Lecture 11: Introduction to Classification

Shukai Gong

From the viewpoint of statistical machine learning, given a set of labeled data $\{\boldsymbol{x}_n \in \mathcal{X}, y_n \in \mathcal{C}\}$, we wish to train a classifier $f : \mathcal{X} \mapsto \mathcal{C}$ that can predict the class label y of a new data point $\boldsymbol{x}$. In this lecture, we will introduce some basic generative or discriminative models for classification.

1 K-Nearest Neighbor Classifier

Suppose we have a set of labeled data $\{\boldsymbol{x}_n, y_n\}$ where y_n has N classes. We can first present y_n as a one-hot vector $\boldsymbol{y}_n$, where the i-th element is 1 and all other elements are 0 when y_n belongs to class i. For a new data point $\boldsymbol{x}$, we can compute its class label $\boldsymbol{y}$ by

$$p(\hat{\boldsymbol{y}}|\boldsymbol{x}) = \frac{1}{|\mathcal{N}_h(\boldsymbol{x})|} \sum_{\boldsymbol{x}_n \in \mathcal{N}_h(\boldsymbol{x})} \boldsymbol{y}_n, \mathcal{N}_h(\boldsymbol{x}) = \text{top-}K(\{\boldsymbol{x}_n : d(\boldsymbol{x}_n, \boldsymbol{x}) \leq h\})$$

Here $\mathcal{N}_h(\boldsymbol{x})$ is the K-nearest neighbor set of $\boldsymbol{x}$, and the probability of $\boldsymbol{x}$ belonging to the i-th class is

$$p(\hat{y}_i|\boldsymbol{x}) = \hat{\boldsymbol{y}}[i]$$

2 Naive Bayes Classifier

Bayesian Classifiers are based on Bayesian decision theory. Suppose that we have N classes $\mathcal{Y} = \{c_1, \cdots, c_N\}$, denote $\lambda_{ij} = \mathbf{I}(i \neq j)$ as the loss of classifying an instance of class c_j as class c_i. The expected loss of classifying an instance of class c_j as class c_i is

$$R(c_i|\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_{ij} P(c_j|\boldsymbol{x}) = 1 - P(c_i|\boldsymbol{x})$$

We wish to find a classifier f that minimizes the expected loss, i.e.

$$f^*(\boldsymbol{x}) = \arg\min_{c_i \in \mathcal{Y}} R(c_i|\boldsymbol{x}) = \arg\max_{c_i \in \mathcal{Y}} P(c_i|\boldsymbol{x})$$

Here f^* is called the Bayes optimal classifier. For each sample $\boldsymbol{x}$, we choose the class c that maximizes the posterior probability $P(c|\boldsymbol{x})$. By Bayes' theorem, we have

$$f^*(\boldsymbol{x}) = \arg\max_{c_i \in \mathcal{Y}} P(c_i|\boldsymbol{x}) = \arg\max_{c_i \in \mathcal{Y}} \frac{P(\boldsymbol{x}|c_i)P(c_i)}{P(\boldsymbol{x})} = \arg\max_{c_i \in \mathcal{Y}} P(\boldsymbol{x}|c_i)P(c_i)$$

Here the prior is $P(c_i)$, the proportion of data points that belong to class c_i in the training set, and the likelihood is $P(\boldsymbol{x}|c_i)$, the joint probability of observing all features of $\boldsymbol{x}$ given that it belongs to class c_i. In the setting of Naive Bayes, we assume **attribute conditional independence, i.e. the features are independent of each other knowing all labels $\mathcal{Y}$.** Denote D_c as the set of data points that belong to class c in training set D, we have

$$f_{\text{NB}}(\boldsymbol{x}) = \arg\max_{c \in \mathcal{Y}} P(\boldsymbol{x}|c)P(c) = \arg\max_{c \in \mathcal{Y}} P(c) \prod_{d=1}^{D} P(x_d|c)$$

$$= \arg\max_{c \in \mathcal{Y}} \frac{|D_c|}{|D|} \prod_{d=1}^{D} P(x_d|c)$$

- Gaussian Naive Bayes: For each $\boldsymbol{x} \in \mathbb{R}^D$, suppose each feature x_d is assumed to be $p(x_d|c) \sim \mathcal{N}(\mu_{cd}, \sigma_{cd}^2)$.

