Classification

Ruwen Qin
Stony Brook University

September 29, 2025

Data used in this module:

- ai4i2020.csv

Python notebook used in this module

- Classification.ipynb

1 Introduction

Considering an entity characterized by an N dimensional feature representation: $\boldsymbol{X} = [X_1, \ldots, X_N]^{\mathrm{T}}$. We would like to infer a target variable of interest, Y, which takes values from a finite set of K categories, $\{c_1, \ldots, c_K\}$. Classification is about developing a classifier: h: $\boldsymbol{X} \rightarrow Y$, which can assign an observation with feature values $\boldsymbol{x} = [x_1, \ldots, x_N]^{\mathrm{T}}$ to a class label y.

There are many classification problems in the real world. For example, given an image captured by a robot in infrastructure inspection, a classifier can determine if the image contains defect. Given measures of machine attributes in operations, a classifier can predict the type of failure the machine will confront. This learning module is developed based on related chapters in [1, 2, 3, 4, 5].

1.1 Bayes Classifier

Define the probability that the class of the response is c_k conditional on the observed feature values is:

$$p_k = P(c_k|\boldsymbol{x}), \tag{1}$$

and $\sum_{k=1}^{K} p_k = 1$.

According to Bayes' theorem,

$$p_k = \frac{f_k(\boldsymbol{x})\pi_k}{\sum_{r=1}^{K} f_r(\boldsymbol{x})\pi_r} \tag{2}$$

where π_k is the prior probability:

$$\pi_k = P(c_k), \tag{3}$$

and $f_k(\boldsymbol{x})$ is the likelihood probability,

$$f_k(\boldsymbol{x}) = P(\boldsymbol{x}|c_k). \tag{4}$$

The Bayes classification rule says that the optimal rule is one that picks the class with the maximum posterior probability:

$$h^*(\boldsymbol{x}) = \arg\max_k p_k = \arg\max_k f_k(\boldsymbol{x})\pi_k. \tag{5}$$

1.2 Three Approaches to Classification

The Bayes rule depends on unknown quantities $f_k(\boldsymbol{x})$ and π_k for $k = 1, \ldots, K$. Therefore, we need to use the data to find some approximation to the Bayes rule.

Generally speaking. there are three major approaches:

- The first approach is density estimation. We use data to estimate the prior and likelihood probabilities and then compute the posterior probability using the Bayes's theorem.
- The second approach is to model the conditional probability p_k directly, for example, as a parametric model.
- The last approach involves constructing a discriminant function that directly assign each vector to a specific class.

Classifiers can be developed in one of these approaches or a mix of them. In this learning module, we will introduce several classification methods.

2 Quadratic and Linear Discriminant Analysis

One approach to classification is to use the density estimation strategy and assume a parametric model for the densities.

Assume $f_k(\boldsymbol{x})$ are multivariate Gaussians:

$$f_k(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^N|\Sigma_k|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)\Sigma_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}}\right\} \tag{6}$$

for $k = 1, \dots, K$, where

$$\boldsymbol{x} = [x_1, \dots, x_N]^{\mathrm{T}} \tag{7}$$

is the feature vector of a data point,

$$\boldsymbol{\mu}_k = \mathrm{E}[\boldsymbol{x}|c_k] = [\mu_{k,1}, \dots, \mu_{k,N}]^{\mathrm{T}} \tag{8}$$

is the mean feature vector for class k, and

$$\Sigma_k = \mathrm{E}[(\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{\mu}_k)|c_k] \tag{9}$$

is the class-k feature covariance matrix. $|\cdot|$ means the determinant of a matrix.

2.1 Quadratic Discriminant Analysis (QDA)

According to (6),

$$\log\left[f_k(\boldsymbol{x})\pi_k\right] = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)\Sigma_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} + \log \pi_k, \tag{10}$$

where the first term on the right hand side, $-\frac{N}{2}\log 2\pi$, is a constant. Therefore, we drop it and define the discriminant function:

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)\Sigma_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} + \log \pi_k. \tag{11}$$

$(\boldsymbol{x} - \boldsymbol{\mu}_k)\Sigma_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}}$ in (11) is called Mahalanobis distance, measuring a point relative to a distribution.

