4 Classification

4.1 ISLR 4.1–4.2: Overview of classification

ISLR Chapter 4 covers some traditional statistical approaches to classification problems: logistic regression and discriminant analysis (linear or quadratic). Modern machine learning approaches will be covered in future lectures.

4.2 ISLR 4.3: Logistic regression models

Let p be the probability of an event. The **odds** of the event is $\frac{p}{1-p}$, and the **log-odds** of the event is $\log(\frac{p}{1-p})$. The function $\text{logit}(p) = \log(\frac{p}{1-p})$ is called the **logit** function. In Statistics, log() refers to the natural logarithm unless defined otherwise.

The probability of the event may change with conditions. Let p_k be the probability of the event under Condition k. The **odds-ratio** (OR) for the event between Condition 1 and Condition 2 is $\frac{p_1/(1-p_1)}{p_2/(1-p_2)}$. An OR is always *between two conditions*. If the two conditions are defined by a binary variable, say sex, with its coding well specified (say 0=F, 1=M), one may say "the OR for sex". If the coding is unclear, "the OR for sex" is ambiguous because it could be the OR for man vs. woman or the OR for woman vs. man; if the former is 2 then the latter is 0.5. For a variable with more than two categories (say, age), a phrase like "the OR for age" does not make sense (unless under a specific context, say, linear effect is assumed and the scale of age is specified).

In **logistic regression**, we model the outcome probability (not the outcome itself) as a function of predictors X:

$$p = \sigma(\beta_0 + \beta_1 X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}, \tag{4.1}$$

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X. \tag{4.2}$$

Here $\sigma(t) = \frac{e^t}{1+e^t}$ is called the *sigmoid function* in neural networks. If $p = \sigma(t)$ then $t = \text{logit}(p)$. The model is fit with the **maximum likelihood** approach to obtain the MLEs for the coefficients. Logistic regression models are not classification models per se, although they can be used to derive classifiers.

Interpretation of coefficients: In (4.2), β_1 is the log-OR for every unit increase in X, and e^{β_1} is the OR for every unit increase in X. The CI for β_1 is often symmetric, while the CI for the OR, e^{β_1}, is often asymmetric.

```r
library(ISLR)
mod = glm(default ~ ., data=Default, family=binomial)
options(scipen=999, digits=3)  ## set output options
cbind(mod$coef, confint(mod))  ## log-OR scale
cbind(exp(mod$coef), exp(confint(mod)))  ## OR scale
```

Consider **an additive model** with two binary predictors, sex (0=F, 1=M) and smoking status (0=NS, 1=S),

$$\begin{aligned}\log\left(\frac{p}{1-p}\right) &= \beta_0 + \beta_1 \cdot \text{sex} + \beta_2 \cdot \text{smoking} \\ &= \begin{cases} \beta_0, & \text{if female non-smoker} \\ \beta_0 + \beta_1, & \text{if male non-smoker} \\ \beta_0 \qquad + \beta_2, & \text{if female smoker} \\ \beta_0 + \beta_1 + \beta_2. & \text{if male smoker} \end{cases}\end{aligned}$$

Then

- e^{β_0} is the odds for female non-smokers (baseline sex, baseline smoking status);
- e^{β_1} is the OR for sex (because it is the OR for male vs. female among non-smokers and among smokers);
- e^{β_2} is the OR for smoking (because it is the OR for smoking vs. non-smoking among females and among males).

Now consider **an interaction model** between sex and smoking,

$$\log(\frac{p}{1-p}) = \beta_0 + \beta_1 \cdot \text{sex} + \beta_2 \cdot \text{smoking} + \beta_3 \cdot \text{sex} \cdot \text{smoking}$$
$$= \begin{cases} \beta_0, & \text{if female non-smoker} \\ \beta_0 + \beta_1, & \text{if male non-smoker} \\ \beta_0 \quad\quad + \beta_2, & \text{if female smoker} \\ \beta_0 + \beta_1 + \beta_2 + \beta_3. & \text{if male smoker} \end{cases}$$

