Foundation of Machine Learning
(CSE4032)
Lecture 06: Linear Methods for Classification

Dr. Kundan Kumar
Associate Professor
Department of ECE

Faculty of Engineering (ITER)
S'O'A Deemed to be University, Bhubaneswar, India-751030

Outline

Linear Methods for Classification

There are two big branches of methods for classification.

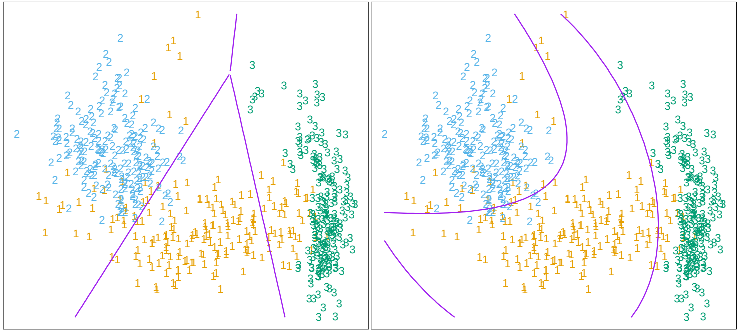
- Generative modeling and
- Discriminative modeling.

Introduction

- Classification is supervised learning for which the true class labels for the data points are given in the training data.
- Setup for supervised learning
  - Training data: $(x_1, g_1), (x_2, g_2), \cdots, (x_N, g_N)$
  - The feature variables $X = (X_1, X_2, \cdots, X_p)$, where each variable X_j is quantitative.
  - The response variable G is categorical. $G \in \{1, 2, \ldots, K\}$
  - Form a predictor $G(x)$ to predict G based on X.
- $G(x)$ divides the input space (feature space) into a collection of regions, each labeled by one class.

Introduction to classification

- See the example partition of the feature space by $G(x)$ in the following plots. For each plot, the space is divided into three pieces, each assigned with a particular class.



Classification Error Rate

- The classification error rate is the number of observations that are misclassified over the sample size.

$$\frac{1}{N}\sum_{i=1}^{N} I\left(\hat{Y}_i \neq y_i\right)$$

where $I\left(\hat{Y}_i \neq y_i\right) = 1$ if $\hat{Y}_i \neq y_i$ and 0 otherwise.

Bayes Classification Rule

- Suppose
  - the marginal distribution of G is specified by the probability mass function (pmf) $p_G(g), \ g = 1, 2, \ldots, K$.
  - The conditional distribution of X given $G = g$ is $f_{X|G}(x \mid G = g)$.
- The training data $\{(x_i, g_i)\}, \ i = 1, 2, \cdots, N$, are independent samples from the joint distribution of X and G,

$$f_{X,G}(x, g) = p_G(g) f_{X|G}(x \mid G = g)$$

Bayes Classification Rule

- Assume the loss of predicting G at $G(X) = \hat{G}$ is $L(\hat{G}, G)$.
- Goal of classification: to minimize the expected loss

$$E_{X,G} L(G(X), G) = E_X \left(E_{G|X} L(G(X), G) \right)$$

- To minimize the left hand side, it suffices to minimize $E_{G|X} L(G(X), G)$ for each X. Hence the optimal predictor:

$$\boxed{\hat{G}(x) = \arg\min_g E_{G|X=x} L(g, G)}$$

- The above decision rule is called the Bayes classification rule.

Bayes Classification Rule

- For $0-1$ loss, i.e.,

$$L(g, g') = \begin{cases} 1 & g \neq g' \\ 0 & g = g' \end{cases}$$

$$\boxed{E_{G|X=x} L(g, G) = 1 - \Pr(G = g \mid X = x)}$$

- The Bayes rule becomes the rule of maximum posterior probability:

$$\begin{aligned} \hat{G}(x) &= \arg\min_{g} E_{G|X=x} L(g, G) \\ &= \arg\max_{g} \Pr(G = g \mid X = x) \end{aligned}$$

Many classification algorithms attempt to estimate $\Pr(G = g \mid X = x)$, then apply the Bayes rule.

