STAT 425: Introduction to Nonparametric Statistics **Winter 2026**

Lecture 8: Nonparametric Classification

Instructor: Yen-Chi Chen

8.1 Introduction

Classification is one of the most important data analysis problems. Much early work on this topic was done by statisticians but in the past 20 years, computer science and machine learning communities have made much much more progress on this topic.

Here are some classical applications of classification.

- **Email spam.** Email service provider such as Google often faced the problem of classifying a new email. The problem they want to address is like: given an email, how do you decide if it is a spam, an ordinary email, or an important one?
- **Image classification.** If you have used Facebook, you may notice that whenever your photo contains a picture of one of your friends, Facebook may ask you if you want to tag your friend, even if you did not manually tell the computer that this is your friend. How do they know that picture is a human and that guy is your friend?

We consider a simple scenario – binary classification. Namely, there are only two possible classes that we will consider. We will just denote the two classes as 0 and 1.

The classification problem can be formalized as follows. Given a feature vector x_0, we want to create a **classifier** c that maps x_0 into 0 or 1. Namely, we want to find a function $c(x_0)$ that outputs only two possible number: 0 and 1. Moreover, we want to make sure that our *classification error* is small. Let y_0 be the actual label of x_0. We measure the classification error using a **loss function** L such that the the loss of making a prediction $c(x_0)$ when the actual class label is y_0 is $L(c(x_0), y_0)$. A common loss function is the **0-1 loss**, which is $L(c(x_0), y_0) = I(c(x_0) \neq y_0)$. Namely, when we make a wrong classification, we loss 1 point and we do not lose anything if we make the correct classification.

How do we find the classifier c? A good news is: often we have a labeled sample (data) $(X_1, Y_1), \cdots, (X_n, Y_n)$ available. Then we will find c using this dataset.

In statistics, we often model the data as an IID random sample from a distribution. We now define several useful distribution functions:

$$\begin{aligned} p_0(x) = p(X = x|Y = 0): &\quad \text{the density of } X \text{ when the actual label is 0,} \\ p_1(x) = p(X = x|Y = 1): &\quad \text{the density of } X \text{ when the actual label is 1,} \\ P(y|x) = P(Y = y|X = x): &\quad \text{the probability of being in the class } y \text{ when the feature is } x, \\ P_Y(y) = P(Y = y): &\quad \text{the probability of observing the class } y, \text{ regardless of the feature value.} \end{aligned} \tag{8.1}$$

Using a probability model, we will define the **risk function**, which is the expected value of the loss function when the input is random. The risk of a classifier c is

$$R(c) = \mathbb{E}(L(c(X), Y)).$$

Ideally, we want to find a classifier that minimizes the risk because such a classifier will minimize our *expected losses*.

Assume that we know the 4 quantities in equation (8.1), what class label will you predict when seeing a feature $X = x$? An intuitive choice is that we should predict the value y that maximizer $P(y|x)$. Namely, we predict the label using the one with highest probability. Such classifier can be written as

$$c_*(x) = \mathsf{argmax}_{y=0,1} P(y|x) = \begin{cases} 0, & \text{if } P(0|x) \geq P(1|x), \\ 1, & \text{if } P(1|x) > P(0|x). \end{cases} \tag{8.2}$$

Is this classifier good in the sense of the classification error (risk)? The answer depends on the loss function. A good news is: this classifier is the optimal classifier for the $0-1$ loss. Namely,

$$R(c_*) = \min_c R(c)$$

when using a $0-1$ loss. However, if we are using other loss function, this classifier will not be the best one (with the smallest expected loss).

