Project 2: Predictive Modelling

STAT GU4243

Applied Data Science

Cynthia Rush
Columbia University

February 19, 2018

Supervised Learning

Data:

$$
\begin{aligned}
&(y_1, x_{11}, x_{12}, \ldots, x_{1p})\\
&(y_2, x_{21}, x_{22}, \ldots, x_{2p})\\
&\ldots\\
&(y_n, x_{n1}, x_{n2}, \ldots, x_{np})
\end{aligned}
$$

Y referred to as '**outputs**', '**responses**', or '**dependent variables**'

X referred to as '**inputs**', '**predictors**', '**features**', or '**independent variables**'

Supervised Learning

In supervised learning, outcome variable Y is given.
In unsupervised learning, there is no label Y.

Learning Tasks

1. **Prediction**: Use X to construct a model to predict Y.
2. **Inference**: Identify subject-matter knowledge by understanding the learned model.

For categorical Y, it is referred to as a 'classification' problem.
For quantitative Y, often called 'regression'.

Assumptions and Terminology

In a **classification problem**, we record measurements
$X_1 = (x_{11}, x_{12}, \dots, x_{1p}), X_2 = (x_{21}, x_{22}, \dots, x_{2p}), \dots$.

We assume:

1. All measurements can be represented as elements of $\mathbb{R}^p$ (p dimensional Euclidean space).
2. Each x_i belongs to exactly one out of K categories, called **classes**. We express this using variables $y_i \in [K]$, called **class labels**:

$$y_i = k \quad \leftrightarrow \quad \text{“}x_i \text{ in class } k\text{”}$$

3. The classes are characterized by the (unknown!) joint distribution of (X, Y), whose density we denote $p(x, y)$.
4. The only information available on the distribution p is a set of example measurements *with* labels,

$$(y_1, x_{11}, x_{12}, \dots, x_{1p}), \dots, (y_n, x_{n1}, x_{n2}, \dots, x_{np}),$$

called the **training data**.

Classifiers

Definition

A **classifier** is a function

$$f : \mathbb{R}^p \to [K] \, ,$$

i.e. a function whose argument is a measurement and whose output is a class label (one of $1, 2, \ldots, K$).

Learning task

Using the training data, we have to estimate a good classifier. This estimation procedure is also called **training**.

A good classifier should generalize well to new data. Ideally, we would like it to perform with high accuracy on data sampled in the same way as the training data (i.e. also from $p(x, y)$).

Simplifying assumption

We can consider the two-class case (K=2), which is also called **binary classification**. In this case, we use the notation

$$Y \in \{-1, +1\} \qquad \text{instead of} \qquad Y \in \{1, 2\}$$

Most classification methods use:

1. Linear or non-linear **decision boundaries** between classes.
2. **Discriminant functions**: for each class $k \in [K]$, define $\hat{f}_k(x)$.
   - Prediction may then be computed as: $\arg\max_k \hat{f}_k(x)$.
   - Most current methods work in this domain.

Outline

What We Won't Talk About

Can't cover every possible classification method you may need to use for the project – there are too many and I don't know them all

What We Will Talk About

- Framework for comparing classification methods: how do we judge performance?
  - Loss functions and risk
  - Test error vs. training error
  - Cross-validation and bootstrap
- General properties of classification methods: bias vs variance, complexity, curse of dimensionality, etc.
- Some classification basics: logistic regression, linear discriminant analysis, kNN
- Some strategies for building better classifiers: boosting, bagging, etc.

Some Lite Decision Theory

Loss Functions

First consider:

- Real-valued, random input $X \in \mathbb{R}^p$
- Real-valued, random $Y \in \mathbb{R}$ (quantitative, not categorical Y for now).

We assume a joint probability distribution $p(x, y)$.

Definition

Want a function $f(X)$ for predicting Y given values of X. This theory requires a loss function:

$$L(Y, f(X)) : \mathbb{R} \times \mathbb{R} \to [0, \infty)$$

for penalizing prediction errors.

Squared Error

The most common and convenient is squared error loss:

$$L(Y, f(X)) = (Y - f(X))^2.$$

Expected Prediction Error

Motivation

It may be a good strategy to allow (even expensive) errors for values of X which are very unlikely to occur.

