CA684 Jia Yuan Yu

2: Supervised Classification

DCU February 21, 2014

These notes are based on the book "A Probabilistic Theory of Pattern Recognition."

1 Stochastic model

In classification, discrimination, or pattern recognition, we often consider a sequence of observation-label pairs

$$(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n), (X_{n+1}, Y_{n+1}), \dots$$

each taking values in $\mathbb{R}^d \times \{0, 1\}$ (for simplicity, we consider two possible labels). In a stochastic model, we assume that this sequence is i.i.d. (independent and identically distributed). In particular, Y_j is sometimes called a Bernoulli random variable, and X_j a random vector.

Machine learning is about decision-making under uncertainty. In the stochastic model, we capture the notion of uncertainty with random variables. Later in the course, we will see other models of uncertainty.

A classifier is a function $g : \mathbb{R}^d \to \{0, 1\}$. Classifier g makes an error on X_j if $g(X) \neq Y_j$. We define the probability of error of a classifier g as

$$L(g) = \mathbb{P}(g(X_j) \neq Y_j),$$

which is independent of j.

1.1 Bayes classifier

The best possible classifier g^* is

$$g^* \in \arg \min_{g:\mathbb{R}^d \to \{0,1\}} \mathbb{P}(g(X_j) \neq Y_j). \tag{1}$$

The function g^* is called the Bayes classifier, and $L(g^*)$ is called the Bayes error. This classifier is unknown when the distribution of (X_j, Y_j) is unknown.

1.2 Learned classifier

Let g_n denote a classifier constructed from $(X_1, Y_1), \dots, (X_n, Y_n)$. The sequence $(X_1, Y_1), \dots, (X_n, Y_n)$ is the training sequence for g_n. The integer n is the size of the training set. The performance of this classifier is given by the (random) probability of error

$$L(g_n) = \mathbb{P}(g_n(X_{n+1}) \neq Y_{n+1} \mid X_1, Y_1, \dots, X_n, Y_n),$$

which is a random variable when the training data is yet unseen.

Before time 1, we decide how to construct the sequence of classifiers $\{g_1, g_2, \ldots\}$. At that time, we have not observed the realizations of any of the observation-label pairs yet. At that time, we view $(X_1, Y_1), (X_2, Y_2), \ldots$ as random variables. However, at time $n + 1$, we have observed the realizations $(x_1, y_1), (x_2, y_2), \ldots$ of the random variables $(X_1, Y_1), (X_2, Y_2), \ldots$ Then, g_n is no longer random, and we apply g_n to X_{n+1}.

1.3 What makes a good sequence of classifiers?

A sequence of classifiers or rule is said to be consistent if

$$\lim_{n\to\infty} \mathbb{E}L(g_n) = L(g^*),$$

i.e., its error converges in expectation to the Bayes error (which is the best we can hope for). A rule is strongly consistent if

$$\mathbb{P}(\lim_{n\to\infty} L(g_n) = L(g^*)) = 1.$$

A rule is said to be universally consistent if it is consistent for every distribution of (X_j, Y_j).

Ideally, we would like to bound

$$\mathbb{P}(L(g_n) \geq L(g^*) + \varepsilon),$$

but such bounds depend on the distribution of (X_j, Y_j). However, we will see that such universal bounds do not exist.

Moreover, we can not compare two rules $\{g_n\}$ and $\{g'_n\}$ universally: For every given rule $\{g_n\}$, we can find a distribution of data (X_j, Y_j) and a rule $\{g'_n\}$ such that

$$\mathbb{E}L(g'_n) < \mathbb{E}L(g_n) \quad \text{for all } n.$$

Hence, it's also meaningless to compare the error rate of rules with simulations. No classification rule is better than another in terms of error rates! However, they may be compared with respect to other properties, like computational complexity, etc., or when the distribution of data is restricted, or when the set of rules is restricted.

1.4 Restriction to a class of classifiers

Consider a class $\mathcal{C}$ of classifiers, e.g., the set of k-nearest neighbour classifiers. Let

$$L \triangleq \inf_{g\in\mathcal{C}} \mathbb{P}(g(X_j) = Y_j).$$

When restricted to $\mathcal{C}$, we can actually pick a classifier that is good with respect to all distributions of the data. Let

$$g_n \in \arg\min_{g\in\mathcal{C}} \frac{1}{n} \sum_{i=1}^{n} 1_{[g(X_i)\neq Y_i]}.$$

This g_n minimizes the empirical risk (a sum of Bernoulli random variables) and for every $\varepsilon > 0$ and distribution of data, its probability of error is bounded by

$$\mathbb{P}(L(g_n) > L + \varepsilon) \leq 8(n^V + 1)e^{-n\varepsilon^2/128},$$

where V is a constant that depends on $\mathcal{C}$ only.