The Bayes rule assigns $\boldsymbol{x}$ to a specific class:

$$h^*(\boldsymbol{x}) = \arg\max_k \delta_k(\boldsymbol{x}). \tag{12}$$

The discriminant function in Eq. (11) shows that, in the feature space,

- the smaller the distance from a data point to the mean of class k, the larger the discriminant function value, if all else are the same.
- the smaller the determinant of class-k co-variance matrix, the tier the distribution, the larger the discriminant function, if all else are the same.
- the larger the prior probability of class k, the larger the discriminant function value, if all else is the same.

The discriminant function $\delta_k(\boldsymbol{x})$ in (11) is quadratic. Therefore, the classification procedure in (11) and (12) is called Quadratic Discriminant Analysis (QDA).

Given a training sample with M data points, we can estimate the prior probability π_k, and parameters of the likelihood probability, Σ_k and $\boldsymbol{\mu}_k$, which are required for determining the discriminant function $\delta_k(\boldsymbol{x})$ in (11).

Denote by $\mathbf{X} \in \mathbb{R}^{N\times M}$ the feature values of training dataset. We partition $\mathbf{X}$ by classes: $\mathbf{X} = \cup_{k=1}^{K}\mathbf{X}_k$, where $\mathbf{X}_k = \{\boldsymbol{x}_i|c_k\}$ represents the attributes of data points in class c_k and M_k is the number of such data points:

$$M_k = \sum_{i=1}^{M} 1\{y_i = c_k\}. \tag{13}$$

Then, for each class k, the prior probability is estimated as:

$$\widehat{\pi}_k = \frac{M_k}{M}, \tag{14}$$

and the parameters of the likelihood function are estimated as:

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{M_k}\sum_{i=1}^{M} 1\{y_i = c_k\}\boldsymbol{x}_i \tag{15}$$

$$\widehat{\Sigma}_k = \frac{1}{M_k - 1}(\mathbf{X}_k - \widehat{\boldsymbol{\mu}}_k)^{\mathrm{T}}(\mathbf{X}_k - \widehat{\boldsymbol{\mu}}_k). \tag{16}$$

2.2 Linear Discriminant Analysis (LDA)

If $\Sigma_1 = \cdots = \Sigma_K = \Sigma$, that is, features have the same covariance matrix across all classes, the discriminant function becomes linear. Let's substitute Σ for Σ_k in the log function of the posterior probability, leading to

$$\begin{aligned}\log f_k(\boldsymbol{x})\pi_k &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}} + \log\pi_k \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma| - \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\Sigma^{-1}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\Sigma^{-1}\boldsymbol{\mu}_k + \log\pi_k.\end{aligned} \tag{17}$$

We drop the first three terms on the right hand side, which are class-independent, to obtain the updated discriminant function:

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\Sigma^{-1}\boldsymbol{\mu}_k + \log\pi_k, \tag{18}$$

which is linear in $\boldsymbol{x}$.

$\boldsymbol{\mu}_k$ in (18) can be estimated using (15) and Σ can be estimated with

$$\widehat{\Sigma} = \frac{1}{M-1}(\mathbf{X}-\widehat{\boldsymbol{\mu}})^{\mathrm{T}}(\mathbf{X}-\widehat{\boldsymbol{\mu}}), \tag{19}$$

where $\widehat{\boldsymbol{\mu}}$ is the estimator for the mean attribute vector using the entire training dataset:

$$\widehat{\boldsymbol{\mu}} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_i = \frac{1}{M}\sum_{k=1}^{K}M_k\,\widehat{\boldsymbol{\mu}}_k. \tag{20}$$

In the learning module "Feature Extraction", we also introduced LDA and derived it in a different approach called Fisher's Linear Discriminant.

3 Naive Bayes (NB)

3.1 Naive Bayes Classifier

Estimating π_k is simple, but estimating the likelihood probability $f_k(\boldsymbol{x})$ is not. In QDA and LDA, $f_k(\boldsymbol{x})$ is assumed a multivariate Gaussian, leading to the requirement of estimating its parameters $\boldsymbol{\mu}_k$ and Σ_k. A more general approach is to estimate $f_k(\boldsymbol{x})$ with some non-parametric density estimator. However, if $\boldsymbol{x}$ is in high dimension, non-parametric density estimation is not reliable.