Derivation of c_* is optimal under $0-1$ loss. Given a classifier c, the risk function $R(c) = \mathbb{E}(L(c(X), Y))$. Using the property of expectation, we can further write it as

$$R(c) = \mathbb{E}(L(c(X), Y)) = \mathbb{E}(\underbrace{\mathbb{E}(L(c(X), Y)|X)}_{(A)}).$$

For the quantity (A), we have

$$\begin{aligned} \mathbb{E}(L(c(X), Y)|X) &= L(c(X), 1)p(Y = 1|X) + L(c(X), 0)p(Y = 0|X) \\ &= I(c(X) \neq 1)p(Y = 1|X) + I(c(X) \neq 0)p(Y = 0|X) \\ &= \begin{cases} p(Y = 1|X) & \text{if } c(X) = 0 \\ p(Y = 0|X) & \text{if } c(X) = 1. \end{cases} \end{aligned}$$

Thus, seeing a feature X, the expected loss we have when predicting $c(X) = 0$ is $P(Y = 1|X)$ whereas when prediction $c(X) = 1$ is $P(Y = 0|X)$. The optimal choice is predicting $c(X) = 0$ if $P(Y = 1|X) \leq P(Y = 0|X)$ and $c(X) = 1$ if $P(Y = 1|X) > P(Y = 0|X)$ (the equality does not matter), which is the classifier c_*.

When a classifier attains the optimal risk (i.e., having a risk of $\min_c R(c)$), it is called a **Bayes classifier**. Thus, the classifier c_* is the Bayes classifier in $0-1$ loss.

For a classifier c, we define its **excess risk (regret)** as

$$\mathcal{E}(c) = R(c) - \min_c R(c).$$

The excess risk is a quantity that measures how the quality of c is away from the optimal/Bayes classifier. If we cannot find the Bayes classifier, we will at least try to find a classifier whose excess risk is small.

8.1.1 Confusion Matrix

Given a classifier and a set of labeled data, we can illustrate the quality of classification using a confusion matrix. In binary classification, a confusion matrix is a 2×2 matrix (you can view it as a contingency table) as follows:

n_{ij} is the number of instances/observations where the predicted label is i and actual label is j.

	Actual label: 0	Actual label: 1
Predicted label: 0	n_{00}	n_{01}
Predicted label: 1	n_{10}	n_{11}

The quantity

$$\frac{n_{10}+n_{01}}{n_{00}+n_{01}+n_{10}+n_{11}},$$

is called the misclassification rate and is an empirical estimate of the risk of the classifier.

If the class label 0 stands for 'normal case' while the label 1 stands for 'anomaly', then we can interpret the confusion matrix as

	Actual label: 0	Actual label: 1
Predicted label: 0	True negative	False negtaive
Predicted label: 1	False positive	True positive

This interpretation is commonly used in engineering problem and medical research for detecting abnormal situation.

8.2 Density Estimation Approach (Naive Bayes)

From the above analysis, we can use a density estimator for classification. This approach is often known as the *naive Bayes* approach.

A key insight is from the Bayes rule:

$$P(y|x) = P(Y=y|X=x) = \frac{p(x,y)}{p(x)} = \frac{p(x|y)P(Y=y)}{p(x)} = \frac{p_y(x)P_Y(y)}{p(x)},$$

where $p(x) = \sum_y p(x,y) = p(x,0) + p(x,1) = p_0(x)P_Y(0) + p_1(x)P_Y(1)$. Thus, the Bayes classifier can be written as

$$\begin{aligned} c_*(x) &= \begin{cases} 0, & \text{if } P(0|x) \geq P(1|x) \\ 1, & \text{if } P(1|x) > P(0|x) \end{cases} \\ &= \begin{cases} 0, & \text{if } \frac{p_0(x)P_Y(0)}{p(x)} \geq \frac{p_1(x)P_Y(1)}{p(x)} \\ 1, & \text{if } \frac{p_1(x)P_Y(1)}{p(x)} > \frac{p_0(x)P_Y(0)}{p(x)} \end{cases} \\ &= \begin{cases} 0, & \text{if } p_0(x)P_Y(0) \geq p_1(x)P_Y(1) \\ 1, & \text{if } p_1(x)P_Y(1) > p_0(x)P_Y(0) \end{cases}. \end{aligned}$$

Thus, if we can estimate $p_0(x), p_1(x)$, and $P_Y(y)$, we can construct a classifier.

$P_Y(y)$ is very easy to estimate. It is the probability of seeing an observation with label y. As a result, a simple estimator is to use the ratio of observations with this label. Namely,

$$\widehat{P}_Y(y) = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = y).$$

$p_y(x)$ is just the conditional density of X given the label being y. Thus, we can simply apply a density estimator to those observations with a class label y.

