CSE 475: Statistical Methods in AI **Monsoon 2019**

SMAI-M-2019 7: Linear Models(I): Regression

Lecturer: C. V. Jawahar *Date: 26 Aug 2019*

7.35 Background

We had seen how a typical ML problems gets formulated. We are looking for a data driven model that can predict y_i given $\mathbf{x}_i$. i.e.,

$$y_i = f(\mathbf{x}, \mathbf{w})$$

Our solution is parameterized by $\mathbf{w}$. During training, an appropriate optimization problem was solved to obtain $\mathbf{w}$ from $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}, \quad i = 1, \ldots, N$ We also appreciated the basic mathematical notions associated with these problems in the last set of lectures. To recollect, we talked about problems and the solutions obtained by solving an optimization problem over the data during the training. What were our problems?

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. 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. (Don't worry. We can extend these ideas to multiclass setting without much effort. We will also see some later on.)

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

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. Popularly these are also called structured prediction problems.

7.36 Modelling and Solving

Now we will systematically look into two very important aspect of the machine learning:

- How do we model the problem and relationship between the variables? (or how do we choose the function class of interest)
- How do we formulate and solve the associated optimization problem for training? What are the associated theoretical issues in optimization? and what are the associated practical issues in implementation?

We will see some of these in the next few lectures.

Linear Vs Non-Linear Models Two major classes of models that we use are:

- Linear Models: Simple models; yet effective in many problems; linear functions.
- Non-Linear Models (eg. Neural Networks): Very effective for complex problems; Hard to interpret

Convex and Non-Convex Optimization Some of the optimization problems lead to globally optimal solutions and performance guarantees. Based on the nature of the objective function and constraints, problems are broadly classified into:

- Convex optimization (eg. MSE in linear regression, SVMs)
- Non-Convext optimization (eg. Training Deep Neural Networks)

7.37 Linear Models in ML

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

$$y = f(x) = \mathbf{w}^T\mathbf{x} + w_0 \tag{7.7}$$

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. For example

$$y = w_3x_3 + w_2x_2 + w_1x_1 + w_0$$
$$y = w_4x_4 + w_3x_3 + w_2x_2 + w_1x_1 + w_0$$

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 7.7 gets simplified as

$$y = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x} \tag{7.8}$$

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 for these two). Hope you appreciate the convenience 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 generalizes 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

$$[x_1, x_2]^T \rightarrow [x_1^2, x_2^2, x_1x_2, x_1, x_2, 1]^T$$
$$\phi(\mathbf{x}) = [x_1^2, x_2^2, x_1x_2, x_1, x_2, 1]^T$$

Such a modification will allow us to learn a new model $\mathbf{w}^T\mathbf{x}$ as

$$w_5x_1^2 + w_4x_2^2 + w_3x_1x_2 + w_2x_1 + w_1x_2 + w_0 \tag{7.9}$$

This is really a quadratic function. Our linear algorithm (that we see soon) is able to learn nonlinear models too, provided $\mathbf{x}$ was augmented and modified to suite this function. The objective of introducing this feature mapping trick ($\phi()$) at this stage is to convince you that the algorithms that we will discuss are very powerful and useful. Linearity does not constrain us too much. We will see this feature map again when we discuss kernels in the context of SVMs.

7.38 Regression and MSE

7.38.1 Line Fitting

We want to start with a simple problem. We are given a number of samples $x_i, y_i)$ $i = 1, \ldots, N$ and our objective is to find a line that fits this data. i.e.,

$$y_i = w_1x_i + w_0$$

Note our problem is to find two coefficients (parameters) w_1 and w_0 that can define the line. Indeed when we fit a line, all points need not be on the line. There could be errors. This leads to

$$y_i = w_1x_i + w_0 + \epsilon_i$$

What should be our goal? We want ϵ_i to be small for all the samples or

$$Minimize \frac{a}{N}\sum_{i=1}^{N}\epsilon_i^2$$

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 discussed 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 $\epsilon_i = y_i - \mathbf{w}^T\mathbf{x}$. Let us write the equations as vectors/matrices.

Though we started with the problem of fitting a line (linear model) in 2D plane, the final problem we have formulated is applicable for any data in d-dimensions. Lines become hyperplane. y_i still remain scalar.

$$Minimize \frac{1}{N}\sum_{i=1}^{N}\epsilon_i^2 = Minimize \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

7.38.2 Mean Squared Error (MSE) Loss

There are N samples with us. We first state our problem a compact matrix form.

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_N}^T \end{bmatrix} \mathbf{w} \tag{7.10}$$

The equation is more compactly written as

$$\mathbf{E} = \mathbf{Y} - \mathbf{X}\mathbf{w}$$

The loss (sum of squared error over all the samples) or mean sum of sqaured error (MSE) is:

$$J = \frac{1}{N}\mathbf{E}^T\mathbf{E}.$$

Why are we dividing by N?

Our objective is to minimize the above loss. What do we have control to modify? $\mathbf{w}$. We minimize over $\mathbf{w}$. i.e, $J()$ is a function of $\mathbf{w}$ and the minima can be obtained by optimizing

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

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.

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

7.38.3 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?

7.38.4 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.

7.39 Probabilistic View Point

We saw how a closed form solution to the regression with MSE loss can be obtained. However, is there a rationale for the MSE loss? Here, we argue that the MLE estimate of the parameters, under certain assumptions, lead to MSE loss/objective.

