Chapter 2: Overview of Supervised Learning

Back to the Contents

2.1 Introduction

The main goal of unsupervised learning is to predict the Output(responses) via Input(predictors).

2.2 Variable Types and Terminology

Overall, there are mainly two kinds of outputs: quantitative outputs(Y) and qualitative outputs(G).

For Y, it means that sime measurements are bigger than others, and the measurements close in value means the variables are close in nature. To predict quantitative outputs, we name the methods as regression.

For G, it means that the values lay in a finite set and there is no explicit ordering among them. To code them, we can use dummy variables or other methods.To predictict qualitative outputs, we name the methods as classification.

There are also other kind of outputs, such as ordered categorical outputs.

Given the data

$$X = (x_1, \cdots, x_N)^T,$$

which is a $N \times p$ matrix, our task is to predict $\hat{Y} \in \mathbb{R}$, or $\hat{G} \in \mathcal{G}$. We have a set of training data: $(x_i, y_i), i = 1, \cdots, N$

2.3 Two Simple Approaches to Prediction: Least Squares and Nearesr Neighbors.

2.3.1 Linear Models and Least Squares

Input: $X = (X_1, \cdots, X_p)^T$, predict Y via $\hat{Y} = X^T\hat{\beta}$ (intercept included). The function $f(X) = X^T\beta$ is a linear function.

To fit this model, we have

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2 = (y - X'\beta)^T(y - X'\beta).$$

After minimizing it by taking derivatives, we get

$$\hat{\beta} = (X'X)^{-1}X'y$$

2.3.2 Nearest Neighbor Methods

We find k observations with x_i closest to x in input space and average their responses by

$$\hat{Y}(x) = \frac{1}{k}\sum_{x_i \in N_k(x)} y_i,$$

where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training sample.

2.3.3 From Least Square to Nearest Neighbors

First we consider the complexity. To fit the least square model, we need to fit p parameters, while for k-NN, a single parameter should be fit. However, for k-NN, if the neighborhoods were non-overlapping, there would be N/k neighborhoods and we would fit one parameter(a mean) in each neighborhood, and N/k is generally bigger than p.

From bias and variance aspects, generally, linear models has low variance and potnetially high bias, while k-NN is wiggly and unstable with high variance and low bias.

Considering where the constructed data came from, we have the two possible scenarios:

Scenario 1 : The training data in each class were generated from bivariate Gaussian distributions with uncorrelated components and different means.

Scenario 2 : The training data in each class came from a mixture of 10 lowvariance Gaussian distributions, with individual means themselves distributed as Gaussian.

For Scenario 1, linear decision boundary is the best one[See Chapter 4]. For 2, other methods can be better, such as k-NN.

Each method has its own situations for which it works best; in particular linear regression is more appropriate for Scenario 1 above, while nearest neighbors are more suitable for Scenario 2. The time has come to expose the oracle!

2.4 Statistical Decision Theory

2.4.1 Theory for Quantitative Output

With joint probability $P(X, Y)$,to find f, such that $f(X)$ predicts Y, we have the EPE(expected prediction error) as follows:

$$EPE(f) = E(L(Y, f(X))),$$

where $L(\cdot, \cdot)$ here is the loss function. If $L(Y, f(X)) = (Y - f(X))^2$, we have

$$EPE(f) = E(Y - f(X))^2 = E_X E_{Y|X}([Y - f(X)]^2|X).$$

To minimize it, let $c := f(X)|_{X=x}$, then compute

$$\min_c E(Y - c)^2$$

and get $c_0 = EY$. Thus, we know $f(x) = E(Y|X)$.

2.4.2 Implementation to k-NN and Linear Model

The nearest-neighbor methods attempt to directly implement this recipe using the training data. At each point x, we might ask for the average of all those y_is with input $x_i = x$. Since there is typically at most one observation at any point x, we settle for

$$\hat{f}(x) = Ave(y_i|x_i \in N_k(x)),$$

where "Ave" denotes average,and $N_k(x)$ is the neighborhood containing the k points in $\mathcal{T}$ closest to x. Two approximations are happening here:

• expectation is approximated by averaging over sample data;

• conditioning at a point is relaxed to conditioning on some region "close" to the target point.

Fact With joint probability $P(X,Y)$, $\hat{f}(x) \to E(Y|X=x)$ as $k, N \to \infty$ and $k/N \to 0$. However, as dimension p increases, the approximations will get worse.

For linear model, we assume the function f of the form $f(x) \approx x^T\beta$, and then plug it in the definition of EPE, we have

$$EPE(f) = E(Y - X'\beta)^2.$$

After minimizing it, we have

$$\hat{\beta} = [E(X'X)]^{-1}E(XY).$$

Note: we do not need it conditioned on X.

So both k-nearest neighbors and least squares end up approximating conditional expectations by averages. But they differ dramatically in terms of model assumptions:

• Least squares assumes $f(x)$ is well approximated by a globally linear function.

• k-nearest neighbors assumes $f(x)$ is well approximated by a locally constant function.

