University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 2: August 30

Lecturer: Alexander Schwing Scribe: Yanli Qian

1 Lecture Overview

The lecture introduced the topic on linear regression. It gave an overview of basic linear regression model and its cost function, as well as the least squares method for minimize the cost. The basic model was further extended to solve high dimension, regularization, and higher order polynomials problem. The aforementioned cases are solved from error view. There exists an alternative probabilistic view with Gaussian distribution. More mathematical computations were involved to demonstrate the consistency of these two views. Finally, the lecture discussed the binary classification problem and the infrequency of using linear regression in this case.

2 Linear Regression Model

2.1 Introduction

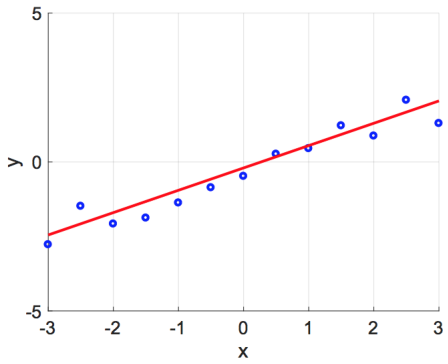
Linear regression is a linear approach for real value prediction. The idea behind is to best fit a dependent variable with respect to a set of independent covariates (or explanatory variables) in a straight line. In other words, it models the relationship between two continuous variables by using a linear equation, and insures that there is the least amount of space between ground truth (e.g. observation data) and the point on linear regression line.

2.2 Simple Linear Regression Model

Suppose we are given a training sample dataset of N pairs (x, y): $\mathfrak{D} = (x^i, y^i)_{i=1}^N$, $x^i, y^i \in \mathbb{R}$, where x^i is the input feature, and y^i is the ground truth. The subscript i means the index into the dataset. We assume a simple linear model with parameters $w_1 \in \mathbb{R}$ and $w_2 \in \mathbb{R}$ so that most of the data points are aligned close to the line. The model can be written as:

$$y = w_1 \cdot x + w_2 \tag{1}$$

And visualize as:



Although we could have more complicated functions to model the relationship, linear function is the simplest model that many complex models are based on.

Now, having the training dataset and the linear model, we want to make the best prediction so that the predicted values are close enough to the ground truth. **Loss function** or **cost function** is used to measure the bias between the predicted value and the actual dataset. Let's define **residual** as subtracting output from the ground truth:

$$r = y^{(i)} - \hat{y}$$

We introduce the **square loss** by squaring the residual. There is another way of taking the absolute value of residual, known as **least absolute deviation** to account for the loss. The reason we are using the square loss here is because the error gets amplified by squaring, so the model will be more sensitive to an outlier. The resultant function is known as the **mean square error (MSE)** or **quadratic loss**:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - w_1 \cdot x^{(i)} - w_2)^2 \tag{2}$$

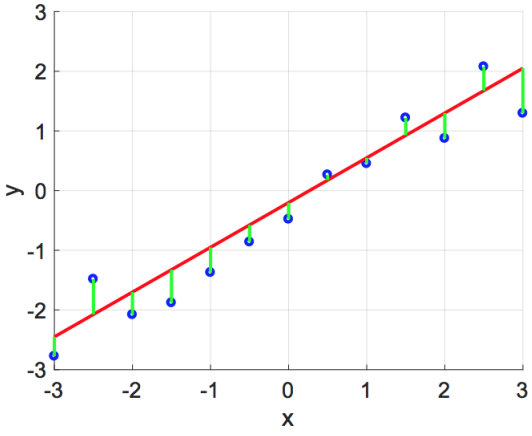
$*\frac{1}{2}$ is used only as mathematics convention.

To minimize the cost, we need to find the proper parameters w_1 and w_2 that satisfy:

$$\underset{w_1,w_2}{argmin}\frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - w_1 \cdot x^{(i)} - w_2)^2 \tag{3}$$

The method is so called **least squares**.

The short green lines in the following figure represent the error, which is the distance between the observed data and the corresponding point on the straight line. It is parallel to y-axis rather than perpendicular to the line.