Linear Methods for Classification

- Decision boundaries are linear.
- Two class problem:
  - The decision boundary between the two classes is a hyperplane in the feature space.
  - A hyperplane in the p dimensional input space is the set:

$$\left\{ x : \alpha_o + \sum_{j=1}^{p} \alpha_j x_j = 0 \right\}$$

  - The two regions separated by a hyperplane:

$$\left\{ x : \alpha_o + \sum_{j=1}^{p} \alpha_j x_j > 0 \right\} \text{ and } \left\{ x : \alpha_o + \sum_{j=1}^{p} \alpha_j x_j < 0 \right\}$$

Linear Methods for Classification

- More than two classes: The decision boundary between any pair of classes k and l is a hyperplane (shown in previous figure).
- How do you choose the hyperplane?
- Example methods for deciding the hyperplane:
  - Linear discriminant analysis.
  - Logistic regression.
- Linear decision boundaries are not necessarily constrained. We can expand the feature space by adding in extra variables formed by functions of the original variables. Linear decision boundaries in the expanded feature space may be nonlinear in the original feature space.

Linear Discriminant Analysis

Introduction

- Let the feature vector be X and the class labels be Y.
- The Bayes rule says that if you have the joint distribution of X and Y, and if X is given, under $0-1$ loss, the optimal decision on Y is to choose a class with maximum posterior probability given X.
- Discriminant analysis belongs to the branch of classification methods called generative modeling, where we try to estimate the within class density of X given the class label.
- Combined with the prior probability (unconditioned probability) of classes, the posterior probability of Y can be obtained by the Bayes formula.

Bayes Rule

- Assume the prior probability or the marginal probability mass function (pmf) for class k is denoted as π_k, $\sum_{k=1}^{K} \pi_k = 1$.
- π_k is usually estimated simply by empirical frequencies of the training set:

$$\hat{\pi}_k = \frac{\text{\# of Samples in class } k}{\text{Total \# of samples}}$$

- You have the training data set and you count what percentage of data come from a certain class. Then we need the class-conditional density of X. Remember this is the density of X conditioned on the class k, or class $G = k$ denoted by $f_k(x)$.

Bayes Rule

- According to the Bayes rule, what we need is to compute the posterior probability:

$$\Pr(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

  This is a conditional probability of class G given X.

- By MAP (maximum a posteriori, i.e., the Bayes rule for $0-1$ loss):

$$\begin{aligned}\hat{G}(x) &= \arg\max_k \Pr(G = k \mid X = x) \\ &= \arg\max_k f_k(x)\pi_k\end{aligned}$$

- Notice that the denominator is identical no matter what class k you are using. Therefore, for maximization, it does not make a difference in the choice of k. The MAP rule is essentially trying to maximize π_k times $f_k(x)$.

Class Density Estimation

- Depending on which algorithms you use, you end up with different ways of density estimation within every class.
- In Linear Discriminant Analysis (LDA) we assume that every density within each class is a Gaussian distribution.
- Linear and Quadratic Discriminant Analysis: Gaussian densities.
  - In LDA, we assume those Gaussian distributions for different classes share the same covariance structure.
  - In Quadratic Discriminant Analysis (QDA) we don't have such a constraint. You will see the difference later.
- General Nonparametric Density Estimates:
  - You can also use general nonparametric density estimates, for instance kernel estimates and histograms.

Class Density Estimation

- Naive Bayes:
  - Assume each of the class densities are products of marginal densities, that is, all the variables are independent.
- There is a well-known algorithm called the Naive Bayes algorithm. Here the basic assumption is that all the variables are independent given the class label. Therefore, to estimate the class density, you can separately estimate the density for every dimension and then multiply them to get the joint density. This makes the computation much simpler.
- X may be discrete, not continuous. Instead of talking about density, we will use the probability mass function (pmf).
- In this case, we would compute a probability mass function for every dimension and then multiply them to get the joint probability mass function.