Definition

The expected prediction error, $\mathrm{EPE}(f)$, of a classifier f is its expected loss under p, that is,

$$\mathrm{EPE}(f) = \mathbb{E}_{(X,Y)}[L(Y, f(X))].$$

Loss Functions

Example

For squared error loss,

$$\text{EPE}(f) = \mathbb{E}_{(X,Y)}(Y - f(X))^2 = \mathbb{E}_X \mathbb{E}_{Y|X}((Y - f(X))^2 | X)$$

Using the above, $f(x) = \mathbb{E}(Y|X = x)$ minimizes $\text{EPE}(f)$, which is known as the regression function.

Interpretation

Tells us, best prediction of Y when $X = x$ is given by $\mathbb{E}(Y|X = x)$, when best is measured by square error loss.

Therefore, the type of loss function considered guides the prediction method.

Some examples....

Squared Error Loss

Want to classify with $f(x) = \mathbb{E}(Y|X=x)$ in order to minimize EPE(f), but we can't calculate this classifier since we don't know $p(x,y)$.

Least Squares

Least squares estimates $f(x) = \mathbb{E}(Y|X=x)$ which minimizes $\mathbb{E}_{(X,Y)}[L(Y, f(X))]$, by minimizing observed or empirical $L(Y, f(X))$ among all linear models.

k-Nearest Neighbors

Nearest neighbors estimates $f(x) = \mathbb{E}(Y|X=x)$ by using the mean of y_i values for $x_i \in N_{\mathsf{k}}(x)$ where $N_{\mathsf{k}}(x)$ is a small neighborhood of x.

Some Nice Properties of kNN

A larger sample means more observations close to x (i.e. $N_{\mathsf{k}}(x)$ is tighter) which produces a more stable $\hat{f}(x)$ (given $\mathbb{E}(Y|X)$ is continuous enough).

Actually, it is not difficult to show that if $N, \mathsf{k} \to \infty$ and $\mathsf{k}/N \to 0$, then $\hat{f}(x) \to \mathbb{E}(Y|X = x)$.

Why not just use kNN, then? Often don't have large sample size, also curse of dimensionality which we address in a bit.

Absolute Loss

We replace the L_2 metric by the L_1 metric giving $L(Y, f(X)) = |Y - f(X)|$.

It can be shown that $f(x) = \text{median}(Y|X = x)$ minimizes

$$\text{EPE}(f) = \mathbb{E}|Y - f(X)|.$$

As before, implies regression and kNN methods:

- Regression: least absolute deviations (LAD) regression or LAR (least absolute residuals).
- kNN: use median instead of average for calculating distance used to find each $N_{\mathsf{k}}(x)$.

Classification Loss

Definition

Want a function $f(X)$ for predicting class Y given values of X. This theory requires a loss function:

$$L(Y, f(X)) : [K] \times [K] \to [0, \infty)$$

for penalizing prediction errors.

Multiple Classes

Some example loss functions:

1. Loss function defined based on a penalty matrix, $\mathbf{L} = [L(k, \ell)]_{K \times K}$ for K classes.
2. Zero-one loss corresponding to the number of misclassifications. Defined as $L(k, \ell) = 1$ if $k \neq \ell$ and 0 otherwise.

EPE

Then,

$$\text{EPE}(f) = \mathbb{E}_{(X,Y)}[L(Y, f(X))] = \mathbb{E}_X \sum_{k=1}^{K} L(k, f(X)) \Pr(k|X).$$

With 0-1 loss, it can be shown that

$$\hat{f}(x) = k \quad \text{if} \quad \Pr(Y = k \mid X = x) = \max_{k' \in [K]} \Pr(Y = k' \mid X = x)$$

minimizes $\text{EPE}(f)$.

Bayes Classifier

"Classify to the most probable class, using the conditional distribution $\Pr(Y|X)$"

- Error rate of this classifier is called the Bayes rate.
- Bayes classifier is best under zero-one loss.
- In practice, can't calculate Bayes classifier becasue don't know $p(x, y)$.

kNN attempts to estimate the Bayes classifier by estimating $\Pr(Y = k|X = x)$ by $\hat{\Pr}(Y = k|N_{\mathsf{k}}(x))$.