Example: kernel density estimator. Using a kernel density estimator (KDE), we obtain

$$\widehat{p}_{y,kde}(x) = \frac{1}{n_y h}\sum_{i=1}^{n} I(Y_i = y) K\left(\frac{X_i - x}{h}\right),$$

where $n_y = \sum_{i=1}^{n} I(Y_i = y)$ is the number of observations with label being y. Note that $\widehat{P}_Y(y) = \frac{n_y}{n}$. Thus, a classifier based on a KDE is

$$\begin{aligned}\widehat{c}_{\mathsf{KDE}}(x) &= \begin{cases} 0, & \text{if } \widehat{p}_{0,kde}(x)\widehat{P}_Y(0) \geq \widehat{p}_{1,kde}(x)\widehat{P}_Y(1) \\ 1, & \text{if } \widehat{p}_{1,kde}(x)\widehat{P}_Y(1) > \widehat{p}_{0,kde}(x)\widehat{P}_Y(0) \end{cases} \\ &= \begin{cases} 0, & \text{if } \sum_{i=1}^{n} I(Y_i = 0) K\left(\frac{X_i - x}{h}\right) \geq \sum_{i=1}^{n} I(Y_i = 1) K\left(\frac{X_i - x}{h}\right) \\ 1, & \text{if } \sum_{i=1}^{n} I(Y_i = 1) K\left(\frac{X_i - x}{h}\right) > \sum_{i=1}^{n} I(Y_i = 0) K\left(\frac{X_i - x}{h}\right) \end{cases}. \end{aligned}$$

The classifier $\widehat{c}_{\mathsf{KDE}}(x)$ is also called the kernel classifier.

Example: density basis approach. We can use the basis approach as well. Assume that we consider M basis and we use the cosine basis $\{\phi_1(x), \phi_2(x), \cdots\}$. The estimator of $p_y(x)$ will be the density estimator using only those observations with a label y. Let

$$\widehat{\theta}_{y,\ell} = \frac{1}{n_y}\sum_{i=1}^{n} I(Y_i = y)\phi_\ell(X_i)$$

be the estimator of the ℓ-th coefficient using only those observations with a label y. The corresponding density estimator is

$$\widehat{p}_{y,M}(x) = \sum_{\ell=1}^{M} \widehat{\theta}_{y,\ell}\phi_\ell(x) = \sum_{\ell=1}^{M} \frac{1}{n_y}\sum_{i=1}^{n} I(Y_i = y)\phi_\ell(X_i)\phi_\ell(x) = \frac{1}{n_y}\sum_{i=1}^{n} I(Y_i = y)\sum_{\ell=1}^{M}\phi_\ell(X_i)\phi_\ell(x).$$

Thus, the corresponding classifier is

$$\begin{aligned}\widehat{c}_{\mathsf{M}}(x) &= \begin{cases} 0, & \text{if } \widehat{p}_{0,M}(x)\widehat{P}_Y(0) \geq \widehat{p}_{1,M}(x)\widehat{P}_Y(1) \\ 1, & \text{if } \widehat{p}_{1,M}(x)\widehat{P}_Y(1) > \widehat{p}_{0,M}(x)\widehat{P}_Y(0) \end{cases} \\ &= \begin{cases} 0, & \text{if } \sum_{i=1}^{n} I(Y_i = 0)\sum_{\ell=1}^{M}\phi_\ell(X_i)\phi_\ell(x) \geq \sum_{i=1}^{n} I(Y_i = 1)\sum_{\ell=1}^{M}\phi_\ell(X_i)\phi_\ell(x) \\ 1, & \text{if } \sum_{i=1}^{n} I(Y_i = 1)\sum_{\ell=1}^{M}\phi_\ell(X_i)\phi_\ell(x) > \sum_{i=1}^{n} I(Y_i = 0)\sum_{\ell=1}^{M}\phi_\ell(X_i)\phi_\ell(x) \end{cases}. \end{aligned}$$

8.3 Regression Approach

If we know the $P(y|x)$, we can build the Bayes classifier and this classifier is the optimal one in terms of the risk function. However, $P(y|x)$ is a population quantity, which is often unknown to us. All we have is a random sample $(X_1, Y_1), \cdots, (X_n, Y_n)$. So the question becomes: how do we estimate $P(y|x)$ using the data?