Linear Discriminant Analysis

- Under LDA, we assume that the density for $\mathbf{X}$, given every class k, is following a Gaussian distribution.
- Here is the density formula for a multivariate Gaussian distribution:

$$\boxed{f_k(x) = \frac{1}{(2\pi)^{p/2}\left|\Sigma_k\right|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}}$$

  p is the dimension and Σ_k is the covariance matrix. In this case, we are doing matrix multiplication. The vector x and the mean vector μ_k are both column vectors.

- For Linear discriminant analysis (LDA):

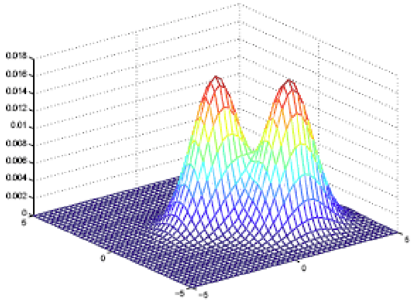
$$\boxed{\Sigma_k = \Sigma, \ \forall k}$$

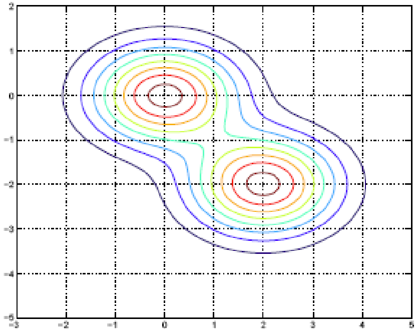
Linear Discriminant Analysis

- In LDA, as we mentioned, you simply assume for different k that the covariance matrix is identical.
- By making this assumption, the classifier becomes linear.
- The only difference from quadratic discriminant analysis is that we do not assume that the covariance matrix is identical for different classes.
- For QDA, the decision boundary is determined by a quadratic function.
- Since the covariance matrix determines the shape of the Gaussian density, in LDA, the Gaussian densities for different classes have the same shape, but are shifted versions of each other (different mean vectors).

Linear Discriminant Analysis

- Example densities for the LDA model are shown below.





Optimal Classifier

- For the moment, we will assume that we already have the covariance matrix for every class. And we will talk about how to estimate this in a moment.
- Let's look at what the optimal classification would be based on the Bayes rule.
- Bayes rule says that we should pick a class that has the maximum posterior probability given the feature vector x. If we are using the generative modeling approach this is equivalent to maximizing the product of the prior and the within class density.
- Since the log function is an increasing function, the maximization is equivalent because whatever gives you the maximum should also give you a maximum under a log function. Next, we plug in the density of the Gaussian distribution assuming common covariance and then multiplying the prior probabilities.

Optimal Classifier

$$\begin{aligned}\hat{G}(x) &= \arg\max_k \Pr(G = k \mid X = x) \\ &= \arg\max_k f_k(x)\pi_k \\ &= \arg\max_k \ln\left(f_k(x)\pi_k\right) \\ &= \arg\max_k \left[-\ln\left((2\pi)^{p/2}|\Sigma|^{1/2}\right) - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \ln(\pi_k)\right] \\ &= \arg\max_k \left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \ln(\pi_k)\right]\end{aligned}$$

NOTE:

$$-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}x^T\Sigma^{-1}x$$

Optimal Classifier

- To sum up, after simplification we obtain this formula:

$$\hat{G}(x) = \arg\max_k \left[x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln(\pi_k) \right]$$

- This is the final classifier. Given any x, you simply plug into this formula and see which k maximizes this.
- Usually the number of classes is pretty small, and very often only two classes. Hence, an exhaustive search over the classes is effective.
- LDA gives you a linear boundary because the quadratic term is dropped.