Then

- e^{β_0} is the odds for female non-smokers (baseline sex, baseline smoking status);
- e^{β_1} is the OR for male non-smokers vs. female non-smokers. That is, e^{β_1} is the OR for sex among non-smokers. The OR for sex among smokers is $e^{\beta_1+\beta_3}$.
- e^{β_2} is the OR for female smokers vs. female non-smokers. That is, e^{β_2} is the OR for smoking among females. The OR for smoking among males is $e^{\beta_2+\beta_3}$.
- e^{β_3} is a ratio of ORs:

$$e^{\beta_3} = \frac{\text{OR for sex among smokers}}{\text{OR for sex among non-smokers}} = \frac{\text{OR for smoking among males}}{\text{OR for smoking among females}}.$$

(1) β_1 and β_2 do not reflect the overall effects of sex and smoking. That is, they are not "main effects".
(2) β_0, β_1, and β_3 are not comparable because they are different concepts: β_0 is a log-odds, β_1 and β_2 are log-ORs, and β_3 is the difference of two log-ORs. Thus, it is meaningless to make statements like $\beta_1 = \beta_3$ even though they may have the same numbers, nor "the interaction effect is stronger than the main effect".

4.3 Simpson's paradox

Collinearity of the predictors can make coefficients change direction. That is, the effect of x_1 becomes opposite (and still significant!) after x_2 is added to the model. In the following example, `student` has a significantly positive effect when it is the only feature in the model but has a significantly negative effect when `balance` is added to the model.

```r
library(ISLR)
summary(glm(default ~ balance, family=binomial, data=Default))$coef
summary(glm(default ~ student, family=binomial, data=Default))$coef
summary(glm(default ~ balance + student, family=binomial, data=Default))$coef
```

This is similar to Simpson's paradox, where the relationship between X and Y appear quite differently depending on whether Z is taken into account or not. In the example above, X and Y are `student` and `default`, while Z is `balance`. Below are a few additional examples of Simpson's paradox:

1. Berkeley admission data in 1973 (Bickel et al., Science, 1975, 187:398–404): Every department admitted approximately equal fractions of male and female applicants, but the overall tally across departments clearly showed a significantly higher admission rate for males than for females. The reason is that the departments with a high admission rate often had a higher proportion of male applicants than females, and the departments with a low admission rate often had a higher proportion of female applicants. Here, X is the sex of an applicant, Y is the person's admission status, and Z is the department the person applied for.

2. Kidney stone surgery example (Julious and Mullee, 1994, BMJ), where two surgeries for kidney stone removal, open surgery (A) and percutaneous nephrolithotomy (B), were compared for their success rate:

	A	B
Success	273 (78%)	289 (83%)
NS	77	61
Total	350	350

	Small stones		Large stones	
	A	B	A	B
Success	81 (93%)	234 (87%)	192 (73%)	55 (69%)
NS	6	36	71	25
Total	87	270	263	80

Here the stone size (Z) is correlated with both surgery type (X) and the outcome (Y). It is likely that stone size is a confounding variable such that small stones tended to be treated with percutaneous nephrolithotomy (B) and also tended to be successfully removed.

3. Death penalty data from 1976 to 1987 (Radelet and Pierce, 1991, Florida Law Review), where 368 Florida homicide records (for which complete data were available) were analyzed with respect to the race of the suspect and whether death penalty (DP) was the final verdict:

	White	AA
DP	53 (11.0%)	15 (7.9%)
No DP	430	176
Total	483	191

	White victim		AA victim	
	White	AA	White	AA
DP	53 (11.3%)	11 (22.9%)	0 (0%)	4 (2.8%)
No DP	414	37	16	139
Total	467	48	16	143

Here, white victim is strongly associated both with white suspect and with death penalty. Thus marginally, white suspect is associated with death penalty. However, within each group of victim race, AA suspect is associated with death penalty.

4.4 ISLR 4.4

ISLR 4.4 covers LDA/QDA, the confusion matrix, and ROC.

4.4.1 LDA and QDA

Suppose the outcome Y has K classes and there are p features X.

Linear discriminant analysis (LDA): For class k, the features $X \sim N_p(\mu_k, \Sigma)$. These K distributions have different means (centers), but they have the same variance–covariance matrix.

