Regression, Least Squares

Burak Bayramli

Abstract — **The purpose of this paper is to communicate the author's knowledge on machine learning field, a subject on which he desires to complete a doctorate degree. In this text, we will go over the purposes, motivation, some history and methods of machine learning while trying to make the case for the statistical perspective and graphical model approach.**

1 Introduction

Machine Learning lets us derive short representations for raw data when we do not have a clear mathematical model for the domain we are interested in. Once we acquire a model using ML methods, we can also use ML applications like classification, regression, clustering, anomaly detection, feature selection, all tools which fall under the umbrella of ML.

Throughout this paper, vector $\mathbf{x}$ will be used to define input data for one dimension, and $\mathbf{X}$ for many dimensions. All elements of vector $\mathbf{x}$ should be assumed as data points, measurements for a single attribute, and should be assumed i.i.d unless stated otherwise.

2 Regression Based Methods

2.1 The Model

A suitable starting point for a machine learning introduction is linear regression. Given data $\mathbf{x}$, we decide to model target function f(x) as a line whose slope and starting point are yet unknown.

$$f(x;\theta) = \sum_{i=1}^{N} \theta_1 x_i + \theta_0 \tag{1}$$

where θ_0 and θ_1 represent unknown variables. We have to search through many possible $f(x;\theta)$'s to find an optimal θ.

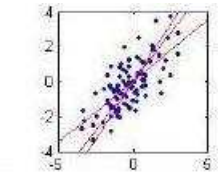

Figure 1: Linear Regression.

First we convert (1) into matrix form so that linear algebra notation is used. Through linear algebra, we will later be able to expand the equation into many dimensions easily.

$$f(x;\theta) = \sum_{i=1}^{N} \begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \tag{2}$$

$$f(\mathbf{x};\theta) = \begin{bmatrix} 1 & x_1 \\ \vdots & \\ 1 & x_N \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \tag{3}$$

Notice the sum in (2) disappeared in (3) and is replaced by added dimensions in the first matrix. Further introduction of linear algebra notation simplifies the equation even further:

$$f(\mathbf{X};\theta) = \mathbf{X}\theta \tag{4}$$

Now for our search, we need to compare the calculated $f(x;\theta)$'s against a given $\mathbf{y}$. This comparison is done utilizing a loss function which defines our success/fail criteria. Let's say we choose sum of square distances between calculated $f(x;\theta)$'s and given $\mathbf{y}$ as our loss function. Based on this loss function, we can calculate overall risk as sum of square distances.

$$R(\theta) = \frac{1}{2N}|\mathbf{y} - \mathbf{X}\theta|^2 \tag{5}$$

Note: We are multiplying with $\frac{1}{2}$ so that it cancels out with ^{2} after derivation. It turns out multiplying with a scalar does not change the location of the minima.

In ML literature, the function $R(\theta)$ is called empirical risk. We will aim to minimize R, which at the same time means we will be searching for the best θ.

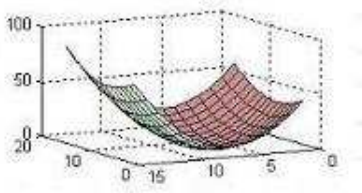

Figure 2: Risk Function in 2D.

The point where risk is lowest is where the rate of change at any point is zero, and the gradient

of $R(\theta)$ equals to zero. Gradient contains all partial derivatives of $R(\theta)$ in vector form. Taking the gradient of a function produces a vector which always points to the direction of largest increase. At the point where the increase is zero, it means we reached a flat region where we have no increase nor decrease.