Optimal Classifier

- To sum up

$$\boxed{\hat{G}(x) = \arg\max_k \left[x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln(\pi_k) \right]}$$

  - Define the linear discriminant function

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \tfrac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln(\pi_k)$$
$$\hat{G}(x) = \arg\max_k \delta_k(x)$$

  - Then, the decision boundary between class k and l is:

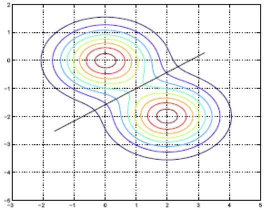
$$\{x : \delta_k(x) = \delta_l(x)\}$$

  - Or equivalently the following holds

$$\ln \frac{\pi_k}{\pi_l} - \frac{1}{2} (\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) + x^T \Sigma^{-1} (\mu_k - \mu_l) = 0$$

Binary Classification

- In binary classification in particular, for instance if we let $(k = 1, l = 2)$, then we would define constant a_0, given below, where π_1 and π_2 are prior probabilities for the two classes and μ_1 and μ_2 are mean vectors.
  - Binary classification $(k = 1, l = 2)$:
  - Define $a_0 = \ln \frac{\pi_1}{\pi_2} - \frac{1}{2}(\mu_1 + \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$
  - Define $(a_1, a_2, \ldots, a_p)^T = \Sigma^{-1}(\mu_1 - \mu_2)$
  - Classify to class 1 if $a_0 + \sum_{j=1}^{p} a_j x_j > 0$; to class 2 otherwise.
  - An example
    - $\pi_1 = \pi_2 = 0.5$
    - $\mu_1 = (0,0)^T, \mu_2 = (2,-2)^T$
    - $\Sigma = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.5625 \end{pmatrix}$
    - Decision boundary: $5.56 - 2.00x_1 + 3.56x_2 = 0.0$



Binary Classification

- We have two classes and we know the within class density.
- The marginal density is simply the weighted sum of the within class densities, where the weights are the prior probabilities.
- Because we have equal weights and because the covariance matrix of two classes are identical, we get these symmetric lines in the contour plot.
- The black diagonal line is the decision boundary for the two classes.
- Basically, if you are given an x above the line, then we would classify this x into the first-class. If it is below the line, we would classify it into the second class.
- Here, we have the prior probabilities for the classes and we also had the within class densities given to us. Of course, in practice you don't have this. In practice, what we have is only a set of training data.
- The question is how do we find the π_k's and the $f_k(x)$?

Estimating the Gaussian Distributions

- Here is the formula for estimating the π_k' s and the parameters in the Gaussian distributions.
- The formula below is actually the maximum likelihood estimator:

$$\hat{\pi}_k = N_k/N$$

where N_k is the number of samples in kth class and N is the total number of points in the training data.
- As we mentioned, to get the prior probabilities for class k, you simply count the frequency of data points in class k.
- Then, the mean vector for every class is also simple. You take all of the data points in a given class and compute the average, the sample mean:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{g_i = k} x^{(i)}$$

Estimating the Gaussian Distributions

- Next, the covariance matrix formula looks slightly complicated. The reason is because we have to get a common covariance matrix for all of the classes.
- First you divide the data points in two given classes according to the given labels.
- If we were looking at class k, for every point, we subtract the corresponding mean which we computed earlier. Then multiply its transpose.
- Remember x is a column vector, therefore if we have a column vector multiplied by a row vector, we get a square matrix, which is what we need.

$$\hat{\Sigma} = \frac{1}{(N-K)} \sum_{k=1}^{K} \sum_{g_i = k} \left(x^{(i)} - \hat{\mu}_k\right) \left(x^{(i)} - \hat{\mu}_k\right)^T$$

Estimating the Gaussian Distributions

- First, we do the summation within every class k, then we have the sum over all of the classes.
- Next, we normalize by the scalar quantity, $N - K$. When we fit a maximum likelihood estimator it should be divided by N, but if it is divided by $N - K$, we get an unbiased estimator.
- Remember, K is the number of classes. So, when N is large, the difference between N and $N - K$ is pretty small.
- Note that $x^{(i)}$ denotes the ith sample vector.
- In summary, if you want to use LDA to obtain a classification rule, the first step would involve estimating the parameters using the formulas above. Once you have these, then go back and find the linear discriminant function and choose a class according to the discriminant functions.