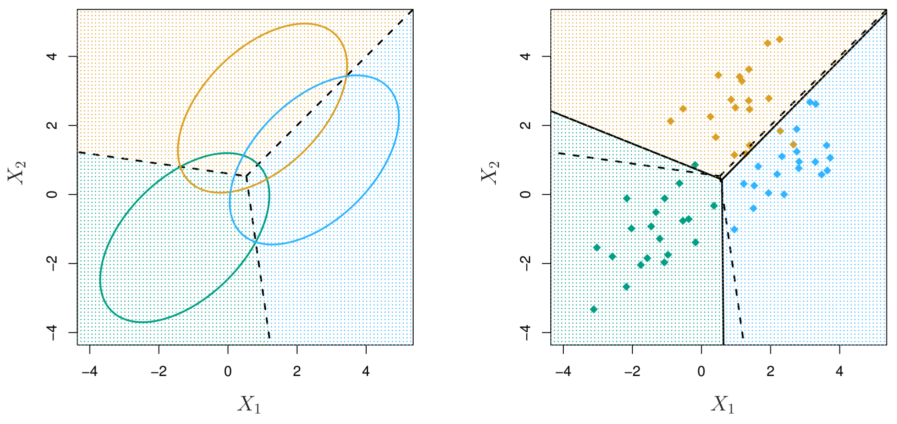
Quadratic discriminant analysis (QDA): For class k, the features $X \sim N_p(\mu_k, \Sigma_k)$. These K distributions have different means and possibly different variance–covariance matrices.

- Both LDA and QDA require multivariate normality (a very strong assumption) for X in every class.
- When $K = 2$ and $p = 1$, LDA has a similar setup as the (equal-variance) t-test. The difference is that the roles of X and Y are switched in LDA, as compared to their roles in the t-test. Similarly, QDA is analogous to unequal-variance t-test.

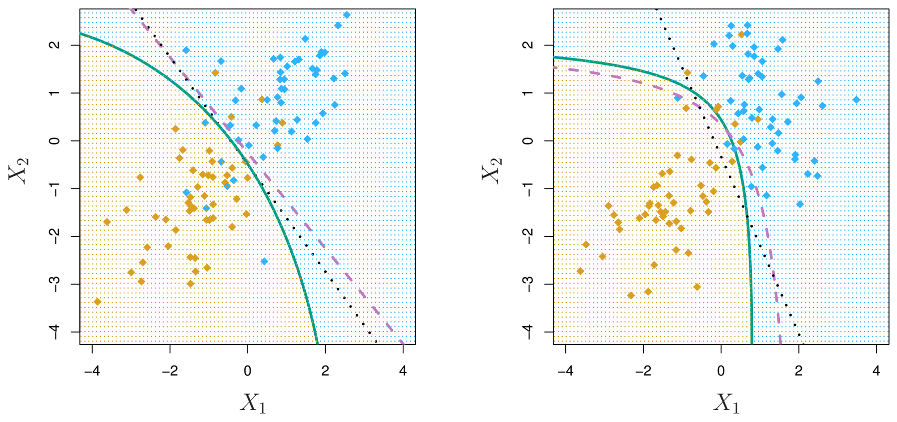
The **prior distribution** should be the true outcome distribution.

- If the data were obtained as a random sample, the prior can be estimated by the sample empirical distribution; otherwise, it may be quite different from the sample empirical distribution.
- When the prior is not specified by the analyst, a random sample is often assumed by software.

In LDA, the K discriminant functions are linear in $\boldsymbol{x}$, and the boundaries are straight lines/planes. (4.13) (4.17) for 1-dimensional X, (4.19) for multi-dimensional X.



In QDA, the K discriminant functions are quadratic in $\boldsymbol{x}$, and the boundaries are not straight lines/planes. (4.23)



Technically, in LDA, given any $\boldsymbol{x}$, when all classes have equal prior probability, we seek to identify the nearest class. Specifically, we seek to identify the class k that has the smallest squared Mahalanobis distance between $\boldsymbol{x}$ and the class center μ_k,

$$d_M^2(\boldsymbol{x}, \mu_k) = (\boldsymbol{x} - \mu_k)^T \Sigma^{-1} (\boldsymbol{x} - \mu_k).$$

- If we have unequal prior probabilities, an offset $-2\log(\pi_k)$ is added to $d_M^2(\boldsymbol{x}, \mu_k)$, and the criterion becomes

  $$d_M^2(\boldsymbol{x}, \mu_k) - 2\log(\pi_k).$$

  This gives more chance to the class that has a higher prior probability.
- In QDA, another offset $\log(|\Sigma_k|)$ is added, and the criterion becomes

  $$d_M^2(\boldsymbol{x}, \mu_k) - 2\log(\pi_k) + \log(|\Sigma_k|).$$

  This gives more chance to the class that has more "concentration" (i.e., a smaller $|\Sigma_k|$).
- The number of parameters in the LDA is $(K-1)(p+1)$; for QDA, $(K-1)(p+1)(\frac{p}{2}+1)$. For example, if $K = 2$ and $p = 10$, then LDA has 11 parameters and QDA has 66 parameters; if $K = 4$ and $p = 10$, then LDA has 33 parameters and QDA has 198 parameters.

