Machine Learning Notes

Presidency University

Ritwick Mondal

9th jan,2022

Contents

Statistical learning plays a key role in many areas of science, finance and industry. Here are some examples of learning problems:

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demo- graphic, diet and clinical measurements for that patient.

Statistical learning plays a key role in many areas of science, finance and industry. Here are some examples of learning problems:

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demo- graphic, diet and clinical measurements for that patient.
- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.

Statistical learning plays a key role in many areas of science, finance and industry. Here are some examples of learning problems:

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demo- graphic, diet and clinical measurements for that patient.
- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.
- Identify the numbers in a handwritten ZIP code, from a digitized image.

Statistical learning plays a key role in many areas of science, finance and industry. Here are some examples of learning problems:

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demo- graphic, diet and clinical measurements for that patient.
- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.
- Identify the numbers in a handwritten ZIP code, from a digitized image.
- Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person's blood.

- Identify the risk factors for prostate cancer, based on clinical and demographic variables.

- Identify the risk factors for prostate cancer, based on clinical and demographic variables.
- We have an outcome measurement, usually **quantitative** (such as a stock price) or **categorical** (such as heart attack/no heart attack), that we wish to predict based on a set of features (such as diet and clinical measurements [1])

There is a set of variables that might be denoted as **inputs**, which are measured or preset. These have some influence on one or more **outputs**.Main purpose is to use the inputs to predict the values of the outputs. This exercise is called **supervised learning**.

We have used the more modern language of machine learning.

In the statistical literature the inputs are often called the **predictors**, a term we will use interchangeably with inputs, and more classically the independent variables. In the pattern recognition literature the term features is preferred, which we use as well. The outputs are called the **responses**, or classically the **dependent variables.**

Variable Types

The outputs vary in nature among the examples.

- In the glucose prediction example, the output is a quantitative measurement, where some measurements are bigger than others, and measurements close in value are close in nature.

Variable Types

The outputs vary in nature among the examples.

- In the glucose prediction example, the output is a quantitative measurement, where some measurements are bigger than others, and measurements close in value are close in nature.
- In the famous Iris discrimination example due to R. A. Fisher, the output is qualitative (species of Iris) and assumes values in a finite set G={Virginica, Setosa and Versicolor}.

Variable Types

The outputs vary in nature among the examples.

- In the glucose prediction example, the output is a quantitative measurement, where some measurements are bigger than others, and measurements close in value are close in nature.
- In the famous Iris discrimination example due to R. A. Fisher, the output is qualitative (species of Iris) and assumes values in a finite set G={Virginica, Setosa and Versicolor}.
- In the handwritten digit example the output is one of 10 different digit classes: $G = 0, 1, ..., 9$.In both of these there is no explicit ordering in the classes, and in fact often descriptive labels rather than numbers are used to denote the classes

Variable Types

The outputs vary in nature among the examples.

- In the glucose prediction example, the output is a quantitative measurement, where some measurements are bigger than others, and measurements close in value are close in nature.
- In the famous Iris discrimination example due to R. A. Fisher, the output is qualitative (species of Iris) and assumes values in a finite set G={Virginica, Setosa and Versicolor}.
- In the handwritten digit example the output is one of 10 different digit classes: $G = 0, 1, ..., 9$.In both of these there is no explicit ordering in the classes, and in fact often descriptive labels rather than numbers are used to denote the classes
- Qualitative variables are also referred to as categorical or discrete variables as well as factors.

- For both types of outputs it makes sense to think of using the inputs to predict the output. Given some specific atmospheric measurements today and yesterday, we want to predict the ozone level tomorrow. Given the grayscale values for the pixels of the digitized image of the handwritten digit, we want to predict its class label.

- Qualitative variables are typically represented numerically by codes. The easiest case is when there are only two classes or categories, such as “success” or “failure,” “survived” or “died.” These are often represented by a single binary digit as 0 or 1, or else by -1 and 1. For reasons that will become apparent, such numeric codes are sometimes referred to as targets. When there are more than two categories, several alternatives are available. The most useful and commonly used coding is via dummy variables. Here a K-level qualitative variable is represented by a vector of K binary variables , only one of which is “on” at a time. Although more compact coding schemes are possible, dummy variables are symmetric in the levels of the factor.

Terminology

- We will typically denote an input variable by the symbol X. If X is a vector, its components can be accessed by subscripts Xj.

Terminology

- We will typically denote an input variable by the symbol X. If X is a vector, its components can be accessed by subscripts Xj.
- Quantitative outputs will be denoted by Y , and qualitative outputs by G (for group).