If we assume feature variables are independent, we can just estimate N 1-dimensional density functions. The joint distribution is simply the product of these 1-dimensional density functions. Although these are apparently over-simplified assumptions, Naive Bayes (NB) classifiers have worked quite well in many real-world situations.

NB classifiers are implemented below.

1. Estimate the density for the jth attribute pertaining to class k, $\widehat{f}_{k,j}(x_j) = \mathrm{P}(x_j|c_k)$, with a density estimator
2. The estimator of the likelihood probability is the product of the N 1-dimensional density estimators: $\widehat{f}_k(\boldsymbol{x}) = \prod_{j=1}^{N}\widehat{f}_{k,j}(x_j)$
3. Estimate the prior probability $\widehat{\pi}_k = \sum_1^M 1\{y_i = c_k\}/M$.
4. The Bayes rule is

$$h^*(\boldsymbol{x}) = \arg\max_k \widehat{\pi}_k\widehat{f}_k(\boldsymbol{x}). \tag{21}$$

3.2 Likelihood Density Estimation

There are various NB classifiers. They differ mainly by the data type of features and so the assumption they make regarding the distribution $f_{k,j}$.

- Gaussian NB: assumes that $f_{k,j}$ is a Gaussian.

$$f_{k,j}(x_j) = \frac{1}{\sqrt{2\pi\sigma_{k,j}^2}} \exp\left\{ -\frac{(x_j - \mu_{k,j})^2}{2\sigma_{k,j}^2} \right\}. \tag{22}$$

  where $\mu_{k,j}$ and $\sigma_{k,j}$ are parameters of the Gussian for feature j in class k, estimated as

$$\widehat{\mu}_{k,j} = \frac{\sum_{i=1}^{M} x_{i,j} 1\{y_i = c_k\}}{M_k}$$

$$\widehat{\sigma}_{k,j} = \frac{\sum_{i=1}^{M} (x_{i,j} 1\{y_i = c_k\} - \widehat{\mu}_{k,j})^2}{M_k - 1}.$$

- Bernoulli NB: implemented for features that are distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued variable.

$$f_{k,j}(x_j) = \mu_{k,j}^{x_j} (1 - \mu_{k,j})^{(1-x_j)} \tag{23}$$

  where $\mu_{k,j}$ is the probability that attribute j appears in a data point pertaining to class k, estimated as

$$\widehat{\mu}_{k,j} = \frac{\sum_{i=1}^{M} x_{i,j} 1\{y_i = c_k\}}{M_k}.$$

- Multinomial NB: implemented for multinomially distributed data. That is, there may be multiple features but each is assumed to be a binomial variable.

$$f_{k,j}(x_j) = \frac{M_k!}{x_j!(M_k - x_j)!} \mu_{k,j}^{x_j} (1 - \mu_{k,j})^{(M_k - x_j)}. \tag{24}$$

  where $\mu_{k,j}$ is the probability that attribute j occurs in a sample pertaining to class k, estimated as

$$\widehat{\mu}_{k,j} = \frac{\sum_{i=1}^{M} 1\{y_i = c_k\} x_{i,j} + \alpha}{\sum_{i=1}^{M} \sum_{j=1}^{N} 1\{y_i = c_k\} x_{i,j} + \alpha N}, \tag{25}$$

  where α is a smoothing factor that accounts for features not present in the training dataset and prevents zero probabilities in inference.

- Categorical NB: implemented for categorically distributed data. It assumes that each feature has its own categorical distribution. Let $s_{j,l}$ denote class l of attribute j, then the density $f_{k,j}(x_j)$ is estimated as

$$\widehat{f}_{k,j}(x_j = s_{j,l}) = \frac{\sum_{i=1}^{M} 1\{x_{i,j} = s_{j,l}, y_i = c_k\} + \alpha}{M_k + \alpha N_j} \tag{26}$$

  where α is a smoothing factor that accounts for features not present in the training dataset and prevents zero probabilities in inference, and n_j is the number of categories of attribute j.

