STAT 452/652: Statistical Learning and Prediction

Owen G. Ward
owen_ward@sfu.ca

Fall 2023

These notes are largely based on previous iterations of this course, taught by Prof. Tom Loughin and Prof. Haolun Shi.

11 Introduction to Classification

(Reading: ISLR 2.1.5, 2.2.3, 4.1)

Goals of this section

- Regression is great for predicting means or numerical responses.
- Categorical responses are different
  - Groups have no magnitude and (usually) no ordering
  - Squared error loss is not relevant
- We have seen categorical measurements used as explanatory variables
- Need methods for predicting categorical *responses*
  - Some are similar to regression methods we have learned
  - Others are quite different

11.1 Terminology and notation for classification

- Response measurement is a *nominal variable*, Y
  - Y takes on values from K possible CLASSES
  - Classes are indexed by $k = 1, \dots, K$
    - Confusingly, ISLR uses "j" to index classes;
    - We will reserve i for different observations, j for different explanatory variables, and k for different classes
  - K is often 2.
- Our "universe" has the same principles as before, but with some different details

1. In regression:
   (a) There was a *continuous* distribution of Y values across the universe, $Y = g(\mathbb{X}) + \delta$.
      i. Not all measurements taken at $\mathbb{X} = x_0$ will equal $g(x_0)$
   (b) True structure $g(\mathbb{X})$ represented the mean of the distribution of Y
   (c) "Irreducible error" δ represented a numerical deviation from $g(\mathbb{X})$ for a given observed value
2. In classification,
   (a) At a point $\mathbb{X} = x_0$ in the universe there is a *discrete* distribution across the K possible Y values,

$$\begin{aligned} p_1(x_0) &= P(Y = 1|\mathbb{X} = x_0) \\ p_2(x_0) &= P(Y = 2|\mathbb{X} = x_0) \\ &\vdots \\ p_K(x_0) &= P(Y = K|\mathbb{X} = x_0) \end{aligned}$$

      i. That is, each possible class has a certain probability of occurring.
      ii. This probability likely depends on values of explanatory variables (both measured and unmeasured)
   (b) The "true class" at any $\mathbb{X} = x_0$ point in the universe is whichever class has the highest probability there
   (c) The "irreducible error" is now represented by the fact that some class other than the most likely class could be observed instead.

- Objective is to learn a function or machine $f(X)$ that will return the *most likely class* $\hat{y}$ at a given value of X
  - Predicting the most likely class gives us the best chance of being right.
  - And estimated $\hat{f}$ is called a CLASSIFIER
- Squared error measures like RSS, $sMSE$, and $MSPE$ are no longer relevant, since both y and $\hat{y}$ are class labels instead of numerical values
  - Don't be confused by our use of numbers $1, 2, \ldots, K$ to *represent* class labels.
  - Class 3 is not equal to class 1 + class 2.
- Instead we measure error most often by the MISCLASSIFICATION RATE
  - What fraction of predictions were the wrong class?
  - Compute $I(y_i \neq \hat{y}_i)$ for each observation
    - $= 0$ if class is correctly predicted, $= 1$ if class is incorrectly predicted
  - Compute average indicator across sample
    - Gives back proportion of incorrect predictions
- Also often produce CONFUSION MATRIX

  - Just a $K \times K$ table of $\hat{Y}$ vs. Y

- Fitting, comparing, selecting, and evaluating models is done using same principles as before.
  - Just substitute misclassification rate for mean squares
  - Can be computed on training, validation, or test sets with exactly the same purposes and interpretations
- Bias-variance tradeoff is still alive and well here!
  - Some models are more flexible and complex than others
  - May be better at guessing which class *occurred* in the training set, but not as good at generalizing to when a different class was *more likely to occur* in the population.

11.2 K-Nearest Neighbours: A Simple Classifier

Focus now on a sample of size n from X and Y. How do we find $\hat{f}$?

- Many possible models and algorithms do classification!! See Figure 1.
- A very simple idea that does not require a model is to classify any point x_0 according to the majority of points in its "neighbourhood"
  - Whichever class is most all around x_0 is a good guess for the best class at x_0

1. K-Nearest-Neighbours (KNN) Algorithm:[1]
   (a) Choose m, the chosen size of the "neighbourhood," in terms of the number of points that will get to "vote" on the class
   (b) Select a point x_0 in the p-dimensional space of X.
   (c) Identify the m closest points to x_0 the neighbourhood and call this collection $\mathcal{N}_m(x_0)$.
   (d) Count the number of points of each class within $\mathcal{N}_m(x_0)$:
   (e) $\hat{Y}(x_0)$ is class with the highest count
2. See Figure 1 for examples with $m = 1$ and $m = 15$
3. This produces very irregular shapes at the boundaries between classes.
4. m is a tuning parameter:
   (a) Matters A LOT!
   (b) Bias/Variance tradeoff
      i. Larger m: low variance and higher bias
      ii. Smaller m: higher variance and low bias
      iii. See Figure 2

[1]Note: People usually use "K" to index the number of neighbours to be used in the prediction. The procedure is then abbreviated as "KNN". We are already using K to be the number of classes. So to avoid confusion, so we will use the ubiquitous "m" to index number of neighbours. But we will still abbreviate it as "KNN" to emphasize the universal term.

Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Chap 2

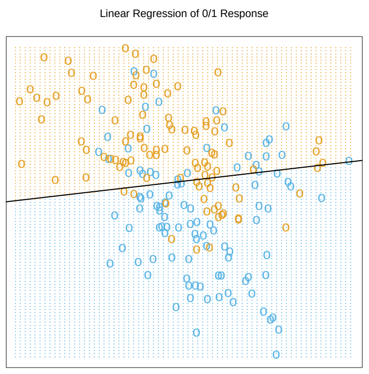

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable* (`BLUE` $= 0$, `ORANGE` $= 1$)*, and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$. *The orange shaded region denotes that part of input space classified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

Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Chap 2

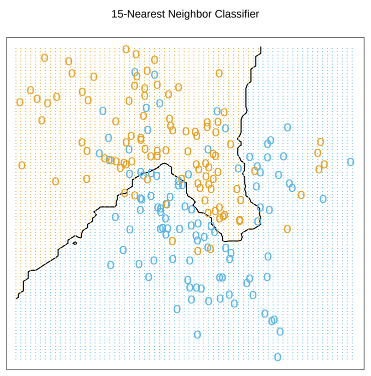

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (`BLUE` $= 0$, `ORANGE` $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Chap 2

