# 1 Linear Regression

## 1.1 Residual and Regular

### 1.1.1 Residual

For given data set $x$ and $y$, we try to find the relation between them therefore make use of our research, which means:

$$D = \{x_i, y_i\}_{i=1}^{N} \qquad find \quad y = f(x)$$

For the simplest case, which is linear regression, we want to find vector w, so that,

$$y = w^T x = \begin{pmatrix} w_0 & w_1 & ... & w_D \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ ... \\ x_D \end{pmatrix} = w_0 + \sum_{j=1}^{D} w_j x_j$$

where vector $w$ is the goal of the process, known as Regression Coefficient, while D stands for the number of dimensions. Note that $w_0$, the Bias, is specifically listed out so that the constant term is not ignored. The Goal, however, is to let the error of the approximation be as little as possible, although we cannot be absolutely sure how close is our model to the real world. Anyway, it is still a practicle model if the real world does not change rapidly and randomly, and if the error of the model under known data set is minimized. Therefore, we have the method of minimizing ERROR. We take the ERROR simply to be the difference between the model and the real data.

$$r = y - \hat{y}$$

where $r$ stands for residual, $y$ stands for real data, $\hat{y}$ stands for estimated data. They are all vectors. Note that in order to prevent the positive and negative value from canceling each other, normally we take 2 kinds of adjustment.

$$r_{L_1} = |y - \hat{y}| \qquad r_{L_2} = (y - \hat{y})^2$$

There, we want the sum of the squared value of the residual to be minimized.

$$RSS = \sum_{i=1}^{N} r_i$$

And we take the $r_i$ that seems fit to us.

### 1.1.2 Probability Explaination of Residual

Here, we are trying to find out why is it the right thing to do that we try to minimize the residual.

**L2 Case** Take:

$$RSS(w) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

If the data has a certain noise:

$$\epsilon \sim N(0,\sigma^2) \quad y = f(x)+\epsilon \qquad y|_{x=x_i} \sim N(f(x_i),\sigma^2)$$

Maximum likelihood estimation:

$$\ln(\prod_{i=1}^{N} p_{y_i|x_i}(y|x=x_i)) = -\frac{N}{2}\ln(2\pi) - N\ln\sigma - \sum_{i=1}^{N}\frac{(y_i - f(x_i))^2}{2\sigma^2}$$

As we can see in the function above, to take the maximun of RHS is to take the minimun of the L2 residual of the model. Therefore, these two are equivalent.

**L1 Case** Take:

$$RSS(w) = \sum_{i=1}^{N} L(y_i,\hat{y_i}) = \sum_{i=1}^{N} |y_i - f(x_i)|$$

If the data has a certain noise:

$$\epsilon \sim Laplace(0,b) \quad y = f(x)+\epsilon \qquad y_{i|x=x_i} \sim Laplace(f(x_i),b) \sim p_{y|x=x_i}(y|x=x_i) = \frac{1}{2b}e^{-\frac{|y-f(x_i)|}{b}}$$

Maximum likelihood estimation:

$$\ln(\prod_{i=1}^{N} p_{y_i|x_i}(y|x=x_i)) = -\ln(2b) - \sum_{i=1}^{N}\frac{|y_i - f(x_i)|}{b}$$

**Differeces between L1 and L2** L1 method is not sensitive to noise while L2 method is very sensitive to noise, which means noise data has a larger weight in L2 model. However, L1 function has a major disadvantage, which is its incontinuity, therefore finding its minimun value can be hard. In order to combine the ups and eliminate the downs of the two models, we have the Huber method.

$$L_\delta(r) = \begin{cases} \frac{1}{2}r^2 & |r| \le \delta \\ \delta|r| - \frac{1}{2}\delta^2 & others \end{cases}$$

### 1.1.3 Over Regression

For given data set, we tend to fit our model into the data set as much as possible. Thus in some cases, we manage to let the model fit every bit of the data set, or nearly achieve that. It is not hard for us to do so, for the simpliest method of polynomial regression. If we have a data set with the amount of $n$, we simply need to construct a model holding $n$th power. However, even if the model seems to fit well enough, under the real circumstance, it will not fit, for every data set has its particularity and we should not try to maximize that in our model. Therefore, we want to hold the training error to be small and also hold the testing error to be small. The case of training error is very small and testing error is large is called Over Regression, meaning that the model is too fit for the training data thus too unfit for the real world.

### 1.1.4 Regular Term

In order to prevent over regression from happening, we have the method of adding a regular term. $R(w)$ is the Regular Term.

