Classification

Zack Treisman

Spring 2022

```
library(tidyverse)
library(GGally)
library(nnet)
library(caret)
library(MASS)
library(e1071)
library(tree)
library(randomForest)
set.seed(3)
```

Philosophy

Thus far the response variables of our models have been numeric.

▶ A binomial glm is used when the response is a binary categorical variable, but the response in the model is the probability of observing the level designated success.

Options to predict a categorical response with more than two levels:

▶ Nearest-neighbor averaging (KNN, `class::knn`[1])
▶ Multinomial logistic regression
▶ Discriminant analysis (linear, quadratic, naive Bayes, . . .)
▶ Classification trees and Random Forests
▶ Support vector machines
▶ Neural networks (aka deep learning)
▶ Other machine learning methods

---

[1] The R syntax `a_library::an_object` lets you use `an_object` from `a_library` without first loading the library. I'll use it in these notes as shorthand - "the function `knn` in the library `class`."

Binomial GLMs

Consider a binary categorical response variable Y (levels 0 and 1), and predictor variables $X = (X_1, X_2, \ldots, X_m)$.

In a binomial GLM, we model the expected probability $p(x) = P(Y = 1 | X = x)$ using

$$\text{logit}(p(x)) = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m$$

where

$$\text{logit}(p(x)) = \log\left(\frac{p(x)}{1 - p(x)}\right).$$

Equivalently,

$$\frac{p(x)}{1 - p(x)} = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m}$$

or

$$p(x) = \frac{e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m}}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m}}.$$

Log-odds and interpreting coefficients

For a probability p, the expression $\frac{p}{1-p}$ is called the **odds**.

Example: Let A be some event with $P(A) = 0.75$.

$$\text{odds}(A) = \frac{0.75}{1 - 0.75} = \frac{3}{1}$$

This is referred to as **3 to 1 odds**, meaning $P(A) = 3P(\text{not } A)$.

An event whose odds are 1 is as likely to happen as not.

We use this to **interpret** the model

$$\text{odds}(Y = 1 | X = x)) = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m}$$

by saying that

- e^{β_0} is the odds that $Y = 1$ when all the $X_i = 0$.
- e^{β_i} is the factor by which the odds that $Y = 1$ changes when x_i increases by 1 with all the other x_j remaining fixed.

Multinomial regression

Now consider Y with levels $1, \ldots, K$. We still want to know $P(Y = k | X = x)$ for an arbitrary level k.

We either choose a reference level or not. The model is the same either way, this only affects the particular parameters that we have to estimate and get to interpret.

Suppose that it makes sense to set level K as a reference level. Then for $k = 1, \ldots, K - 1$

$$P(Y = k | X = x) = \frac{e^{\beta_{k0} + \beta_{k1} x_1 + \cdots + \beta_{km} x_m}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_{l1} x_1 + \cdots + \beta_{lm} x_m}}$$

and

$$P(Y = K | X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_{l1} x_1 + \cdots + \beta_{lm} x_m}}$$

Log-odds in Multinomial Regression

The same algebra as in the binary case shows that the odds of a particular level, relative to the chosen reference level are

$$\frac{P(Y = k|X = x)}{P(Y = K|X = x)} = e^{\beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{km}x_m}$$
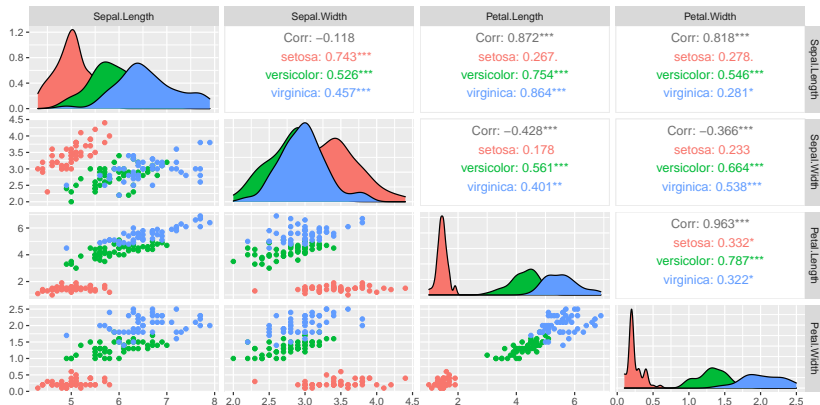
and so the interpretations of the parameters are similar:

- $e^{\beta_{k0}}$ is the odds that $Y = k$ rather than $Y = K$ when all the $X_i = 0$.
- $e^{\beta_{ki}}$ is the factor by which the odds that $Y = k$ rather than $Y = K$ changes when x_i increases by 1 with all the other x_j remaining fixed.