Here is an example of LDA analysis in R:

```
library(MASS)  ## for lda(), qda()
library(ISLR)  ## for the Default dataset
library(caret) ## for confusionMatrix()

lda1 = lda(default ~ balance + income, data=Default)
names(lda1)
## by default the prior is the class distribution
lda1$prior; table(Default$default)
## compare observed with predicted
table(Default$default, predict(lda1)$class)
```

An example of QDA analysis, which yields 5 more correct classifications than the LDA:

```
qda1 = qda(default ~ balance + income, data=Default)
qda1$prior; table(Default$default)
table(Default$default, predict(qda1)$class)
```

Example: QDA boundary when there is a single predictor:

```
mu1=0; s1=2; pi1=0.5  ## true distribution for class 1 (red)
mu2=3; s2=1; pi2=0.5  ## true distribution for class 2 (green)
curve(pi1*dnorm(x, mu1, s1) +pi2*dnorm(x, mu2, s2), -4, 7, ylim=c(0,0.3))
curve(pi1*dnorm(x, mu1, s1), col=2, add=T)
curve(pi2*dnorm(x, mu2, s2), col=3, add=T)
```

```
## There are 2 threshold values for the QDA !?
pred1 = function(x) ((x-mu1)/s1)^2-2*log(pi1)+log(s1^2) <((x-mu2)/s2)^2-2*log(pi2)+log(s2^2)
curve(0.3*pred1(x), n=1001, lty=2, add=T)

## Enlarge the far right portion to see why.  Red crosses above green there!
curve(pi1*dnorm(x, mu1, s1) +pi2*dnorm(x, mu2, s2), 6, 8)
curve(pi1*dnorm(x, mu1, s1), col=2, add=T)
curve(pi2*dnorm(x, mu2, s2), col=3, add=T)
```

Example: QDA boundary with two predictors:

```
mu1=c(0,0); s1=1.2; pi1=0.5  ## true distribution for class 1
mu2=c(3,2); s2=1; pi2=0.5    ## true distribution for class 2
grid=seq(-3, 6, .01)
x1grid = rep(grid, length(grid))
x2grid = rep(grid, each=length(grid))
d1 = pi1 * dnorm(x1grid, mu1[1], s1) * dnorm(x2grid, mu1[2], s1)
d2 = pi2 * dnorm(x1grid, mu2[1], s2) * dnorm(x2grid, mu2[2], s2)
library(rgl)
plot3d(x1grid, x2grid, d1, col='lightgrey')  ## class 1 distribution
plot3d(x1grid, x2grid, d2, col='grey', add=T) ## class 2 distribution
```

4.4.2 Confusion matrix and related measures

Tables 4.6, 4.7.

In classification problems with 2 classes (positive/negative), we can tally the observed (true) and predicted classes into a 2×2 table, also called a “confusion matrix”.

(Also see https://en.wikipedia.org/wiki/Sensitivity_and_specificity, where the confusion matrix is shown with rows for predicted classes and columns for true classes.)

		Predicted		
		Positive	Negative	Total
True	Positive	TP	FN	trueP
	Negative	FP	TN	trueN
	Total	predP	predN	n

Let P and N denote the true class, and P^* and N^* the predicted class. We can define some simple measures of performance as conditional probabilities:

- $\Pr(P^*|P)$ = TP/trueP : TPR, sensitivity, power, recall, 1−FNR, 1−Type II error rate
- $\Pr(N^*|N)$ = TN/trueN : TNR, specificity, 1−FPR, 1−Type I error rate
- $\Pr(P|P^*)$ = TP/predP : PPV, precision, 1−FDR
- $\Pr(N|N^*)$ = TN/predN : NPV, 1−FOR

Some other simple measures of performance are functions of one of these four measures. For example, false positive rate is FPR=1−TNR; false discovery rate is FDR=1−PPV.

