STAT 425: Introduction to Nonparametric Statistics **Winter 2018**

Lecture 13: Classification: Multivariate Approaches

Instructor: Yen-Chi Chen

In the previous lecture, we have learned that we can construct an optimal classifier if we know the true underlying probability model. Even we do not know the true model, we can estimate the model nonparametrically and then train a classifier using the estimated density function or regression function.

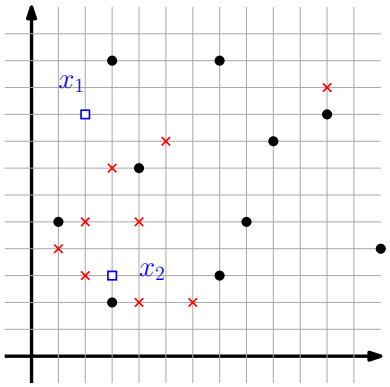
This approach is mathematically great but practically problematic because in reality the dimension of the data is often large (i.e., the feature/covariate X has many values). Recall that the MSE of a nonparametric density/regression estimator often has a convergence rate of the order $O(n^{-\frac{4}{4+d}})$. This rate is extremely slow when d is large. Thus, converting a nonparametric estimator to a classifier may not work in practice.

Now we are going to introduce a couple of other classifiers that does not require an explicit estimation of the density function or the regression function.

13.1 k-NN Approach

The k-NN approach can be applied to classification as well. 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 as 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.

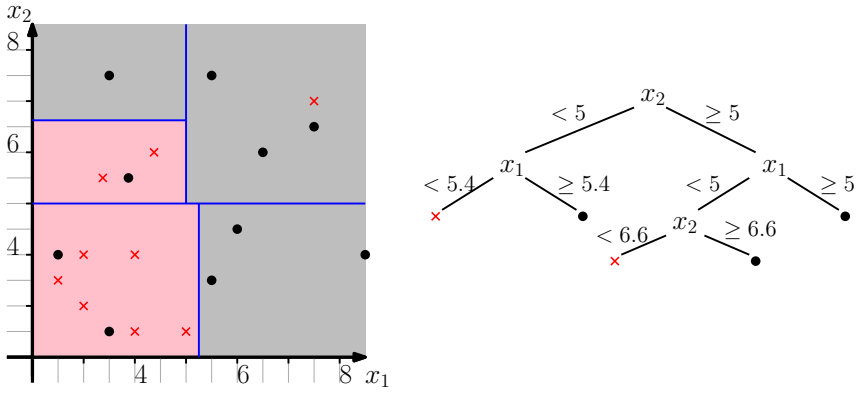
How do we choose k? The choice of k is often done by a technique called cross-validation. The basic principle is: we split the data into two parts, use one part to train our classifier and evaluate the performance on the other part. Repeating the above procedure multiple times and applying it to each 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.

13.2 Decision Tree

Decision tree is another common approach in classification. 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.

13.3 Logistic Regression

The logistic regression is a regression model that is commonly applied to classification problems as well. Like the method of converting a regression estimator to a classifier, the logistic regression use a regression function as an intermediate step and then form a classifier. We first talk about some interesting examples.

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 such as the Newton-Raphson[1].

13.4 Training Classifiers as an Optimization Problem

Even without any probabilistic model, classification problem can be viewed as an optimization problem. The key element is to replace the risk function $\mathbb{E}(L(c(X), Y)$ by the empirical risk

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

[1] some references can be found: https://www.cs.princeton.edu/~bee/courses/lec/lec_jan24.pdf

Consider a collection of classifiers $\mathcal{C}$. Then the goal is to find the best classifier $c^* \in \mathcal{C}$ such that the the empirical risk $\widehat{R}_n(c)$ is minimized. Namely,

$$c^* = \underset{c \in \mathcal{C}}{\mathsf{argmin}}\ \widehat{R}_n(c).$$

Ideally, this should work because

$$\mathbb{E}(\widehat{R}_n(c)) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} L(c(X_i), Y_i)\right) = \mathbb{E}(L(c(X_1), Y_1)) = R(c) \tag{13.1}$$

is the actual risk function. Namely, $\widehat{R}_n(c)$ is an unbiased estimator of the risk function.