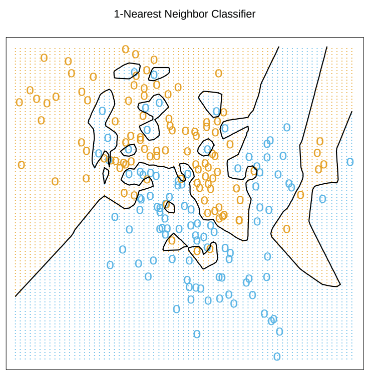

FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (`BLUE` $= 0$, `ORANGE` $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Chap 2

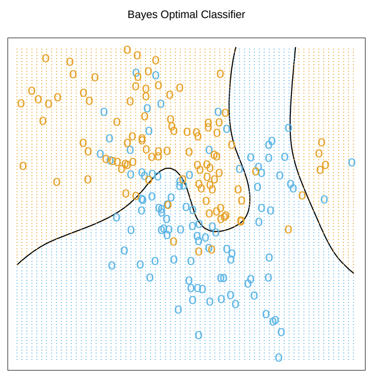

FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Figure 1: Simulated data for a classification problem, and three classifiers: Top left: Linear; top right: 15 nearest neighbour; bottom left: 1 nearest neighbour; bottom right: Bayes classifier based on known model

42 2. Statistical Learning

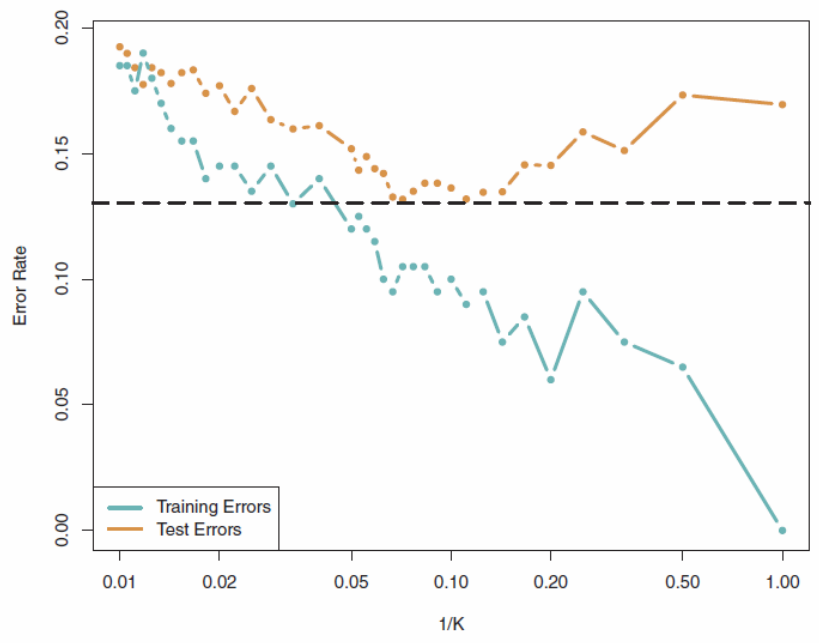

FIGURE 2.17. *The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.*

Figure 2: Training and test error rates for KNN using various values of m from a simulated example. X-axis is $1/m$, so model complexity increases from left to right. (From ISLR.)

11.3 Notes on KNN

KNN is not always a useful tools, but some people really like it

1. KNN struggles in high dimensions
   (a) Larger m may create big neighbourhoods that span regions with very different class probability distributions
   (b) Problem gets worse as p gets larger because points less dense, have to look farther for m neighbours.
   (c) Adds potential bias and variance from distribution and variability of X
2. Complicated computation, slow for large n, p
3. Need to *standardize* the features (sd=1) before using
   (a) Want "distance" to have same meaning in all directions: relative to spread of data
   (b) Categorical features have no "distance" metric
      i. Need to be turned into contrasts (e.g., indicators) and standardized

4. *Asymptotically* the error of the 1-NN is twice the optimal (Bayes) error (see later)
   (a) Application of this knowledge: fit a 1-NN and measure test error to estimate target error for other methods as half that much
   (b) Need large $n : p$ ratio to resemble asymptotics
5. Somewhat sensitive to presence of irrelevant features
   (a) Helps considerably if you can do something to remove worthless variables.
6. Performance *can* be surprisingly good.
   (a) Insensitive to outliers if m not too close to 1

Example: KNN on Wheat Kernel Data (`Sec11_Nearest_Neighbour_Wheat.R`)

To use KNN we introduce a new data set about Wheat measurements (the original source is Martin, Herrman, Loughin, and Oentong (1998), *Cereal Chemistry.* The description is amended from Bilder and Loughin (2014) *Analysis of Categorical Data with R*). The presence of sprouted or diseased kernels in wheat can reduce the value of a wheat producer's entire crop. It is important to identify these kernels after harvested but prior to sale. To facilitate this identification process, automated systems have been developed to separate healthy kernels from the rest. Improving these systems requires better understanding of the measurable ways in which healthy kernels differ from kernels that have sprouted prematurely or are infected with a fungus ("Scab"). To this end, an observational study measures numerous physical properties of kernels—density, hardness, size, weight, and moisture content—on a sample of wheat kernels from two different classes of wheat, hard red winter ("hrw") and soft red winter ("srw"). Each kernel's condition was also classified as "Healthy,""Sprout, "or "Scab"by human visual inspection. In the data provided by the authors of this paper, we have measurements from 275 wheat kernels.