Some of these measures may not be meaningfully estimated if the data are not a random sample. For example, in a case-control study, trueP and trueN are predetermined and trueP/n is often much higher than prevalence $\pi = \Pr(P)$. In such a study, PPV and NPV cannot be estimated as above. (They can be estimated if prevalence π is known: $PPV = \frac{\pi TRP}{\pi TPR + (1-\pi)FPR} = \frac{r}{r+1}$, where $r = \frac{\pi}{1-\pi}\frac{TPR}{FPR}$; $NPV = \frac{(1-\pi)TNR}{\pi FNR + (1-\pi)TNR} = \frac{s}{s+1}$, where $s = \frac{1-\pi}{\pi}\frac{TNR}{FNR}$.)

Ideally we would like all these four probabilities to be high. Unfortunately, **a high value in one measure does not guarantee a high value in any of the other three**. We use sensitivity as an example.

(1) Sensitivity and specificity often move on opposite directions; that is, a high $\Pr(P^*|P)$ may correspond to a low $\Pr(N^*|N)$. This can be shown in an ROC curve.
(2) Sensitivity (recall) and PPV (precision) often move on opposite directions; that is, a high $\Pr(P^*|P)$ may correspond to a low $\Pr(P|P^*)$. This is called precision–recall tradeoff.
(3) A high $\Pr(P^*|P)$ does not imply a high $\Pr(N|N^*)$. This is surprising because in pure logic, if A implies B, then not B must imply not A. Unfortunately, in probabilistic reasoning, there is no guarantee that if $\Pr(B|A)$ is high then $\Pr(notA|notB)$ is high. Thus, it is wrong to claim that "a highly sensitive test is deemed effective at ruling out a disease when negative".
(4) Similarly, a high $\Pr(N^*|N)$ does not guarantee a high $\Pr(P|P^*)$, and thus it is wrong to claim that "a highly specific test is effective at ruling in a disease when positive".

Prevalence, $\pi = \Pr(P)$, has a large impact on PPV/NPV. A high prevalence often leads to a high PPV and a low prevalence leads to a high NPV. For example, if the positive class has a low prevalence $\pi = 0.01$, even a test with a high sensitivity TPR=0.95 and a high specificity TNR=0.95 has a low PPV=0.16 ($r = \frac{\pi}{1-\pi}\frac{TPR}{FPR} = \frac{0.01}{0.99}\frac{0.95}{0.05} = 0.19$ and $PPV = \frac{r}{r+1} = 0.16$); if the positive class has a high prevalence $\pi = 0.9$, a test with TPR=0.95 and TNR=0.95 has NPV=0.68 ($s = \frac{1-\pi}{\pi}\frac{TNR}{FNR} = \frac{0.1}{0.9}\frac{0.95}{0.05} = 2.11$, $NPV = \frac{s}{s+1} = 0.68$).

To balance some of these measures, composite measures of **classification performance** are defined:

- F-score: $F_1 = 2pr/(p+r)$, the harmonic average of precision and recall. A harmonic average is close to the smaller value. Its extension, $F_\beta = (1+\beta^2)pr/(\beta^2 p + r)$, is the harmonic average of one portion of precision and β^2 portions of recall.
- G-score (Fowlkes–Mallows index): $G = \sqrt{pr}$, the geometric average of precision and recall. When $p \neq r$, $F_1 < G$.
- Informedness (Youden's J): $J = sens + spec - 1$, which is $sens - (1 - spec) = y - x$ in the ROC curve. So a higher J corresponds to a larger distance from the diagonal line $y = x$ in the ROC curve. It is shown above that a classifier with a high sensitivity and a high specificity may still have a low PPV (e.g., when prevalence π is low).
- Markedness: $PPV + NPV - 1$. Similar to J, when prediction rate $\Pr(P^*)$ is low, a classifier with a high PPV and a high NPV may have a low sensitivity.

The confusion matrix requires a designation of positive and negative classes. In some applications, there may not be a natural choice of positive and negative classes. For example, suppose we use demographic and socioeconomic variables to predict the party, Republican or Democrat, a person will vote for in 2020. Suppose we designate voting Republican as being positive (scenario 1), and calculate various measures of performance as above. If we switch the designation and label voting Democrat as positive (scenario 2), sensitivity in scenario 1 becomes specificity in scenario 2 and specificity in scenario 1 becomes sensitivity in scenario 2. Similarly PPV and NPV are also swapped. In this situation, measures F_1 and G may not make sense because they are designed to focus on the performance over the positive class.