$$P(x_d|c) = \frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x_d - \mu_{cd})^2}{2\sigma_{cd}^2}\right)$$

  By MLE, we can estimate $\hat{\mu}_{cd} = \frac{1}{|D_c|} \sum\limits_{\boldsymbol{x} \in D_c} x_d$, $\hat{\sigma}_{cd}^2 = \frac{1}{|D_c|} \sum_{\boldsymbol{x} \in D_c} (x_d - \hat{\mu}_{cd})^2$.

- Multinomial Naive Bayes: For each $\boldsymbol{x} \in \mathbb{N}^D$, suppose

$$\begin{aligned} P(\boldsymbol{x}|c) &= \frac{n!}{x_1! x_2! \cdots x_D!} \prod_{d=1}^{D} p_{c,d}^{x_d} \\ &= \frac{\Gamma(\sum\limits_{d=1}^{D} x_d + 1)}{\prod\limits_{d=1}^{D} \Gamma(x_d + 1)} \prod_{d=1}^{D} p_{c,d}^{x_d} = \frac{(\sum\limits_{d=1}^{D} x_d)!}{\prod\limits_{d=1}^{D} (x_d)!} \prod_{d=1}^{D} p_{c,d}^{x_d} \end{aligned}$$

  Then

$$P(c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|c)P(c)}{P(\boldsymbol{x})} \propto P(\boldsymbol{x}|c)P(c) \propto P(c) \prod_{d=1}^{D} p_{c,d}^{x_d}$$

$$\Rightarrow \log P(c|\boldsymbol{x}) \propto \log P(c) + \sum_{d=1}^{D} x_d \log p_{c,d} = \log P(c) + [\log p_{c,1}, \cdots, \log p_{c,D}] \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} \triangleq \log P(c) + \boldsymbol{\theta}_c^T \boldsymbol{x}$$

- Bernoulli Naive Bayes: For each $\boldsymbol{x} \in \{0, 1\}^D$, suppose

$$p(\boldsymbol{x}|c) = \prod_{d=1}^{D} p_{c,d}^{x_d} (1 - p_{c,d})^{1 - x_d}$$

3 Linear Discriminant Analysis

As is shown in Figure 1, given a training set $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ where y_i are class labels, we want to find out an optimal line $y = \boldsymbol{w}^\top \boldsymbol{x}$ that maximizes the separation between different classes and minimizes the seperation within classes of data points projected onto $y = \boldsymbol{w}^\top \boldsymbol{x}$. For a new data point $\boldsymbol{x}^*$ to be classified, we project it on the same line and assign it to the class that is closest to it. Essentially, we want to find out the optimal $\boldsymbol{w}$ that seperates the projections of data points from different classes as much as possible.

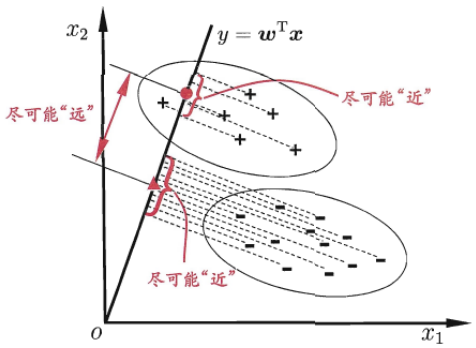

Figure 1: Linear Discriminant Analysis

LDA can both serve as a dimensionality reduction technique and a classifier.

- First, LDA finds a low-dimensional representation of the data (by projection vector $\boldsymbol{w}$ or projection matrix $\boldsymbol{W}$) that maximizes the ratio of between-class covariance to within-class covariance.
- Then, by likelihood ratio test, LDA classifies a new data point $\boldsymbol{x}^*$ by comparing the likelihood of $\boldsymbol{x}^*$ belonging to different classes. For example, if $\log \frac{P(\boldsymbol{x}^*|y^*=c_1)}{P(\boldsymbol{x}^*|y^*=c_0)} > 0$, then $\boldsymbol{x}^*$ is classified as c_1. $P(\boldsymbol{x}^*|y^*=c_i)$ is often modeled as Gaussian.