The data are available in the spreadsheet **wheat.csv**. This is an "easy" problem in the sense that the number of explanatories is relatively small (6). The data have a few quirks, and code is supplied to smooth them out and create two sets. A summary of the full data is below.

```
> summary(wheat)
 class        density          hardness
 hrw:143   Min.   :0.7352   Min.   :-44.080
 srw:132   1st Qu.:1.1358   1st Qu.:  0.689
           Median :1.2126   Median : 24.465
           Mean   :1.1885   Mean   : 25.564
           3rd Qu.:1.2687   3rd Qu.: 45.606
           Max.   :1.6454   Max.   :111.934
      size            weight          moisture
 Min.   :0.5973   Min.   : 8.532   Min.   : 6.486
 1st Qu.:1.8900   1st Qu.:21.982   1st Qu.: 9.540
 Median :2.2303   Median :27.610   Median :11.909
 Mean   :2.2047   Mean   :27.501   Mean   :11.192
 3rd Qu.:2.5125   3rd Qu.:32.882   3rd Qu.:12.538
 Max.   :4.3100   Max.   :46.334   Max.   :14.514
      type       classnum
 Healthy:96   Min.   :1.00
 Scab   :83   1st Qu.:1.00
 Sprout :96   Median :1.00
```

```
                 Mean   :1.48
                 3rd Qu.:2.00
                 Max.   :2.00
```

KNN There are lots of functions that do KNN. There is a `knn()` function in the `class` package that is loaded automatically into `R`. There is also a `knn()` function in the "fast nearest neighbor" (`FNN`) package. We'll use the latter. One can also tune using `knn.cv()` in both packages, using leave-one-out, or using the `train()` function in `caret` for the first function. Included is a tuning script and follow-up plot.

The program first scales the explanatory variables by subtracting off the mean of the training set and dividing by the standard deviation of the training set. Then all variables contribute equally to distance measures, and the test data have the same transformation applied as the training data.

Then we fit a 1-NN model ($m = 1$) to show how the basic code works. The `knn()` function uses the matrix or data frame of training data in `train =` to produce predicted values for a matrix or data frame in `test =`. They can be the same data if training error is desired. The response factor-class object is listed as `cl =` . The number of NNs is `k =` . We can use the code to compute training and test misclassification rates for 1-NN (we know that training error should be 0), as well as confusion matrices. The results are below.

```
> # Fit the 1-NN function using set 1 to train AND test
> #   (compute training error)
> knnfit.1.1 <- knn(train=x.1, test=x.1, cl=set1[,6], k=1)
>
> # Create Confusion Matrix and misclass rate
> table(knnfit.1.1, set1[,6],  dnn=c("Predicted","Observed"))
         Observed
Predicted Healthy Scab Sprout
  Healthy      71    0      0
  Scab          0   61      0
  Sprout        0    0     68
> (misclass.knn1.1 <-
+          mean(ifelse(knnfit.1.1 == set1[,6], yes=0, no=1)))
[1] 0
>
> # Fit the 1-NN function using set 1 to train and set2 to test
> #   (compute test error)
> knnfit.1.2 <- knn(train=x.1, test=x.2, cl=set1[,6], k=1)
> # Create Confusion Matrix and misclass rate
> table(knnfit.1.2, set2[,6],  dnn=c("Predicted","Observed"))
         Observed
Predicted Healthy Scab Sprout
  Healthy      13    4     12
  Scab          2   12      1
  Sprout       10    6     15
> (misclass.knn1.2 <-
+          mean(ifelse(knnfit.1.2 == set2[,6], yes=0, no=1)))
[1] 0.4666667
```

We see that the 1-NN provides perfect training prediction, which happens because each response observation is the predicted value for every x_0 it is closest to, including itself. The test misclassification rate is a poor 47%, largely because the classifier does not distinguish well between `Healthy` and `Sprout` classes.

Next we run the `knn.cv()` function to do use leave-one-out CV to compute the error rate for

$m = 1, 2, \ldots, 40$. We also compute an approximate standard error for these error rates, assuming that the prediction errors are independent and using the standard error for the binomial proportion. The result is in Figure 3

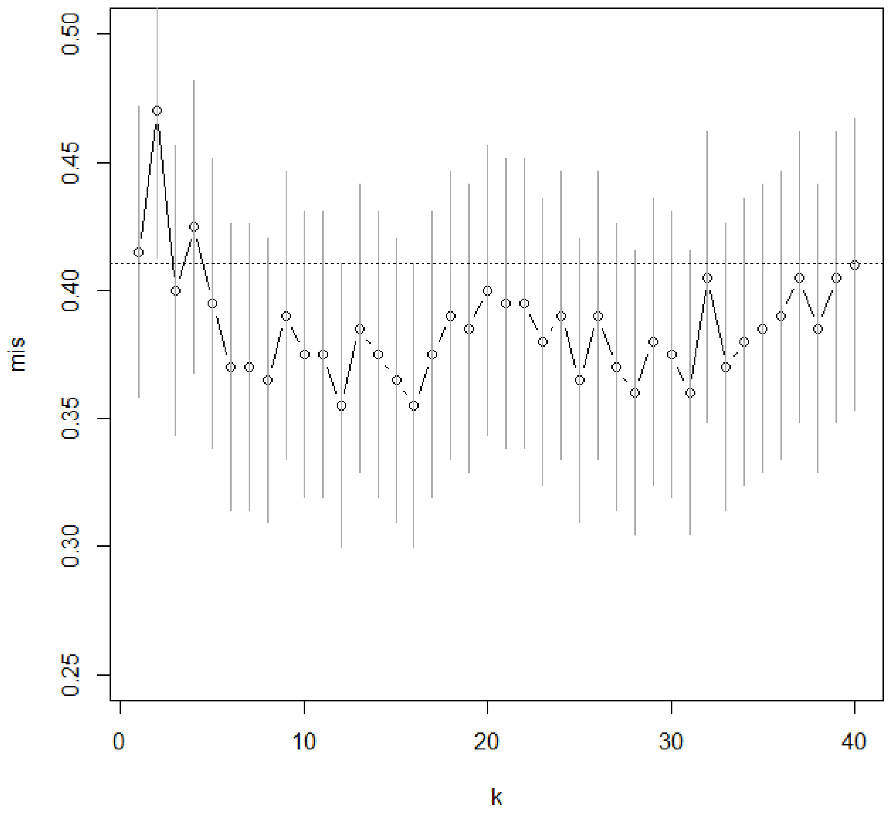

Figure 3: CV error rates for KNN on wheat training data using $m = 1, 2, \ldots, 40$.

The misclassification rate is minimized at $m = 12$, whereas applying a "1-se rule" and using the least complex model with error rate within 1SE of the minimum suggests $m = 40$. The test set error rates for these values are also high: 0.49 and 0.51. This may just be a data set where there are too many unnecessary variables that influence the measurement of distance between observations but do not improve predictions

What to take away from this

1. Classification is a new problem where we try to predict a categorical variable
2. The "true structure" is a probability distribution across all possible classes at each location in the universe.
   (a) The class with highest probability it the best class for that location.
3. There are lots of methods for doing classification

4. We introduced one simple one, K-nearest neighbours, that uses observations closest to a point in space to determine the prediction at that point.