**L2 Loss + L2 Regular: Ridge**

$$J(w,\lambda)=\sum_{i=0}^{N} r^2+\lambda\sum_{j=1}^{N} w_j^2 \qquad where \quad R(w)=\lambda\sum_{j=1}^{N} w_j^2$$

The process of constructing the model becomes finding the minimun value of $J$. Note that we do not add punishment AKA regulation upon the Bias term and $j$ starts at 1. It is important for us to choose an appropriate $\lambda$. It is better to nondimensionalize the data set upon all dimensions. Same thing holds for the others.

**L2 Loss + L1 Regular: Lasso**

$$R(w)=\sum_{j=1}^{N}|w_j|$$

**L2 Loss + Both Regular: Elastic Net**

$$R(w)=\sum_{j=1}^{N}(\rho|w_j|+\frac{1-\rho}{2}w_j^2)$$

#### 1.1.5 Probability Explaination of Regular

## 1.2 Analytic Solution of OLS

**OLS Formular**

$$J(w)=\sum_{i=1}^{N}(y_i-f(x_i))^2$$

**Goal**

$$\hat{w}=argmin_w J(w)$$

**Requirement**

$$\frac{\partial J(w)}{\partial w}=0$$

**Normal Equation**

$$J(w)=||y-Xw||_2^2=(y-Xw)^T(y-Xw)=y^Ty-y^TXw-w^TX^Ty+w^TX^TXw$$
$$\frac{\partial J(w)}{\partial w}=-2X^Ty+2X^TXw=0 \qquad X^TXw=X^Ty$$
$$\hat{w}_{OLS}=(X^TX)^{-1}X^Ty$$

## 1.3 Gradient Descent Method(GD)

**Why do We Need It?** The analytical method takes running time of $O(N^2D)$, for matrix $M\times D$. Sometimes it is too slow to use the analytical method, as well as the memory of the computer might not be enough for a single set of data.

**How Do We Do It?** The Gradient Descent Method is simply to calculate the gradient of the function at a initial point so that the fastest descending direction of the function can be found. Then let the function go towards that direction and take the gradient again, so and so forth. We continue iterating the process until the function value is close enough to the last result.

Normally, we take the minmun value of the function, and if we want a max of the function, we simply take the min of its negative version. Therefore, we have the following steps.

fundamental equation:

$$x_{t+1} = x + \Delta x = x_t - \eta \nabla f(x) \qquad for \quad \eta \geq 0$$

**under the OLS case:**

$$J(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2 = ||y - Xw||_2^2 = (y - Xw)^T (y - Xw)$$

gradient:

$$\nabla J(w) = -2X^T y + 2X^T Xw = -2X^T (y - Xw)$$

plug into the iteration:

$$w_{t+1} = w_t - \eta \nabla J(w_t) = w_t + 2\eta X^T (y - Xw_t)$$

in which $(y - Xw_t)$ is the residua of the prediction.

With the equations above, we substitude 0 or a small random data set into the equations as initial values, then iteratively calculate the $w$ according to a certain learning rate $\eta$, until it satisfies the breaking condition. The breaking condition can be:

1. Iterating time meets the maximun value;

2.

$$\frac{J(w_t) - J(w_{t+1})}{J(w_t)} \leq \epsilon; \tag{1}$$

There, we have the full process of simple Gradient Discent Method. Mind that the choice of the learning rate is important. If learning rate is too large, in the end, there might be oscillation in the result.

**Ridge Regression**

$$J(w) = ||y - Xw||_2^2 + \lambda ||w||_2^2$$
$$\nabla J(w) = -2X^T y + 2X^T Xw + 2\lambda w$$

**Stochastic Gradient Descent(SGD)** When the sample is too complex or the gradients upon all dimensions are too alike, it is not so appropriate that we still apply the simple GD method to the sample. Therefore, we now apply the Stochastic GD method.We simply take index t by some approach, usually randomly.

$$\nabla J(w_t) = \nabla L(y_t, f(x_t; w_t)) + \lambda \nabla R(w_t)$$

## 1.4 Coordinate Descent Method

### 1.4.1 Subderivative

In some cases like Lasso, when absolute value is involved and the function is not continuous at some points, the original concept of derivative and maximizing function is not valid. At the incontinuous point, there could be many lines which satisfy the feature of tangency, which is that around the certain point, the line does not cross the function as well as has a single intersection point with the function. There can be a group of lines here, whose slope make up the set of the subderivative. According to the defination, the subderivative is noted as $\partial f(x_0)$, also:

$$\partial f(x_0) \in [a, b] \qquad where \quad a = \lim_{x \to x_0^-} \frac{f(x) - f(x_0)}{x - x_0}, b = \lim_{x \to x_0^+} \frac{f(x) - f(x_0)}{x - x_0}$$

The upper and lower bound of subderivative is determined by the left and right limit of the derivative. For example, take $f(x) = |x|$.

$$\partial f(x) = \begin{cases} -1 & \text{(x ¡ 0)} \\ [-1, 1] & \text{(x = 0)} \\ 1 & \text{(x ¿ 0)} \end{cases}$$

Obviously, if the function is continuous, the subderivative set contains only its derivative. Similarly, we define Subgradient for multivariable functions. If 0 is included in the Subgradient set, we have the critical point.