4 Logistic Regression

We studied logistic regression in the learning module "Regression", which estimates the conditional probability $p_k = P(c_k|\boldsymbol{x})$ by fitting a parametric model to data. We will not repeat it in this learning module. Logistic regression and LDA are almost the same in that they both lead to a linear decision rule. In LDA, we estimate the joint probability $f(\boldsymbol{x}|y)f(y)$, which equals $f(y|\boldsymbol{x})f(\boldsymbol{x})$. In logistic regression, we estimate the conditional probability $f(y|\boldsymbol{x})$ only and ignores $f(\boldsymbol{x})$.

5 Support Vector Machine (SVM)

Let's consider a non-probablistic classifier in the form of a decision boundary

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i). \tag{27}$$

where $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i)$ is a kernel function measuring the similarity of a new data point, $\boldsymbol{x}$, to be tested with a training data point i, $\boldsymbol{x}_i$, in the feature space. By adding additional suitable constraints, many of the coefficients λ_i's are zeros so that the prediction in test only depends on a subset of the training data points called *support vectors*. A resulting model is called support vector machine (SVM).

5.1 Large Margin Classifiers

Let's consider a linear decision boundary for a binary classification task:

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 \tag{28}$$

where $\boldsymbol{x}$ is the N-dimensional feature vector, $\boldsymbol{w}$ is the N-dimensional coefficient vector, and w_0 is the bias. If one assume data points are in two clases $y \in \{1, -1\}$ and they are linearly separately, $f(\boldsymbol{x}) = 0$ split the feature space into two regions: $f(\boldsymbol{x}) > 0$ is the region for $y = 1$ and $f(\boldsymbol{x}) < 0$ is the region for $y = -1$. That is, $yf(\boldsymbol{x}) \geq 0$.

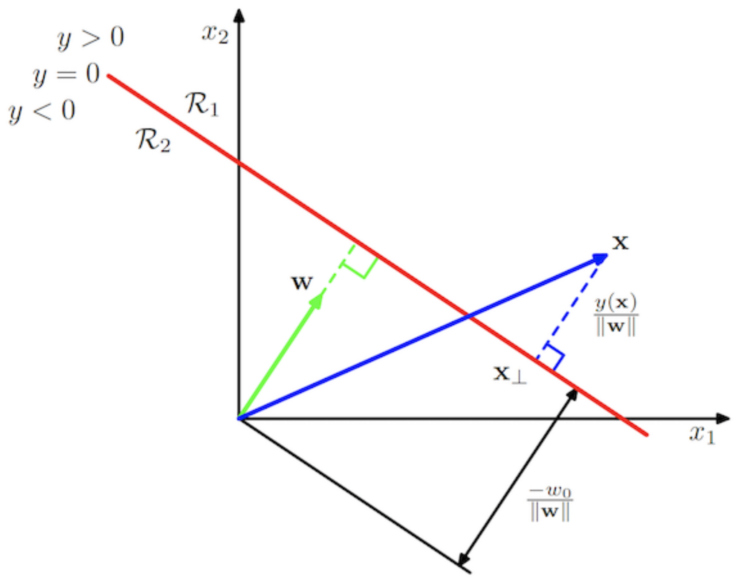

Figure 1: A linear decision boundary[4]

Ideally, we would like to pick a decision boundary that has maximum *margin*. The margin is the distance of the closet point to the decision boundary. Figure 1 illustrates a decision boundary on a 2D feature space. $\boldsymbol{w}$ is perpendicular to the decision boundary[1], indicating that $\boldsymbol{w}$ decides the direction of the boundary. w_0 is the distance from the boundary to the origin. A point in the feature space, $\boldsymbol{x}$, can be decomposed

$$\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{29}$$

[1] $\boldsymbol{w}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$ for any two points, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, on the decision boundary. $\boldsymbol{x}_1 - \boldsymbol{x}_2$ is along the direction of $f(\boldsymbol{x}) = 0$. Therefore, $\boldsymbol{w}$ must be perpendicular to $f(\boldsymbol{x}) = 0$.

where $\boldsymbol{x}_\perp$ is the orthogonal projection of $\boldsymbol{x}$ onto the boundary. $r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ is the projection of $\boldsymbol{x}$ along the direction of $\boldsymbol{w}$, where r is the length of $\boldsymbol{x}$ along $\boldsymbol{w}$. Since