3.1 Binary Classification Case

Given a dataset $\{\boldsymbol{x}_n, y_n\}_{n=1}^N, y_i \in \{0,1\}$, denote $X_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ as the data points, mean vector and covariance matrix of class i. When all data points are projected onto $y = \boldsymbol{w}^\top \boldsymbol{x}$, the centers of the two classes are projected onto $\boldsymbol{w}^\top \boldsymbol{\mu}_0$ and $\boldsymbol{w}^\top \boldsymbol{\mu}_1$, and the covariance for two classes of projections are $\boldsymbol{w}^\top \boldsymbol{\Sigma}_0 \boldsymbol{w}$ and $\boldsymbol{w}^\top \boldsymbol{\Sigma}_1 \boldsymbol{w}$ respectively.

Maximizing the separation of projections between classes can be achieved by maximizing the distance between the centers of two classes, i.e. $\|\boldsymbol{w}^\top \boldsymbol{\mu}_1 - \boldsymbol{w}^\top \boldsymbol{\mu}_0\|_2^2$, and minimizing the separation of projections within each class can be achieved by minimizing the sum of the covariance matrices of two classes, i.e. $\boldsymbol{w}^\top(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}$. Therefore, the optimization problem can be formulated as

$$\max_{\boldsymbol{w}} J = \max_{\boldsymbol{w}} \frac{\|\boldsymbol{w}^\top \boldsymbol{\mu}_1 - \boldsymbol{w}^\top \boldsymbol{\mu}_0\|_2^2}{\boldsymbol{w}^\top(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}} = \frac{\boldsymbol{w}^\top(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{w}}{\boldsymbol{w}^\top(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}}$$

Before we solve this optimization problem, we first define:

- **Within-class scatter matrix:** $\boldsymbol{S}_w = \boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1 = \sum_{\boldsymbol{x} \in X_0} (\boldsymbol{x} - \boldsymbol{\mu}_0)(\boldsymbol{x} - \boldsymbol{\mu}_0)^\top + \sum_{\boldsymbol{x} \in X_1} (\boldsymbol{x} - \boldsymbol{\mu}_1)(\boldsymbol{x} - \boldsymbol{\mu}_1)^\top$
- **Between-class scatter matrix:** $\boldsymbol{S}_b = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top$

Then our optimization problem can be rewritten as

$$\max_{\boldsymbol{w}} J = \max_{\boldsymbol{w}} \frac{\boldsymbol{w}^\top \boldsymbol{S}_b \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w}}$$

Note that J is a Rayleigh quotient where the denominator and numerator are both quadratic forms of $\boldsymbol{w}$. The solution to this optimization problem is **irrelevant to the magnitude of $\boldsymbol{w}$ but only to its direction.**(i.e. if $\boldsymbol{w}^*$ is a solution, $\lambda \boldsymbol{w}^*$ is also a solution for any $\lambda \neq 0$). Without loss of generality, we can assume $\boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w} = 1$, then the optimization problem can be rewritten as

$$\max_{\boldsymbol{w}} \boldsymbol{w}^\top \boldsymbol{S}_b \boldsymbol{w} \quad \text{s.t.} \quad \boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w} = 1$$

By setting up the Lagrangian, we have

$$\begin{aligned} \mathcal{L} &= \boldsymbol{w}^\top \boldsymbol{S}_b \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w}) \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} &= 2\boldsymbol{S}_b \boldsymbol{w} - 2\lambda \boldsymbol{S}_w \boldsymbol{w} = 0 \\ \Rightarrow \boldsymbol{S}_b \boldsymbol{w} &= \lambda \boldsymbol{S}_w \boldsymbol{w} \end{aligned}$$

Plugging $\boldsymbol{S}_b \boldsymbol{w} = \lambda \boldsymbol{S}_w \boldsymbol{w}$ back into constraint, we have

$$\boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w} = \boldsymbol{w}^\top \lambda \boldsymbol{S}_w \boldsymbol{w} = \lambda$$