The `iris` data

A famous data set for testing classification methods.

```
ggpairs(iris[,1:4], aes(color=iris$Species))
```

Multinomial regression example

A tool for multinomial regression is `nnet::multinom`.

```
iris$Species <- relevel(iris$Species, ref = "versicolor") # set the reference level
mlr_iris <- multinom(Species~., data=iris) # Species in terms of everything else
```

```
summary(mlr_iris)
```

```
## Call:
## multinom(formula = Species ~ ., data = iris)
##
## Coefficients:
##           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa      -0.4917771     2.205180    6.313709   -10.370916   -5.135036
## virginica -42.3514649    -2.471389   -6.655958     9.391305   18.207938
##
## Std. Errors:
##           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa       42.21659   100.974966  162.889191   100.537988   48.061685
## virginica    25.52521     2.394038    4.461874     4.703818    9.688497
##
## Residual Deviance: 11.89917
## AIC: 31.89917
```

Multinomial regression is unstable when classes are well separated.
This is the case for setosa. Note the high standard errors.

Predictions from multinomial regression

Even though the standard errors on the model coefficients are quite high for setosa, the model almost always makes correct predictions if we choose the level with the highest probability.

```
mlr.pred <- predict(mlr_iris)
table(iris$Species, mlr.pred)
```

```
##              mlr.pred
##               versicolor setosa virginica
##   versicolor          49      0         1
##   setosa               0     50         0
##   virginica            1      0        49
```

Discriminant Analysis

Another way to approach classification is to model the distribution of X for each class, then apply Bayes' theorem. We'll follow James et al. (2021) and use the following terminology and notation:

- $f_k(x) = P(X = x | Y = k)$ is the **distribution** of X for class k.
- $\pi_k = P(Y = k)$ is the **prior** probability of class k.
- $p_k(x) = P(Y = k | X = x)$ is the **posterior** probability of class k given values of the predictors X.

Bayes' theorem says

$$p_k(x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Logistic regression was one method of estimating the probabilities $p_k(x)$. This is another. As before, given $X = x$, we can choose the class k for Y such that $p_k(x)$ is largest.

Linear Discriminant Analysis

Two strong assumptions make finding the largest $p_k(x)$ much easier.

- ▶ The distributions $f_k(x)$ are normal. $f_k(x) = \frac{1}{\sigma_k\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$
- ▶ The variance is the same for all classes: $\sigma_k = \sigma$.

Taking logs of the resulting formulas for $p_k(x)$ and discarding terms that don't depend on k gives the *discriminant*:

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k).$$

This is a linear function of x. It is easy to compute and $p_k(x)$ is greatest when $\delta_k(x)$ is greatest. Find the *decision boundary* between class j and class k by solving $\delta_j(x) = \delta_k(x)$.

- ▶ The above is written assuming that there is only a single predictor x. For multiple predictors, the math is similar, with x replaced by the vector of predictors and σ by the covariance matrix. Decision boundaries are linear.

LDA illustration
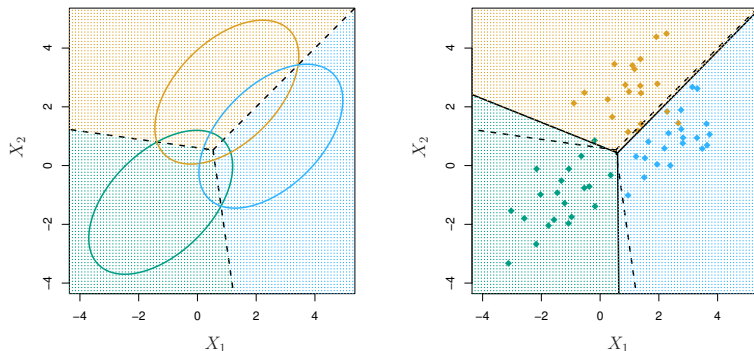
This graphic is from James et al. (2021). It shows simulated data from Gaussian distributions indicated by the ovals. Dotted lines are optimal decision boundaries. Solid lines are decision boundaries from LDA.