5. It's not a great technique but can be very flexible and adaptable to any true structure

Problem Set 13: Introduction to Classification

Concepts

1. *Establishing a "baseline" error rate.* Suppose that we have a classification problem with K classes, and suppose that the proportions of observations in each class are $p_1, p_2, \dots, p_K$. Suppose that class Q has the largest proportion, so that $p_Q > p_m$ for all other $m \neq Q$.

   If you had no explanatory variables and still had to do prediction, you would use a NAIVE CLASSIFIER that always assigns most common class to all predictions. In our problem, **what would be the misclassification rate for the naive classifier?**

   This is sometimes called the BASELINE ERROR RATE for the problem, and represents a guess at the worst error rate you expect and "real" classifier to have, assuming that future samples have the same distribution of classes as this one.

2. *Difficulties with classifying unbalanced responses.* Suppose you have a classification problem with $K = 2$, and that 95% of the responses are class 1. **What is the baseline error rate for this problem?**

   It is often the case that the baseline error rate is hard to beat with a "real" classifier, because correctly classifying a portion of the class-2 data often causes an even larger number of class-1 data to be misclassified. For example, if the ratio of class 1 to class 2 is 95:5, then correctly classifying even one or two class-2 observations may cause 5 or 10 class-1 responses to be misclassified. For this reason, we may choose to use other measures besides total misclassifications to judge a classifier. We will talk about these more later.

Application

DATA: Vehicle Image Recognition[2], **vehicle.csv**

This data set is described in subsection 8.7 of Izenman, with a longer description here: http://archive.ics.uci.edu/ml/datasets/Statlog+%28Vehicle+Silhouettes%29 . Please read through this description briefly to get a general idea of what these data are about. Details aren't important. The basic purpose of the study was to improve on image analysis to recognize different shapes. Four model cars—a 2-door car (OPEL, coded as `1`), a 4-door car (SAAB, coded as `2`), a double-decker bus (BUS, coded as `3`), and a minivan (VAN, coded as `4`)—were viewed in silhouette from different angles and 18 different measurements were taken of these images. The goal is to use these 18 measurements to identify which type of car made the picture. So this is a classification problem. Remember the coding of the response variable, because it will help with later interpretations.

The data set can be read using

```
read.csv("<file location>\\vehicle.csv")
```

The explanatory variables have long names, and the response is `class`, with coding 1, 2, 3, 4 as described above.

[2]This dataset comes from the Turing Institute, Glasgow, Scotland.

1. Get to know the data
   (a) Read the data and **print a summary**.
   (b) Notice that `class` has been read as numeric. Convert it to a factor using `vehdata$class = factor(vehdata$class, labels=c("2D", "4D", "BUS", "VAN"))` Show a summary of this new version of `class`.
   (c) **Print the correlation matrix for the explanatory variables. Comment on any strong correlations (maybe beyond ± 0.7 and note especially any beyond ± 0.9)**

2. Split the data using the code below, where `set1` will be the training set for future analyses and `set2` the test set:

```
set.seed(46685326, kind="Mersenne-Twister")
perm <- sample(x=nrow(vehdata))
set1 <- vehdata[which(perm <= 3*nrow(vehdata)/4), ]
set2 <- vehdata[which(perm > 3*nrow(vehdata)/4), ]
```

   Print the top 6 observations from each data set.

3. Run a KNN analysis on the training data using $m = 1$.
   (a) **Show the confusion matrix for the test data. Comment on how well separated the four classes are. In particular, are there classes that are easier/harder to separate?**
   (b) Compute and **report the test misclassification rate and approximate standard error.**

4. Reset the seed to `set.seed(9910314, kind="Mersenne-Twister")`. Tune the KNN using CV error with `knn.cv()`. Use a grid from $m = 1, \ldots, 40$. This may take a few seconds or minutes.
   (a) Plot the validation error with standard errors against the number of neighbours. **Show the plot and comment: is there a clear best m or is there a broad range of similar values, according to the SE?**
   (b) Report the value of m with lowest error, as well as the one selected by the 1SE rule.
   (c) Compute the test misclassification rate with both of these parameter values. **Report both error rates, using only one more digit than the first digit in their standard errors.** (For example, if the SE is 0.00375, the first digit in the SE 3 after the decimal, so report error to 4 after the decimal.)

      Keep these error rates handy. You will use them for comparing all classification methods.

      It would be better to use a more rigourous method like multiple reps of CV for computing error rates to make an "arena" for comparing methods. Classification is no different from regression in this way. However, this process is time consuming, and you have already practiced doing this on regression. I want to keep the classification assignments lighter, because they will accumulate rather quickly.

12 Linear Methods for Classification

(Reading: ISLR 4.2, 4.3, 4.4)

12.1 Simple Linear Methods for Classification

- As we have seen, the simplest form of decision boundary between classes is a linear boundary
- There are numerous ways to make linear decision boundaries between classes
  - When true structure of class probabilities forms boundaries between true classes are (approximately) linear, expect these methods to do well
    - Low bias
    - (potentially) Low variance
  - When boundaries are highly nonlinear, then these methods will have high bias and potentially poor misclassification rates.

Linear regression (not actually useful)

A bad way to create linear boundaries is to fit a linear regression to indicator functions for each response

1. Convert classification variable into numerical by creating indicators for group membership
   (a) $Y_1 = I(Y = 1), \ldots, Y_K = I(Y = K)$
   (b) These add to 1 for every observation, see Table 1.

Y	Y_1	Y_2	Y_3	Y_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Table 1: Converting categories into indicator variables.

2. It turns out mathematically, that the mean value of an indicator function equals the probability of a 1.
   (a) Sample proportion is average of sample of 1's and 0's
3. Could try using a linear model separately fitting each Y_k vs X and estimate parameters by least squares
4. Then $\hat{f}(X) = \hat{p}_k(X) = \widehat{\Pr}(Y = k|X)$
5. Decision boundaries where pairs of $\hat{p}_k(X)$'s are equal creates linear boundaries
   (a) At $X = x_0$ classify according to which k has the largest $\hat{p}_k(x_0)$

Problems

1. Linear regression does not constrain $\hat{p}_k(\mathbf{x})$ to lie between 0 and 1!

   (a) Simple example with $p = 1$ and $K = 2$ demonstrates this. See Figure 4.