So **$\boldsymbol{w}$ is the eigenvector of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ and the corresponding to the largest eigenvalue $\lambda_{\max}$.** From another perspective, note that $\boldsymbol{S}_b \boldsymbol{w} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{w}\right)$ is parallel to $\boldsymbol{\mu_1} - \boldsymbol{\mu}_0$, meaning that

$$\boldsymbol{S}_w \boldsymbol{w} = \frac{1}{\lambda} \boldsymbol{S}_b \boldsymbol{w} \propto (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \Rightarrow \boldsymbol{w} \propto \boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

Since $\boldsymbol{w}$ only indicates the direction, the magnitude of $\boldsymbol{w}$ is unimportant.

3.2 Decision Rule of Binary Classification

Now we will prove that in binary classification task, when the two types of data are Gaussian distributed and homoskedastic, LDA generates the Bayesian optimal classifier. Before introducing the proof, several assumptions are made:

- **Multivariate Normality:** The data points $\boldsymbol{x} \in X_i$ are drawn from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, i.e. $P(\boldsymbol{x}|y=i) = \dfrac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\dfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)$.
- **Homoskedasticity:** $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$ for LDA (No such restriction for QDA).

Since the Bayes optimal classifier is defined as $f^*(\boldsymbol{x}) = \arg\max_{y\in\{0,1\}} P(\boldsymbol{x}|y)P(y)$, essentially we cares about the relative magnitude of $P(\boldsymbol{x}|y=1)P(y=1)$ and $P(\boldsymbol{x}|y=0)P(y=0)$.

$$\log \frac{P(\boldsymbol{x}|y=1)P(y=1)}{P(\boldsymbol{x}|y=0)P(y=0)} = \log \frac{P(\boldsymbol{x}|y=1)}{P(\boldsymbol{x}|y=0)} + \log \frac{P(y=1)}{P(y=0)} = \log \frac{P(\boldsymbol{x}|y=1)}{P(\boldsymbol{x}|y=0)}$$

since the two classes share the same prior probability by assumptions.

$$\begin{aligned}\frac{P(\boldsymbol{x}|y=1)}{P(\boldsymbol{x}|y=0)} &= \frac{\dfrac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\dfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right)}{\dfrac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\dfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)\right)} \\ &= \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)\right)\end{aligned}$$

For LDA, since $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$, we have

$$\begin{aligned}\frac{P(\boldsymbol{x}|y=1)}{P(\boldsymbol{x}|y=0)} &= \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)\right) \\ &= \exp\left(-\frac{1}{2}(\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - 2\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + 2\boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0)\right) \\ &= \exp\left(\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - \frac{1}{2}(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0)\right) \\ &= \exp\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)\right)\end{aligned}$$

Normally we set the threshold for decision to be 0, i.e.

$$y_{\boldsymbol{x}} = 1: \quad \log \frac{P(\boldsymbol{x}|y=1)}{P(\boldsymbol{x}|y=0)} > 0 \iff (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} > \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0) = 0$$

If we denote $\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$, then the decision rule can be rewritten as

$$y_{\boldsymbol{x}} = 1: \quad \boldsymbol{w}^\top \boldsymbol{x} > \boldsymbol{w}^\top \frac{(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)}{2}$$

Surprisingly, the $\boldsymbol{w}$ derived from the Bayesian viewpoint is exactly the $\boldsymbol{w} = \boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ we derived from LDA optimization problem. ($\boldsymbol{S}_w = \boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1 = 2\boldsymbol{\Sigma}$ here, the magnitude difference can be neglected)

3.3 Multiclass Case

Suppose we have a dataset $\{\boldsymbol{x}_n, y_n\}_{n=1}^N, \boldsymbol{x}_i \in \mathbb{R}^D, y_i \in \{1, \cdots, K\}$, and we want to lower the dimensionality of the data points to d' ($d' \leq K-1$). We need to find a d-dimensional hyperplane $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{d'}] \in \mathbb{R}^{D\times d'}$

that maximizes the within-class seperation and minimizes the between-class seperation.