Here are some concrete examples.

Linear classifier. Assume that we consider a linear classifier:

$$c_{\beta_0,\beta}(x) = I(\beta_0 + \beta^T x > 0),$$

where β_0 is a number like the intercept and β is the vector like the slope of each covariate/feature. Namely, how we assign the class label purely depends on the value of $\beta_0 + \beta^T x$. If this value is positive, then we give it a label 1. If the value is negative, then we give it a label 0. Then the set

$$\mathcal{C}_{\mathsf{lin}} = \{c_{\beta_0,\beta} : \beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^d\}$$

is a collection of all linear classifier. The idea of empirical risk minimization is to find the classifier in $\mathcal{C}_{\mathsf{lin}}$ such that the empirical risk $\widehat{R}_n(\cdot)$ is minimized. Because every classifier is indexed by the two quantities (parameters) β_0, β, finding the one that minimizes the empirical risk is equivalent to finding the best β_0, β minimizing $\widehat{R}_n(\cdot)$.

Logistic regression. Similar to the linear classifier, the logistic regression can be viewed as a classifier of the form

$$\widetilde{c}_{\beta_0,\beta}(x) = I\left(\frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}} > \frac{1}{2}\right).$$

Then we can define the set

$$\mathcal{C}_{\mathsf{logistic}} = \{\widetilde{c}_{\beta_0,\beta} : \beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^d\}$$

as the collection of all classifiers from a logistic regression model. The MLE approach becomes an empirical risk minimization method using a particular loss function – the log-likelihood function.

Decision tree (fixed k, fixed leaves). The decision tree classifier $\widehat{c}_{\mathsf{DT}}$ can be viewed as a classifier from the empirical risk minimization as well. For simplicity, we assume that the regions/leaves of the decision trees $R_1, \cdots, R_k$ are fixed. Then any decision tree classifier can be written as

$$c_{\mathsf{DT}}(x) = \sum_{j=1}^{k} \alpha_j I(x \in R_j),$$

where $\alpha_1, \cdots, \alpha_k$ are quantities/parameters that determine how we will predict the class label of region $R_1, \cdots, R_k$, respectively. And the collection of all possible classifiers will be

$$\mathcal{C}_{\mathsf{DT}} = \{c_{\mathsf{DT}}(x) : \alpha_j \in \{0,1\}, j = 1, \cdots, k\}.$$

Note that there are only 2^k classifiers in the set $\mathcal{C}_{\mathsf{DT}}$. The estimator $\widehat{c}_{\mathsf{DT}}$ is just the one that minimizes the empirical loss $\widehat{R}_n(\cdot)$ with a $0-1$ loss.

Decision tree (fixed k, non-fixed leaves). When the regions/leaves are not fixed, the collection of all possible decision tree classifiers is much more complex. Here is an abstract way of describing such a collection. Recall that at each split of the tree, we pick one feature and a threshold (e.g., $x_1 > 10$ versus $x_1 \leq 10$), every split can be represented by two indices: the feature index (which feature is this split occurs), and the threshold index (what is the split level). Thus, a split is characterized by a pair (m, λ), where $m \in \{1, 2, \cdots, d\}$ and $\lambda \in \mathbb{R}$. If a tree has k leaves, there will be $k-1$ splits. So any decision tree classifier is indexed by

$$\alpha_1, \cdots, \alpha_k, (m_1, \lambda_1), \cdots, (m_{k-1}, \lambda_{k-1}).$$

