# Regression

## Linear Regression

### 1. Form

$$Y = \mathbf{X}\beta + \epsilon$$

where $Y = (y_1, y_2, \ldots, y_n)^T$, $\beta = (\beta_1, \beta_2, \ldots, \beta_p)^T$ and $\mathbf{X}$ is the data matrix with an extra column of ones on the left to account for the intercept.

### 2. Assumptions

- Linear relationship
- Multivariate normality + Independence + Constant variance

$$\epsilon \overset{iid}{\sim} N(0, \sigma^2)$$

| Assumptions | Diagnosis | Solutions |
| --- | --- | --- |
| Linearity | Scatter plot | 1. Apply a nonlinear transformation; 2. Try a nonlinear form to fit |
| Normality | QQ plot of residuals/non-parametric tests(KS or AD test) | Box-cox transformation on dependent variable |
| Independence | Residual vs time/Durbin Watson test/VIF | 1. Time series model like ARCH, ARMA or ARIMA; 2. Ridge Regression/Lasso Regression |
| Constant Variance | Residual vs predicted values | Log transformation |

*Ref: [http://people.duke.edu/~rnau/testing.htm](http://people.duke.edu/~rnau/testing.htm)*

### 3. Loss Function & Estimation

$$RSS = \Sigma_{i=1}^{n}(y_i - X_i\beta)^2$$

The goal is to minimize RSS. The mean square estimation of $\beta$ is:

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

This only exists when $\mathbf{X}^T X$ is invertible. This requires $n \geq p$ (because if a matrix is invertible, it should be full rank).

- What if $n \leq p$ ?

  We could introduce **L1 regulariation** to solve the problem. Then the estimator would become

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

*Ref: https://zhuanlan.zhihu.com/p/44612139*

- In practise: it would take long time to solve the inverse for a matrix with high dimension, therefore, we always use gradient descent to get the estimator in practise.

**4. Goodness of Fit**

- F-test: test whether a group of variables is important
- T-test: test whether a variable is important
- Variable selection:
  - Forward: start from a null model, include variables one at a time, minimize the RSS at each step.
  - Backward: start from a full model, eliminate variables one at a time, choosing the one with the largest t-test p-value at each step.
  - Mixed: start from a null model, include variables one at a time, minimizing the RSS at each step. If the p-value for some variables goes beyond a threshold, eliminate that variable.
- Model selection: AIC/BIC
- R square: $corr^2(Y, \hat{Y})$ always increases as we add more variables.
- RSE: residual standard error does not always imporve with more predictors:

$$RSE = \sqrt{\frac{1}{n-p-1}RSS}$$

- MSE:

$$MSE = \frac{1}{n}RSS$$

**5. Additionals**

- How to encode dummy variables?

  Different ways of encoding would bring different interpretations for parameters. In order to get corresponding results, we have to carefully encode our dummy variables.

- How to solve overfitting problem when variables are too many ?
  - regularization
  - variable selection
  - dimension reduction (feature extraction)

# Classification

## Logistic Regression

Logistic regression is a method of classification which means its dependent variable should be qualitive.

**1. Form**

$$y = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

where $y = P(y = 1|\mathbf{x})$. The form could be also written as

$$ln\frac{p}{1-p} = w^T\mathbf{x} + b$$

we call $\frac{p}{1-p}$ as **odds** and $ln\frac{p}{1-p}$ as **logit**.

**2. Loss Function & Estimation**

$$Loss = \frac{1}{m}\Sigma_{i=1}^{m} - y_i log(\hat{y}_i) - (1-y_i)log(1-\hat{y}_i)$$

Here we use logarithmic loss function (a.k.a 0-1 classification form of cross entropy) as our loss function. And **minimizing this loss function is actually equivalent to maximizing the log-likelihood**.

- Why don't we use MSE as loss function?

  If we use MSE as loss function, the deriative of the loss function must include the deriative of sigmoid.

  $$loss' = (\hat{y} - y)sigmoid'x$$

  And we know the deriative of sigmoid would be 0 when $x$ is big which would lead to **gradient vanishing** issue. However, logarithmic loss function would not have this problem. The derivative of logarithmic loss function looks as below

  $$loss' = \frac{1}{N}\Sigma_{i=1}^{N} x_i(y_i - p(x_i))$$

  The advantage of logarithmic loss function is that when the predicted value is far away from the truth, graident would become big and then the training process would be speeded up.