There, we can apply the gradient descent method to incontinuous functions, changing the ending condition into the set is small enough and includes 0. The gradient here is not always descending, therefore, we take all of the calulated $f(x_t)$ and take their minimun.

### 1.4.2 Coordinate Descent Method

Instead of taking all of the dimensions into consideration at the same time, this time, we only take one dimension each time and find the minimun value. Here, we take the example of Lasso.

$$J(w) = ||y - Xw||_2^2 + \lambda ||w||_1$$

For the jth dimension:

$$\begin{aligned}
\frac{\partial}{\partial w_j} RSS(w) &= \frac{\partial}{\partial w_j} \sum_{i=1}^{N} (y_i - (w_{-j}^T x_{i,-j} + w_j x_{ij}))^2 \\
&= -2 \sum_{i=1}^{N} (y_i - w_{-j}^T x_{i,-j} - w_i x_{ij}) x_{ij} \\
&= 2 \sum_{i=1}^{N} x_{ij}^2 w_j - 2 \sum_{i=1}^{N} (y_i - w_{-j}^T x_{i,-j)}) x_{ij} \\
&= a_j w_j - c_j
\end{aligned}$$

## 1.5 Evaluation

### 1.5.1 Rooted Mean Square Error(RMSE)

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

### 1.5.2 Mean Absolute Error(MAE)

$$MAE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

### 1.5.3 Median Absolute Error(MedAE)

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y}_1|, ..., |y_N - \hat{y}_N|)$$

### 1.5.4 Mean Squared Logarithmic Error(MSLE)

$$MSLE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\ln(1 + y_i) - \ln(1 + \hat{y}_i)^2$$

### 1.5.5 $R^2$ Score

$$SS_{res}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, SS_{tot}(y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$
$$R^2(y, \hat{y}) = 1 - \frac{SS_{res}(y, \hat{y})}{SS_{tot}(y)}$$

### 1.5.6 Methods

Ideally, it can be better if we have a certain data sample to build a model and have extra data sample to examine it. However, we only have one data sample in the real world, therefore we have to divide it into training data and testing data. Here, we introduce a method called Cross Validation. Simply, we divide the sample into N parts and try N times with each time having a different part as the testing data and the rest as training data.

$$\text{For the } k\text{th time} : e_k = e(y_{test_k}, f(X_{test\_k}; \hat{w}_k; \lambda))$$
$$\text{Gather the result for certain } \lambda : e_\lambda = \frac{1}{K}\sum_{k=1}^{K} e_k$$

Furthermore, it is better that we try different $\lambda$ and get a fitter result.

**Bootstrap** For the Cross Validation mentioned above, we are taking one peice of data sample each time and make sure the same data will not be taken again. Here, we change our way of taking subsamples and take randomly $\frac{1}{N}$ of the sample each time, regardless of whether the data has been taken before. Therefore, after N times of taking subsamples, the probability of a certain peice of data having not been taken for a single time is $(1 - \frac{1}{N})^N$, which goes to 0.368, which means only around 63.2% of the sample is taken with Bootstrap.

## 1.6 Exploratory Data Analysis(EDA)

### 1.6.1 Basic Information

By the time we start, there are some basic things we need to figure out.

1. Dimension of the sample
2. Amount of the sample
3. Physical meaning of each dimension
4. Type of each dimension, categorical features or discrete features or numeric feature

### 1.6.2 Behavior of Each Dimension

1. Display basic statistical magnitude, mean, variance, quantiles.
2. A glimpse of distributions of each dimension by a histogram. Most of the numeric features are usually normally distrubuted. If not, it might be that some data out of a certain range is counted into the lower and upper bound, thus both ends might behave unordinarily large, therefore might be dumped.
3. A glimpse of correlations between dimensions. Build the correlation matrix with heat map. For the dimensions that are strongly linearly correlated perceived from the matrix, print out the scatter diagram of the two dimensions.

## 1.7 Feature Engineering(FE)

Delete the data that is obviously out of range, where the $3\sigma$ rulemight be applied.

Saparate the sample into the input and output parts. It might be atempted that we apply some function, log, exponential, power, etc, upon the output sample.

Encode the discrete featured data, where-hot Encoding is generally favored. For the discrete featured data having k possible outcomes, we expand it into k dimensions of discrete featured data having only 0 or 1 as outcome.

Standardization.

$$X_{standard} = \frac{X - \mu}{\sigma}$$

from sklearn.preprocessing import StandardScaler

A StandardScaler class must be created in advance. Then call fit$_t ransformwithinputdatathatrequiredstandardizati$

Save the result of FE.