Methods Using Discriminant Analysis

Linear Discriminant Analysis (LDA)

- LDA assumes a Gaussian mixture with common covariance matrix, Σ.
- Want to estimate $\Pr(Y = k|X = x)$. By Bayes' Rule:

$$\Pr(Y = k \,|\, X = x) \propto \Pr(X = x \,|\, Y = k) \Pr(Y = k).$$

- When comparing two classes, it is sufficient to look at the log-ratio:

$$\log \frac{\Pr(Y = k \,|\, X = x)}{\Pr(Y = \ell \,|\, X = x)} = \log \frac{\Pr(G = k)}{\Pr(G = \ell)} - \frac{1}{2}(\mu_k - \mu_\ell)^T \Sigma^{-1} (\mu_k - \mu_\ell) + x^T \Sigma^{-1} (\mu_k - \mu_\ell)$$

- Above implies, that the decision boundary between classes k and ℓ is linear in x; in p dimensions a hyperplane
- The linear discriminant function is then

$$\delta_k(x) = \log \Pr(Y = k) - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + x^T \Sigma^{-1} \mu_k$$

  and $\hat{f}(x) = \arg\max_k \delta_k(x)$.
- Natural ways to estimate $\hat{\pi}_k$ (with N_k/N), $\hat{\mu}_k$ (with $\sum_{y_i = k} x_i / N_k$), and $\hat{\Sigma}$ (with the pooled estimate of the covariance matrix).

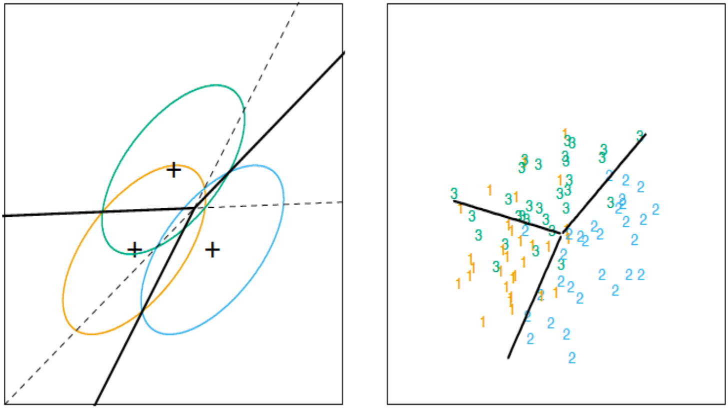

FIGURE 4.5. *The left panel shows three Gaussian distributions, with the same covariance and different means. Included are the contours of constant density enclosing 95% of the probability in each case. The Bayes decision boundaries between each pair of classes are shown (broken straight lines), and the Bayes decision boundaries separating all three classes are the thicker solid lines (a subset of the former). On the right we see a sample of 30 drawn from each Gaussian distribution, and the fitted LDA decision boundaries.*

Quadratic Discriminant Functions

If we assume different covariance matrices Σ_K for the classes $k = 1, \ldots, K$, the discriminant function based on log-likelihood-ratio will be quadratic.

$$\delta_k(x) = \log Pr(Y = k) - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)$$

Thus, the discriminant boundary ($\{x : \delta_k(x) = \delta_\ell(x)\}$) is a quadratic function.

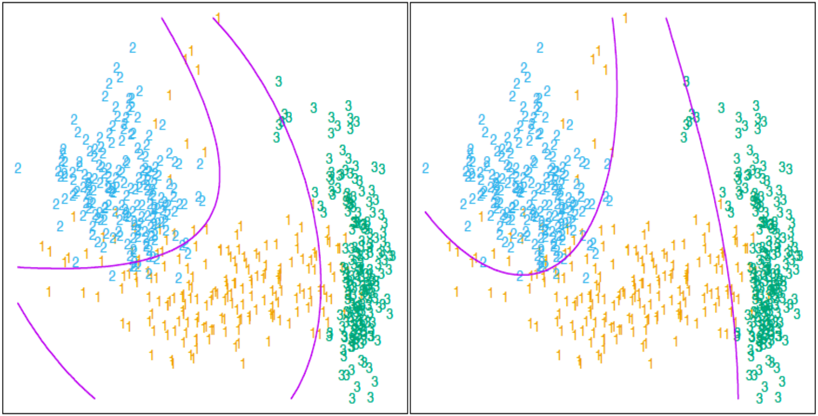

FIGURE 4.6. *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

Method of Nearest Neighbors

- Define $N_{\mathsf{k}}(x)$ as the neighborhood of x containing the k 'nearest' points in the training set.
- A distance metric is (implicitly) needed for kNN methods – most popular choice is Euclidean distance.
- Prediction is then $\hat{f}(x) = \frac{1}{\mathsf{k}} \sum_{x_i \in N_{\mathsf{k}}(x)} y_i$.
- Obviously residual sum of squares minimized at $\mathsf{k} = 1$ (all training classifications are correct).
- Models fit using kNN is less rigid than linear methods like LDA and generate non-linear prediction functions.