- How to estimate parameters?
  - Gradient Descent: first-order deriative

    $$w^{t+1} = w^t - \alpha\frac{\partial L}{\partial w^t}$$

  - Newton's algorithm: second-order deriative

    $$w^{t+1} = w^t - \alpha\frac{L'}{L''}$$

**3. Additionals**

- Is it linear or non-linear classfication?

Logistic regression is a linear classifier because the predicted log-odds could be formed as a linear function of $x$. However, non-linear classfication, such as neural networks, there is no way to summarize the output of a neural network in terms of a linear function of $x$.

- Can it apply to multi-classes problem?

  Yes. As for k-classes problem, we could fit k-1 logistic regression models to classify k-classes.

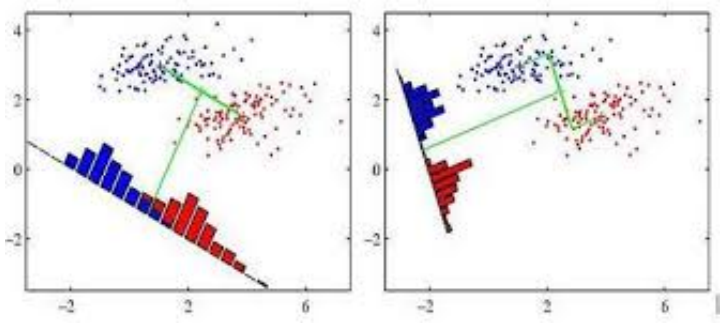
- Can it be used to non-linear problem?

  Yes. Think about Kernel used in SVM. For non-linear problem, we could fit kernel logistic regression.

## Linear Discriminant Analysis (LDA)

**1. Goal**

The goal of LDA (binary case) is that : given dataset, we need to find a line $y = w^T x$ which could make the projection of same class points to be close to each other and conversely the projection of different class points to be as far as possible.



*Another equivalent explanation is: we have to make the covariance of same class points to be as small as possible, in the meanwhile, the center of different classes to be as far as possible. This explaination could be easier to understant if we take the normality assumption of LDA into account.*

**2. Loss Function & Estimation**

Based on the goal of LDA, we could construct a loss function like

$$\begin{aligned} Loss &= \frac{||w^T\mu_0 - w^T\mu_1||_2^2}{w^T\Sigma_0 w + w^T\Sigma_1 w} \\ &= \frac{w^T(\mu_0-\mu_1)(\mu_0-\mu_1)^T w}{w^T(\Sigma_0+\Sigma_1)w} \end{aligned}$$

Here, we could let $S_w = \Sigma_0 + \Sigma_1$ and $S_b = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T$.

According to Lagrange multiplier and the property of vector dot product (dot product is scalar which has no direction), we could get

$$w = S_w^{-1}(\mu_0 - \mu_1)$$

*In practise, considering the stability of results, we usually apply SVD on $S_w$ to get its inverse*.

