They are iterative algorithms that train a network in steps. One step of training goes like this:
  1. Sample some training data and run it through the network to make predictions.
  2. Measure the loss between the predictions and the true values.
  3. Finally, adjust the weights in a direction that makes the loss smaller.

The Optimizer – Stochastic Gradient Descent (2)

- Dsa

Epoch

- Each iteration's sample of training data is called a minibatch (or often just "batch"), while a complete round of the training data is called an epoch.
- The number of epochs you train for is how many times the network will see each training example.
- Every time SGD sees a new minibatch, it will shift the weights (w the slope and b the y-intercept) toward their correct values on that batch.
- Batch after batch, the line eventually converges to its best fit.
- You can see that the loss gets smaller as the weights get closer to their true values.

Learning Rate and Batch Size

- Notice that the line only makes a small shift in the direction of each batch (instead of moving all the way).
- The size of these shifts is determined by the learning rate. A smaller learning rate means the network needs to see more minibatches before its weights converge to their best values.
- The learning rate and the size of the minibatches are the two parameters that have the largest effect on how the SGD training proceeds.
- Their interaction is often subtle and the right choice for these parameters isn't always obvious. (We'll explore these effects in the exercise.)

Adam

- Fortunately, for most work it won't be necessary to do an extensive hyperparameter search to get satisfactory results.

- **Adam** is an SGD algorithm that has an adaptive learning rate that makes it suitable for most problems without any parameter tuning (it is "self tuning", in a sense).

- Adam is a great general-purpose optimizer.

Adding the Loss and Optimizer

- After defining a model, you can add a loss function and optimizer with the model's compile method:

```
model.compile(
    optimizer="adam",
    loss="mae",
)
```