2.4.3 Bayes Classifier

For outputs which are categorical variable G, the loss function can be represented as $L_{K\times K}$, where $K = card(\mathcal{G})$.$L_{i,j}$ is the price paid for classifying G belonging to $\mathcal{G}_\rangle$ as$\mathcal{G}_j$.(0 on the diagonal and non-negative elsewhere)

More commonly, we use zero-one loss function which can be represented as

$$L = J_{2\times 2} - Diag(1)_{2\times 2} \text{ or } J_{3\times 3} - Diag(1)_{3\times 3},$$

which is 0 on the diagonal and 1 elsewhere.

With $P(G,X)$, we have

$$EPE = E_X \sum_{k=1}^{K} L(\mathcal{G}_k, \hat{G}(X))P(\mathcal{G}_k|X).$$

With 0-1 loss function, we can minimize it point-wisely, which is known as Bayes-optimal classifier.

The error rate of this classifier is called Bayes rate.

2.5 Local Methods in High Dimensions.

In this section, it is explained that why k-NN will lead to worse results as dimension p increases as well as why linear model can be more stable when rigid assumptions are satisfied.

There are three main points listed to explain why k-NN will fail when p is large:

(1). To capture r proportion of data to form a local average, we need to cover more of the range of each input variable, as p increases.

For example, for a hypercube,r proportion of each dimension means r^p in the whole input space. So, if we need r proportion data of the whole input space, $r^{1/p}$ data on each dimension will be needed. As p increases, $r^{1/p}$ will also increase converging to 1.

(2). Sampling density decreases exponentially, when p increases.

(3). Average distance from other observations increase, and thus, the estimate tends to be 0 more.

For example, consider N data points uniformly distributed in a p-dimensional unit ball centered at the origin. Suppose we consider a nearest-neighboor estimate at the origin, The median distance from the origin to the closest data point is give by

$$d(p, N) = (1 - \frac{1}{2}^{1/N})^{1/p}$$

For $N = 500, p = 10, d(p, N) \approx 0.52$, more than halfway to the boundary. Hence most data points are closer to the boundary of the sample space than to any other data point. An intuitive understanding to this example is similar to the example of the first point.

Now, we discuss why linear model can be more robust to p if the linear assumptions hold.

The linear assumption:

$$Y = X^T\beta + \varepsilon, \varepsilon \sim N(0, \sigma^2).$$

Now, we have

$$EPE(x_0) = \sigma^2 + E_{\mathcal{T}} x_0'(X'X)^{-1}x_0\sigma^2.$$

As N is large enough,$\mathcal{T}$ is selected randomly,assume $E(X) = 0$, $X'X \to NCov(X)$, we have

$$E_{x_0}EPE(x_0) \sim \sigma^2 + \sigma^2(p/N).$$

This is linearly proportional to p. As N is large and σ^2 is small, the growth of it can be ignored when p is increasing.

Overall, By relying on rigid assumptions, the linear model has no bias at all and negligible variance, while the error in 1-nearest neighbor is substantially larger. However, if the assumptions are wrong, all bets are off and the 1-nearest neighbor may dominate.

2.6 Statistical Models, Supervised Learning and Function Approximation

To overcome the dimensionality problems,we anticipate using other classes of models for $f(x)$. Now, er discuss a frame work for incorporating them into the prediction problem.

2.6.1 A Statistical Model for the Joint Distribution $P(X, Y)$

Suppose our data arose from a statistical model

$$Y = f(X) + \varepsilon,$$

where the random error ε has $E(\varepsilon) = 0$ and is independent of X.

Note that for this model, $f(x) = E(Y|X = x)$, and in fact the conditional distribution $P(Y|X)$ depends on X only through the conditional mean $f(x)$.

This additive error model is typically not used for qualitative outputs G. In this case, the target function $p(X)$ is the conditional density $P(G|X)$, and is modeled directly.

2.6.2 Supervised Learning

In this section, the task of supervised learning is briefly introduced. That is to learn f by example through a teacher.

2.6.3 Function Approximation

The function $f(x)$ has domain equal to the p-dimensional input subspace, and is related to the data via a model such as $y_i = f(x_i) + \varepsilon_i$. For convenience, in this chapter we will assume the domain is $\mathbb{R}^p$, a p-dimensional Euclidean space, although in general the inputs can be of mixed type.

Note: many of the approximations we will encounter as associated a set of parameters θ that can be modified to suit the data at hand.

For additive error models, RSS seems to be a reasonalbe criterion. However, least square is not the only criterion used and in some cases it would not make much sense, although it is generally convenient.

A more general principle for estimation is maximum likelihood estimation.

2.7 Structured Regression Models

First we discuss the difficulty of the problem. Consider

$$RSS(f) = \sum_{i=1}^{N} (y_i - f(x_i))^2,$$

minimizing it without any constrain leads to infinitely many solutions: any functions $\hat{f}$ passing all the training points. And the results turn out to be useless. Thus, we need some constrains of f when finding the function f.