Terminology

- We will typically denote an input variable by the symbol X. If X is a vector, its components can be accessed by subscripts Xj.
- Quantitative outputs will be denoted by Y , and qualitative outputs by G (for group).
- We use uppercase letters such as X, Y or G when referring to the generic aspects of a variable.

Terminology

- We will typically denote an input variable by the symbol X. If X is a vector, its components can be accessed by subscripts Xj.
- Quantitative outputs will be denoted by Y , and qualitative outputs by G (for group).
- We use uppercase letters such as X, Y or G when referring to the generic aspects of a variable.
- Observed values are written in lowercase like the ith observed value of X is written as x_i (where x_i is again a scalar or vector).

- Matrices are represented by bold uppercase letters; for example, a set of N input p-vectors x_i, $i = 1, ..., N$ would be represented by the $N \times p$ matrix $\mathbf{X}$. In general, vectors will not be bold, except when they have N components.

- Matrices are represented by bold uppercase letters; for example, a set of N input p-vectors x_i, $i = 1, \ldots, N$ would be represented by the $N \times p$ matrix $\mathbf{X}$. In general, vectors will not be bold, except when they have N components.
- This convention distinguishes a p-vector of inputs x_i for the ith observation from the N-vector x_j consisting of all the observations on variable X_j. Since all vectors are assumed to be column vectors, the ith row of X is ${x_i}^T$, the vector transpose of x_i.

Regression and Classification

- This distinction in output type has led to a naming convention for the prediction tasks: **regression** when we predict **quantitative outputs**, and **classification** when we predict **qualitative outputs**. We will see that these two tasks have a lot in common, and in particular both can be viewed as a task in function approximation.

Regression and Classification

- This distinction in output type has led to a naming convention for the prediction tasks: **regression** when we predict **quantitative outputs**, and **classification** when we predict **qualitative outputs**. We will see that these two tasks have a lot in common, and in particular both can be viewed as a task in function approximation.
- For the moment we can loosely state the learning task as follows: given the value of an input vector X, make a good prediction of the output Y, denoted by $\widehat{Y}$ (pronounced "$y - hat$"). If Y takes values in $\mathbb{R}$ then so should $\widehat{Y}$; likewise for categorical outputs, $\widehat{G}$ should take values in the same set G.

- For a two-class G, one approach is to denote the binary coded target as Y , and then treat it as a **quantitative output**. The predictions $\widehat{Y}$ will typically lie in $[0, 1]$, and we can assign to $\widehat{G}$ the class label according to whether $\widehat{Y} > 0.5$. This approach generalizes to K-level qualitative outputs as well.

- For a two-class G, one approach is to denote the binary coded target as Y , and then treat it as a **quantitative output**. The predictions $\widehat{Y}$ will typically lie in $[0, 1]$, and we can assign to $\widehat{G}$ the class label according to whether $\widehat{Y} > 0.5$. This approach generalizes to K-level qualitative outputs as well.
- We need data to construct prediction rules, often a lot of it. We thus suppose we have available a set of measurements (x_i, y_i) or (x_i, g_i), $i = 1, ..., N$, known as the **training data**, with **which to construct our prediction rule**.

- In this section we develop two simple but powerful prediction methods : the linear model fit by **least squares** and the **k-nearest-neighbor** prediction rule. The linear model makes huge assumptions about structure and yields stable but possibly inaccurate predictions. The method of k-nearest neighbors makes very mild structural assumptions: Its predictions are often accurate but can be unstable.

Linear Models and Least Squares

Given a vector of inputs $X^T = (X_1, X_2, ..., X_p)$, we predict the output Y via the model $\widehat{Y} = \widehat{\beta_0} + \sum_{i=1}^{p} \widehat{\beta}_i X_i$

Here term $\widehat{\beta_0}$ is also known an **bias in machine learning.**

Often it is convenient to include the constant variable 1 in X, include $\widehat{\beta_0}$ in the vector of coefficients $\widehat{\beta}$, and then write the linear model in vector form as an inner product $\widehat{Y} = X^T \widehat{\beta}$.

Here we are modeling a single output, so $\widehat{Y}$ is a scalar.

in general $\widehat{Y}$ can be a K–vector, in which case β would be a $p \times K$ matrix of coefficients.

If $p = 2$ then $(X, \widehat{Y})$ represents a line.

- In the $(p+1)$-dimensional input–output space, $(X, \widehat{Y})$ represents a **hyperplane**. If the constant is included in X, then the hyperplane includes the origin and is a subspace; if not, it is an affine set cutting the Y -axis at the point $(0, \widehat{\beta_0})$. From now on we assume that the intercept is included in $\widehat{\beta}$.