It turns out that this is a problem we know have to solve. Here is just one hint: because the response variable Y only takes two possible values $\{0, 1\}$, it is actually a Bernoulli random variable! Thus, $\mathbb{E}(Y) = P(Y = 1)$, which implies

$$\mathbb{E}(Y|X = x) = P(Y = 1|X = x) = P(1|x). \tag{8.3}$$

Namely, $P(1|x)$ is the regression function! Using the fact that $P(0|x) + P(1|x) = 1$, an estimator of $P(1|x)$ leads to an estimator of $P(y|x)$ for both $y = 0$ and $y = 1$.

Thus, as long as we have a regression estimator, we can convert it into a classifier. Here is one example of using kernel regression. Let

$$\widehat{m}_K(x) = \frac{\sum_{i=1}^n Y_i K\left(\frac{X_i - x}{h}\right)}{\sum_{i=1}^n K\left(\frac{X_i - x}{h}\right)}$$

be the kernel regression. Then

$$\widehat{P}_K(1|x) = \widehat{m}_K(x), \quad \widehat{P}_K(0|x) = 1 - \widehat{m}_K(x).$$

Thus, a classifier based on kernel regression is

$$\begin{aligned}
\widehat{c}_K(x) &= \begin{cases} 0, & \text{if } \widehat{P}_K(0|x) \geq \widehat{P}_K(1|x), \\ 1, & \text{if } \widehat{P}_K(1|x) > \widehat{P}_K(0|x) \end{cases} \\
&= \begin{cases} 0, & \text{if } 1 - \widehat{P}_K(1|x) \geq \widehat{P}_K(1|x), \\ 1, & \text{if } \widehat{P}_K(1|x) > 1 - \widehat{P}_K(1|x). \end{cases} \\
&= \begin{cases} 0, & \text{if } \widehat{P}_K(1|x) \leq \frac{1}{2}, \\ 1, & \text{if } \widehat{P}_K(1|x) > \frac{1}{2}. \end{cases} \\
&= \begin{cases} 0, & \text{if } \widehat{m}_K(x) \leq \frac{1}{2}, \\ 1, & \text{if } \widehat{m}_K(x) > \frac{1}{2}. \end{cases}
\end{aligned}$$

Namely, the classifier will output 1 whenever the estimated regression function is greater than half and 0 otherwise.

Will this classifier be a good one? Intuitively, it should be true. If we have a good regression estimator, the corresponding classifier should also be good. In fact, we have the following powerful result linking the quality of a regression estimator and the excess risk.

Theorem 8.1 *Assume we use the* $0-1$ *loss. Let* $\widehat{m}$ *be a regression estimator and* $\widehat{c}_m$ *be the corresponding classifier. Then*

$$\mathcal{E}(\widehat{c}_m) \leq 2 \int |\widehat{m}(x) - m(x)| dP(x) \leq 2\sqrt{\int |\widehat{m}(x) - m(x)|^2 dP(x)}.$$

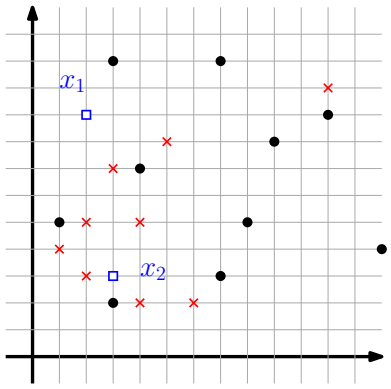
Namely, if we have a regression estimator whose overall quality is good, the corresponding classifier will also have a small excess risk (i.e., perform comparably well compared to the optimal classifier).

8.4 k-NN Approach

The regression method essentially shows that our classifier should be based on the probability $P(Y = 1|x)$ versus $P(Y = 0|x)$. Namely, $c(x) = 1$ if the the observed data's class labels around $X = x$ has a majority of $Y = 1$.

The k-NN approach offers a natural application of this insight. The idea is very simple – for a given point x_0, we find its k-th nearest data points. Then we compare the labels of these k points and assign the label of x_0 according to the majority label in these k points.

