Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 56

Hello everyone. This is the first lecture of week 12, the final week of this course. In the past 11 weeks, we have learned extensively about many algorithms in both unconstrained and constrained optimization. In constrained optimization, we learned in detail about certain algorithms for non-linear programming and certain algorithms for linear programming.

We had gradient descent, the conjugate gradient algorithm, Newton's method, and quasi-Newton methods, all for unconstrained optimization. In constrained optimization for non-linear programs, we discussed the penalty method and the augmented Lagrangian method. For linear programming, we discussed the simplex method, affine scaling, and Karmarkar's algorithm. That is quite a lot of methods.

A basic question we have not addressed much is where these algorithms are actually used. Of course, many industries have optimization problems to solve. You type values into a computer, and it gives you the answer. That is fine. This entire week, I plan to take up a particular application, specifically in machine learning, as it is used very extensively. We will discuss the role of optimization algorithms in machine learning.

The plan is as follows. I will first explain the application in some detail because I want you to understand what it is. I will start from the basics of machine learning. I will explain what machine learning is, give some examples, and converge on a particular technique, a particular method in machine learning. Then, we will discuss how the optimization algorithms we have learned in the past eleven weeks are useful for finding the solution for this particular method. We will also take a publicly available dataset.

I will start with what machine learning is. Many of you may know this, but since at least some of you would like an outline, I will go through it quickly. Let me start with an example. Consider the problem of determining whether a given email is spam or not. The machine finds this out by itself. Many of you use Gmail or other popular mail services. When you receive an email, Gmail itself decides whether that particular mail is spam with near 100 percent accuracy, perhaps 99.5%. When you get 100 or 1000 emails, about 995 or 999 are classified correctly. Only one or two might be wrongly classified, where a spam mail is classified as not spam, or a not-spam mail is classified as spam. How does this happen? How does the machine know by itself that a mail is spam?

If you are familiar with machine learning, you know it is used in this problem. This problem is called spam detection. These are some examples of machine learning. The spam detection problem, put simply, is this: when given an email, classify it as either spam or not spam.

How would you make a computer do this? A simple answer is to ask this question instead: suppose you have a child, perhaps someone in first standard, a five or six-year-old kid. How would you teach him this? He possibly does not know much about computers, but assume he knows what an email is. How would you teach him to determine, upon seeing an email, whether it is spam or not?

I would do the following. I would give him many examples. I would show him a particular email and say, "This is how spam looks." For example, a message saying "you get \$2 million" is possibly spam. Then I would show him some normal mails which are not spam. The kid sees hundreds, thousands of mails like this. Eventually, when the next mail comes, he will not need my help. He will see the mail and say it is spam because it has phrases like "dollar 2 million." Otherwise, he will say it is a normal mail, not spam.

What has happened? You have given some labeled data. Labeled data consists of data and a label. The data is the email, and the label is whether it is spam or not. You gave many examples of spam and not-spam mails. The child saw many such mails and, based on that, learned to classify a new mail as spam or not. We are going to do the same for a computer. We will give it many labeled data. We will give it emails initially with a label, saying particular mails are spam and particular mails are not spam.

Put in mathematical terms, you are giving something called data points. Assume you give email 1, and then also give a label 1. Label 1 could be spam or not spam. Then you give email 2 with label 2, which is possibly spam or not spam. This continues for maybe n emails, thousands of emails. This collection is called the dataset. A dataset consists of n data points.

We usually write it this way in general. For a spam detection problem, and in general for any supervised learning problem, you have a dataset with labeled data. The data is (x_1, y_1), which is like email 1 and label 1. x_1 could be large; it is like a vector. Here, it is a vector of words. x_i, for that matter, will be a vector belonging to $\mathbb{R}^p$.

y_i, based on the problem, could belong to the real numbers. In this case, it is not that; it is actually 0 or 1. This is because the given example is a classification problem. You classify it as 0 or 1. We could have a regression problem where the label is a real number. This is categorical data.

You have n data points where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$.

It need not be $\mathbb{R}$; it could be a set of values $a_1, a_2, ..., a_k$.

If it is real numbers, it is a regression problem. If it is a set of categories, it is a classification problem. The spam detection problem is a classification problem. A regression problem example is stock market prediction. Based on previous data, you predict the stocks for the following days. Suppose you are given Sensex data for the last few years with many other parameters, not just the Sensex data, such as how many companies are trading, how many people are investing, macro details like GDP growth, and so on. You are given all these data points on a daily basis and the stock value. You must predict the stock value the next day. That is the stock market prediction problem. Here, you have labeled data because for the set of features—the number of people trading, the economic condition, etc.—these are the independent variables. On the right, you have a dependent variable, the stock market price, which is not 0 or 1; it could be any real number. That is a regression problem.