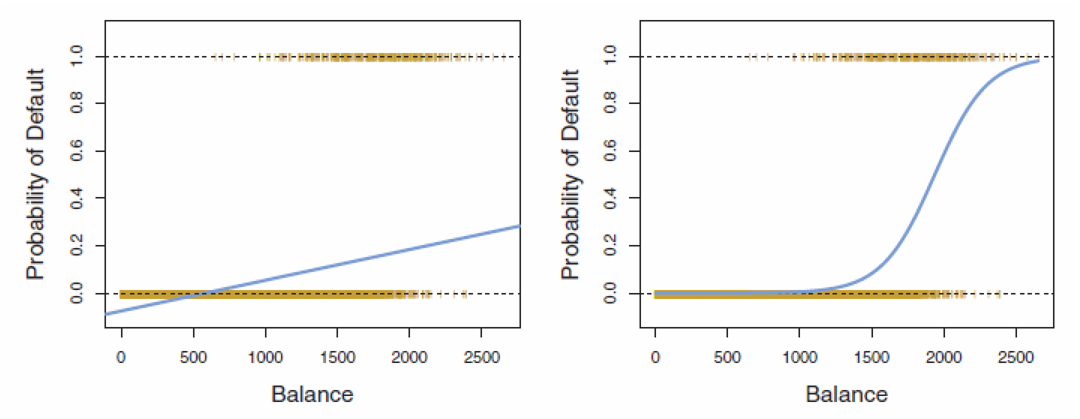

FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default`*(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using logistic regression. All probabilities lie between* 0 *and* 1.

Figure 4: Comparison of linear and logistic regression classifiers. From ISLR.

   (b) Modeling $E(Y)$, not thinking of it as a probability

2. Linear regression has even more problems with $K > 2$

   (a) Can miss classes completely depending on how they are arranged in X

12.2 Logistic Regression

Logistic regression is form of regression developed specifically for binary responses ($K = 2$):

- With $K = 2$, note that $p_2(x) = 1 - p_1(x)$
  - So only need to model one of these
- Rewrite Y as a single indicator function, Y =1 (“Success”) or 0 (“Failure”)
- Probability of success = $p(X)$
- Assume that Y has a Bernoulli/Binomial distribution
- Assume a linear model for “logit” (log-odds of success) instead of mean

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

- Estimate parameters $(\beta_0, \beta_1, \ldots, \beta_p)$ using “maximum likelihood” (ML)

  - $\hat{p}(\mathbf{x}) = \frac{\exp(\hat{\beta}_0+\hat{\beta}_1 X_1+\ldots+\hat{\beta}_p X_p)}{1+\exp(\hat{\beta}_0+\hat{\beta}_1 X_1+\ldots+\hat{\beta}_p X_p)} = \frac{1}{1+\exp[-(\hat{\beta}_0+\hat{\beta}_1 X_1+\ldots+\hat{\beta}_p X_p)]}$

- When $p = 1$, probability curve looks like right side of Figure 4.
  - Can make decision boundary at all x where $\hat{p}(X) = 0.5$
  - Choose class 1 if $\hat{p}(X) > 0.5$
- When $p > 1$, creates linear (or hyperplane) boundaries between classes

Extension to $K > 2$ classes

Not quite as well known but still pretty classic stuff

1. Use multinomial distribution, where class probabilities depend on X, $p_k(X)$, $k = 1, \ldots, K$
2. Use "multi-response logit" link, comparing each class to some baseline category (doesn't matter which one; predictions are the same)
   (a) For example, if the last class is chosen as the baseline, then make $K - 1$ logit ratios
   $$\log\left(\frac{p_k(X)}{p_K(X)}\right) = \beta_{k0} + \beta_{k1}X_1 + \ldots + \beta_{kp}X_p,\ k = 1, \ldots, K-1$$
   (b) Results in probabilities
   $$\Pr(G = k|X) = \frac{\exp[\beta_{k0} + \beta_{k1}X_1 + \ldots + \beta_{kp}X_p]}{1 + \sum_{l=1}^{K-1} \exp[\beta_{l0} + \beta_{l1}X_1 + \ldots + \beta_{lp}X_p]},\ k = 1, \ldots, K-1$$
   and
   $$\Pr(G = K|X) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp[\beta_{l0} + \beta_{l1}X_1 + \ldots + \beta_{lp}X_p]}$$
      i. Parameters again estimated by ML
3. Also produces linear boundaries (can be proved mathematically)
4. Then the classifier $\hat{f}(x)$ returns class k if $\hat{p}_k(x)$ is the largest class probability.
5. Results in $(p + 1)(K - 1)$ parameters.
   (a) In vehicle image data, this is $19 \times 3 = 57$

Notes on Logistic Regression

1. One advantage of logistic regression is that wealth of inference procedures that follow it, thanks to ML estimation (assuming no model selection)
   (a) Confidence intervals for parameters and for probability estimates
   (b) Tests for individual parameters or groups of them in ANOVA-style tests
   (c) Variable-selection methods
      i. All subsets, stepwise based on information criteria, which were developed explicitly for ML estimates
      ii. LASSO has a version that works with ML to shrink estimates and select variables

EXAMPLE: Logistic Regression on Wheat Data (`Sec12_Logistic_Regression_Wheat.R`)

The package `nnet` that does neural nets has a function that fits multinomial models, `multinom()`. Evidently, it just skips the hidden layer and combines the inputs directly into a linear combination, transformed by a sigmoidal output function. It can be slow and fail to converge with too few iterations. Also, it requires that explanatory variables be scaled to lie approximately between 0 and 1, or else it may give very strange results.

```
> summary(mod.fit)
Call:
multinom(formula = type ~ ., data = set1.rescale, trace = TRUE)

Coefficients:
       (Intercept)   density  hardness     size      weight
Scab     16.096363 -18.81505 -4.915380 6.736446 -18.2989406
Sprout    8.761246 -12.50927 -4.052665 1.803399  -0.9999576
         moisture    classnum
Scab    0.1109796 -0.77462360
Sprout -1.6636584 -0.04881767

Std. Errors:
       (Intercept)  density hardness     size   weight
Scab      2.518195 3.176054 2.086737 3.423757 3.861902
Sprout    1.981611 2.593197 1.574296 2.264907 1.797390
       moisture  classnum
Scab   1.599686 0.8898911
Sprout 1.076341 0.6012836

Residual Deviance: 250.127
AIC: 278.127
>
> # Adding a function to perform LR Tests.
> #  Legitimate here since I am fitting one model
> library(car)
> Anova(mod.fit)
Analysis of Deviance Table (Type II tests)

Response: type
         LR Chisq Df Pr(>Chisq)
density    68.440  2  1.375e-15 ***
hardness    8.402  2    0.01498 *
size        3.874  2    0.14413
weight     38.271  2  4.893e-09 ***
moisture    3.195  2    0.20244
classnum    0.873  2    0.64620
```