2 Plug-in classifier

Consider the pair of random variables (X, Y) taking values in $\mathbb{R}^d \times \{0, 1\}$. Let $x \in \mathbb{R}^d$ denote a possible realization of X. We denote the conditional probability of the event $\{Y = 1\}$ given that $X = x$ by

$$\eta(x) \triangleq \mathbb{P}(Y = 1 \mid X = x).$$

The Bayes classifier (recall (1)) can be expressed as

$$g^*(x) = \begin{cases} 1, & \text{if } \eta(x) > 1/2, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

A plug-in classifier uses an approximation $\hat{\eta}$ to η:

$$g(x) = \begin{cases} 1, & \text{if } \hat{\eta}(x) > 1/2, \\ 0, & \text{otherwise.} \end{cases}$$

3 Nearest neighbour classifiers

For every point x, we can compute its distances to the n data points:

$$\|x - X_1\|, \ldots, \|x - X_n\|.$$

We can reorder the data points such that

$$\|x - X_{i_1}\| \leq \ldots \leq \|x - X_{i_n}\|.$$

Let $N(x, k)$ denotes the set of k nearest neighbours of x, i.e.,

$$N(x, k) = \{X_{i_1}, \ldots, X_{i_k}\}.$$

Given n training samples, the k-NN classifier is the function

$$g_n(x) = \begin{cases} 1, & \text{if } \sum_{i=1}^n w_{n,i}(x,k) 1_{[Y_i=1]} > \sum_{i=1}^n w_{n,i}(x,k) 1_{[Y_i=0]}, \\ 0, & \text{otherwise,} \end{cases}$$

where

$$w_{n,i}(x, k) = \begin{cases} 1/k, & \text{if } X_i \in N(x, k), \\ 0, & \text{otherwise.} \end{cases}$$

Basically, we compare two weighted sums of Bernoulli random variables, i.e., we do majority voting between the k nearest neighours. We can break ties arbitrarily, or choose k odd to avoid ties. Computing $g(x)$ directly takes time $O(nkd)$. With preprocessing (partitioning, tree structure), we can compute $g(x)$ faster.

Theorem 3.1. *For odd and fixed k and all data distribution,*

$$\lim_{n\to\infty} \mathbb{E}L(g_n) = L_{kNN},$$
$$L_{kNN} \le L(g^*) + 1/\sqrt{ek}.$$

See (PTPR Theorem 5.2 and 5.6) for proof.

Later, we will see that this rule is universally consistent, i.e., $\lim_{n\to\infty} \mathbb{E}L(g_n) = L(g^*)$ if $k, n \to \infty$ and $k/n \to 0$.

4 Linear classifiers

We denote the elements of the vector $x \in \mathbb{R}^d$ by $(x^1, \ldots, x^d)$. A linear classifier with weights $a_0, a_1, \ldots, a_d$ is

$$g(x) = \begin{cases} 1, & \text{if } \sum_{i=1}^d a_i \cdot x^i + a_0 > 0, \\ 0, & \text{otherwise.} \end{cases}$$

The probability of error of g is $L(g)$. We define the empirical risk estimate

$$\hat{L}_n(g) = \frac{1}{n} \sum_{i=1}^n 1_{[g(X_i)\neq Y_i]}.$$

From $X_1, \ldots, X_n$, pick d points $X_{i_1}, \ldots, X_{i_d}$ (in general positions with probability one). Let a denote the coefficents of the hyperplane $\sum_{i=1}^d a_i \cdot x^i + a_0 = 0$ that passes through these d points. Each such hyperplane gives you two classifiers. In total, we have $2\binom{n}{d}$ classifiers, call this class $\mathcal{C}$. Consider the following linear classifier

$$\hat{g} \in \arg\min_{g\in\mathcal{C}} \hat{L}_n(g).$$

This classifier is good (we'll see how good later in the VC theory section). Observe that for every linear classifier g', we can find a $g \in \mathcal{C}$ such that the two classifiers agree on all data points except possibly for d such points. However, the above optimization problem is hard.

5 Get your hands dirty

Homework:

1. Verify that (2) and (1) really give the same g^*.
2. Install Python, Scipy (`www.scipy.org`), scikit-learn (`scikit-learn.org/stable/index.html`).
3. Follow a Numpy tutorial (`wiki.scipy.org/Tentative_NumPy_Tutorial`) in iPython.

4. In class, we will go over an example (`scikit-learn.org/stable/tutorial/basic/tutorial.html#introduction`)
5. Try a different example from class (e.g., `scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html`)
6. Think about your final project (20 % of final mark); what data set would you like to use? (For example, Dublin city makes available `www.dublinked.ie/datastore/datastore.php`, City of Cologne `sumo-sim.org/userdoc/Data/Scenarios/TAPASCologne.html`, or elsewhere.)
7. You can choose to do a project on contemporary topics in machine learning (for ideas, see `nips.cc/Conferences/2013/Program/schedule.php?Session=Workshops`).