Logistic Regression

Logistic Regression

- Logistic regression for classification is a discriminative modeling approach, where we estimate the posterior probabilities of classes given X directly without assuming the marginal distribution on X.
- It preserves linear classification boundaries.
- A review of the Bayes rule shows that when we use $0-1$ loss, we pick the class k that has the maximum posterior probability:

$$\hat{G}(x) = \arg\max_k \Pr(G = k \mid X = x)$$

- The decision boundary between classes k and l is determined by the equation:

$$\Pr(G = k \mid X = x) = \Pr(G = l \mid X = x)$$

that is the x's at which the two posterior probabilities of k and l are equal.

Logistic Regression

- If we divide both sides by $\Pr(G = l \mid X = x)$ and take the log of this ratio, the above equation is equivalent to:

$$\log \frac{\Pr(G = k \mid X = x)}{\Pr(G = l \mid X = x)} = 0$$

- Since we want to enforce a linear classification boundary, we assume the function above is linear (below):

$$\log \frac{\Pr(G = k \mid X = x)}{\Pr(G = l \mid X = x)} = a_0^{(k,l)} + \sum_{j=1}^{p} a_j^{(k,l)} x_j$$

- This is the basic assumption of logistic regression (simple indeed).

Logistic Regression

- We use the superscript (k, l) on the coefficients of the linear function because for every pair of k and l, the decision boundary would be different, determined by the different coefficients.
- For logistic regression, there are restrictive relations between $a^{(k,l)}$ for different pairs of (k, l). We don't really need to specify this equation for every pair of k and l.
- Instead, we only need to specify it for $K - 1$ such pairs.

Logistic Regression

- Assumptions
  - Let's look at our assumptions. If we take class K as the base class, the assumed equations are:

$$\log \frac{Pr(G=1 \mid X=x)}{Pr(G=K \mid X=x)} = \beta_{10} + \beta_1^T x$$
$$\log \frac{Pr(G=2 \mid X=x)}{Pr(G=K \mid X=x)} = \beta_{20} + \beta_2^T x$$
$$\vdots$$
$$\log \frac{Pr(G=K-1 \mid X=x)}{Pr(G=K \mid X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

  - This indicates that we don't have to specify the decision boundary for every pair of classes.

Logistic Regression

- We only need to specify the decision boundary between class j and the base class K. (You can choose any class as the base class - it doesn't really matter.)
- Once we have specified the parameters for these $K-1$ log ratios, then for any pair of classes (k, l), we can derive the log ratios without introducing new parameters:

$$\log \frac{\Pr(G = k \mid X = x)}{\Pr(G = l \mid X = x)} = \beta_{k0} + \beta_{l0} + (\beta_k - \beta_l)^T x$$

- Number of parameters: $(K-1)(p+1)$
- For convenience, we will denote the entire parameter set by θ and arrange them in this way:

$$\theta = \{\beta_{10}, \beta_1^T, \beta_{20}, \beta_2^T, \ldots, , \beta_{(K-1)0}, \beta_{K-1}^T\}$$

Logistic Regression

- The log ratios of posterior probabilities are called log-odds or logit transformations.
- Under these assumptions, the posterior probabilities are given by the following two equations:

$$\Pr(G=k \mid X=x)=\frac{\exp\left(\beta_{k0}+\beta_k^T x\right)}{1+\sum_{l=1}^{K-1}\exp\left(\beta_{l0}+\beta_l^T x\right)} \text{ for } k=1,\ldots,K-1$$

$$\Pr(G=K \mid X=x)=\frac{1}{1+\sum_{l=1}^{K-1}\exp\left(\beta_{l0}+\beta_l^T x\right)}$$

- For $\Pr(G=k \mid X=x)$ given above, obviously
  - These must sum up to 1: $\sum_{k=1}^{K}\Pr(G=k \mid X=x)=1$
  - A simple calculation shows that the assumptions are satisfied.