Take the data in the following picture as an example. There are two classes: black dots and red crosses. We are interested in the class label at the two blue boxes (x_1 and x_2).



Assume we use a 3-NN classifier $\widehat{c}_{3-NN}$. At point x_1, its 3-NN contains two black dots and one red cross, so $\widehat{c}_{3-NN}(x_1) =$ black dot. At point x_2, its 3-NN has one black dots and two red crosses, so $\widehat{c}_{3-NN}(x_1) =$ red cross. Note that if there are ties, we will randomly assign the class label attaining the tie.

The k-NN approach is simple and easy to operate. It can be easily generalize to multiple classes – the idea is the same: we assign the class label according to the majority in the neighborhood.

How do we choose k? The choice of k is often done by a technique called cross-validation. The basic principle is: we split the data into two parts, use one part to train our classifier and evaluate the performance on the other part. Repeating the above procedure multiple times and applying it to each k, we can obtain an estimate of the performance. We then choose the k with the best performance. We will talk about this topic later.

8.5 Decision Tree

Decision tree is another common approach in classification. It also utilizes the fact that an ideal classifier $c(x) = 1$ if the majority observed labels around $X = x$ is $Y = 1$. The concept of 'data around $X = x$' is then described by a tree on the covariate/feature space.

A decision tree is like a regression tree – we partition the space of covariate into rectangular regions and then assign each region a class label. If the tree is given, the class label at each region is determined by the majority vote (using the majority of labels in that region).

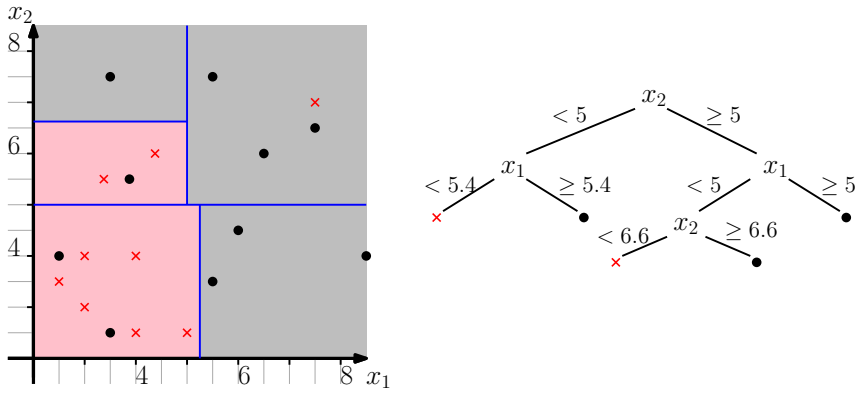
The following picture provides an example of a decision tree using the same data as the one in the previous section.

In the left panel, we display the scatterplot and regions separated by a decision tree. The background color denotes the estimated label. The right panel displays the tree structure.

In the case of binary classification (the class label $Y = 0$ or 1), the decision tree can be written as follows. Let $(X_1, Y_1), \cdots, (X_n, Y_n)$ denotes the data and $R_1, \cdots, R_k$ is a rectangular partition of the space of the covariates. Let $R(x)$ denotes the rectangular regions where x falls within. The decision tree is

$$\widehat{c}_{DT}(x) = I\left(\frac{\sum_{i=1}^n Y_i I(X_i \in R(x))}{\sum_{i=1}^n I(X_i \in R(x))} > \frac{1}{2}\right).$$

Here, you can see that the decision tree is essentially a classifier converted from a regression tree.



8.6 Logistic Regression

The logistic regression is a regression model that is commonly applied to classification problems as well. It is becoming more and more popular since people utilize the logistic regression to construct a smooth loss function for classification problem, which is particularly useful for training a complex classifier like a neural net. We being with an introduction on the usual modeling concept of logistic regression.

We first talk about some interesting examples when attempting to model the probability.