In cases when taking the gradient results in an algebraically messy formula, approximation methods are used to iteratively *move* in the direction of fastest increase, trying to reach the region where gradient equals zero. In the current case, the algebra will come out clean, as we shall see below.

$$\begin{aligned}
\bigtriangledown_\theta R\left(\theta\right) &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\bigtriangledown_\theta \left(\frac{1}{2N}||\mathbf{y}-\mathbf{X}\theta||^2\right) &= 0 \\
\frac{1}{2N}\bigtriangledown_\theta \left(\left(\mathbf{y}-\mathbf{X}\theta\right)^T\left(\mathbf{y}-\mathbf{X}\theta\right)\right) &= 0 \\
\frac{1}{2N}\bigtriangledown_\theta \left(\left(\mathbf{y}^T-\left(\mathbf{X}\theta\right)^T\right)\left(\mathbf{y}-\mathbf{X}\theta\right)\right) &= 0 \\
\frac{1}{2N}\bigtriangledown_\theta \left(\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{X}\theta-\left(\mathbf{X}\theta\right)^T\mathbf{y}+\theta^T\mathbf{X}^T\mathbf{X}\theta\right) &= 0
\end{aligned}$$

The second and third term are really the same because for vector multiplication, $\mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u}$.

$$\begin{aligned}
\frac{1}{2N}\bigtriangledown_\theta \left(\mathbf{y}^T\mathbf{y}-2\mathbf{y}^T\mathbf{X}\theta+\theta^T\mathbf{X}^T\mathbf{X}\theta\right) &= 0 \\
\frac{1}{2N}\left(-2\mathbf{y}^T\mathbf{X}+2\theta\mathbf{X}^T\mathbf{X}\right) &= 0 \\
\frac{1}{N}\left(-\mathbf{y}^T\mathbf{X}+\theta\mathbf{X}^T\mathbf{X}\right) &= 0 \\
\theta\mathbf{X}^T\mathbf{X} &= \mathbf{y}^T\mathbf{X} \\
\theta &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{y}^T\mathbf{X} \qquad (6)
\end{aligned}$$

Once θ is calculated, it can be plugged in $f(x;\theta)$ to predict future data.

2.2 Multiple Dimensions

Linear regression can be easily extended to multiple dimensions.

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & \dots & x_1(D) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & \dots & x_N(D) \end{bmatrix}$$

$$R\left(\theta\right) = \frac{1}{2N}\left|\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \mathbf{X}\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_D \end{bmatrix}\right|^2$$

This risk calculation is for N data points and D dimensions. Calculation for θ can still be performed through Equation (6). The result will be a N-dimensional hyperplane fit to our training data.

2.3 Basis Functions

We also may choose to model the training data through so-called basis functions. With this method, a polynomial, sinusiodal, or radial basis function is selected, then every data point $\mathbf{x}_{i\in N}$ is fed through the basis function calculating a new $\mathbf{x}_i$. Then regression becomes an act of finding the optimal combination of N basis functions. Radial basis functions are particularly popular, as they allow learning in many dimensions.

Calculating a radial basis function means inserting a "bump" for every data point. The bump is shown as

$$\phi_i(\mathbf{x}) = exp\left(-\frac{1}{2\sigma}\left|\mathbf{x}-\mathbf{x}_i\right|^2\right)$$

where $\mathbf{x}$ can be in one or many dimensions. $\mathbf{x}$ here represents one data point in many dimensions unlike previously, when it represented all data points in one vector.

The width of the bump is controlled through σ which is held constant throughout the calculations. Risk then becomes

$$R(\theta) = \frac{1}{2N}|\mathbf{y}-Q\theta|^2 \qquad (7)$$

where Q is

$$\begin{bmatrix} 1 & \phi_0(x_1) & \dots & \phi_D(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \phi_0(x_N) & \dots & \phi_D(x_N) \end{bmatrix}$$

We can still use (6) to perform the regression.

2.4 Problems With Regression

A critical issue with regression is the absence of a firm mathematical foundation for choosing a modeling method for each problem. As we have seen, the loss function was pulled out of a hat. The loss function could easily have been based on the cube of distances instead of squares. Certain approaches tried to address this issue by dividing the training data in two parts, one part used for training, the other to measure the performance of the training, hence evaluating each model. Ideally however, we would like to use one modeling technique for diverse set of problems.