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 \\ &= \boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + w_0 \\ &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_\perp + r\|\boldsymbol{w}\| + w_0, \end{aligned} \tag{30}$$

and $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_\perp = 0$, we have

$$r = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|} + \frac{-w_0}{\|\boldsymbol{w}\|}. \tag{31}$$

Figure 1 indicates the distance from $\boldsymbol{x}$ to the decision boundary is $\frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|}$. Therefore, to maximize the length of this distance is about solving the following optimization problem:

$$\max_{\boldsymbol{w},w_0} \frac{1}{\|\boldsymbol{w}\|} \min_{i\in\{1,\dots,M\}} (y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0)). \tag{32}$$

Scaling $\boldsymbol{w}$ and w_0 in (32) does not change the value of the objective function. Therefore, ons can force

$$\min_{i\in\{1,\dots,M\}} (y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0)) = 1 \tag{33}$$

so that (32) becomes a quadratic programming (QP) problem:

$$\begin{aligned} &\min_{\boldsymbol{w},w_0} \|\boldsymbol{w}\|^2 \\ &\text{s.t.} \\ &\quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1, i = 1, \dots, M. \end{aligned} \tag{34}$$

To solve the QP problem in (34), one can construct a Lagrangian by introducing Lagrangian multipliers $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_M]^{\mathrm{T}} \geq 0$,

$$\mathrm{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{M} \lambda_i(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) - 1), \tag{35}$$

which is the lower boundary for the objective function in (34). We can maximize the Lagrangian function with respect to decision variables $\boldsymbol{w}, w_0, \boldsymbol{\lambda}$ to find the optimal solution to (34). First, the $\boldsymbol{w}$ and w_0 are optimized out by solving the following linear system:

$$\nabla_{\boldsymbol{w}}\mathrm{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \boldsymbol{w} - \sum_{i=1}^{M} \lambda_i y_i \boldsymbol{x}_i = 0, \tag{36}$$

$$\frac{\partial \mathrm{L}(\boldsymbol{w}, \mathrm{w}_0, \boldsymbol{\lambda})}{\partial w_0} = \sum_{i=1}^{M} \lambda_i y_i = 0. \tag{37}$$

Therefore,

$$\widehat{\boldsymbol{w}} = \sum_{i=1}^{M} \lambda_i y_i \boldsymbol{x}_i, \tag{38}$$

$$\sum_{i=1}^{M} \lambda_i y_i = 0. \tag{39}$$

Bringing them into the Lagrangian in (35) leads to the Lagrangian dual function:

$$\mathrm{L}(\boldsymbol{\lambda}) = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^{M} \lambda_i. \tag{40}$$

Ultimately, we will solve the following standard QP problem:

$$\begin{aligned} &\max_{\boldsymbol{\lambda}} -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^{M}\lambda_i \\ \text{s.t.}& \\ &\sum_{i=1}^{M}\lambda_i y_i = 0 \\ &\lambda_i \geq 0, \quad i = 1, \ldots, M. \end{aligned} \tag{41}$$

Solution to (41), denoted by $\widehat{\boldsymbol{\lambda}}$, must also satisfy the Karush–Kuhn–Tucker (KKT) conditions:

$$\begin{aligned} \lambda_i &\geq 0 \\ y_i f(\boldsymbol{x}_i) - 1 &\geq 0 \\ \lambda_i(y_i f(\boldsymbol{x}) - 1) &= 0 \end{aligned} \tag{42}$$

for $i = 1, \ldots, M$. The KKT conditions in (42) says that either $\widehat{\lambda}_i = 0$ or the constraint $y_i f(\boldsymbol{x}_i) = 1$ is active. $y_i f(\boldsymbol{x}_i) = 1$ means the data point i is on the maximum margin of SVM. Those data points are support vectors. $\mathcal{S}$ denotes the set of support vectors.

Finally, the SVM classifier is

$$\begin{aligned} f^*(\boldsymbol{x}) &= f(\boldsymbol{x}|\widehat{\boldsymbol{w}}, \widehat{w}_0) \\ &= \widehat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{x} + \widehat{w}_0 \\ &= \sum_{i\in\mathcal{S}}\widehat{\lambda}_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\left(y_i - \widehat{\lambda}_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i\right), \end{aligned} \tag{43}$$

which shows that only a portion of the training data points (i.e., support vectors) are used in predicting the class label given the feature vector $\boldsymbol{x}$.

