Stochastic gradient descent

February 5, 2024

Contents

1 Intro

1.1. In this note we will discuss variants of gradient descent.

1.2. The basic setup is the same as we have used in our discussion of statistical learning theory.

- The sample space will be denoted by X and the space of labels will be denoted by Y. We will denote the product space by $Z = X \times Y$.
- We will consider a joint distribution on $X \times Y$ which will be denoted by D.
- We consider a general loss function
$$l : H \times Z \to \mathbb{R}_{\geq 0}$$
which is measurable in Z.
- The true error function is the function
$$\begin{aligned} L_D &: H \to \mathbb{R}_{\geq 0} \\ h &\mapsto \mathbb{E}_D[l(h,z)] = \int_Z l(h,z)dz \end{aligned}$$

1.3. There are standard choices for loss functions. A standard way to obtain a loss function from training data is to use maximum likelihood estimation.

2 Gradient descent

2.1. Recall the gradient descend algorithm, which is an algorithm to compute the minimum of a (sub)differentiable function.

Algorithm 1 Gradient descent

```
Parameters: integer $T > 0$, $\eta_1, \dots \eta_T \in \mathbb{R}$
initialize: $w^{(1)} = 0$
  for $t = 1, \dots, T$ do
      $w^{(t)} \leftarrow w^{(t-1)} - \eta_t \nabla f(w^{(t-1)})$
  end for

return : $w = \frac{1}{T} \sum^T w^{(t)}$
```

2.2. Remember that our goal is to minimize the generalization error. Hence we would like to minimize the true error function L_D. Let's be more precise for the case of neural networks. Here, the hypothesis space is given by $H_{V,E,\rho}$. Since we have fixed an architecture for our neural network, a choice of weights (and biases) will determine a neural network. Thus, we can view the true error as a function

$$L_D : \mathbb{R}^{|E|} \to \mathbb{R}_{\geq 0}$$

2.3. Since we want to minimize the true error function, we would like to apply the gradient descend algorithm to L_D. However, the distribution D is unknown, hence we don't have access to L_D and its gradient.

2.4. A variation is to minimize the empirical risk instead. Suppose we are give trainings data $S \in Z^m$. Recall that the empirical risk is given by

$$L_S : H \to \mathbb{R}_{\geq 0}, \quad h \mapsto \frac{1}{m} \sum l(h, z)$$

Statistical learning theory tells us that minimizing the empirical risk is a good approach to learning. <inimizing the empirical risk using gradient descend is also called *batch gradient descend*. Note that in the algorithm, we need to compute the gradient $\nabla L_S(h)$ and thus by linearity

$$\nabla L_S(h) = \frac{1}{m} \sum \nabla l(h, z)$$

Thus we need to evaluate our model at each sample and compute the gradient at each sample which is in general very inefficient.

Algorithm 2 Batch Gradient descent

```
Parameters: integer $T > 0$, $\eta_1, \dots \eta_T \in \mathbb{R}$
initialize: $w^{(1)} = 0$
  for $t = 1, \dots, T$ do
    $w^{(t)} \leftarrow w^{(t-1)} - \eta_t \nabla L_S(w^{(t-1)})$
  end for

return : $w = \frac{1}{T} \sum^T w^{(t)}$
```

3 Stochastic gradient descend

3.1. The key insight of stochastic gradient descend is that we don't really need to know the gradient of the true error function. It suffices to have an unbiased estimator of the true gradient. To be more precise, suppose we have a function $f : \mathbb{R}^n \to \mathbb{R}$ that we want to minimize using gradient descend. Thus we need to compute the gradient $\nabla f(w)$ at a point $w \in \mathbb{R}^n$. Given such a $w \in \mathbb{R}^n$, let's say we sample a point $v \in \mathbb{R}^n$ according to some conditional distribution P. The condition to be an unbiased estimator is then that

$$\mathbb{E}_P(v|w) = \nabla f(w)$$

This leads to the following general version of stochastic gradient descend.

Algorithm 3 Stochastic Gradient descent

```
Parameters: integer $T > 0$, $\eta_1, \dots \eta_T \in \mathbb{R}$
initialize: $w^{(1)} = 0$
  for $t = 1, \dots, T$ do
    Sample $v$ according to some distribution such that $\mathbb{E}_P(v|w^{t-1}) = \nabla f(w^{t-1})$
    $w^{(t)} \leftarrow w^{(t-1)} - \eta_t v$
  end for

return : $w = \frac{1}{T} \sum^T w^{(t)}$
```

3.2. In practice, how do we find such an estimator? It turns out that this is actually quite easy. Suppose for the moment that we know the distribution D. If we sample a single point $z \in Z$ according to D, then $\nabla l(h, z)$ is an unbiased estimator. To see this note that in our case the gradient operator commutes with taking the integral and thus we have

$$\mathbb{E}_D \nabla(l(h, z)) = \nabla \mathbb{E}_D(l(h, z)) = \nabla L_D(h)$$

3.3. But then, we still don't know D. However, we do have access to points $z \in Z$ which are sampled from D. Our trainings data! Now there are two heuristics for stochastic gradient descend.

- To obtain good results, we should use all the trainings data available
- Running stochastic gradient descend several times leads to better results.

To implement these heuristics one uses the concept of *epochs*. An epoch is finished whenever the algorithm has seen every training point. Thus, instead of steps, we have the number of epochs as parameter resulting in the following algorithm:

```
Algorithm 4 Stochastic Gradient descent with epochs
Parameters: integer $E > 0$, $\eta \in \mathbb{R}$
initialize: $w^{(1)} = 0$
  for $e = 1, \dots, E$ do
      for $z \in S$ do
          $w^{(t)} \leftarrow w^{(t-1)} - \eta \nabla l(h, z)$
      end for
  end for

return : $w = \frac{1}{T} \sum^T w^{(t)}$
```

3.4. In contrast to batch gradient descend, which also sees all the training data, we only need to evaluate once to take a step in the gradient descend algorithm instead of evaluating everything to take a single step.

4 Minibatch Stochastic Gradient Descend

4.1. There is a middle ground between stochastic gradient descend and batch gradient descend called minibatch gradient descend. This can also take into account hardware advantages in contrast to stochastic gradient descend. It uses another estimator for the gradient of the true error function. Again, for stochastic gradient descend we just need some probability space to estimate the gradient. This time we take subspaces of our training space, i.e. we choose $k << m$ and consider the random variable

$$Z^k \to \mathbb{R}_{\geq 0}, \quad (z_1, \dots, z_k) \mapsto \frac{1}{k} \sum_i \nabla l(h, z_i)$$

We have

$$\mathbb{E}_{D^k} \frac{1}{k} \sum_i \nabla l(h, z_i) = \frac{1}{k} \nabla \sum E_D[l(h, z)]$$

Thus we have another estimator for the gradient of the true loss function. We then obtain the following algorithm.

4.2. In conclusion, minibatch gradient descend takes a step for each batch and will do so for the number of epochs we specified. While computing the gradient we can take advantage and compute several gradients of the sum at once.