- In the $(p+1)$-dimensional input–output space, $(X, \widehat{Y})$ represents a **hyperplane**. If the constant is included in X, then the hyperplane includes the origin and is a subspace; if not, it is an affine set cutting the Y -axis at the point $(0, \widehat{\beta_0})$. From now on we assume that the intercept is included in $\widehat{\beta}$.
- In this approach, we pick the coefficients β to minimize the residual sum of squares $RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T\beta)^2$, where $RSS(\beta)$ is a quadratic function of the parameters, and hence its minimum always exists, but may not be unique. The solution is easiest to characterize in matrix notation. We can write $RSS(\beta) = (y - X\beta)^T(y - X\beta)$.$X$ is an $N \times p$ matrix with each row an input vector, and y is an $N-$vector of the outputs in the training set.

- In the $(p+1)$-dimensional input–output space, $(X, \widehat{Y})$ represents a **hyperplane**. If the constant is included in X, then the hyperplane includes the origin and is a subspace; if not, it is an affine set cutting the Y -axis at the point $(0, \widehat{\beta_0})$. From now on we assume that the intercept is included in $\widehat{\beta}$.
- In this approach, we pick the coefficients β to minimize the residual sum of squares $RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$, where $RSS(\beta)$ is a quadratic function of the parameters, and hence its minimum always exists, but may not be unique. The solution is easiest to characterize in matrix notation. We can write $RSS(\beta) = (y - X\beta)^T (y - X\beta)$.$X$ is an $N \times p$ matrix with each row an input vector, and y is an $N-$vector of the outputs in the training set.
- If $X^T X$ is nonsingular, then the unique solution is given by $\widehat{\beta} = (X^T X)^{-1} X^T y$.

An Example

An Example

Let's look at an example of the linear model in a classification context. Figure in next page, shows a scatterplot of training data on a pair of inputs X_1 and X_2.Let The output class variable G has the values $BLUE$ or $ORANGE$. There are 100 points in each of the two classes. The linear regression model was fit to these data, with the response Y coded as 0 **for BLUE** and **1 for ORANGE**. The fitted values $\widehat{Y}$ are converted to a fitted class variable $\widehat{G}$ according to the rule $\widehat{G}$

$$= \begin{cases} ORANGE & \widehat{Y} > 0.5 \\ BLUE & \widehat{Y} \leq 0.5 \end{cases}$$

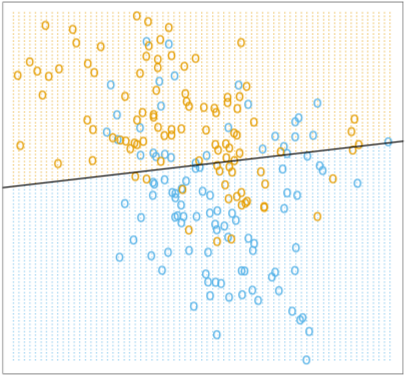


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE.

- The hyperplane $x^T\beta = 0.5$ divides all of $\mathbb{R}^n$ into 3 mutually exclusive and exhaustive sets.

- The hyperplane $x^T\beta = 0.5$ divides all of $\mathbb{R}^n$ into 3 mutually exclusive and exhaustive sets.
- The set of points in $\mathbb{R}$ classified as ORANGE corresponds to $\{x : x^T\beta > 0.5\}$,The set of points in $\mathbb{R}$ classified as BLUE corresponds to $\{x : x^T\beta < 0.5\}$ indicates **open half spaces**. This two predicted classes are separated by the **decision boundary** $\{x : x^T\beta = 0.5\}$, which is linear in this case.

- The hyperplane $x^T\beta = 0.5$ divides all of $\mathbb{R}^n$ into 3 mutually exclusive and exhaustive sets.
- The set of points in $\mathbb{R}$ classified as ORANGE corresponds to $\{x : x^T\beta > 0.5\}$,The set of points in $\mathbb{R}$ classified as BLUE corresponds to $\{x : x^T\beta < 0.5\}$ indicates **open half spaces**. This two predicted classes are separated by the **decision boundary** $\{x : x^T\beta = 0.5\}$, which is linear in this case.
- We see that for these data there are several **misclassifications** on both sides of the decision boundary. Perhaps our linear model is too rigid— or are such errors unavoidable? Remember that these are errors on the training data itself, and we have not said where the constructed data came from.

Consider 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 low- variance Gaussian distributions, with individual means themselves distributed as Gaussian.

- mixture of Gaussians is best described in terms of the generative model. One first generates a discrete variable that determines which of the component Gaussians to use, and then generates an observation from the chosen density.

