Optimization For Machine Learning with Python

Rajiv Sambasivan

November 3, 2017

Outline

This session will be mostly implementation oriented rather than theory oriented. However, to set up a context, I will talk about:

1. Need for optimization in machine learning.
2. Some canonical forms of optimization problems.
3. Some important characteristics of problems to be aware of during implementation.
4. A brief overview of implementation challenges.

Need for Optimization

Consider the least squares problem - fitting a plane:

$$Y = w.x + \eta$$

Here:

- X are the predictors, Y is the response, $\eta \sim \mathbf{N}(0,1)$ is the noise.
- Analytical solution exists $\implies \hat{Y} = \left(X^T X\right)^{-1} X^T.Y$
- We can use this as long as we can compute $\left(X^T X\right)^{-1}$
- Matrix inversion for a matrix of size N is associated with time complexity $O(N^3)$ so large datasets (large N) presents a problem if we want to use the analytical solution.

Need for Optimization... continued

- Another perspective to solving this problem is through the framework of Empirical Risk Minimization.
- This framework has the following template.
- We have an estimator or classifier that predicts a label $\hat{Y}$ associated with a given set of predictors X. We have the correct labels Y. This permits us to evaluate the quality of our estimator by examining how different or alike $\hat{Y}$ and Y are. Mathematically this is described by a loss function.
- The best parameters (w) for our estimator are obtained by minimizing the collective loss for the given data.

Need for Optimization... continued

- The implementation of minimizing the loss function requires us to exploit certain mathematical properties associated with the solution.
- The function values do not change in the (small) neighborhood of the minimum.
- We exploit this property and find the minimum using various optimization algorithms.
- Based on the characteristics of our problem, we can pick a particular optimization algorithm.

Optimization Problem Characteristics

The basic template for most algorithms is the following:

- Start at some point in the parameter space (this is w for the linear least squares problem).
- Based on the characteristics of your problem, prescribe some model to approximate the function in the neighborhood of a point. Examples - using the first derivative only, using the first and second derivatives.
- Calculate the next value of w to evaluate the function. This will depend on the model used in the previous step.
- When loss values do not change in successive evaluations, we are done. We have determined the optimal parameter values.

Terminology

The problem context has been defined. A glossary of key conceptual ideas is needed to navigate the documentation.

- Use of derivatives: How does the optimization method use derivatives to approximate the loss function in the neighborhood of a point in the parameter space? - derivative free, first order, second order
- Use of constraints: Does the problem incorporate constraints on the parameter values? - strictly positive, satisfy some inequality etc. Accordingly the problem belongs to the unconstrained or constrained family.
- Nature of the function to be minimized: - does it involve only continuous values or only discrete values or both. Do we even have an explicit form for the function? Is the objective function noisy?

Problem Characteristics

- Taking the effort to make note of problem characteristics can help us pick the right optimization choice for our problem
- Treating the optimization step is a black box completely is not a good idea for large problems
- Getting a handle on the condition number associated with the problem will help - broadly this tells us how much the output for the algorithm changes for a small change in the input. Low numbers are good, high numbers are bad.
- Pre-conditioning the input helps - we will look at how normalization of the input can affect the convergence in the practical examples.
- Objective function characteristics are important - is the objective function convex? is it strongly convex? Can we assume second order derivatives to exist?

Application - Gradient Descent

Let us make these ideas concrete and examine the most common technique to minimize a loss function - gradient descent.

- This is a first order method. This implies that we use the first derivative to approximate the function (using the Taylor series expansion) in neighborhood of a point.
- We need to provide the method a starting point for the parameter θ that we want to optimize.
- The algorithm is straightforward:

```
1. Set $\theta_0 = initial\ value$
2. do {
       $$\theta_{updated} = \theta_{old} + \alpha.\nabla(loss(\theta_{old}))$$
   }
   while $\theta$ does not change much
```

- Here ∇ represents the gradient, α is the learning rate - we will discuss this.

Application - Gradient Descent

Let us examine the basic gradient descent algorithm in a little more detail.

- α controls how far along the gradient we advance in each iteration. There are variants of gradient descent that adjust this parameter as we proceed through the algorithm
- The loss function, $J(\theta)$, is actually the expected loss

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{i=N} loss(i,\theta)$$

- Note that the loss function involves iterating the entire dataset for each parameter update - expensive step for large datasets.

References I