Comparison with Linear Regression on Indicators

- Similarities:
  - Both attempt to estimate $\Pr(G = k \mid X = x)$.
  - Both have linear classification boundaries.
  - Posterior probabilities sum to 1 across classes.
- Difference:
  - Linear regression on indicator matrix: approximate $\Pr(G = k \mid X = x)$ by a linear function of x. $\Pr(G = k \mid X = x)$ is not guaranteed to fall between 0 and 1.
  - Logistic regression: $\Pr(G = k \mid X = x)$ is a nonlinear function of x. It is guaranteed to range from 0 to 1 .

Fitting Logistic Regression Models

- How do we estimate the parameters? How do we fit a logistic regression model?
- We need a certain optimization criterion for choosing the parameters.
- Optimization Criterion:
  - What we want to do is to find parameters that maximize the conditional likelihood of class labels G given X using the training data.
  - We are not interested in the distribution of X, instead our focus is on the conditional probabilities of the class labels given X.
  - Given point x_i, the posterior probability for the class to be k is denoted by:

$$p_k(x_i;\theta) = \Pr(G = k \mid X = x_i;\theta)$$

  - Given the first input x_1, the posterior probability of its class, denoted as g_1, is computed by:

$$\Pr(G = g_1 \mid X = x_1).$$

Fitting Logistic Regression Models

- Since samples in the training data set are assumed independent, the posterior probability for the N sample points each having class $g_i, i = 1, 2, \cdots, N$, given their inputs $x_1, x_2, \cdots, x_N$ is:

$$\prod_{i=1}^{N} \Pr(G = g_i \mid X = x_i)$$

- In another word, the joint conditional likelihood is the product of the conditional probabilities of the classes given every data point.
- The conditional log-likelihood of the class labels in the training data set becomes a summation:

$$\ell(\theta) = \sum_{i=1}^{N} \log \Pr(G = g_i \mid X = x_i) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

Binary Classification

- Let's look at binary classification first.
- In fact, when we go to more than two classes the formulas become much more complicated, although they are derived similarly. We will begin by looking at this simpler case.
- For binary classification, if $g_i =$ class 1, denote $y_i = 1$; if $g_i =$ class 2, denote $y_i = 0$. All we are doing here is changing the labels to 1's and 0's so that the notation will be simpler.
  - Let $p_1(x; \theta) = p(x; \theta)$
  - Then $p_2(x; \theta) = 1 - p_1(x; \theta) = 1 - p(x; \theta)$

  because the posterior probabilities of the two classes have to sum up to 1.

Binary Classification

- Since $K = 2$ we only have one linear equation and one decision boundary between two classes, the parameters $\theta = \beta_{10}, \beta_1$, (remember, β_1 is a vector). And, we denote $\beta = (\beta_{10}, \beta_1)^T$ which is a column vector.
- Now, let's try to simplify the equation a little bit.
  - If $y_i = 1$, i.e., $g_i = 1$

$$\begin{aligned} \log p_{g_i}(x;\beta) &= \log p_1(x;\beta) \\ &= 1 \cdot \log p(x;\beta) \\ &= y_i \log p(x;\beta) \end{aligned}$$

  - If $y_i = 0$, i.e., $g_i = 2$

$$\begin{aligned} \log p_{g_i}(x;\beta) &= \log p_2(x;\beta) \\ &= 1 \cdot \log(1 - p(x;\beta)) \\ &= (1 - y_i) \log(1 - p(x;\beta)) \end{aligned}$$