This is the data given to you. The task is, given a test vector x_0, find y_0.

When the vector is x_1, the label is y_1; when the vector is x_2, the label is y_2; ...; when the vector is x_n, the label is y_n.

Now, you are given a completely new vector x_0 and asked to find y_0. That is the task.

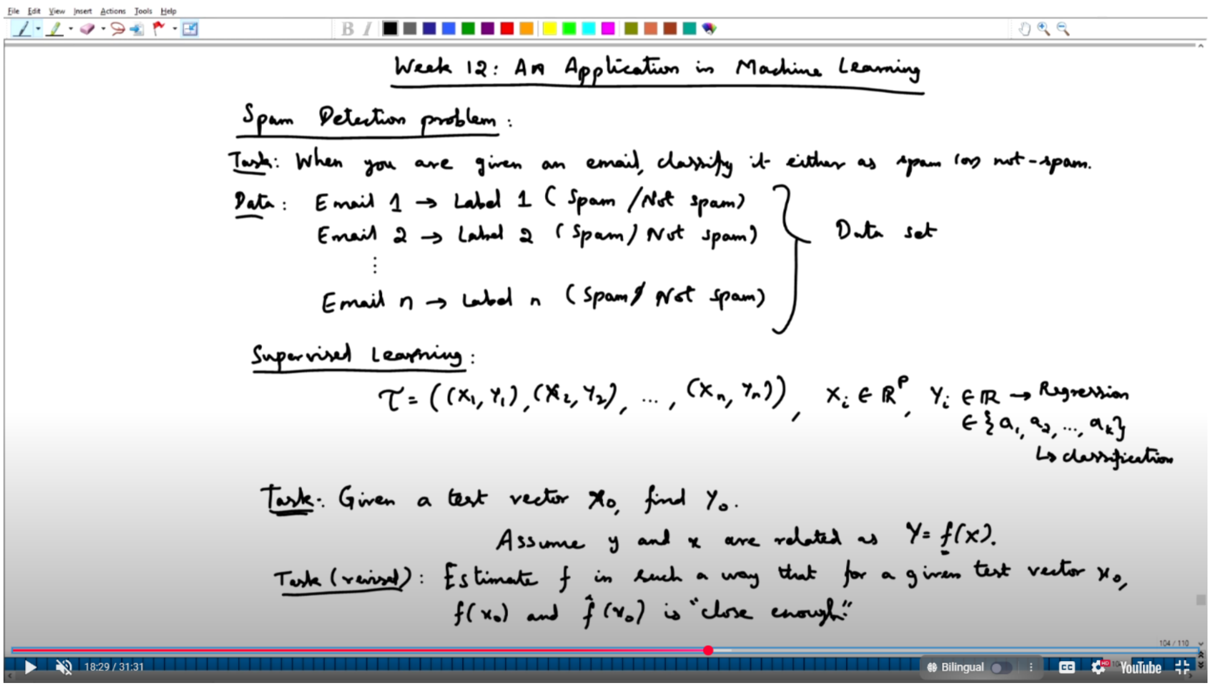
How will you find that? This seems to be a very superficial problem. You have the example of spam detection where you were given email x_1 with label y_1 (spam or not spam), similarly up to x_n with label y_n. Now you are given a new x_0, a new email, and asked to classify it as spam or not. Similarly, for stock market prediction, you have a set of features for which you are given the stock price for hundreds or thousands of days. On a new day, you have the set of parameters and must predict y_0.

How do you do this? You assume that y and x are related in the form of a function. Assume that y and x are related as $y = f(x)$.

The revised task is, given x_0, estimate $\hat{f}(x_0)$ such that $\hat{f}(x_0)$ is close to the true $f(x_0)$.

This is a better way. For a given test vector x_0, $\hat{f}(x_0)$ should be close enough to $f(x_0)$. "Close enough" is in quotes because I have not defined what it means or how close is close.

(Refer Slide Time 18:29)



Week 12: An Application in Machine Learning

Spam Detection problem:

Task: When you are given an email, classify it either as spam (or) not-spam.

Data: Email 1 → Label 1 (Spam / Not spam)
Email 2 → Label 2 (Spam / Not spam)
⋮
Email n → Label n (Spam / Not spam)

Data set

Supervised Learning:

$$\mathcal{T} = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)), \quad x_i \in \mathbb{R}^p, \quad y_i \in \mathbb{R} \rightarrow \text{Regression}$$
$$\in \{a_1, a_2, \dots, a_k\} \hookrightarrow \text{classification}$$

Task: Given a test vector x_0, find y_0.