The first part of the output gives the logic regression coefficients for the two logit models. At the end is a test for the significance of each variable, testing the null hypotheses that a given variable could be dropped completely (from both logits) avainst the alternative that it cannot. We see that `density` and `weight` seem to be dominant variables here, while `hardness` may also be important. Other variables seem less important, given the rest of the model.

Below we show a sample of the estimated class probabilities from the test set, along with the test set confusion matrix. While the confusion matrix is better looking than the ones from KNN, there are still relatively large numbers of misclassifications across the entire table.

```
> pred.probs.2 <- predict(mod.fit, newdata=set2.rescale,
+                          type="probs")
> round(head(pred.probs.2),3)
   Healthy  Scab Sprout
3    0.623 0.022  0.355
11   0.396 0.403  0.201
16   0.686 0.017  0.297
17   0.147 0.759  0.094
18   0.760 0.021  0.218
19   0.207 0.650  0.143
>
> # Test set confusion matrix
> table(set2$type, pred.class.2, dnn=c("Obs","Pred"))
         Pred
Obs        Healthy Scab Sprout
  Healthy       16    4      5
  Scab           4   14      4
  Sprout         7    6     15
```

The `glmnet` package that fits the LASSO fits a different version of the problem. It does a logistic regression on each binary indicator $Y_1, Y_2, \dots Y_k$. So it provides a set of coefficients for all K classes instead of for $K-1$ comparisons with the baseline class. This is not the same analysis. It might be fine in its own right, but it is not the usual multinomial regression.

Finally, we show the list of all test error rates achieved so far in Table 2

Method	Training	Test	Test "SE"
KNN-1	0	0.47	0.06
KNN-Opt(12)	0.28	0.49	
Logistic	0.28	0.40	
Logist-glmnet	0.28	0.40	
Logist-LASSO-min	0.28	0.43	

Table 2: The test error rates.

Using the binomial distribution to approximate the number of misclassified observations in the test set, we can calculate that the standard error for an estimated error rate between 0.4 and 0.5 is about 0.06. So we see that the test error rate from Logistic is more than a standard error lower than from KNN.

12.3 Linear Discriminant Analysis

- Logistic regression is useful, but it has one really painful quirk: **Parameter estimates and boundaries become unstable when classes have little overlap.**
- When classes in sample are completely separated, parameter estimation in logistic regression actually fails
  - Slope wants to be infinite
  - Infinitely many intercepts provide identical perfect fit
- This is frustrating, because (near-) separable classes is the case where classification methods ought to be easy to fit!
- Fortunately, there are other, very different ways to make linear boundaries

LDA Concepts

- Linear discriminant analysis (LDA) is a very popular tool for classification
- LDA operates in the reverse direction from regression-type methods like logistic
  - Regression methods model the distribution of the response, given the value of the explanatory variable
    - Logistic regression models $P(Y|X)$ using bi-/multinomial
  - LDA instead models distribution of the explanatory variables, given the response
    - Consider each class separately—what does distribution of X look like?
    - Then use a conditional probability formula called BAYES' RULE to estimate the probability of each class, given X
- For example, suppose we wanted to understand the effects of, say, blood pressure on death due to contracting Covid-19.
  - We look at distribution of blood pressure for people who die from Covid-19
  - We look at distribution of blood pressure for people who survive Covid-19
  - For any particular BP, x, the ratio of these two distributions at x gives a measure of how likely x is to have come from the death group than vs. the survive group. See Figure 5

140 4. Classification

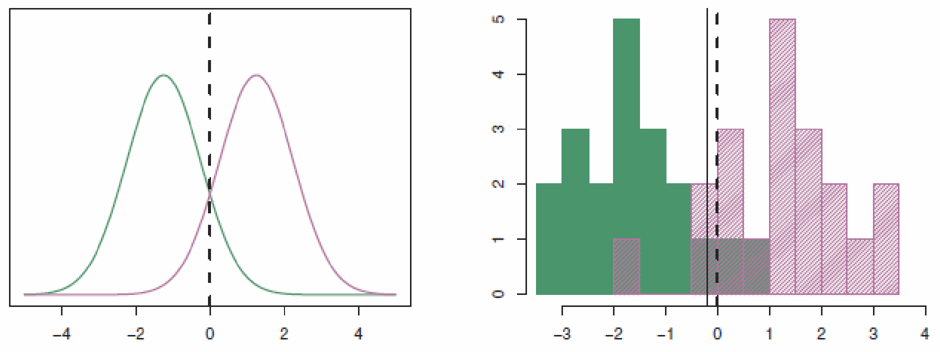

FIGURE 4.4. Left: *Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary.* Right: *20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.*

Figure 5: Example of conditional distributions of X for $K = 2$ classes. When each class is equally likely, Bayes rule says to classify according to highest curve. (From ISLR.)