LDA example (model fitting)

The `lda` command from `MASS` performs LDA. The LD1 and LD2 vectors make a nice basis for the plane spanned by the centroids of the 3 species.

```
lda.mod <- lda(Species~., data = iris)
lda.mod
```

```
## Call:
## lda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
## versicolor      setosa  virginica
##  0.3333333   0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor        5.936       2.770        4.260       1.326
## setosa            5.006       3.428        1.462       0.246
## virginica         6.588       2.974        5.552       2.026
##
## Coefficients of linear discriminants:
##                    LD1         LD2
## Sepal.Length -0.8293776  0.02410215
## Sepal.Width  -1.5344731  2.16452123
## Petal.Length  2.2012117 -0.93192121
## Petal.Width   2.8104603  2.83918785
##
## Proportion of trace:
##    LD1    LD2
## 0.9912 0.0088
```

LDA example (model evaluation)

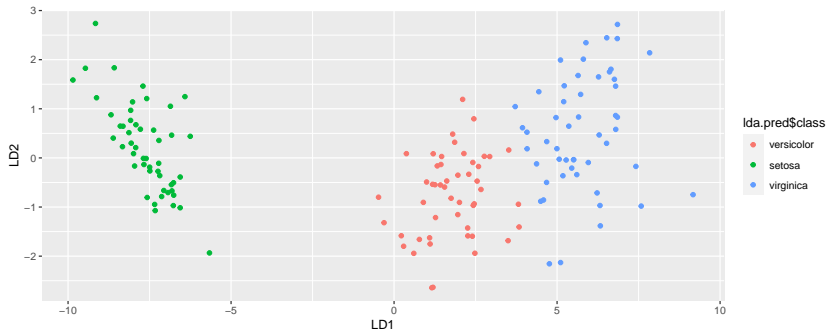The model is correct in all but 3 cases.

```
lda.pred <- predict(lda.mod)
mean(lda.pred$class==iris$Species)
```

```
## [1] 0.98
```

```
sum(lda.pred$class!=iris$Species)
```

```
## [1] 3
```

```
lda.obs <- as.data.frame(lda.pred$x)
ggplot(lda.obs, aes(LD1, LD2, color=lda.pred$class))+geom_point()
```

# LDA example (posterior probabilities)

Posterior probabilities are also available from `predict.lda`.

```
lda.post <- as.data.frame(lda.pred$posterior)
ggplot(lda.obs, aes(LD1, LD2))+
  geom_point(aes(size=lda.post$setosa, color="setosa"), alpha=0.5)+
  geom_point(aes(size=lda.post$virginica, color="virginica"), alpha=0.5)+
  geom_point(aes(size=lda.post$versicolor, color="versicolor"), alpha=0.5)+
  labs(size="Confidence", color="Species")
```
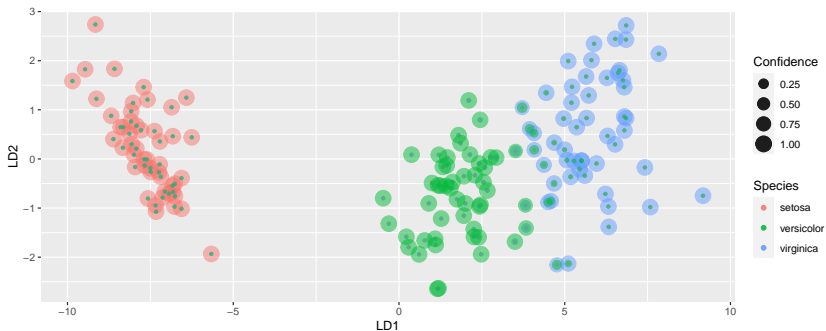
Quadratic Discriminant Analysis
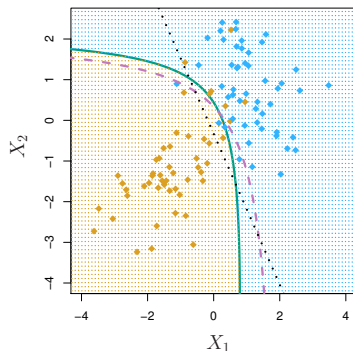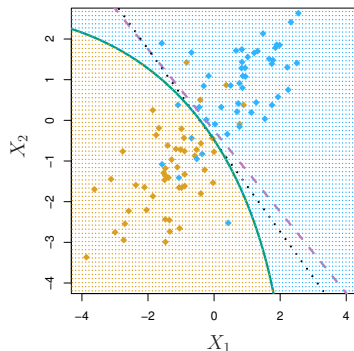
Keeping the normality assumption but relaxing the condition that all of the variances are equal leads to *quadratic discriminant analysis*. The decision boundaries are quadratic instead of linear.