Using regression for classification also poses certain problems. In figure 3, we see that ideal classification would be the dashed line.

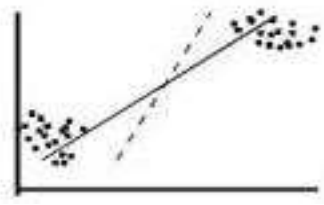

Figure 3: Classification as Regression

However, the points that are furthest to the right and left are being penalized the most by the algorithm, so the regression naturally wants to pull the line towards the furthest points. The result is the solid line shown. An engineering solution for this is using a squashing function triying to smooth out the regression line. This reverses the penalization effect for the larger points. A popular squashing function is the sigmoid function $g(z) = (1 + exp(-z))^{-1}$ and sign function used for building perceptrons.

2.5 Neural Networks

With the introduction of squashing functions regression becomes more usable. On top of this, introducing several layers of regression each utilizing different squashing function lets us handle more complex classification problems as seen in figure 4. This approach is also called a Neural Network.

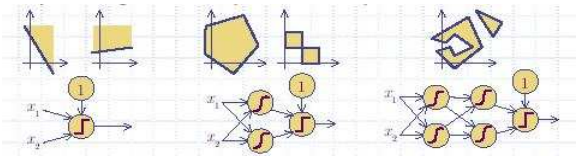

Figure 4: Different Layers for a Neural Network.

With the introduction of squashing function in the network, finding the minima using the gradient becomes algebraically hard to derive. Therefore, we use an approximation technique such as gradient descent. For a multi-layered neural network, gradient descent is applied layer by layer, and idea which forms the basis of the BACKPROP algorithm. BACKPROP has been used successfully for many problems over the years.

Despite the added power of neural networks, certain limitation still remain. For one, NNs are difficult to analyze. As we add a new layer to the network, the consequences of this action on the network cannot be determined. There are also problems that are still impossible to solve using NNs, such as learning the equation $x^2 + y^2 = 1$, which is not a function but a *relation*. There is also the infamous Projectile Cannon problem which is a case of distal learning, and is also impossible to learn using a regression based approach. Problems such as these are much easily addressed using a statistical approach, such as Bayesian Networks.

3 Statistical Approach

3.1 Introduction

Statistical approach allows us to capture more about the training data. In two dimensions, when we do regression, we learn only a slope. With statistical approach when we fit a Gaussian distribution, we learn the center of data distribution μ (mean), a width σ (variance), and a rotation of the ellipse Σ (covariance).

Statistical approach also allows us to use all the tools of probabily theory and statistics. This means the foundation of the model is rigourous, and the applications will have to revert less to randomly applied solutions. This will later help in analysing and improving our model as well.

3.2 The Method

With probability models, instead of learning an exact value for a variable, we may choose to learn a distribution on the same variable. A y would be replaced by $p(y)$. In comparison, using y instead of $p(y)$ is akin to using a distribution with all its weight on one single point.

Introducing more unknown variables into distribution results in a joint distribution. In discrete case, the joint distribution table size is a finite quantity.

Dependence and independence between random variables play a factor in joint distribution table size. If two events, therefore their random variables are related, it is more accurate for our model to reflect this dependence. If for simplication's sake however, all variables were made dependent, the result will be an exponential blow up of the joint distribution table. As an example, a distribution over D unknown binary random variables all dependent on eachother results a table of size 2^D. In comparison, if we make all variables as independent, the size drops to $2D$.

Perhaps we can start with $2D$ and slowly introduce dependencies. Dependency between variables are represented with $p(y|x)$ which reads "the probability of y given x". The joint distribution is

$p(x, y) = p(y|x)p(x)$ and is a well known equality in probability theory. Since writing down all dependencies algebraicly might be confusing, graph notation is also used. The joint distribution $p(x, y, z)$ seen in figure 5 will be equal to product of all probabilities, $p(x)p(y|x)p(z|x)$.