There are also measures of **agreement** (e.g., inter-rater reliability) between the observed and predicted classes. These measures are defined directly on the 2×2 table, not through the conditional probabilities. They reflect how concentrated the data are on the diagonal:

- Phi coefficient = Matthews correlation coefficient = $\sqrt{\chi^2/n}$ = Pearson's CC.
- Kappa, or Cohen's kappa, $\kappa = (c_o - c_e)/(n - c_e)$, where c_o = TP+TN and c_e is the expected count on the diagonal under independence between the observed and the predicted. A variation of Cohen's kappa is Scott's pi, for which c_e is computed under the assumption that the two marginal distributions are the same.
- Accuracy (TP+TN)$/n$ = 1−misclassification rate. It is the fraction of agreement. Under severe class imbalance, a high accuracy can be easily achieved with a classifier that assigns everything to the majority class. Accuracy $= \frac{c_o}{n} \geq \kappa$, where equality holds only when TP+TN $= n$.

The `confusionMatrix()` function in the R `caret` package displays columns as truth and rows as predicted values, same as the layout in the above wikipedia page. The `sklearn.metrics.confusion_matrix()` in Python is the same as `table()` in R; their display depends on the order of the two input variables.

```
confusionMatrix(predict(lda1)$class, Default$default)  ## wrong!
confusionMatrix(predict(lda1)$class, Default$default, positive="Yes")
confusionMatrix(predict(lda1)$class, Default$default, positive="Yes", mode="everything")

confusionMatrix(predict(qda1)$class, Default$default, positive="Yes", mode="everything")
```

4.4.3 ROC Curve and its AUC

ROC curves (and ROC AUCs) are useful for methods that generate a quantitative measure (e.g., a score or a probability) for each observation.

An ROC curve can be drawn for two variables: a is binary and b is quantitative. For example, an ROC curve is often drawn for logistic regression, with a being the observed outcome and b the estimated probability. It reflects the performance of all possible classification models using the quantitative variable b to make predictions.

To draw an ROC curve, we first sort b, and then for every classification model resulting from dichotomizing b ($b \le t$ and $b > t$ for any given t), we calculate the correponding sensitivity and specificity and plot sensitivity against $1-$ specificity (equivalently, power against type I error rate, or TPR against FPR). For n observations, there are at most $n+1$ points on the ROC curve: $n-1$ dichotomizations and 2 extreme classifiers, all predicted as N and all predicted as P, which correspond to $(0,0)$ and $(1,1)$ on the ROC curve.

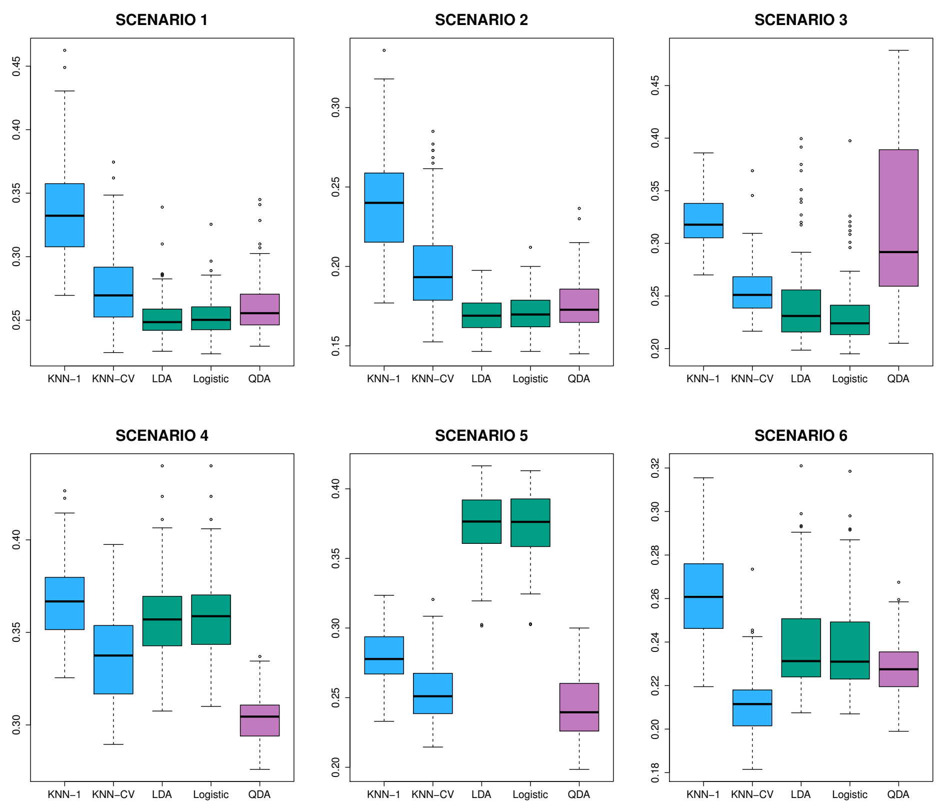
The area-under-curve (**AUC**) of an ROC curve is a summary measure of the relationship between a and b. It can be viewed as a measure of concondance between a and b, because it is the same as the C-index (concordance index), which is defined as the fraction of randomly selected P–N pairs for which the direction of their b values is consistent with that of their a values (a tie is counted as a half). The ROC AUC is also related to Wilcoxon–Mann–Whitney test.