▶ Smaller data sets do better with LDA.

QDA example (model fitting)

QDA doesn't do dimension reduction as a byproduct like LDA.

```
qda.mod <- qda(Species~., data = iris)
qda.mod
```

```
## Call:
## qda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
## versicolor     setosa  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor        5.936       2.770        4.260       1.326
## setosa            5.006       3.428        1.462       0.246
## virginica         6.588       2.974        5.552       2.026
```

QDA example (model evaluation)

The predictions from QDA are the same as LDA on these data.

```
qda.pred <- predict(qda.mod)
sum(qda.pred$class!=iris$Species)
```

```
## [1] 3
```

```
qda.post <- as.data.frame(qda.pred$posterior)
ggplot(lda.obs, aes(LD1, LD2))+
  geom_point(aes(size=qda.post$setosa, color="setosa"), alpha=0.5)+
  geom_point(aes(size=qda.post$virginica, color="virginica"), alpha=0.5)+
  geom_point(aes(size=qda.post$versicolor, color="versicolor"), alpha=0.5)+
  labs(size="Confidence", color="Species")
```
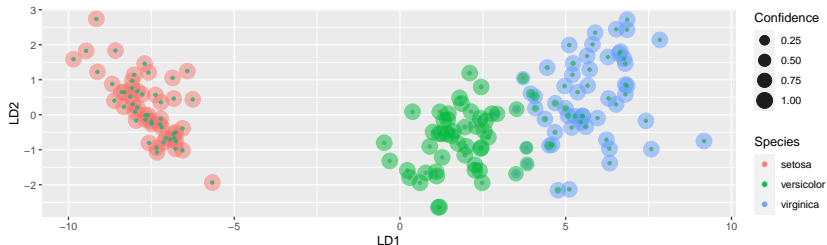
Naive Bayes

With multiple predictors $x = (x_1, \ldots, x_m)$, assuming that the distribution $f_k(x)$ is a product of terms involving only one x_i at a time gives a form for $\delta_k(x)$ that is useful for large numbers of predictors. ($m >> 0$)

```
nb.mod <- naiveBayes(Species~., data=iris)
nb.pred <- predict(nb.mod, newdata = iris)
sum(nb.pred!=iris$Species)
```

```
## [1] 6
```

```
ggplot(lda.obs, aes(LD1, LD2, color=nb.pred))+geom_point()+
  labs(color="Naive Bayes Prediction")
```
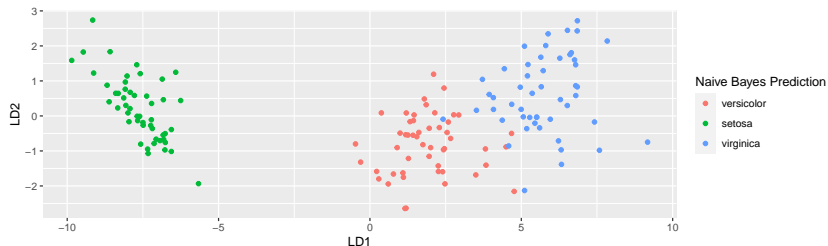
Trees

Decision trees are very intuitive.

```
tree.iris=prune.tree(tree(Species~.,iris), best=4); summary(tree.iris)
```

```
##
## Classification tree:
## snip.tree(tree = tree(Species ~ ., iris), nodes = c(7L, 12L))
## Variables actually used in tree construction:
## [1] "Petal.Length" "Petal.Width"
## Number of terminal nodes:  4
## Residual mean deviance:  0.1849 = 26.99 / 146
## Misclassification error rate: 0.02667 = 4 / 150
```

```
plot(tree.iris); text(tree.iris,pretty=0)
```

Tree Building

Building a tree consists of recursively splitting the data into regions using thresholds on the predictors.