5.2 Soft Margin Classifiers

If data are not linearly separatable, we introduce a slack variable ξ_i (≥ 0) to replace the hard constraint $y_i f(\boldsymbol{x}_i) \geq 1$ with the soft constraint $y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i$. Consequently, the optimization problem in (34) becomes

$$\begin{aligned} &\min_{\boldsymbol{w}, w_0, \boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C\sum_{i=1}^{M}\xi_i \\ \text{s.t.}& \\ &y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1 - \xi_i, \qquad i = 1, \ldots, M \\ &\xi_i \geq 0, \qquad\qquad\qquad\qquad i = 1, \ldots, M \end{aligned} \tag{44}$$

where $C(\geq 0)$ is a hyperparameter that controls how many points are allowed to vialate the margin constraint. If $C \to \infty$, it becomes the hard margin classifier.

The corresponding Lagragian is

$$\mathrm{L}(\boldsymbol{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\eta}) = \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\lambda_i(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) - 1 + \xi_i) - \sum_{i=1}^{M}\eta_i\xi_i \tag{45}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_M]^{\mathrm{T}} \geq 0$ and $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_M]^{\mathrm{T}} \geq 0$ are Lagragian multipliers. By solving the linear system: $\nabla_{\boldsymbol{w}}\mathrm{L} = 0$, $\frac{\partial \mathrm{L}}{\partial w_0} = 0$, and $\nabla_{\boldsymbol{\xi}}\mathrm{L} = 0$, $\boldsymbol{w}$, w_0, and $\boldsymbol{\xi}$ in the Lagragian are optimized out, leading to the same Lagrangian dual function $\mathrm{L}(\boldsymbol{\lambda})$ as in (40). Finally, we solve the following QP dual problem:

$$
\begin{aligned}
&\max_{\boldsymbol{\lambda},\boldsymbol{\eta}} -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^{M}\lambda_i \\
&\text{s.t.} \\
&\sum_{i=1}^{M}\lambda_i y_i = 0 \\
&C - \lambda_i - \eta_i = 0, \quad i = 1,\ldots,M \\
&\lambda_i, \eta_i \geq 0, \quad i = 1,\ldots,M
\end{aligned} \tag{46}
$$

While the objective function in the soft margin case is identical to that in the hard margin case, the constraint on λ_i is different.

The KKT conditions for the soft margin case are:

$$
\begin{aligned}
\lambda_i &\geq 0 \\
y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) - 1 + \xi_i &\geq 0 \\
\lambda_i(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) - 1 + \xi_i) &= 0 \\
\eta_i &\geq 0 \\
\xi_i &\geq 0 \\
\eta_i\xi_i &= 0
\end{aligned} \tag{47}
$$

for all data point in the training set (i.e., $i = 1.\ldots,M$).

Moreover,

$$
C - \lambda_i - \eta_i = 0 \tag{48}
$$

The value of λ_i tells the position of the corresponding training data point in the feature space.

- $\lambda_i = 0$: the data point is outside the margin
  On one hand, $\lambda_i = 0$ indicates $y_i f(\boldsymbol{x}_i) > 1 - \xi_i$. On the other hand, $\lambda_i = 0$ indicates $\eta_i = C > 0$ and so ξ_i must be zero. Therefore, the data point is outside the margin and it can be ignored.
- $0 < \lambda_i < C$: the data point is on the margin
  This indicates that $y_i f(\boldsymbol{x}_i) - 1 = \xi_i$. And, $\lambda_i < C$ indicates that $\eta_i > 0$ and so $\xi_i = 0$. Therefore, the data point is on the margin.
- $\lambda_i = C$: the data point can be inside the margin
  This indicates that $y_i f(\boldsymbol{x}_i) - 1 = \xi_i$. Moreover, $\lambda_i = C$ indicates $\eta_i = 0$ and so $\xi_i > 0$. Therefore, the data point is allowed to be inside the margin.

to be continued...

References

[1] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*, volume 4. AMLBook New York, 2012.

[2] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[4] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[5] Larry Wasserman. *All of statistics: a concise course in statistical inference*, volume 26. Springer, 2004.