  - If death and survival were equally likely overall then Bayes' rule would classify according to which group is more likely to have a blood pressure measurement at x
  - More generally, we also factor in the total fractions of the population in each group and formalize all of this into probability calculation

  - End result is exactly $p_k(x) = p(Y = k|X = x)$

- In Discriminant Analysis, we *assume* that X has a MULTIVARIATE NORMAL DISTRIBUTION (MVN) in each class
  - Each variable X_j has a normal distribution within each class
    * For class k, mean is μ_{jk} and variance is σ^2_{jk}
    * The means and variance may differ across classes in general (more on this in a minute)
  - Each pair of variables $X_j, X_{j'}$ has correlation within each class $\rho_{jj',k}$
  - See Figure 6

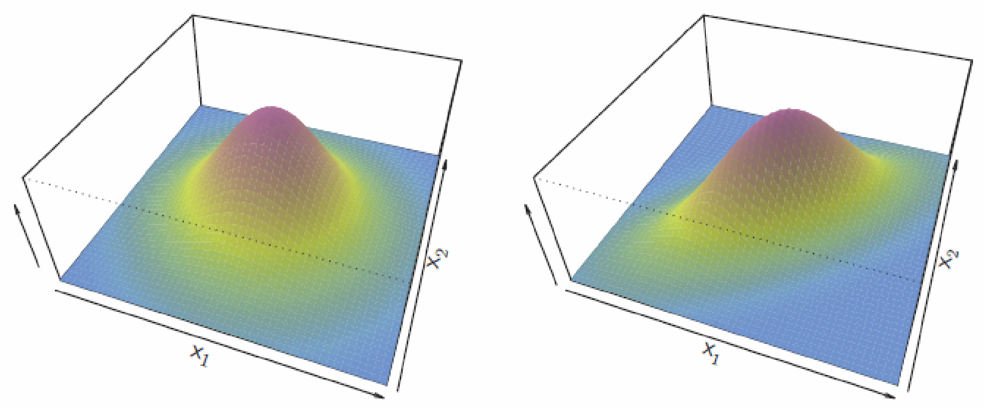

FIGURE 4.5. *Two multivariate Gaussian density functions are shown, with* $p = 2$. Left: *The two predictors are uncorrelated.* Right: *The two variables have a correlation of* 0.7.

Figure 6: 3D pictures of normal distributions for $p = 2$ with different correlations. (From ISLR.)

  - FYI: Normal is also called "Gaussian" (ISLR does calls it Gaussian)

- **Linear** DISCRIMINANT ANALYSIS further assumes that the variances and correlations are not changing across k, but that the means may.
  - Identical variances and correlations forces the distributions have the same "shape" (See Figure 7)
  - Decision boundaries turn out to be linear

- Practically speaking, LDA produces $K - 1$ "linear discriminants", say $Z_1, \ldots, Z_{K-1}$
  - Linear combinations of explanatory variables, like Z in PCA, PLS
    * First one "discriminates" the K classes (normal distributions) as much as possible
    * Second discriminates what first did not, etc through $K - 1$
  - Plotting combinations of them can be interesting.

- If we knew the population means, variances, and correlations, then LDA would return Bayes classifier

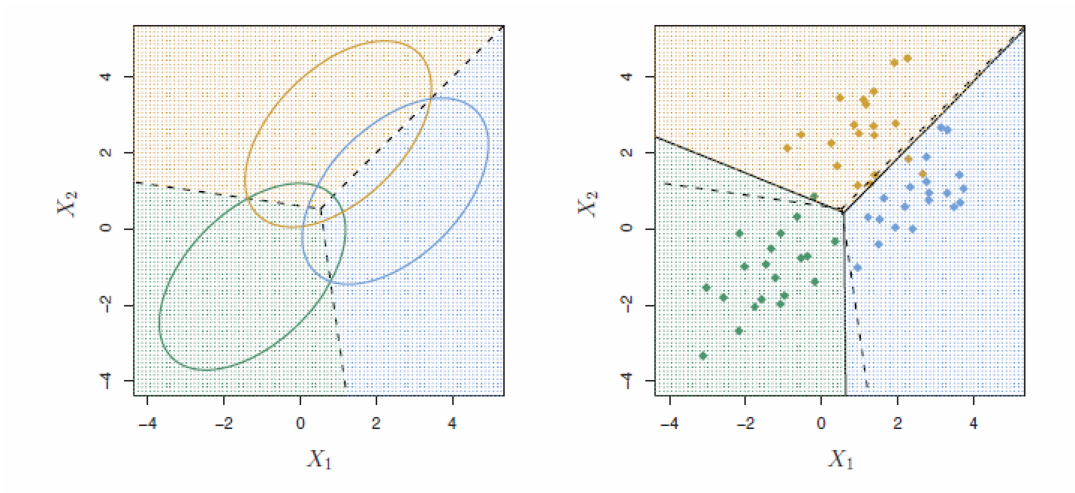

FIGURE 4.6. *An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with* $p = 2$, *with a class-specific mean vector and a common covariance matrix.* Left: *Ellipses that contain 95 % of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries.* Right: 20 *observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.*

Figure 7: 3D pictures of normal distributions for $p = 2$ with different correlations. (From ISLR.)

- In practice, have to estimate parameters, so merely approximate Bayes classifier
    - K means
    - p variances
    - $p(p-1)/2$ correlations
- As n grows relative to $K + p(p+1)/2$, LDA test error rate approaches error rate of Bayes Classifier

Quadratic Discriminant Analysis

- LDA is great if the the variance and correlation structure for X is the same in all classes
    - If it's not, then LDA is biased
        - True decision boundaries are not linear
        - LDA can't adapt
- Alternative version is *not* to assume that variance/correlation structures are identical across classes
    - Let them be different in each class
    - Results in decision boundaries that are quadratic surfaces
    - Quadratic Discriminant Analysis (QDA)
- Which is better, LDA or QDA?

  - Answer is usual bias/variance tradeoff
  - Quadratic can better match certain surfaces than linear, and can choose to be flat if covariance matrices between two classes are about the same
    * Hence reduced bias
  - But extra parameters to be estimated, not all of which may be necessary
    * Has $(K-1)p(p+1)/2$ more parameters than LDA
    * Hence extra variance

- Both of these tools are useful and are in frequent use in practice.
  - However, both rely on assumption of normality for X in each class
  - Still useful even when normal assumptions are obviously false, as with binary variables
    * But no longer capable of mimicking Bayes classifier
    * Take on bias that more flexible methods might be able to eliminate

EXAMPLE: Discriminant Analysis on Wheat Data (`Sec12_Discrim_Wheat.R`)