- mixture of Gaussians is best described in terms of the generative model. One first generates a discrete variable that determines which of the component Gaussians to use, and then generates an observation from the chosen density.
- The region of overlap is inevitable, and future data to be predicted will be plagued by this overlap as well. In the case of mixtures of tightly clustered Gaussians the story is different. A linear decision boundary is unlikely to be optimal, and in fact is not. The optimal decision boundary is nonlinear and disjoint, and as such will be much more difficult to obtain.

- mixture of Gaussians is best described in terms of the generative model. One first generates a discrete variable that determines which of the component Gaussians to use, and then generates an observation from the chosen density.
- The region of overlap is inevitable, and future data to be predicted will be plagued by this overlap as well. In the case of mixtures of tightly clustered Gaussians the story is different. A linear decision boundary is unlikely to be optimal, and in fact is not. The optimal decision boundary is nonlinear and disjoint, and as such will be much more difficult to obtain.
- We now look at another classification and regression procedure that is in some sense at the opposite end of the spectrum to the linear model, and far better suited to the second scenario.

Nearest-Neighbor Methods

Nearest-neighbor methods use those observations in the training set T closest in input space to x to form $\widehat{Y}$. Specifically, the $k-$nearest neighbor fit for $\widehat{Y}$ is defined as follows : $\widehat{Y}(x) = \frac{1}{k}\sum_{xi \in N_k(x)} y_i$
where $N_k(x)$ is the **neighborhood** of x defined by the k closest points x_i in the training sample. **Closeness implies a metric**, which for the moment we assume is Euclidean distance. So, in words, we find the k observations with x_i closest to x in input space and average their responses.

- In Figure 2.2 we use the same training data as in Figure 2.1.we use the same training data as in Figure 2.1, and use 15-nearest-neighbor averaging of the binary coded response as the method of fitting.

- In Figure 2.2 we use the same training data as in Figure 2.1.we use the same training data as in Figure 2.1, and use 15-nearest-neighbor averaging of the binary coded response as the method of fitting.
- Thus $\hat{Y}$ is the proportion of ORANGE's in the neighborhood, and so assigning class ORANGE to $\hat{G}$ if $\hat{Y} > 0.5$ amounts to a majority vote in the neighborhood.

- In Figure 2.2 we use the same training data as in Figure 2.1.we use the same training data as in Figure 2.1, and use 15-nearest-neighbor averaging of the binary coded response as the method of fitting.
- Thus $\hat{Y}$ is the proportion of ORANGE's in the neighborhood, and so assigning class ORANGE to $\hat{G}$ if $\hat{Y} > 0.5$ amounts to a majority vote in the neighborhood.
- The colored regions indicate all those points in input space classified as BLUE or ORANGE by such a rule, in this case found by evaluating the procedure on a fine grid in input space.

- In Figure 2.2 we use the same training data as in Figure 2.1.we use the same training data as in Figure 2.1, and use 15-nearest-neighbor averaging of the binary coded response as the method of fitting.
- Thus $\hat{Y}$ is the proportion of ORANGE's in the neighborhood, and so assigning class ORANGE to $\hat{G}$ if $\hat{Y} > 0.5$ amounts to a majority vote in the neighborhood.
- The colored regions indicate all those points in input space classified as BLUE or ORANGE by such a rule, in this case found by evaluating the procedure on a fine grid in input space.
- We see that the **decision boundaries that separate the BLUE from the ORANGE regions are far more irregular** and respond to local clusters where one class dominates.

15-Nearest Neighbor Classifier

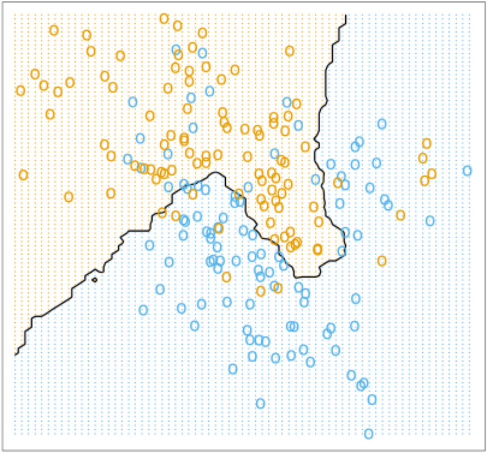


FIGURE 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

From Least Squares to Nearest Neighbors

- The linear decision boundary from **least squares** is very smooth, and apparently stable to fit. It does appear to rely heavily on the assumption that a linear decision boundary is appropriate. In language we will develop shortly, it has **low variance** and **potentially high bias**.