Binary Classification

- Since either $y_i = 0$ or $1 - y_i = 0$, we can add the two (at any time, only one of the two is nonzero) and have:

$$\log p_{g_i}(x;\beta) = y_i \log p(x;\beta) + (1 - y_i) \log(1 - p(x;\beta))$$

- The reason for doing this is that when we later derive the logistic regression we do not want to work with different cases. It would be tedious in notation if we have to distinguish different cases each time.
- This unified equation is always correct for whatever y_i you use.

Binary Classification

- Next, we will plug what we just derived into the log-likelihood, which is simply a summation over all the points:

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{N} \log p_{g_i}(x_i;\beta) \\ &= \sum_{i=1}^{N} \left[y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta))\right]\end{aligned}$$

- There are $p+1$ parameters in $\beta = (\beta_{10}, \beta_1)^T$.
- Assume a column vector form for β :

$$\beta = \begin{bmatrix}\beta_{10} & \beta_{11} & \beta_{12} & \dots & \beta_{1,p}\end{bmatrix}^T$$

Binary Classification

- Here we add the constant term 1 to x to accommodate the intercept and keep this in matrix form.

$$x = [1 \quad x_{,1} \quad x_{,2} \quad , \ldots , x_{,p}]^T$$

- Under the assumption of the logistic regression model:

$$p(x;\beta) = \Pr(G = 1 \mid X = x) = \frac{\exp\left(\beta^T x\right)}{1 + \exp\left(\beta^T x\right)}$$

$$1 - p(x;\beta) = \Pr(G = 2 \mid X = x) = \frac{1}{1 + \exp\left(\beta^T x\right)}$$

- we substitute the above in $\ell(\beta)$:

$$\ell(\beta) = \sum_{i=1}^{N} \left[y_i \beta^T x_i - \log\left(1 + e^{\beta^T x_i}\right)\right]$$

Binary Classification

- The idea here is based on basic calculus. If we want to maximize $\ell(\beta)$, we set the first order partial derivatives of the function $\ell(\beta)$ with respect to β_{1j} to zero.

$$\begin{aligned}\frac{\partial \ell(\beta)}{\beta_{1j}} &= \sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} \frac{x_{ij} e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \\ &= \sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} p(x;\beta) x_{ij} \\ &= \sum_{i=1}^{N} x_{ij} \left(y_i - p\left(x_i;\beta\right)\right)\end{aligned}$$

for all $j = 0, 1, \ldots, p$.

Binary Classification

- And, if we go through the math, we will find out that it is equivalent to the matrix form below.

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i \left(y_i - p\left(x_i; \beta\right)\right)$$

  Here x_i is a column vector and y_i is a scalar.
- To solve the set of $p+1$ nonlinear equations $\frac{\partial \ell(\beta)}{\partial \beta_{1j}} = 0, j = 0, 1, \ldots, p$, we will use the Newton-Raphson algorithm.
- The Newton-Raphson algorithm requires the second-order derivatives or the so-called Hessian matrix (a square matrix of the second order derivatives) which is given by:

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i x_i^T p\left(x_i; \beta\right)\left(1 - p\left(x_i; \beta\right)\right)$$

Binary Classification

- Starting with β^{old} , a single Newton-Raphson update is given by this matrix forumla:

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2 \ell(\beta)}{\partial\beta\partial\beta^T}\right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta}$$

- Basically, if given an old set of parameters, we update the new set of parameters by taking β^{old} minus the inverse of the Hessian matrix times the first order derivative vector. These derivatives are all evaluated at β^{old} .

References

- The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, Hastie, Tibshirani, and Friedman, Springer.
- In Introduction to Statistical Learning with Application in R, Second Edition, James, Witten, Hastie, and Tibshirani, Springer.

Thank you!