Namely, given a set of these values, we can construct a unique decision tree classifier. Thus, the collection of all decision tree with k leaves can be written as

$$\mathcal{C}_{\mathsf{DT}}(k) = \{c_{\mathsf{DT}}(x) = c_{\alpha,m,\lambda}(x) : \alpha_j \in \{0,1\}, (m_\ell, \lambda_\ell) \in \{1, 2, \cdots, d\} \times \mathbb{R}, j = 1, \cdots, k, \ell = 1, \cdots, k-1\}.$$

In reality, when we train a decision tree classifier with a fixed number k, the regions are also computed from the data. Thus, we are actually finding $\widehat{c}_{\mathsf{DT}}$ such that

$$\widehat{c}_{\mathsf{DT}} = \underset{c \in \mathcal{C}_{\mathsf{DT}}(k)}{\mathsf{argmin}} \ \widehat{R}_n(c).$$

Decision tree (both k and leaves are non-fixed). If we train the classifier with k being un-specified, then we are finding $\widehat{c}_{\mathsf{DT}}$ from $\bigcup_{k \in \mathbb{N}} \mathcal{C}_{\mathsf{DT}}(k)$ that minimizes the empirical risk. However, if we really consider all possible k, such an optimal decision tree is not unique and may be problematic. When $k > n$, we can make each leave contains at most and the predicted label is just the label of that observation. Such a classifier has 0 empirical risk but it may have very poor performance in future prediction because we are **overfitting** the data. Overfitting the data implies that $\widehat{R}_n(c)$ and $R(c)$ are very different, even if $\widehat{R}_n(c)$ is an unbiased estimator of $R(c)$!

Why will this happen? $\widehat{R}_n(c)$ is just the sample-average version of $R(c)$, right? Is this contradict to the law of large number that $\widehat{R}_n(c)$ converges to $R(c)$?

It is true that $\widehat{R}_n(c)$ is an unbiased estimator of $R(c)$ and yes indeed the law of large number is applicable in this case. BUT a key requirement for using the law of large number is that we assume c is fixed. Namely, if the classifier c is fixed, then the law of large number guarantees that the empirical risk $\widehat{R}_n(c)$ converges to the true risk function $R(c)$.

However, when we are finding the best classifier, we are consider many many many possible classifiers c. Although for a given classifier c the law of large number works, it may not work when we consider many classifiers. The empirical risk minimization works if

$$\sup_{c \in \mathcal{C}} \left| \widehat{R}_n(c) - R(c) \right| \xrightarrow{P} 0$$

Namely, the convergence is *uniform* for all classifiers in the collection that we are considering. In the next few lectures we will be talking about how the above uniform convergence may be established.

Remark.

- **Regression problem.** The empirical risk minimization method can also be applied to regression problem. We just replace the classifier by the regression function and the loss function can be chosen as the L_2 loss (squared distance). In this formulation, we obtain

$$R(m) = \mathbb{E}\left(\|Y - m(X)\|^2\right)$$

and

$$\widehat{R}_n(m) = \frac{1}{n}\sum_{i=1}^{n} \|Y_i - m(X_i)\|^2.$$

Estimating a regression function can thus be written as an empirical risk minimization – we minimizes $\widehat{R}_n(m)$ for $m \in \mathcal{M}$ to obtain our regression estimator. The set $\mathcal{M}$ is a collection of many regression functions. Similar to the classification problem, we need a uniform convergence of the empirical risk to make sure we have a good regression estimator.

- **Penalty function.** Another approach to handle the difference $\sup_{c\in\mathcal{C}} |\widehat{R}_n(c) - R(c)|$ is to add an extra quantity to $\widehat{R}_n(c)$ so that such a uniform difference is somewhat being controlled. Instead of minimizing $\widehat{R}_n(c)$, we minimizes

$$\widehat{R}_n(c) + \mathcal{P}_\lambda(c),$$

where $\mathcal{P}_\lambda$ is a penalty function. This is just the penalized regression approach but now applying it to a classification problem. When the penalty function is chosen in a good way, we can make sure the optimal classifier/regression estimator from the (penalized) empirical risk minimization indeed has a very small risk.