When b is binary, the ROC curve only has three points, $(a_0, b_0) = (\text{FP/trueN}, \text{TP/trueP})$, $(0,0)$ and $(1,1)$; its AUC is $\frac{1}{2}a_0b_0 + \frac{1}{2}(b_0+1)(1-a_0) = \frac{1}{2}(b_0 - a_0 + 1)$, which is half of Youden's J. Note that $b_0 - a_0$ is the vertical distance of (a_0, b_0) from the diagonal line $y = x$.

4.5 ISLR 4.5 Comparisons of LDA/QDA/logistic/KNN

For the KNN methods, two versions were evaluated: 1-NN, k-NN with k chosen from cross-validation. Consider a two-class (i.e., binary) outcome with centers $\mu_1 \ne \mu_2$, and 2 predictors, (x_1, x_2). Consider 6 scenarios. Each scenario was evaluated with 100 replications.

1. $\binom{x_1}{x_2} \sim N\left(\mu_k, \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)\right)$; 20 obs in each class. LDA is the best, followed closedly by logistic regression.
2. $\binom{x_1}{x_2} \sim N\left(\mu_k, \left(\begin{smallmatrix}1 & -0.5\\ -0.5 & 1\end{smallmatrix}\right)\right)$; 20 obs in each class. LDA is the best, followed closedly by logistic regression.
3. $x_1 \sim t_d$, $x_2 \sim t_d$, and they are independent; 50 obs in each class. The LDA assumptions are violated. Logistic regression is the best, followed closedly by LDA. The QDA results deteriorated considerably as a consequence of non-normality.
4. For class 1, $\binom{x_1}{x_2} \sim N\left(\mu_1, \left(\begin{smallmatrix}1 & 0.5\\ 0.5 & 1\end{smallmatrix}\right)\right)$; for class 2, $\binom{x_1}{x_2} \sim N\left(\mu_2, \left(\begin{smallmatrix}1 & -0.5\\ -0.5 & 1\end{smallmatrix}\right)\right)$. QDA is the best, followed by kNN-CV.
5. $\binom{x_1}{x_2} \sim N\left(\mu, \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)\right)$. Then $p(y=1|x_1,x_2) = \frac{e^{\beta_1 x_1^2 + \beta_2 x_2^2 + \beta_3 x_1 x_2}}{1+e^{\beta_1 x_1^2 + \beta_2 x_2^2 + \beta_3 x_1 x_2}}$. QDA is the best, followed closedly by kNN-CV. The linear methods had very poor performance.
6. Same as 5, but with the responses sampled from a more complicated nonlinear function. kNN-CV wins, but 1-NN gave the worst results. This shows that "even when the data exhibits a complex nonlinear relationship, a nonparametric method such as KNN can still give poor results if the level of smoothness is not chosen correctly."



4.6 The MNIST dataset

The MNIST dataset has been extensively used as a test bed for machine learning methods. Because the same test set has been used, new methods claiming better performance is probably in the territory of overfitting. Nonetheless, it is still a very good starting point for trying machine learning methods.

The MNIST dataset contains 70,000 handwritten digits in 28×28 resolution in greyscale. See http://yann.lecun.com/exdb/mnist/ for its history and the performance of various methods. There is some imbalance in the outcome categories. The digit "1" is the most frequent, 7877; "7" is the 2nd most frequent, 7293. The digit "5" is least frequent, 6313; "4" and "8" are 2nd/3rd least frequent, 6824/6825.

