CSE 475: Statistical Methods in AI **Monsoon 2018**

Lec 3: Linear Methods-I

Lecturer: C. V. Jawahar *Date: Aug. 8, 2018*

3.1 Problem Space

Let us start with our basic problem. We are given a number of examples in the form $\{\mathbf{x_i}, y_i)\}$ for $i = 1, \ldots, N$. For the simplicity, we assume that $\mathbf{x_i}$ is a real d-dimensional vector. And y_i is a scalar (say real number or an integer). Our interest is in finding a function $f()$ such thar $f(\mathbf{x_i})$ is same (or very similar) as y_i.

(At this moment, we will deal with the requirement of the "very similar" to be as close as possible for all the N training samples that we have. Later we should also make sure that the same function will work for all the samples that we come across in the future also (i.e., unseen samples).)

For example, $\mathbf{x_i}$ could be an email that is represented with a real vector. and y_i could be 0 or 1 corresponding to "spam" (1) or not (0). In this case it is a classification problem. One may also formulate the problem as classification of the email into "spam" (0), "personal" (1) and "professional" (2). In this case, this is a multiclass classification problem. You may also predict "how important/urgent" is an email by looking at the content. In this case, then y_i is a real number (say in the raneg $[0-1]$ where 0 means least urgent and 1 means extremely urgent.

3.1.1 Classification

In classification problems y_i is an integer. What value we assign is of not much significance. For example, some people use 0 and 1, while some where else we use -1 and $+1$ as the class IDs. The choice is often based on some conviniences (simpler form of equations!!). In both these cases, it is a "binary" classification. In many cases we also call these classes as ω_1 and ω_2.

Multi class classification where the number of classes is more than 2 is a popular case. Though multiclass classification is very popular and important in practice, many discussions in the linear classifiers assume that the number of classes is only 2. That makes the life simple.

3.1.2 Regression

In the case of regression, y_i is real quantity. It could be a real vector or a simple real number. Let us assume it as a real number at this stage.

3.1.3 Others

There are many other situations where the space of y has structure. For example, y is a graph or a string. Such problems are beyond our interest at this stage.

3.2 Linear Models

Let us first start with a specific, (simple), and yet effective class of models/functions.

$$y = f(x) = \mathbf{w}^T\mathbf{x} + w_0 \tag{3.1}$$

If $d = 2$ then the model is something like

$$y = w_2x_2 + w_1x_1 + w_0$$

The additional term w_0 is required to make sure that it covers all the possible "lines" including the ones that does not pass through the origin.

This is not limited to 2D samples. Lines naturally gets extended to planes and hyperplanes.

3.2.1 Augmented Vectors and Feature Maps

The notation (and many math that come later) could be simpler if we do not have w_0. We do that by augmenting the vector $\mathbf{x}$ with an additional quantities. i.e., the new $\mathbf{x}$ is the old $\mathbf{x}$ with an additional 1 concatenated at the end. Similarly the $\mathbf{w}$ is augmented with w_0 and the equation 3.1 gets simplified as

$$y = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \tag{3.2}$$

Note that we did not introduce a different notation with and without augmentation. This is to make the notations simpler. (Text books may be using different notations). Hope you appreciate the convinience of augmenting with 1.

What more we can do with augmentation (or explicit feature maps)? We can infact create a new vector from the old ones. This also generelizes the trick of augmentation.

Consider that our original vector is $[x_1, x_2]^T$. We now know how to create a new vector $[x_1, x_2, 1]$. Why not $[x_1^2, x_2^2, x_1x_2, x_1, x_2, 1]$? Such a modification will allow us to learn a new model $\mathbf{w}^T\mathbf{x}$ as

$$w_5x_1^2 + w_4x_2^2 + w_3x_1x_2 + w_2x_1 + w_1x_2 + w_0 \qquad (3.3)$$

which is really a quadratic function. Our linear algorithm (that we see soon) is able to learn nonlinear models too. The objective of introducing this at this stage is to convince that the algorithms that we will discuss are very powerful and useful. Linearity does not constrain us too much.

3.3 How do we formulate ?

Now let us come back to our problem. We are given examples $\{(\mathbf{x_i}, y_i)\}$ $i = 1, \dots, N$. What do we want to do? We want to find the most appropriate $\mathbf{w}$.

Indeed we may not find a single $\mathbf{w}$ that can make $y_i = \mathbf{w}^T\mathbf{x}$ for all i. This could be due to various reasons including errors, noise or uncertainty in the data. Therefore we want to model the problem as find the "most appropriate" $\mathbf{w}$. This naturally lead to an optimization problem. Most appropriate in what sense? We need to define an appropriate sense or objective that can be computed. This is the objective function. In machine learning, we also use the term loss function or error function frequently. All these are used with very similar meanings.