Example. In graduate school admission, we are wondering how a student's GPA affects the chance that this applicant received the admission. In this case, each observations is a student and the response variable Y represents whether the student received admission ($Y = 1$) or not ($Y = 0$). GPA is the covariate X. Thus, we can model the probability

$$P(\mathsf{admitted}|\mathsf{GPA} = x) = P(Y = 1|X = x) = q(x).$$

Example. In medical research, people are often wondering if the heretability of the type-2 diabetes is related to some mutation from of a gene. Researchers record if the subject has the type-2 diabetes (response) and measure the mutation signature of genes (covariate X). Thus, the response variable $Y = 1$ if this subject has the type-2 diabetes. A statistical model to associate the covariate X and the response Y is through

$$P(\text{subject has type-2 diabetes}|\text{mutation signature} = x) = P(Y = 1|X = x) = q(x).$$

Thus, the function $q(x)$ now plays a key role in determining how the response Y and the covariate X are associated. The logistic regression provides a simple and elegant way to characterize the function $q(x)$ in a 'linear' way. Because $q(x)$ represents a *probability*, it ranges within $[0, 1]$ so naively using a linear regression will not work. However, consider the following quantity:

$$O(x) = \frac{q(x)}{1 - q(x)} = \frac{P(Y = 1|X = x)}{P(Y = 0|X = x)} \in [0, \infty).$$

The quantity $O(x)$ is called the *odds* that measures the contrast between the event $Y = 1$ versus $Y = 0$. When the odds is greater than 1, we have a higher change of getting $Y = 1$ than $Y = 0$. The odds has an interesting asymmetric form– if $P(Y = 1|X = x) = 2P(Y = 0|X = x)$, then $O(x) = 2$ but if $P(Y = 0|X = x) = 2P(Y = 1|X = x)$, then $O(x) = \frac{1}{2}$. To symmetrize the odds, a straight-forward approach is to take (natural) logarithm of it:

$$\log O(x) = \log \frac{q(x)}{1 - q(x)}.$$

This quantity is called *log odds*. The log odds has several beautiful properties, for instance when the two probabilities are the same ($P(Y=1|X=x) = P(Y=0|X=x)$), $\log O(x) = 0$, and

$$P(Y=1|X=x) = 2P(Y=0|X=x) \Rightarrow \log O(x) = \log 2$$
$$P(Y=0|X=x) = 2P(Y=1|X=x) \Rightarrow \log O(x) = -\log 2.$$

The logistic regression is to impose a linear model to the log odds. Namely, the logistic regression models

$$\log O(x) = \log \frac{q(x)}{1-q(x)} = \beta_0 + \beta^T x,$$

which leads to

$$P(Y=1|X=x) = q(x) = \frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}.$$

Thus, the quantity $q(x) = q(x;\beta_0,\beta)$ depends on the two parameter β_0, β. Here β_0 behaves like the intercept and β behaves like the slope vector (they are the intercept and slope in terms of the log odds).

When we observe data, how can we estimate these two parameters? In general, we will use the maximum likelihood approach to estimate them. You can view the (minus) likelihood function as the loss function in the classification (actually, we will use the log-likelihood function as the loss function). And the goal is to find the parameter via minimizing such a loss.

Recall that we observe IID random sample:

$$(X_1, Y_1), \cdots, (X_n, Y_n).$$

Let $p_X(x)$ denotes the probability density of X; note that we will not use it in estimating β_0, β. For a given pair X_i, Y_i, recalled that the random variable Y_i given X_i is just a Bernoulli random variable with parameter $q(x = X_i)$. Thus, the PMF of Y_i given X_i is

$$\begin{aligned}\mathcal{L}(\beta_0,\beta|X_i,Y_i) = P(Y=Y_i|X_i) &= q(X_i)^{Y_i}(1-q(X_i))^{1-Y_i} \\ &= \left(\frac{e^{\beta_0+\beta^T X_i}}{1+e^{\beta_0+\beta^T X_i}}\right)^{Y_i}\left(\frac{1}{1+e^{\beta_0+\beta^T X_i}}\right)^{1-Y_i} \\ &= \frac{e^{\beta_0 Y_i+\beta^T X_i Y_i}}{1+e^{\beta_0+\beta^T X_i}}.\end{aligned}$$