2.3 Vector Annotation

Least squares estimates the line slope w_1 and the y-intercept w_2. For computational convenience, We can trun the formula (3) into vector form:

$$\underset{w_1,w_2}{argmin}\frac{1}{2}\left\|\begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} - \begin{bmatrix} x^1 & 1 \\ \vdots & \vdots \\ x^N & 1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}\right\|_2^2$$

and simplify it as:

$$\underset{\boldsymbol{w}}{argmin}\ \underbrace{\frac{1}{2}\left\|\boldsymbol{Y}-\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w}\right\|_2^2}_{\text{cost/loss function}} \tag{4}$$

There are two options to solve the $\boldsymbol{w}$ in (5). We take the derivative with respect to $\boldsymbol{w}$ of the cost function, either set it to zero or do the gradient descent. These two ways depend on the number of data points and the dimension. Here we simply set it to zero and find out the optimal solution. Derivative:

$$\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w}^* - \boldsymbol{X}\boldsymbol{Y} = 0$$

Solution:

$$\boldsymbol{w}^* = \left(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\right)^{-1}\boldsymbol{X}\boldsymbol{Y} \tag{5}$$

2.4 More Extensions

2.4.1 Higher Dimensional Problems

Simple linear regression models a set of two-dimensional datapoints, which maps one independent variable to one dependent variable. Suppose that x is no longer real value, but some complex data (e.g. high dimension vector). We have $\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^N$, ($\boldsymbol{x}^i \in \mathbb{R}^d$), and the corresponding $y^1, y^2, ..., y^N$, ($y^i \in \mathbb{R}$). The number of w value increases to $d+1$. The simple linear model now turns to:

$$y^{(i)} = w_0 + \sum_{k=1}^{d}(\boldsymbol{x}_k^{(i)} w_k)$$

Let $\boldsymbol{Y} \in \mathbb{R}^N$, $\boldsymbol{X}^{\mathrm{T}} \in \mathbb{R}^{N\times(d+1)}$, and $\boldsymbol{w} \in \mathbb{R}^{d+1}$, the corresponding vector form for optimizing the solution is:

$$\underset{\boldsymbol{w}}{argmin}\frac{1}{2}\left\|\boldsymbol{Y}-\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w}\right\|_2^2 \tag{6}$$

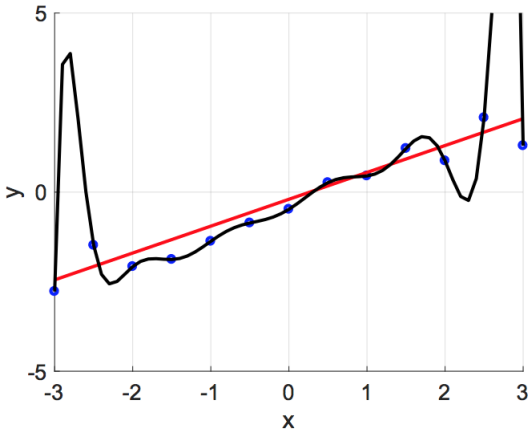
Compared (4) and (6), it is obviously to see that except for the dimension change of $\boldsymbol{x}$, they are same. The solution to the above formula has the same form as (5):

$$\boldsymbol{w}^* = \left(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\right)^{-1}\boldsymbol{X}\boldsymbol{Y} \tag{7}$$

This solution is valid when there exists the inverse of $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$. The number of data points N, in this case, has to be $N > d+1$.

2.4.2 Regularization

In the higher dimensional problems, we assume that $N > d+1$, where d is the input dimension/feature. However, if there is the case when $N < d+1$, the matrix can no longer be inverted. With more dimensions or features added, the model may fit the training data very well but fail to predict new data. It encounters the problem of **overfitting** when detail and noise are learnt during the training step. Thus, lower the performance of data fitting. This is showing as the black line in the figure below:



There are several ways to avoid overfitting:

- Add more data points
- Reduce the number of dimensions
- Cross-validation
- Regularization

Regularization is a technique to prevent overfitting by introducing an additional terms. It will keep all dimensions but reduce the magnitude of the parameters $\boldsymbol{w}$. A multiple of weighted $\boldsymbol{w}$ is added to penalize the cost function. The model becomes:

$$\underset{\boldsymbol{w}}{argmin} \underbrace{\frac{1}{2}\left\|\boldsymbol{Y}-\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w}\right\|_2^2+\frac{C}{2}\left\|\boldsymbol{w}\right\|_2^2}_{\text{cost function}} \tag{8}$$

C is the hyper parameter, a regularization constant which controls the trade-off between whether we would like to fit the training data well (e.g. large C) or keep the parameters as small as possible (e.g. small C).