In general the constraints imposed by most learning methods can be described as complexity restrictions of one kind or another. This usually means some kind of regular behavior in small neighborhoods of the input space. That is, for all input points x sufficiently close to each other in some metric, $\hat{f}$ exhibits some special structure such as nearly constant, linear or low-order polynomial behavior. The estimator is then obtained by averaging or polynomial fitting in that neighborhood.

The strength of the constraint is dictated by the neighborhood size. The larger the size of the neighborhood, the stronger the constraint, and the more sensitive the solution is to the particular choice of constraint. For example, local constant fits in infinitesimally small neighborhoods is no constraint at all; local linear fits in very large neighborhoods is almost a globally linear model, and is very restrictive.

The nature of the constraint depends on the metric used. Some methods, such as kernel and local regression and tree-based methods, directly specify the metric and size of the neighborhood. The nearest-neighbor methods discussed so far are based on the assumption that locally the function is constant; close to a target input x_0, the function does not change much, and so close outputs can be averaged to produce $\hat{f}(x_0)$. Other methods such as splines, neural networks and basis-function methods implicitly define neighborhoods of local behavior.

One fact should be clear by now. Any method that attempts to produce locally varying functions in small isotropic neighborhoods will run into problems in high dimensions — again the curse of dimensionality.

2.8 Classes of Restricted Estimators

The variety of nonparametric regression techniques or learning methods fall into a number of different classes depending on the nature of the restrictions imposed. Here three croad classes are described.

2.8.1 Roughness Penalty and Bayesian Methods

Here the class of functions is controlled by explicitly penalizing $RSS(f)$ with a roughness penalty

$$PRSS(f; \lambda) = RSS(f) + \lambda J(f).$$

The user-seected penalty $J(f)$ here controls some property of f.It will be large for functions f that vary too rapidly over small regions of input space. The amount of penalty is dictated by $\lambda \geqslant 0$. For $\lambda = 0$, no penalty is imposed, and any interpolating function will do.

Penalty function, or regularization methods, express our prior belief that the type of functions we seek exhibit a certain type of smooth behavior, and indeed can usually be cast in a Bayesian framework.

2.8.2 Kernel Methods and Local Regression

These methods can be thought of as explicitly providing estimates of the regression function or conditional expectation by specifying the nature of the local neighborhood, and of the class of regular functions fitted locally. The local neighborhood is specified by a kernel function $K_\lambda(x_0, x)$, which assigns weights to points x in a region around x_0.

In general, we can define a local regression estimate of $f(x_0)$ as $f_{\hat{\theta}}(x_0)$, where $\hat{\theta}$ minimizes

$$RSS(f_\theta, x_0) = \sum_{i=0}^{N} K_\lambda(x_0, x_i)(y_i - f_\theta(x_i))^2,$$

and f_θ is some parameterized function, such as low-ordered polynimial.

Note: basically, $K_\lambda(x_0, x)$ is larger when x is closer to x_0. It means that the points in the neighborhood of x_0 are considered to be more important.

2.8.3 Basis Functions and Dictionary Methods

Here the model of function f is a linear expansion of basis functions

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x),$$

where each of the h_m is a function of the input x, and the term linear here refers to the action of the parameters θ. In some cases, the sequence of basis functions is prescribed, such as a basis for polynomials in x of total degree M.

The adaptively chosen basis function methods are known as dictionary methods, where one has available a possibly infinite set or dictionary $\mathcal{D}$ of candidate basis functions from which to choose, and models are built up by employing some kind of search mechanism.

2.9 Model Selection and the Bias-Variance Tradeoff

For the three classes drescribed in the last section, we need to determine:

- the multiplier of the penalty term;
- the width of the kernel;
- or the number of basis functions.

The k-NN regression fit $\hat{f}_k(x_0)$ usefully illustrates the competing forces that affect the predictive ability of such approximations. Suppose the data arise from a model $Y = f(X) + \varepsilon$, with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$. For simplicity here we assume that the values of x_i in the sample are fixed in advance. The expected prediction error at x_0 can be decomposed:

$$
\begin{aligned}
EPE_k(x_0) &= E[(Y - \hat{f}_k(x_0))^2 | X = x_0] \\
&= \sigma^2 + [Bias^2(\hat{f}_k(x_0)) + Var_{\mathcal{T}}(\hat{f}_k(x_0))] \\
&= \sigma^2 + [f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)})]^2 + \frac{\sigma^2}{k}.
\end{aligned}
$$

The first term σ^2 is beyond our control, even if we know the true $f(x_0)$.

The second term will likely increase with k, if the true function is reasonably smooth.

The second term is simply the variance of an average here, and decreases as teh inverse of k.

Thus, as k varies, there is a bias-variance tradeoff.

More generally, as the model complexity of our procedure is increased, the variance tends to increase and the squared bias tends to decrease. The opposite behavior occurs as the model complexity is decreased. For k-nearest neighbors, the model complexity is controlled by k.