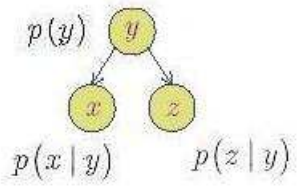

Figure 5: Graphical Model.

Notice $p(x, y, z)$ does not equal $p(x)p(y)p(z)$ according to this model. That would only be the case if all variables were independent.

When dealing with probability distributions, we can also represent model variables as random variables with other variables depending on them. As an example, let's take the problem of classifying points in figure 6.

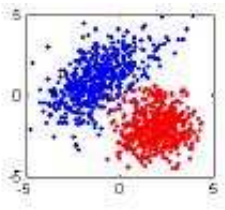

Figure 6: Sample Data.

In the example, $x \in \mathbf{R}^2$ is input and $y \in \{0, 1\}$ is the output. Since y is binary, we can model $p(y)$ as a Bernoulli distribution,

$$p(y_i|\alpha) = \alpha_i^y (1 - \alpha)^{1-y_i} \tag{8}$$

where $i = 1...N$. Using a distribution instead of a table allows us to manipulate the distribution algebraically, plug it in to methods such as Maximum Likelihood, take its derivative and so on.

For x's, we will pick a Gaussian distribution

$$p(x_i|y_i, \mu, \Sigma) = N(x_i|\mu_y, \Sigma_y) \tag{9}$$

The graphical model for the data will look like in figure 7.

We see that μ, α, Σ are also made random variables. Looking at the graph, the algebraic joint distribution $p(x, y, \mu, \alpha, \Sigma)$ can be shown as

$$p(y|\alpha)p(x|y, \mu, \Sigma)p(\Sigma)p(\mu)p(\alpha) \tag{10}$$

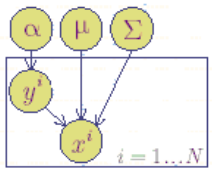

Figure 7: Model.

For our classification needs, we need to turn Equation (10) into a version that answers $p(x, y|\alpha, \mu, \Sigma)$. From pure probability theory, we proceed algebraically

$$p(x, y|\alpha, \mu, \Sigma) = \frac{p(x, y, \alpha, \mu, \Sigma)}{p(\alpha)p(\mu)p(\Sigma)} \tag{11}$$

Now we replace the nominator in (11) with the joint distribution taken from the graphical model in (10).

$$\begin{aligned} p(x, y|\alpha, \mu, \Sigma) &= \frac{p(y|x)p(x|y, \mu, \Sigma)p(\alpha)p(\mu)p(\Sigma)}{p(\alpha)p(\mu)p(\Sigma)} \\ &= p(y|x)p(x|y, \mu, \Sigma) \end{aligned} \tag{12}$$

3.3 Maximum Likelihood

In order to find the parameter which are most likely to fit traning data, we will use a method called Maximum Likelihood. This requires taking the product of (12) for N data points. The best estimation for the unknown parameters are at the point when this final super joint probability is at its heightest value.

$$Likelihood = \prod_{i=1}^{N} p(y|x)p(x|y, \mu, \Sigma) \tag{13}$$

The point when likelihood is heightest is when the derivative of likelihood equals to zero.

In order to simply the derivative later, we take the log of terms inside (13). This will turn the product into a sum. We can do this because the log of the formula in (13) has the same rate of change, therefore the final result will not be effected.

$$l = \sum_{i=1}^{N} log(p(y_i|x_i)) + \sum_{i=1}^{N} log(p(x_i|y_i, \mu, \Sigma)) \tag{14}$$

We can break apart the last term into two sums. We need to marginalize over y in two chunks, one for each possible value of y.

$$l = \sum_{i=1}^{N} log(p(y_i|\alpha)) + \sum_{y_i \in 0} log(p(x_i|\mu_0, \Sigma_0)) + \sum_{y_i \in 1} log(p(x_i|\mu_1, \Sigma_1)) \quad (15)$$