We can still use the least squares algorithm to optimize the cost function, and get the result of the optimal $\boldsymbol{w}$:

$$\boldsymbol{w}^* = (\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} + C\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{Y} \tag{9}$$

2.4.3 Higher Order Polynomials

The models we talk so far are all first order polynomials in one or more variables. Sometimes, a plot of data points may distributed in a nonlinear relationship so we can have higher order polynomials

regression to account for this problems. Such a model for $x^i \in \mathbb{R}, y^i \in \mathbb{R}$, is:

$$y^{(i)} = w_2 \cdot (x^{(i)})^2 + w_1 \cdot x^{(i)} + w_0$$

The least squares in matrix form:

$$arg \min_{w_0,w_1,w_2} \frac{1}{2} \left\| \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} - \begin{bmatrix} (x^1)^2 & x^1 & 1 \\ \vdots & \vdots & \vdots \\ (x^N)^2 & x^N & 1 \end{bmatrix} \cdot \begin{bmatrix} w_2 \\ w_1 \\ w_0 \end{bmatrix} \right\|_2^2$$

In which:

$$\underbrace{\begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix}}_{\boldsymbol{Y} \in \mathbb{R}^N} \quad \underbrace{\begin{bmatrix} (x^1)^2 & x^1 & 1 \\ \vdots & \vdots & \vdots \\ (x^N)^2 & x^N & 1 \end{bmatrix}}_{\phi^T \in \mathbb{R}^{N \times M}} \quad \underbrace{\begin{bmatrix} w_2 \\ w_1 \\ w_0 \end{bmatrix}}_{\boldsymbol{w} \in \mathbb{R}^M}$$

The cost function in vector form is:

$$J(\theta) = \frac{1}{2} \left\| \boldsymbol{Y} - \phi^T \cdot \boldsymbol{w} \right\|_2^2 \tag{10}$$

Optimal solution $\boldsymbol{w}^*$ for the above cost function is:

$$\boldsymbol{w}^* = (\phi \phi^T)^{-1} \phi \, \boldsymbol{Y} \tag{11}$$

2.4.4 Generalized Model

In general, suppose we have:

- $x^{(i)}$ is some data
- $\theta(x^{(i)}) \in \mathbb{R}^M$ is a transformation into a feature vector

we can generate the model of y:

$$y^{(i)} = \phi(x^{(i)})^T \boldsymbol{w}$$

and use the least squares to find the optimum:

$$\underset{\boldsymbol{w}}{argmin} \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - \phi(x^{(i)})^T \boldsymbol{w})^2$$

$$\boldsymbol{w}^* = (\boldsymbol{\phi} \boldsymbol{\phi}^T)^{-1} \boldsymbol{\phi} \, \boldsymbol{Y} \quad \text{where} \quad \boldsymbol{\phi} = [\phi(x^1), \ldots, \phi(x^N)] \in \mathbb{R}^{M \times N}$$

2.5 Probabilistic View of Linear Regression

From the error view of linear regression, we use the least squares method to optimize the model. In this part, we are going to view from probabilistic perspective to illustrate that the least squares algorithm is a reasonable choice.

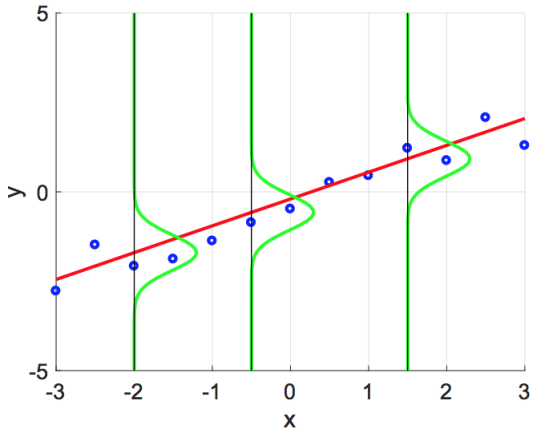
Given the dataset $\mathfrak{D} = \left(x^{(i)}, y^{(i)}\right)_{i=1}^{N}$, and suppose that the error (difference between the predicted value and the observed point) distributed i.i.d(independently and identically distributed). We can say that the error has a zero mean **Gaussian distribution** with variance σ^2. Then we observe that $y^{(i)}$ at point $x^{(i)}$ follows the Gaussian distribution $(\boldsymbol{w}^T x^{(i)}, \sigma^2)$. The distribution of $y^{(i)}$ can be written as:

$$y^{(i)}|x^{(i)}; \boldsymbol{w} \sim N(\boldsymbol{w}^T x^{(i)}, \sigma^2)$$

The probability of $y^{(i)}$ given $x^{(i)}$ and parameterized by $\boldsymbol{w}$ is:

$$p(y^{(i)}|x^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{1}{2\sigma^2}\big(y^i - \boldsymbol{w}^T\phi(x^{(i)})\big)^2\Big)$$

and visualize it as:



We can estimate $\boldsymbol{w}$ using the principal of **Maximum Likelihood**. The function is:

$$p(\mathfrak{D}) = \prod_{(x^{(i)},y^{(i)})\in\mathfrak{D}} p(y^{(i)}|x^{(i)}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\Big(-\frac{1}{2\sigma^2}\left\|\boldsymbol{Y} - \boldsymbol{\phi}^T\boldsymbol{w}\right\|_2^2\Big)$$

Instead of maximizing the entire likelihood function, we can maximize the increasing exponential part. Moreover, without changing the monotonicity of function, it is much easier to do it with taking the log likelihood:

$$\begin{aligned}
\log p(\mathfrak{D}) &= \log \prod_{(x^{(i)},y^{(i)})\in\mathfrak{D}} p(y^{(i)}|x^{(i)}) \\
&= \log \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\Big(-\frac{1}{2\sigma^2}\left\|\boldsymbol{Y} - \boldsymbol{\phi}^T\boldsymbol{w}\right\|_2^2\Big) \\
&= \log \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} - \frac{1}{2\sigma^2}\left\|\boldsymbol{Y} - \boldsymbol{\phi}^T\boldsymbol{w}\right\|_2^2
\end{aligned}$$

Since we want to find $arg\ \min_{\boldsymbol{w}} p(\mathfrak{D})$, we take the derivative of the log likelihood. The negative sign within the exponential turns the $arg\ max$ to $arg\ min$. Maximizing $\log p(\mathfrak{D})$ gives:

$$arg\ \max_{\boldsymbol{w}} p(\mathfrak{D}) = arg\ \min_{\boldsymbol{w}} \frac{1}{2} \left\| \boldsymbol{Y} - \boldsymbol{\phi}^T \boldsymbol{w} \right\|_2^2 \tag{12}$$

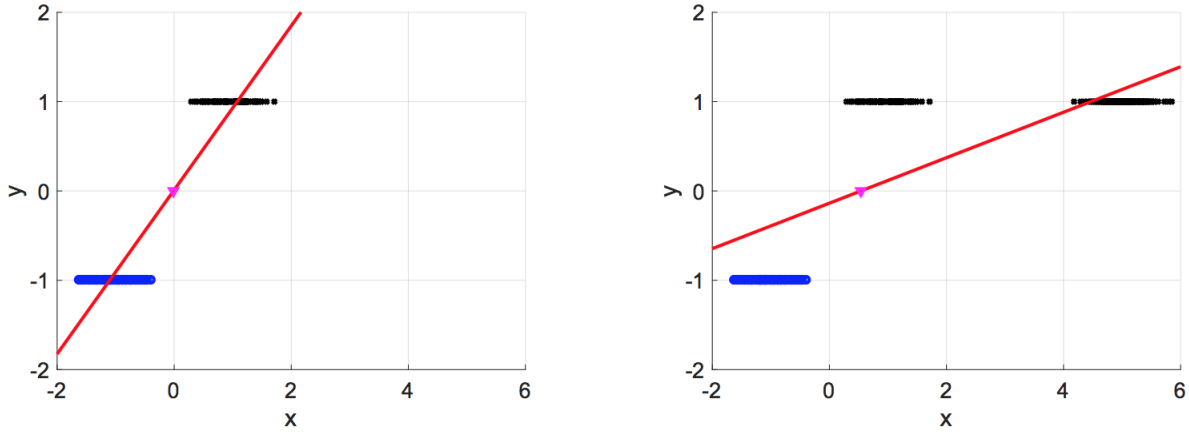
It is obviously to see that (12) has the same form as (4). Therefore, maximizing the log likelihood gives the same result as the minimizing the square loss by applying least squares method.