We still inherit the notation of $X_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ from the binary classification case. Denote $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$ as the overall mean vector, and m_i as the number of data points in class i. Similarily, we define

- **Within-class scatter matrix:** $\boldsymbol{S}_w = \sum_{i=1}^{K} \sum_{\boldsymbol{x}\in X_i} (\boldsymbol{x}-\boldsymbol{\mu}_i)(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top$
- **Between-class scatter matrix:** $\boldsymbol{S}_b = \sum_{i=1}^{K} m_i(\boldsymbol{\mu}_i-\boldsymbol{\mu})(\boldsymbol{\mu}_i-\boldsymbol{\mu})^\top$

Our optimization goal is now

$$\begin{aligned}
\max_{\boldsymbol{w}_1,\cdots,\boldsymbol{w}'_d} \frac{\sum_{i=1}^{d'} \boldsymbol{w}_i^\top \boldsymbol{S}_b \boldsymbol{w}_i}{\sum_{i=1}^{d'} \boldsymbol{w}_i^\top \boldsymbol{S}_w \boldsymbol{w}_i} &= \max_{\boldsymbol{W}} \frac{\mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{S}_b \boldsymbol{W})}{\mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{S}_w \boldsymbol{W})} \\
\iff \max_{\boldsymbol{W}} \mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{S}_b \boldsymbol{W}) \quad &\text{s.t.} \quad \mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{S}_w \boldsymbol{W}) = 1
\end{aligned}$$

Consider Lagrangian

$$\begin{aligned}
\mathcal{L} &= \mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{S}_b \boldsymbol{W}) + \lambda(1 - \mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{S}_w \boldsymbol{W})) \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} &= \boldsymbol{W}^\top(\boldsymbol{S}_b + \boldsymbol{S}_b^\top) - \lambda \boldsymbol{W}^\top(\boldsymbol{S}_w + \boldsymbol{S}_w^\top) = 2\boldsymbol{W}^\top(\boldsymbol{S}_b - \lambda \boldsymbol{S}_w) = 0 \\
\Rightarrow \boldsymbol{S}_b \boldsymbol{W} &= \lambda \boldsymbol{S}_w \boldsymbol{W}
\end{aligned}$$

Similarily, we choose $\boldsymbol{W}$ to be the eigenvectors of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ corresponding to the d' largest eigenvalues.

If classification is required besides dimension reduction, there are a number of alternative techniques available.

- **One against the rest:** Train K classifiers, each of which maximizes the separation between one class and the rest.
- **Pairwise classification:** Train $K(K-1)/2$ classifiers, each of which maximizes the separation between two arbitary classes.

4 Logistic Regression

Logistic Regression is a special case of GLM where the link function is sigmoid (logit). Consider a binary classification case where the output labels are $y \in \{0, 1\}$. Given a set of labeled data $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, the LR model is

$$y = \sigma(\boldsymbol{x}^\top \boldsymbol{\beta}) = \frac{1}{1+\exp(-\boldsymbol{x}^\top \boldsymbol{\beta})} \iff \log \frac{y}{1-y} = \boldsymbol{x}^\top \boldsymbol{\beta}$$

The labels y are viewed as posterior probability of label y on data point $\boldsymbol{x}$, i.e. $P(y=1|\boldsymbol{x})$, so that

$$\log \frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})} = \boldsymbol{x}^\top \boldsymbol{\beta} \Rightarrow \begin{cases} P(y=1|\boldsymbol{x}) = \dfrac{1}{1+\exp(-\boldsymbol{x}^\top \boldsymbol{\beta})} = \sigma(\boldsymbol{x}^\top \boldsymbol{\beta}) \\ P(y=0|\boldsymbol{x}) = 1 - P(y=1|\boldsymbol{x}) = 1 - \sigma(\boldsymbol{x}^\top \boldsymbol{\beta}) \end{cases}$$