From Least Squares to Nearest Neighbors

- The linear decision boundary from **least squares** is very smooth, and apparently stable to fit. It does appear to rely heavily on the assumption that a linear decision boundary is appropriate. In language we will develop shortly, it has **low variance** and **potentially high bias**.
- On the other hand, the $k-$**nearest-neighbor** procedures do not appear to rely on any stringent assumptions about the underlying data, and can adapt to any situation. However, any particular subregion of the decision boundary depends on a handful of input points and their particular positions, and is thus wiggly and unstable—**high variance** and **low bias**.

From Least Squares to Nearest Neighbors

- The linear decision boundary from **least squares** is very smooth, and apparently stable to fit. It does appear to rely heavily on the assumption that a linear decision boundary is appropriate. In language we will develop shortly, it has **low variance** and **potentially high bias**.
- On the other hand, the $k-$**nearest-neighbor** procedures do not appear to rely on any stringent assumptions about the underlying data, and can adapt to any situation. However, any particular subregion of the decision boundary depends on a handful of input points and their particular positions, and is thus wiggly and unstable—**high variance** and **low bias**.
- Each method has its own situations for which it works best; in particular **linear regression is more appropriate for Scenario 1** , while **nearest neighbors are more suitable for Scenario 2.**

An example of generating 10 means from a bivariate Gaussian distribution

- First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, I)$ and labeled this class **BLUE**.

An example of generating 10 means from a bivariate Gaussian distribution

- First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, I)$ and labeled this class **BLUE**.
- Similarly, 10 more were drawn from $N((0,1)^T, I)$ and labeled class **ORANGE**.

An example of generating 10 means from a bivariate Gaussian distribution

- First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, I)$ and labeled this class **BLUE**.
- Similarly, 10 more were drawn from $N((0,1)^T, I)$ and labeled class **ORANGE**.
- Then for each class we generated 100 observations as follows: for each observation, we picked an mk at random with probability $1/10$.

An example of generating 10 means from a bivariate Gaussian distribution

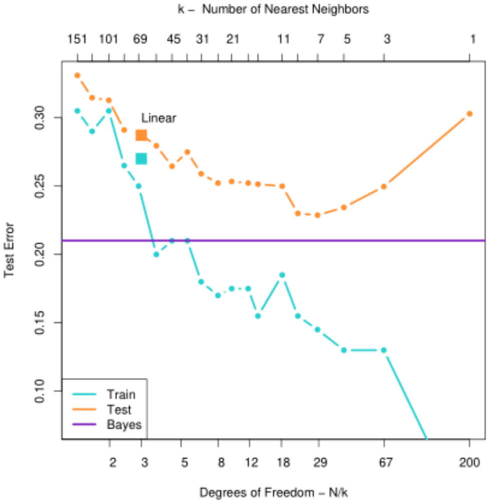
- First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, I)$ and labeled this class **BLUE**.
- Similarly, 10 more were drawn from $N((0,1)^T, I)$ and labeled class **ORANGE**.
- Then for each class we generated 100 observations as follows: for each observation, we picked an mk at random with probability $1/10$.
- Then We generated a $N(m_k, I/5)$, thus leading to a mixture of Gaussian clusters for each class.

An example of generating 10 means from a bivariate Gaussian distribution

- First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, I)$ and labeled this class **BLUE**.
- Similarly, 10 more were drawn from $N((0,1)^T, I)$ and labeled class **ORANGE**.
- Then for each class we generated 100 observations as follows: for each observation, we picked an mk at random with probability $1/10$.
- Then We generated a $N(m_k, I/5)$, thus leading to a mixture of Gaussian clusters for each class.
- Figure in the next page shows the results of classifying $10,000$ new observations generated from the model.

An example of generating 10 means from a bivariate Gaussian distribution

- First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, I)$ and labeled this class **BLUE**.
- Similarly, 10 more were drawn from $N((0,1)^T, I)$ and labeled class **ORANGE**.
- Then for each class we generated 100 observations as follows: for each observation, we picked an mk at random with probability $1/10$.
- Then We generated a $N(m_k, I/5)$, thus leading to a mixture of Gaussian clusters for each class.
- Figure in the next page shows the results of classifying $10,000$ new observations generated from the model.
- We compare the results for least squares and those for k-nearest neighbors for a range of values of k.


k – Number of Nearest Neighbors
151 101 69 45 31 21 11 7 5 3 1
Linear
Test Error
0.30
0.25
0.20
0.15
0.10
Train
Test
Bayes
2 3 5 8 12 18 29 67 200
Degrees of Freedom – N/k

References

The Elements of Statistical Learning Data Mining, Inference, and Prediction (2nd edition) (12print 2017) by Hastie, Tibshirani & Friedman.pdf