The data have been packaged in a few ways, either as a single dataset, as training/test sets, or as training/validation/test. The files are often python friendly. HOML Chapter 3 uses a Matlab version from mldata.org: http://mldata.org/repository/data/download/matlab/mnist-original/. Scikit-Learn provides a function `fetch_mldata` to download and save it in `$HOME/scikit_learn_data/mldata/`. This "MNIST original" version, mnist-original.mat, comes as a single dataset with no pre-splitting.

Its `data` part is a 2-D array with dimensions 784×70000. Note that $784 = 28 \times 28$. The 784 pixels are ordered so that the first 28 pixels are the first row (from left to right), the next 28 pixels are the second row, etc. This is called *row-major order*, which is used by Python's numpy (also by C). R uses *column-major order* (same as Fortran, Matlab, Julia).

The pixel intensity is an integer between 0 and 255. `reshape(28,28)` makes it ready to view with `imshow`. Scikit-Learn's `fetch_mldata` by default transposes arrays and its output is 70000×784. Scipy's `scipy.io.loadmat` won't transpose data.

```python
import numpy as np
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
mnist   ## like a python dictionary (a Perl hash table, or an R list)
X, y = mnist["data"], mnist["target"]  ## X and y are numpy arrays
```

```
print(X.shape)  ## (70000, 784)
print(y.shape)  ## (70000,)
print(np.unique(y, return_counts=True))  ## tally the counts

some_digit_image = X[36000].reshape(28, 28)  ## reshape(nrow, ncol)
import matplotlib
import matplotlib.pyplot as plt
plt.imshow(some_digit_image, cmap = matplotlib.cm.binary, interpolation="none")
plt.axis("off")
```

To load the data into R, one can use `R.matlab`, which does not require python installed in your computer, or `reticulate`, which runs python behind R. My code using `R.matlab`:

```
library(R.matlab)
aa = readMat("/home/lic3/scikit_learn_data/mldata/mnist-original.mat")
str(aa)
names(aa); attributes(aa)
X = aa$data
y = aa$label
class(X); dim(X)  ## (784, 70000)
class(y); dim(y)  ## (1, 70000)
```

My code using `reticulate`:

```
library(reticulate)
#use_condaenv("r-tensorflow")  ## if you have installed keras in a conda environment
scipy <- import("scipy")
aa = scipy$io$loadmat('/home/lic3/scikit_learn_data/mldata/mnist-original.mat')
str(aa)

X = aa$data
y = aa$label
class(X); dim(X)  ## 784, 70000
class(y); dim(y)  ## 1, 70000
```

Another version, http://deeplearning.net/data/mnist/mnist.pkl.gz, contains three subsets for training/validation/test. The book NNDL (by Nielsen) uses this version in its demonstration. In this version, the pixel intensity values are rescaled to be between 0 and 1. To load it into Python:

```
import pickle, gzip
with gzip.open('mnist.pkl.gz', 'rb') as f:
    (x_train, y_train), (x_valid, y_valid), (x_test, y_test) = pickle.load(f, encoding="latin1"
```

To load it into R, we can use `reticulate`:

```
library(reticulate)
library(zeallot)
my_py_load_object <- function(filename) {
  pickle <- import("pickle")
  gzip <- import("gzip")
  handle <- gzip$open(filename, "rb")
  on.exit(handle$close(), add = TRUE)
  pickle$load(handle, encoding="latin1")  ## to read cPickle format
}

mypklfile = '/home/lic3/Work/Teaching/Datasets/MNIST/mnist.pkl.gz'
aa = my_py_load_object(mypklfile)
str(aa)
c(c(x_train2, y_train2), c(x_valid2, y_valid2), c(x_test2, y_test2)) %<-% aa
```

There is a third version, https://s3.amazonaws.com/img-datasets/mnist.npz. The R `keras` package provides `dataset_mnist()` to download and save it to $HOME/.keras/datasets/mnist.npz. This version contains two subsets: `train` and `test`. Their `x` parts are 3-D arrays with dimensions $60000 \times 28 \times 28$ and $10000 \times 28 \times 28$.

4.7 Assignment

1. **Homework**: ISLR Chapter 4 Exercises 13 (`Boston` dataset in MASS)
2. HOML Chapter 3 presents various classification methods using the MNIST dataset as an example. Try out the Python code from that book.