Note that here we construct the likelihood function using only the conditional PMF because similarly to the linear regression, the distribution of the covariate X does not depends on the parameter β_0, β. Thus, the log-likelihood function is

$$\begin{aligned}\ell(\beta_0,\beta|X_1,Y_1,\cdots,X_n,Y_n) &= \sum_{i=1}^n \log \mathcal{L}(\beta_0,\beta^T|X_i,Y_i) \\ &= \sum_{i=1}^n \log\left(\frac{e^{\beta_0 Y_i+\beta^T X_i Y_i}}{1+e^{\beta_0+\beta^T X_i}}\right) \\ &= \sum_{i=1}^n \beta_0 Y_i + \beta^T X_i Y_i - \log\left(1+e^{\beta_0+\beta^T X_i}\right).\end{aligned}$$

Our estimates are

$$\begin{aligned}\widehat{\beta}_0, \widehat{\beta} &= \underset{\beta_0,\beta}{\mathsf{argmax}}\ \ell(\beta_0, \beta | X_1, Y_1, \cdots, X_n, Y_n) \\ &= \underset{\beta_0,\beta}{\mathsf{argmin}}\ -\ell(\beta_0, \beta | X_1, Y_1, \cdots, X_n, Y_n) \\ &= \underset{\beta_0,\beta}{\mathsf{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{-\ell(\beta_0, \beta | X_i, Y_i)}_{\text{loss function}}}_{\text{empirical estimate of the loss function}},\end{aligned}$$

where the loss function is

$$-\ell(\beta_0, \beta | X_i, Y_i) = \beta_0 Y_i + \beta^T X_i Y_i - \log\left(1 + e^{\beta_0 + \beta^T X_i}\right).$$

$\widehat{\beta}_0, \widehat{\beta}$ does not have a closed-form solution in general so we cannot write down a simple expression of the estimator. Despite this disadvantage, such a log-likelihood function can be optimized by a gradient ascent approach.

8.6.1 Smooth Loss Function for Classification

The logistic regression offers another loss function for classification, which can be very useful in training a classifier from the data. To see this, assume that our classifier $c(x)$ has a parameter vector $\theta \in \mathbb{R}^p$, so we write it as $c_\theta(x)$. We assume that $x \in \mathbb{R}^d$.

Examples.

1. **Half space classifier.** A half space classifier is of the form:

   $$c_\theta(x) = I(\theta^T x - 1 > 0).$$

   You can see that the decision boundary of this classifier is the hyperplane $\{x : \theta^T x = 1\}$, where on one side of this plane the classifier outputs 1 while on the other side of plane the classifier output 0. Note: you can show that the half space classifier is equivalent to the logistic regression classifier.

2. **Radial classifier.** A radial classifier centered at μ with a radius ρ is

   $$c_\theta(x) = c_{\mu,\rho}(x) = I(\|x - \mu\| \le \rho),$$

   which outputs 1 if x is within ρ distance to μ.

3. **Multi-center radial classifier.** We can extend the radial classifier to multiple centers with multiple radius. Let $\mu_1, \cdots, \mu_K \in \mathbb{R}^d$ be K centers and $\rho > 0$ be the radius. The classifier is

   $$\begin{aligned}c_\theta(x) &= I(\|x - \mu_1\| \le \rho \text{ or } \cdots \text{ or } \|x - \mu_K\| \le \rho) \\ &= I\left(\min\{\|x - \mu_k\| : k = 1, \cdots, K\} \le \rho\right).\end{aligned}$$

   Namely, this classifier outputs 1 if x is close to any one of the K centers.

Recall that in our training/estimating of a classifier, we are trying to minimize the empirical risk

$$\min_c \widehat{R}(c), \qquad \widehat{R}(c) = \frac{1}{n}\sum_{i=1}^{n} L(c(X_i), Y_i).$$

When classifiers are indexed by the parameter θ, this problem becomes the search problem of optimal θ such that

$$\min_{\theta} \widehat{R}(\theta), \qquad \widehat{R}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(c_\theta(X_i), Y_i).$$

When using the 0-1 loss, the above numerical minimization run into a problem. Under the 0-1 loss,

$$L(c_\theta(X_i), Y_i) = I(c_\theta(X_i) \neq Y_i),$$

which is a discrete output for a continuous parameter θ. Namely, we CANNOT use the conventional method that we take derivative of $\widehat{R}(\theta)$ with respect to θ to find the optimal solution. This also makes numerical methods such as gradient descent infeasible!