**3. Bayesian Explanation**

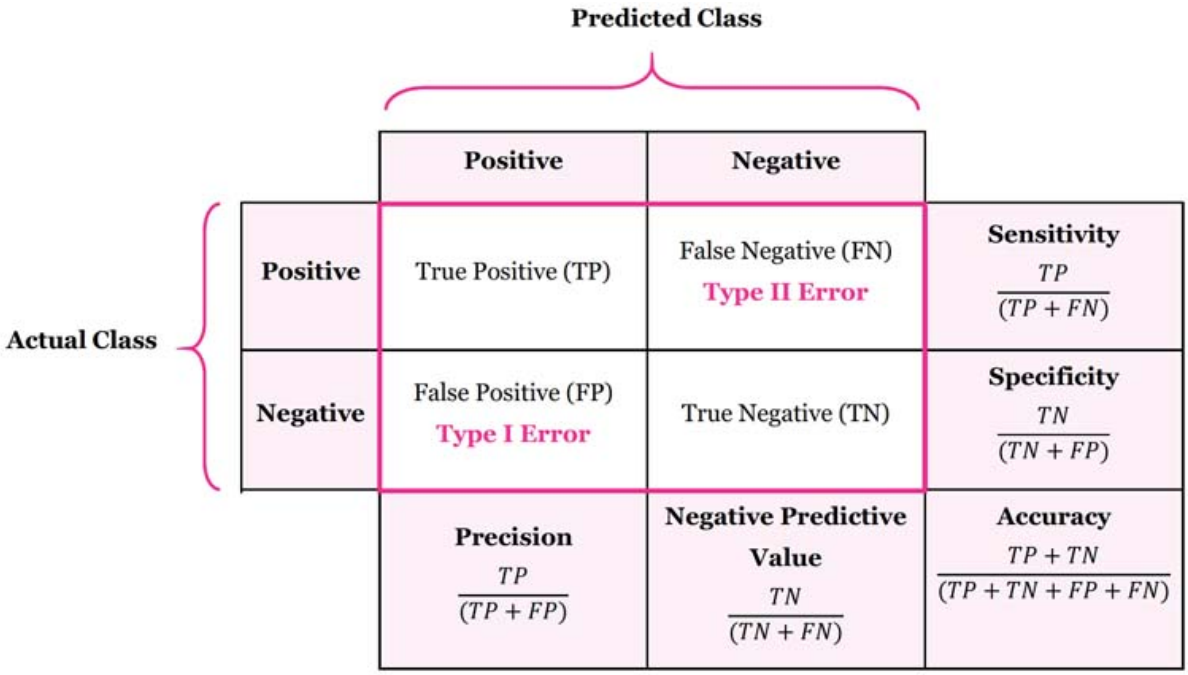
- What we want to know:

$$\hat{P}(Y=k|X=x)=\frac{\hat{P}(X=x|Y=k)\hat{P}(Y=k)}{\Sigma_j\hat{P}(X=x|Y=j)\hat{P}(Y=j)}$$

- What we suppose:
  - Suppose $P(X=x|Y=k)=f_k(x)$ to be a **multivariate normal distribution**.
  - Suppose the convariance matrix is same to all categories.
  - Suppose we know $P(Y=k)=\pi_k$ exactly.
- After expanding RHS of the equation, we could know that LDA has linear decision boundaries.
- How about QDA?
  - QDA is a non-linear classifier.
  - It **does not** require the convariance matrix is same to all categories.

**4. Evaluation**

- Confusion Matrix




  - Type I Error: reject the true null hypothesis.
  - Type II Error: fail to reject the false null hypothesis.

  *Note: threshold is the hidden parameter needs to be careful.*

- ROC curves
  - Display the performance of the method for any choice of threshold. That means, we could use it as a method of choosing best-fit threshold.
  - The area under the curver (AUC) measures the quality of the classifier. The closer AUC is to 1, the better.

**5. Multi-class LDA**

As for N classes, $i-th$ class has $m_i$ samples. We could define the global scatter matrix:

$$
\begin{aligned}
S_t &= S_b + S_w \\
&= \Sigma_{i=1}^{m}(x_i - \mu)(x_i - \mu)^T
\end{aligned}
$$

And we could also define:

$$
S_w = \Sigma_{i=1}^{N}\Sigma_{x \in X_i}(x_i - \mu_i)(x_i - \mu_i)^T
$$

Then we have

$$
\begin{aligned}
S_b &= S_t - S_w \\
&= \Sigma_{i=1}^{N} m_i(\mu_i - \mu)(\mu_i - \mu)^T
\end{aligned}
$$

Similar to binary case, we have to maximize $\frac{tr(W^T S_b W)}{tr(W^T S_w W)}$ . In the end, we have to solve $S_b W = \lambda S_w W$.

The closed-form solution of W is the matrix composed of the top d' eigenvectors having the d largest eigenvalues of $S_w^{-1} S_b$. Here, $d' \leq N - 1$.

*Note: since we could take W as a project matrix and d' is always smaller than the original dimension d, in this way, LDA is also an useful way of dimension reduction.*

## Two Main Issues on Classification

- How to train build a multi-classes classifier?
  - One vs One: N(N-1)/2 binary classifiers + vote
  - One vs Rest: N binary classifiers
  - Many vs Many: Error correcting output codes + compute distance
- Imbalance dataset (ex: 0 dataset is larger than 1)
  - undersampling: eliminate some 0 samples.
  - oversampling: add some 1 samples. For example, SMOTE

    *Note: here we could not simply resample 1s because it could lead to overfitting.*
  - Threshold-moving: changing threshold of binary classifier.