The distribution of the output labels is Bernoulli, so the log-likelihood function for an arbitary data point $\boldsymbol{x}_i$ is $\log f(y_i|\boldsymbol{x}_i,\boldsymbol{\beta}) = \log\left(P(y_i|\boldsymbol{x}_i)^{y_i}(1-P(y_i|\boldsymbol{x}_i))^{1-y_i}\right) = y_i \log P(y_i|\boldsymbol{x}_i) + (1-y_i)\log(1-P(y_i|\boldsymbol{x}_i))$. The parameter $\boldsymbol{\beta}$ can be estimated by MLE,

$$\begin{aligned}
\boldsymbol{\beta} &= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{N} \log f(y_i|\boldsymbol{x}_i,\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left(y_i \log P(y_i|\boldsymbol{x}_i) + (1-y_i)\log(1-P(y_i|\boldsymbol{x}_i))\right) \\
&= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left(y_i \log \sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta}) + (1-y_i)\log(1-\sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta}))\right) \\
&\triangleq \arg\max_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) \qquad \textbf{(Essentially Cross-Entrophy Loss)}
\end{aligned}$$

There is no closed-form solution to $\boldsymbol{\beta}$, usually we use gradient descent or Newton's method to solve it.

$$\begin{aligned}
&\text{Gradient Descent: } \boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \eta \nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}^{(t)}) \\
&\text{Newton's Method: } \boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \left(\nabla^2_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}^{(t)})\right)^{-1} \nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}^{(t)})
\end{aligned}$$

And the decision rule for LR is

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{x}^\top \boldsymbol{\beta})} > 0.5 \iff \boldsymbol{x}^\top \boldsymbol{\beta} > 0$$

4.1 Softmax Regression

Softmax Regression is a generalization of LR to multiclass classification. Given a set of labeled data $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$ where $\boldsymbol{y}_i \in \{0,1\}^K$ is an one-hot vector. The distribution of the output labels is Multinomial, so the log-likelihood function for an arbitary data point $\boldsymbol{x}_i$ is

$$\log f(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{\beta}) = \log\left(\prod_{k=1}^{K} P(y_{ik}|\boldsymbol{x}_i)^{y_{ik}}\right) = \sum_{k=1}^{K} y_{ik} \log P(y_{ik}|\boldsymbol{x}_i)$$

The cell label y_{ik} within the one-hot vector $\boldsymbol{y}_i$ are still viewed as the posterior probability of label y_{ik} on data point $\boldsymbol{x}_i$, i.e. $P(y_{ik}=1|\boldsymbol{x}_i)$, but **the link function switches from sigmoid to softmax**,

$$P(y_{ik}=1|\boldsymbol{x}_i) = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)}{\sum_{k=1}^{K} \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)} = \text{softmax}(\boldsymbol{x}_i^\top \boldsymbol{\beta})_k$$

By MLE, the parameter $\{\boldsymbol{\beta}_k\}_{k=1}^K$ can be estimated by

$$\{\boldsymbol{\beta}_k\}_{k=1}^K = \arg\max_{\{\boldsymbol{\beta}_k\}_{k=1}^K} \sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log P(y_{ik}=1|\boldsymbol{x}_i,\boldsymbol{\beta}_k)$$

However, the optimal $\{\boldsymbol{\beta}_k\}_{k=1}^K$ is not unique in that softmax is shift-invariant

$$\text{softmax}(\boldsymbol{x}_i^\top \boldsymbol{\beta})_k = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)}{\sum_{k=1}^{K} \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)} = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)}{\sum_{k=1}^{K} \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)} \cdot \frac{\exp(\Delta)}{\exp(\Delta)} = \text{softmax}(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \Delta)_k = \text{softmax}(\boldsymbol{x}_i^\top \boldsymbol{\beta'})_k$$

5 Comparison between LDA and Logistic Regression

	LDA	Logistic Regression
Model	**Generative:** First obtain $p(\boldsymbol{x}\|y)$ explicitly, then determine $p(y\|\boldsymbol{x})$ in the projected space	**Discriminative:** Model $p(y\|\boldsymbol{x})$ directly
Assumption	$p(\boldsymbol{x}\|y=i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$	GLM assumption
Decision Rule	The logarithm of likelihood ratio indicates the confidence of classification implicitly	$p(y\|\boldsymbol{x})$ indicates the confidence of classification directly

Table 1: Comparison between LDA and Logistic Regression

References

- Linear Discriminant Analysis
- Machine Learning, Zhi-Hua Zhou