In fact, the problem is two-folded, not only the loss function is non-continuous, the classifier itself is also non-continuous. Both non-continuities have made the numerical optimization of minimizing $\widehat{R}(\theta)$ very challenging!

The logistic regression offers a remedy to this problem. There are two key insights from the logistic regression. First, instead of directly considering the classifier, we may consider a generative model inside the classifier. Specifically, we model $q(x) = P(Y = 1|x)$ or the odds $O(x) = \frac{P(Y=1|x)}{P(Y=0|x)}$ or the log-odds $\log O(x)$ rather than directly modeling $c(x)$. The probability/odds/log-odds is a continuous quantity that allows a differentiable model. Note that in general, we model the log-odds since it can take any values on the whole real line.

Suppose we model the log-odds as

$$\log O(x) = f_\theta(x).$$

This implies the probability

$$P(Y = 1|x) = \frac{e^{f_\theta(x)}}{1 + e^{f_\theta(x)}}.$$

Instead of using the 0-1 loss, we consider the smooth loss function from the logistic regression model–the negative log-likelihood function:

$$\begin{aligned} -\ell(\theta|X_i, Y_i) &= -Y_i \log P(Y_i = 1|X_i) - (1 - Y_i) \log P(Y_i = 0|X_i) \\ &= -Y_i f_\theta(X_i) - Y_i \log\left[1 + e^{f_\theta(X_i)}\right] - (1 - Y_i) \log\left[1 + e^{f_\theta(X_i)}\right] \\ &= -Y_i f_\theta(X_i) - \log\left[1 + e^{f_\theta(X_i)}\right]. \end{aligned}$$

So the *empirical risk under logistic loss function* is

$$\widehat{R}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \ell(\theta|X_i, Y_i) = \frac{1}{n}\sum_{i=1}^{n} -Y_i f_\theta(X_i) - \log\left[1 + e^{f_\theta(X_i)}\right].$$

As long as our model f_θ is differentiable with respect to θ, $\widehat{R}(\theta)$ will be differentiable. This enables a fast numerical optimization of training a classifier.

The modern neural nets for classification is mostly trained under the above construction. The model $f_\theta(x)$ is constructed by a neural net.

8.6.2 Basis Logistic Regression

As a concrete example, we consider the basis approach. Let $\phi(x) \in \mathbb{R}^p$ be a vector-valued basis functions such that

$$\{\phi_1(x), \cdots, \phi_p(x)\}$$

is a basis. For simplicity, we assume that $\phi_1(x) = 1$ so that the first parameter represents the intercept.

The basis logistic regression is very straightforward. We model the log-odds:

$$\log O(x) = \theta^T \phi(x).$$

This implies that

$$P(Y = 1|X = x) = \frac{e^{\theta^T \phi(x)}}{1 + e^{\theta^T \phi(x)}}$$

and the log-likelihood is

$$\ell(\theta|x, y) = \theta^T \phi(x) y + \log\left(1 + e^{\theta^T \phi(x)}\right).$$

So the empirical risk is

$$\widehat{R}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \ell(\theta|X_i, Y_i) = \frac{1}{n}\sum_{i=1}^{n} -\theta^T \phi(X_i) Y_i - \log\left[1 + e^{\theta^T \phi(X_i)}\right].$$

The estimator of θ is

$$\widehat{\theta} = \mathsf{argmin}_\theta \quad \widehat{R}(\theta).$$

Note that you can show that $\widehat{R}(\theta)$ is a convex function of θ so gradient descent algorithm finds the minimizer quickly.

8.6.3 Complexity of classifiers

In Section 8.6.1, we have seen various classifiers indexed by a parameter. Geometrically, each classifier can be viewed as a method to partition the whole sample space of X into two parts: one for the label 0 and the other for label 1. Note that each part may contains multiple disjoint regions (e.g., the multi-center radial classifier). When we change the underlying parameter, the partition changes.

In fact, a common measure of the complexity of a classifier is based on such region changes. A classifier with a higher complexity means that we can change the underlying parameter to form flexible regions for labeling as 1 (or 0). This is formalized in the concept of VC-dimension, which we will talk more in the next lecture.