Our problem is then to find $\mathbf{w}$ that minimizes

$$Total{-}Loss = \sum_{i=1}^{N} Loss{-}Per{-}Sample = \sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)$$

Why do we have to sum over i? Why not products? That is also possible. We do summation so that differentiation is easier later. (do you remember how to differential $u + v$ and uv?). There may be many different ways in which you can define loss functions.

Q: Do you see any other advantage or disadvatage of products over sum? Which will be more sensitive to outliers? (samples which may have very larger error).

3.3.1 Loss Function

Consider the regression problem in 1D. You have (x_i, y_i). By augmenting 1, x_i becomes a 2 dimensional vector $\mathbf{x}_i$. Our problem is to find the vector $\mathbf{w} = [w_1, w_0]^T$ such that the model is $y = w_1x_1 + w_0$. Look at this as a line fitting.

Error or loss in this case is the difference between the model prediction and actual.

$$L(\mathbf{w}, \mathbf{x_i}, y_i) == (y_i - \mathbf{w}^T\mathbf{x_i})^2$$

We square the error such that no loss is negative. This is required since we will now be adding the loss from different examples to get the total loss.

And the total loss or objective is then sum over all the examples

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x_i})^2 \qquad (3.4)$$

Note that we have only one $\mathbf{w}$ for all the samples.

Similarly for a classification problem, loss can be 1 if the classification is wrong and 0 if the classification is correct.

Let us assume that the classifier is

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} > 0 \\ -1 & \text{else} \end{cases} \qquad (3.5)$$

$$J = \frac{1}{N}\sum_{i=1}^{N}(1 - y_i \cdot f(\mathbf{x_i})) \qquad (3.6)$$

- Q: If we had used a $0 - 1$ convention for the classes, how the equation could have been written?
- Q: The problem with this loss is that this is not differentiable. Why?

3.3.2 Optimization

The optimization problem we need to solve is

$$minimze \;\; J(\mathbf{w}) \qquad (3.7)$$

Though we know what problem to solve, very often we can not find the "best w" in practice. There are two prominent classes of optimization problems:

- Convex optimization. A well behaved class of problem. You can find the optima. And often efficiently.
- Non-Convex optimization problem. This is a class of nasty optimization problems. Unfortunately, we will encounter them very frequently. In this case, very often, we have to be happy with "a" minima/solution and not the best solution.

More details of these classes of problems is beyond the scope of this course.

3.4 Regression, Line Fitting and MSE Solution

Let us now see how do we find the best $\mathbf{w}$ for our line fitting problem in 1D. i.e., (x_i, y_i). As dsicussed above, by augmenting, we got the problem as $(\mathbf{x_i}, y_i)$. We want a model $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. Or we want to minimize the error $e_i = y_i - \mathbf{w}^T\mathbf{x}$. Let us write the equations as vectors/matrices.

$$\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_N}^T \end{bmatrix} \mathbf{w} \tag{3.8}$$

The right hand side is more compactly $\mathbf{Y} - \mathbf{X}\mathbf{w}$

The total loss (sum of squared error) or mean sum of sequred error (equation 3.4) is.

$$J(\mathbf{w}) = \frac{1}{N}[\mathbf{Y} - \mathbf{X}\mathbf{w}]^T[\mathbf{Y} - \mathbf{X}\mathbf{w}]$$

where $\mathbf{Y}$ is a $N \times 1$ vector. $\mathbf{X}$ is a $N \times d$ matrix and $\mathbf{w}$ is a $d \times 1$ vector.

$$J(\mathbf{w}) = \frac{1}{N}(\mathbf{Y}^T\mathbf{Y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T(\mathbf{X^T}\mathbf{Y}))$$

3.4.1 Problem and the closed form solution

Our problem is to find the "best" $\mathbf{w}$. Note that the only variable in the loss function is $\mathbf{w}$. We can differentiate the loss function with respect to $\mathbf{w}$ and equate to zero to get the minima. This leads to

$$2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{Y} = 0$$

or

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

Since we started by finding a $\mathbf{w}$ that suits $\mathbf{Y} = \mathbf{X}\mathbf{w}$. We can also look $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ as the pseudo inverse of $\mathbf{X}$.

- Q:Why do we have to have a pseudo inverse? Why not a regular inverse?
- Q:Under what situations the inverse in the above equation can not be computed? When can this happen for the above MSE problem.
- Q;If all the samples were on the line itself. i.e., all the errors e_is were zero. What do we know about the matrix $(\mathbf{X}^T\mathbf{X})$?
- Q: If $N = 1$, is the problem (easily) solvable? what could happen to ths solution?

3.4.2 Discussions

The above formulation is very effective. However, here are some points woth noting.

- This assume all the samples are available before we start. (not when samples come over time in an online manner).
- The inverse of a $d \times d$ matrix is not very attractive. Specially when d is large.
- As a line fitting in 2D, this is not very intuitive, some time we want the orthogonal distance to the line to be minimized.