3 Linear Regression for Classification

In classification problems, we want to classify a set of sample x into categories y, in which y takes a limited number of discrete values. Consider a **binary classification** where y is either -1 or 1 ($y^i \in \{-1, 1\}$), rather than a continuous quantity. We can borrow the idea from linear regression by modeling y as:

$$y = w_1 x + w_0$$

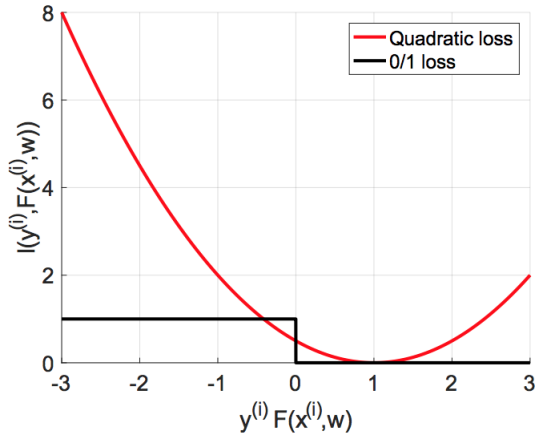
and set the threshold at $y = 0$



The left-hand side figure shows a perfect classification where the majority of data are classified as them ought to be. If another sample with larger x value is added and we run linear regression again. We can learn from the right-had side figure that the decision boundary shifts and the hypnosis based on linear model does not work anymore.
Doing classification is somehow related to optimize misclassifications, since you only care about whether the model predicts the right class or not. On the contrast, doing regression underlines the importance of minimizing the distortion between the predict value and the ground truth. We will use **quadratic loss** as the common penalty function for regression.
In this case when $y^{(i)} \in \{-1, 1\}$, the quadratic loss function can be written as:

$$\begin{aligned} l\big(y_i, \phi(x^{(i)})^T \boldsymbol{w}\big) = & \quad \frac{1}{2}\Big(y^{(i)} - \phi(x^{(i)})^T \boldsymbol{w}\Big)^2 \\ \overset{(y^{(i)})^2=1}{=} & \quad \frac{1}{2}\Big(1 - y^{(i)} \underbrace{\phi(x^{(i)})^T \boldsymbol{w}}_{F(x^{(i)}, \boldsymbol{w})}\Big)^2 \\ & \underbrace{\qquad\qquad\qquad}_{F(x^{(i)},\, \boldsymbol{w},\, y^{(i)})} \end{aligned}$$

$y^{(i)}F(x^{(i)}, \boldsymbol{w})$ implies We can also plot the quadratic loss vs. 0/1 loss



From the plot, we notice that we pay a high price when the $y^{(i)}F(x^{(i)}, \boldsymbol{w})$ value is extremely large, and we cannot restrict the loss value between 0 and 1. Therefore, results the line shift in the linear model.
It is hard to optimize classification error directly. To fix this issue, We have logistic regression, which will be introduced in the next lecture, to work for the classification problem.

4 Applications

In real life, linear Regression can be applied to:

- Stock trading
- Sports betting
- Flight time prediction

5 Summary

- Linear regression predicts a continues variable (e.g. temperatures, prices, etc.)
- Simple linear regression model gives:

$$y = w_1 \cdot x + w_2, \quad x, y \text{ have to real values}$$

- Linear regression optimizes the quadratic cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - w_1 \cdot x^{(i)} - w_2)^2$$

- There are two ways for finding the optimum:
  - Error view: least squares method
  - Probabilistic view: Maximum likelihood estimation
- Linear regression is not suitable for classification (e.g. discrete value) problem

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambrigde, Massachusetts, 2012.

[3] A. Ng. Cs229 lecture notes. url: http://cs229.stanford.edu/notes/cs229-notes1.pdf.

[4] S. Russel. Machine learning methodology: Overfitting, regularization, and all that. url: https://people.eecs.berkeley.edu/ russell/classes/cs194/f11/lectures/CS194

[5] R. Shukla. L1 vs. l2 loss function. url: http://rishy.github.io/ml/2015/07/28/l1-vs-l2-loss/.