The process is *top-down* and *greedy*.

- ▶ Top-down because we start with unlabeled data and perform the best split.
- ▶ Greedy because we don't look ahead and make any sub-optimal split that will then allow for better splits further down the tree.

After building a tree, we need to prune it back to avoid overfitting. The degree of pruning is a parameter that can be tuned using cross validation.

Tree Building Details

To determine the best split, we use *Gini index* or *cross-entropy*. Both are defined for a region (subset of the data domain) corresponding to a node/leaf of the tree.

▶ The Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k)$$

where $\hat{p}_k$ is the proportion of training observations in the region that are from the kth class.

▶ The cross-entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

These two measures are similar numerically. Small values indicate that the region in question is homogeneous.

Unfortunately . . .

Single trees do poorly at predicting compared to the other methods we have studied.

Random Forests and Bagging

By building a lot of trees, adding noise to the process as we do so, and then aggregating the predictions we can build very effective predictive models.

- *Bagging* (bootstrap aggregation) in a general technique for building many models and averaging them. Bootstrap samples of the data are used to create replicates of the data for training models.
- The *Random Forest* algorithm builds trees on bootstrapped training samples, but at each step, only a random subset of the predictors are considered as options for defining a split.
- These additions of noise result in models with lower variance and higher predictive power.

Random Forest Example

```
west_fork <- read.csv("data/FieldPlots_WithSpatial.csv")
west_fork$bc_sqrt_7 <- factor(west_fork$bc_sqrt_7)
rf.mod <- randomForest(bc_sqrt_7~.,
                       data = west_fork[, c(4,8:32)], importance=TRUE)

accuracy <- function(tab)sum(diag(tab))/(sum(tab))
rf.tab <- table(predict(rf.mod),west_fork$bc_sqrt_7)
rf.tab
```

```
##
##      1  2  3  4  5  6  7
##   1  6  0  5  0  0  3  2
##   2  0 91  6 20 10  4  0
##   3 22  9 59  8  0  3 16
##   4  0  9  4 20  2  0  4
##   5  0  2  0  0  4  0  0
##   6  0  0  0  0  0  0  0
##   7  0  0  2  0  0  0  1
```
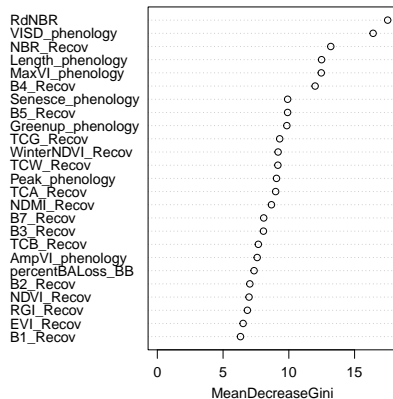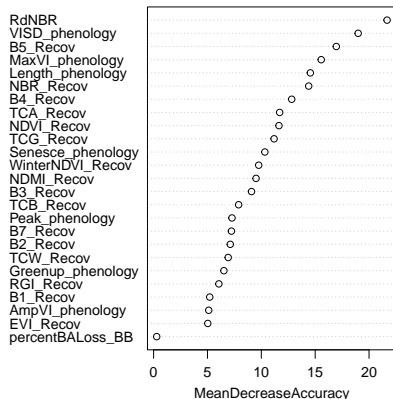
```
accuracy(rf.tab)
```

```
## [1] 0.5801282
```

Variable Importance

Interpretation of a Random Forest is challenging, but we can get relative levels of variable importance by looking at decreases in the Gini index over all the splits where a particular variable is used.

```
varImpPlot(rf.mod)
```

rf.mod

References

James, G., D. Witten, T. Hastie, and R. Tibshirani. 2021. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer US. https://books.google.com/books?id=g5gezgEACAAJ.