The package `MASS` has functions `lda()` and `qda()` that do linear and quadratic discriminant analysis, respectively. As usual, there are a variety of helper functions for examining the results. If we first standardize the explanatory variables to have mean 0 and variance 1 in the combined data, then when the separate class means are estimated, we can see which ones are most different. See below.

```
> set1s <- apply(set1[,-6], 2, scale)
> set1s <- data.frame(set1s,type=set1$type)
> lda.fit.s <- lda(data=set1s, type~.)
> lda.fit.s
Call:
lda(type ~ ., data = set1s)

Prior probabilities of groups:
Healthy    Scab  Sprout 
  0.355   0.305   0.340 

Group means:
            density   hardness       size     weight
Healthy  0.70102694  0.1265443  0.2897454  0.3743367
Scab    -0.82241704  0.1705631 -0.7327091 -0.9532671
Sprout   0.00580186 -0.2851323  0.3547549  0.4642851
           moisture      classnum
Healthy  0.15005446  0.0721051156
Scab    -0.07828193 -0.0845800172
Sprout  -0.08645102  0.0005870271

Coefficients of linear discriminants:
                LD1         LD2
density  -0.84687425 -0.73968176
hardness -0.14168390 -0.44763549
size      0.07351008  0.21444413
weight   -1.01267254  0.53878781
moisture  0.01751022 -0.35676216
classnum -0.02125836  0.07436082
```

```
Proportion of trace:
   LD1    LD2
0.8999 0.1001
```

Agreeing with the logistic regression analysis, the class mean for `density` and for `weight` seem most different from one another. We also see the coefficients that create the linear discriminants, Z_1 and Z_2. Figure 8 plots the three classes against these two linear discriminants. We can see that the first one (`LD1`) seems to mostly separate Scab from the other groups, while the second tries—with limited success—to separate healthy and sprout.

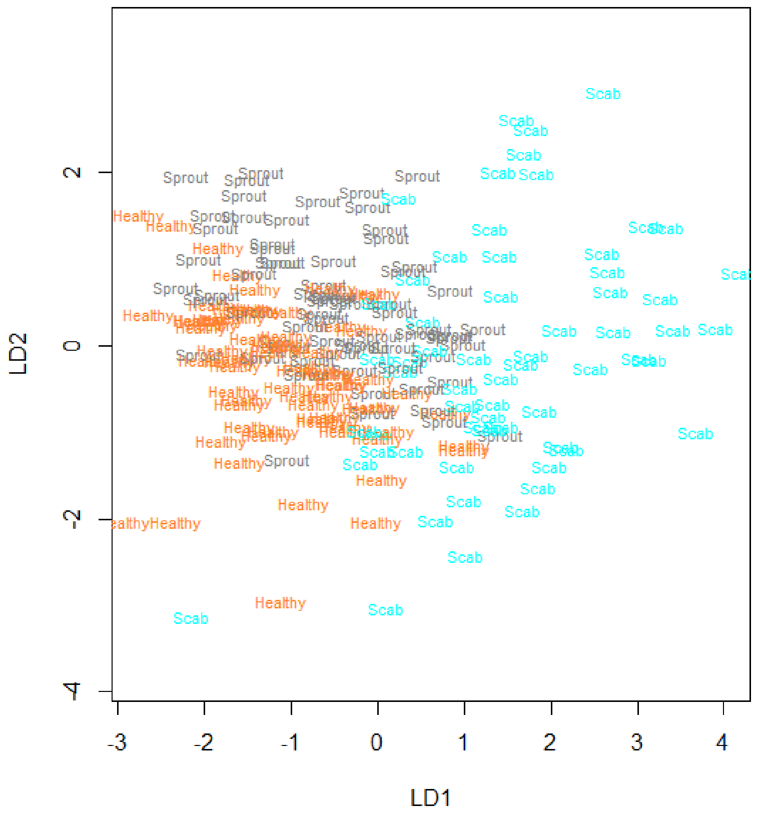

Figure 8: Plot of three classes on the two linear discriminants for the Wheat Data

The test error for LDA is in Table3. We also run QDA in the program, and its test error is also below. LDA is mostly indistinguishable from logistic regression, and QDA does not seem to help.

What to take away from this

1. Linear methods for classification are well-studied and easy to use (sort of).
2. There are many more bells and whistles than what I've shown, so follow-up diagnostics are possible
    (a) In particular, especially when $K > 2$ it can sometimes be instructive to look at the misclassified cases

Method	Training	Test	Test "SE"
KNN-1	0	0.47	0.06
KNN-Opt(12)	0.28	0.49	
Logistic	0.28	0.40	
Logist-glmnet	0.28	0.40	
Logist-LASSO-min	0.28	0.43	
LDA	0.28	0.39	
QDA	0.26	0.43	

Table 3: Test Error

i. Do they tend to be all from certain groups or are they spread out more equally?

3. Extensions have been proposed to make them more robust or better classifiers.

   (a) These extensions represent additional tools to consider.

Problem Set 14: Linear Classification Methods

Application

Return to the Vehicle data used in the previous lecture. Use the same split as before.

```
set.seed(46685326, kind="Mersenne-Twister")
perm <- sample(x=nrow(vehdata))
set1 <- vehdata[which(perm <= 3*nrow(vehdata)/4), ]
set2 <- vehdata[which(perm > 3*nrow(vehdata)/4), ]
```

1. Run logistic regression using `multinom()`.

   (a) Scale the training data to lie between 0 and 1, and use the same min and max values to scale the test data. Run `summary()` in each scaled data set to confirm that you are doing this correctly. **Report the summary of the first 3 variables in each set.**

   (b) Run the logistic regression model using all explanatory variables.

      i. Run `Anova()` on the object. **Report the table of test results and comment on which variables seem to be important or unimportant.**

      ii. Compute and **report training and test error**. **Does test error seem better or worse than optimal KNN?** (Use the standard error computed before to help you make a sensible comment here.)

      iii. **Report the confusion matrix and comment sensibly on what it tells you.**

2. Run a LASSO version of logistic regression, using CV to estimate optimal shrinkage in the logistic regression parameters using minimum CV-error.

   (a) **Report a list of the variables included/excluded in each logit. Does the pattern seem somewhat consistent with the ANOVA results from earlier? Explain in a sentence.**

   (b) Compute and **report training and test error**. **How does test error compare to other methods?**

3. Run LDA on these data.

   (a) Make a colour plot of the classes against pairs of linear discriminants. The `plot()` function will do this for you automatically. Use these colours:
   ```
   class.col <- ifelse(set1$class==1, y = 53,
   n = ifelse(set1$class==2, y = 68, n = ifelse(set1$class==3,y=203,n=464)))
   ```
   Present the plot and write a sentence for each linear discriminant, explaining how it seems to separate classes.

   (b) **Report training and test error. How does test error compare to other methods?**

4. Run QDA on these data. **Report training and test error. How does test error compare to other methods?**