Assume y and x are related as $y = f(x)$.

Task (revised): Estimate f in such a way that for a given test vector x_0, $f(x_0)$ and $\hat{f}(x_0)$ is "close enough."

To refine this statement and say what "close enough" means, I will define a loss function. Define a loss function

$L : Y \times Y \to \mathbb{R}$. L is for loss.

Y is the space from which y is taken. A better way to write this is: $x_i \in X$, a subset of $\mathbb{R}^p$, and y_i is in some Y, a subset of $\mathbb{R}$ or a set of categories.

There are some popular loss functions. For example, squared loss is of the form

$L(y, \hat{y}) = ||y - \hat{y}||^2$.

Absolute loss = $|y - \hat{y}|$.

These are used for regressions in general.

You also have the 0-1 loss, which is 0 if $y = \hat{y}$ and 1 if $y \neq \hat{y}$.

This is usually used for classification problems. If you classify spam as spam, the loss is 0.

If you classify spam as not spam, or not spam as spam, the loss is 1. If you classify not spam as not spam, the loss is 0. These are some nice loss functions. You can have many others.

Now, I have defined a loss function. I will revise the task again.

Given a test vector x_0, estimate $\hat{f}(x_0)$ such that the loss $L(f(x_0), \hat{f}(x_0))$ is minimized.

This is basically what you need to find.

If you follow this closely, you see we have an optimization problem. You are finding a function f, an estimation $\hat{f}$, such that the loss function $L(f(x_0), \hat{f}(x_0))$ is minimized.

Roughly, this is an optimization problem because in optimization, you minimize some function of some variables, possibly subject to constraints.

$L(f(x_0), \hat{f}(x_0))$ is a function of $\hat{f}$, and you minimize it with respect to $\hat{f}$.

In very general terms, any supervised learning problem has this form. I have not pinpointed a particular one.

You have labeled data $(x_1, y_1), ..., (x_n, y_n)$.

You assume a relation $y = f(x)$.

Given a test vector x_0, your task is to

minimize $L(f(x_0), \hat{f}(x_0))$ for some given loss function L.

This is the general structure for any supervised learning problem. You can see that any supervised learning problem has an optimization problem embedded within it. To find $\hat{f}$ that minimizes this loss function, you must solve an optimization problem. To solve an optimization problem, you use optimization algorithms. That is why we need optimization algorithms in the first place.

I will speak further about the functions.

This problem assumes y and x are related as $y = f(x)$.

Considering any general function f is very hard to solve. What we do is, for different methods, we make simplifying assumptions on this function f.

I am not sure how many of you have heard of the popular method OLS, ordinary least squares, or in general linear regression or linear classification problems. The OLS method assumes f to be linear in x. Instead of any general function f, it assumes f is linear and then solves for the coefficients of the linear function. Not just OLS; OLS is the most popular method that assumes f is linear. Other methods like ridge regression, lasso, and a few other shrinkage methods are also popular. In general, we have some simplifying assumptions on f. Some methods do not have this, but many do.

To cite a few methods for now: OLS (ordinary least squares) and ridge regression assume that f is linear. You might also see CART, classification and regression trees, also called decision trees. These assume that f is constant in axis-parallel regions. I can cite many such examples.

Let me explain what a simplifying assumption means.

If f is linear, then we have

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$, where X is the vector $(x_1, x_2, ..., x_p)$.

Recall that each x_i is a p-dimensional vector. Each example has a p-dimensional vector. If y is linear in x, it must be of this form: a constant plus β_1 times the first element of the vector, plus β_2 times the second element, plus ... plus β_p times the p-th element.

If the loss is squared loss, then we need to

minimize $L(f(x_0), \hat{f}(x_0)) = ||f(x_0) - \hat{f}(x_0)||^2$.

But $f(x_0) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$.

In f, you have $\beta_0, \beta_1, ..., \beta_p$. In $\hat{f}$, you have $\hat{\beta}_0, ..., \hat{\beta}_p$.

This quantity becomes $||y - \hat{\beta}_0 - \Sigma\, \hat{\beta}_i x_i\, ||^2$, for i=1 to p.

This is the quantity we must minimize.

Now we have a very nice optimization problem to look at. We want to minimize a mathematical quantity that is nicely defined. This gives a nice example of how optimization problems can solve certain machine learning applications.

We will continue in the next lecture on this same example where we have the simplifying assumption that f is linear. We will discuss different methods before converging to the method we are looking for. Thank you.