Now, if we take the gradient of the likelihood according to α, Σ, μ (seperately) and set these equations equal to zero, we get equations for most likely parameter values. Let's do $\frac{\partial l}{\partial \alpha}$ as an example. Because we are taking derivative with respect to α, all the terms of (15) go away except for the first term. Then we have

$$\begin{aligned} \frac{\partial l}{\partial \alpha} &= \frac{\partial}{\partial \alpha} \sum_{i=1}^{N} log(p(y_i|\alpha)) \\ 0 &= \frac{\partial}{\partial \alpha} \sum_{i=1}^{N} log\left(\alpha_i^y (1-\alpha)^{1-y_i}\right) \\ 0 &= \frac{\partial}{\partial \alpha} \sum_{i=1}^{N} y_i log(\alpha) + (1-y_i) log(1-\alpha) \end{aligned}$$

From the problem domain, we know that some y_i's are 0, some are 1. Zero values would cancel out $y_i log(\alpha)$, one values would cancel out the $(1 - y_i) log(1 - \alpha)$. So, in order to drop y_i's from the equation, we change the equation to

$$\begin{aligned} 0 &= \frac{\partial}{\partial \alpha} \left(\sum_{i \in class1} log(\alpha) + \sum_{i \in class0} log(1-\alpha) \right) \\ 0 &= \sum_{i \in class1} \frac{1}{\alpha} - \sum_{i \in class0} \frac{1}{1-\alpha} \end{aligned}$$

Now, let N_0 be the quantity of y_i's in class 0, N_1 the quantity of y_i's in class 1.

$$0 = \frac{N_1}{\alpha} - \frac{N_0}{1-\alpha}$$
$$\alpha = \frac{N_1}{N_1 + N_0} = \frac{N_1}{N}$$

This result supports the intuition that dividing the number of y_i's such that $y_i = 1$ in the training set with the total number of traning points results in the value for α. We could guessed this, but it's nice to see verification for it through Maximum Likelihood mathematically.

The rest of the unknowns, namely $\mu_0, \mu_1, \Sigma_0, \Sigma_1$ can be derived using the same idea. Taking the derivative of (15) with respect the unknowns one by one, and equating the results to zero will give us formulas for each unknown. The derivation will not be shown here due to lack of space. The final result will give all the unknown parameters of Equation (9). Using (9) in conjunction with $p(y)$ we derived earlier, we can finally use (12) for classification. We should note here that in ML literature, $p(y)$ is called *prior guess*.

3.4 Classification

For classification we need one final equation, namely $p(y|x)$ which is known as posterior in ML terminology. With the posterior known, when we are given a new x we have not seen before, we can plug it into such as $p(y|x = newvalue)$ and calculate the likelihood of the new y.

Let's derive the posterior $p(y|x)$. From probability theory,

$$p(y|x) = \frac{p(x,y)}{p(x)} \quad (16)$$

We need $p(x)$ for this to work. Let's take $p(x, y)$ and marginalize over y.

$$\begin{aligned} p(y|x) &= \frac{p(x,y)}{\sum_y p(x,y)} \quad (17) \\ &= \frac{p(x,y)}{p(x,y=0) + p(x,y=1)} \quad (18) \end{aligned}$$

Now the act of classification becomes calculating $p(y = 1|x)$, and $p(y = 0|x)$ seperately for two possible values of y. Whichever probability value has the higher value, will be our classification answer for the new data point x.

4 Acknowledgements

All figures in this paper with the exception of figure 3 are taken from Prof. Tony Jebara's Machine Learning lecture notes at `http://www.cs.columbia.edu/ jebara/4771`. A list of Machine Learning literature I own is presented below.

References

[1] C. Bishop *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] R. O. Duda, P. E. Hart & D. G. Stork *Pattern Classification (2nd ed)*, Wiley, 2000.

[3] M. Jordan, C. Bishop, *Introduction to Graphical Models*, not yet published.

[4] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.