15-Nearest Neighbor Classifier

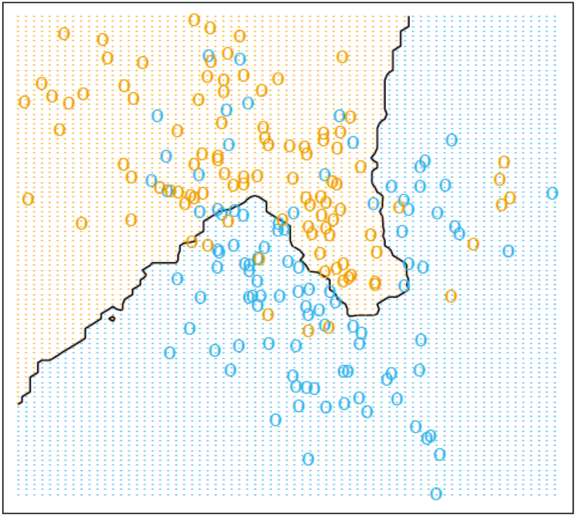

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

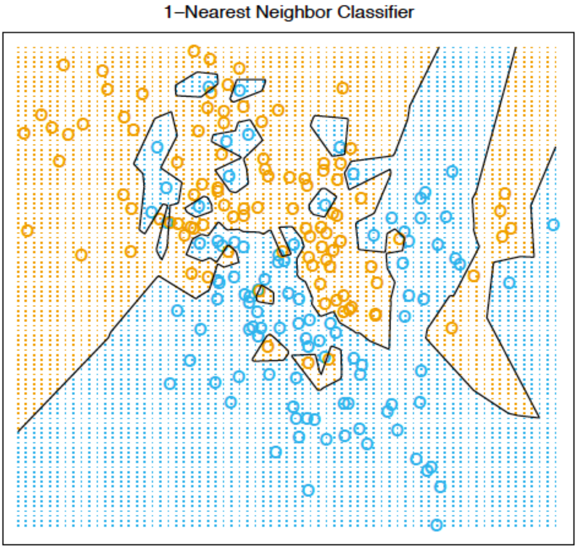

FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Bayes Optimal Classifier

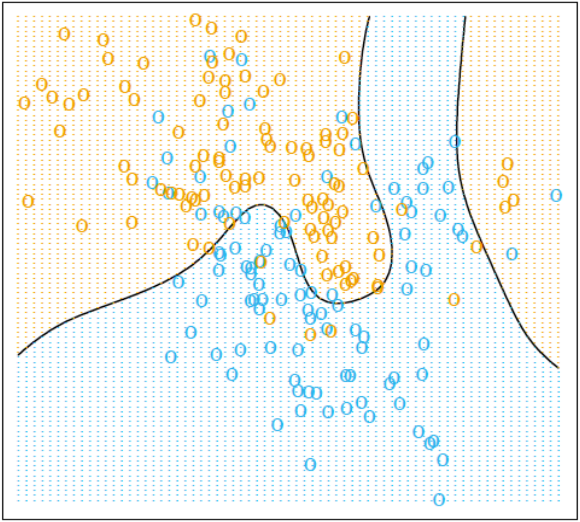

FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Properties

Linear methods

Low variance, high bias

- Unbiased if true model is linear, biased if true model is non-linear.
- Stable (i.e., individual observations not very influential) and smooth.
- Nice analytical results.

Nearest Neighbor methods

High variance, low bias

- No assumption on the model.
- High variance in the predictions since each $\hat{Y}$ is calculated only by a few observations.

Variants of the Two Methods

- Kernel methods can be applied to distance metrics. E.g. smoother kernel to replace the kNN "kernel".
- Local weights, or weights varying across dimensions, to make linear methods less rigid.
- Linear models fit to a basis expansion of the original inputs: this expands the class of models considered.