We know that the problem is modelled as

$$y_i = \mathbf{w}^T\mathbf{x} + \epsilon_i$$

We assume that the ϵ_i are distributed IID (Independently and Identically Distributed) according to a Gaussian/Normal distribution with zero mean and variance σ^2

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\epsilon_i^2}{2\sigma^2}\right)$$

This leads to

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y_i - \mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right)$$

This represents the distribution of y_i given $\mathbf{x_i}$ and parameterized by $\mathbf{w}$. We model this as $\mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2)$

Our problem is now to find the MLE estimate of $\mathbf{w}$ from the data. Given the data $(\mathbf{X}, Y)$, we can do a maximum likelihood estimate of $\mathbf{w}$.

$$L(\mathbf{w}) = L(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \mathbf{w})$$

$$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y_i - \mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right)$$

Instead of maximizing $L(\mathbf{w})$, we maximize the log-likelihood.

$$l(\mathbf{w}) = \log\ L(\mathbf{w}) = \log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y_i - \mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right)$$

$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y_i - \mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right)$$

$$= K\frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x})^2$$

Where K is a constant.

This imply now that the MSE objective we had is only a scaled version of the MLE estimate objective we have here. Both view points come closer again.

$$\Rightarrow \min_{\mathbf{w}} J(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

In other words, under the appropriate probabilistic assumptions of the data, least-squares linear regression (MSE) is exactly the same as the maximum likelihood estimate of $\mathbf{w}$.

Another interesting point to note at this stage is that our final choice of $\mathbf{w}$ did not depend on what was σ^2.

7.40 Lasso and Ridge Regression

Let us revisit our regression model

$$y_i = \mathbf{w}^T \mathbf{x}_i + \epsilon_i$$

and our objective

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

The solution to this was arrived as

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

Bias and variance If we had sampled many different sub-sets of samples and did linear regression, say for M times, what we should have obtained? Indeed we should have obtained M different estimates of the regression coefficients $\mathbf{w}_i$ and errors J_i for $i = 1, \ldots, M$. Note that we often are limited by a small number of samples to work with.

- Variance is the amount by which estimate of the solution/function will change if a different training sub-set was employed for training.
- Bias is known as the assumptions made by a model or designer to make the target function simpler to learn. Low bias suggests less assumptions about the form of the target function. High-bias suggests more assumptions about the form of the target function.

We are interested in errors from bias and variance. The bias is an error from erroneous assumptions in the learning algorithm. It simply means how far away is our estimated values from actual values. The variance is an error from sensitivity to small fluctuations in the training set. It is a measure of spread or variations in our predictions. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

To build models that can generalize well and reduce errors on the test data (unseen data), we need to improve the simple regression we learned.

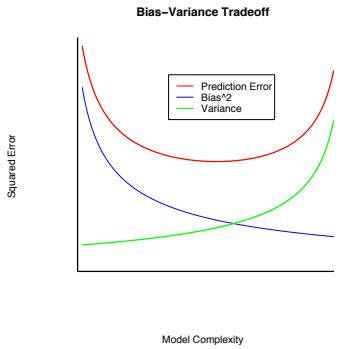

Figure 7.2: Bias-Variance View

Ridge Regression In ridge regression, we improve the objective as

$$J = \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x})^2 + \lambda \sum_{j=1}^{d} w_j^2$$

$$= \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x})^2 + \lambda ||\mathbf{w}||_2^2.$$

What happens when λ is very low (say zero) and very high (say infinity)? When λ is zero, this reduces to the normal MSE based regression and when λ is infinity, $\mathbf{w}$ reduces to zero. Both these extreme cases need to be the best. We need to find the best λ for our data/problem. (See figure). This is often done by cross validation.

Let us come back to ridge regression objective. It can be done as:

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

Optimization of the above objective function leads to:

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

- The bias increases as λ (amount of shrinkage) increase.
- The variance decreases as λ increases

(See figure)

Lasso Regression Lasso is another variant of the regression where the objective has $L1$ norm of the $\mathbf{w}$ instead of $L2$.

$$J = \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x})^2 + \lambda \sum_{i=1}^{d} |w_j|$$

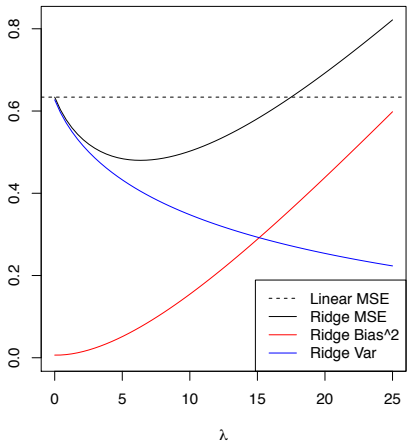

Figure 7.3: Bias-Variance View of Ridge Regression

$$= \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x})^2 + \lambda ||\mathbf{w}||_1.$$

In Lasso, we not only penalize the high value of some of the coefficients, optimization also leads to zero for irrelevant variables.

The change in the norm of the penalty may seem like only a minor difference, however the behavior of the $L1$-norm is significantly different than that of the $l2$-norm.

7.41 Challenges with Data Size

7.41.1 Incremental Setting and Large Data Setting

In the last sections, we saw the problem of regression with Mean Square Error (MSE) Loss/Objective getting solved directly with a closed form solution as:

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

what all influence the computational complexity of this solution?

- When N is very large? Do we have to use all the data?
  - Stochastic Versions
  - Work on subsets that on full data at a time
- When N is large and computational capabilities are limited
  - Iterative solution scheme → "Gradient Discent"
- When all the data can not be stored/available
  - Online variants (see home works)

These issues, lead to the settings like:

Large Data Sets Where does the number of samples N come into the picture?

Incremental Setting Another situation of importance is when we get newer samples on a regular basis.