Lab02 - Questions and exercises

Statistical learning main topics:

1. Main families of approaches (inference/interpretation vs prediction, parametric vs non-parametric, regression vs classification vs clustering)
2. Reducible/irreducible error (in regression and classification)
3. Bias/variance tradeoff
4. Error curves (bias, variance, training/test/irreducible error), see Exercise on the book
5. Bayes classifier and Bayes error rate

List of useful material for this class:
https://github.com/asadoughi/stat-learning (solutions to the exercises in the book)

Bias/variance tradeoff
https://theclevermachine.wordpress.com/2013/04/21/model-selection-underfitting-overfitting-and-the-bias-variance-tradeoff/
http://scott.fortmann-roe.com/docs/BiasVariance.html

Homework exercise from last year:
http://davide.eynard.it/2015/01/05/statistical-learning-with-r-part-1-overfitting/

---

(1) Main families of approaches

See also Question 2, Section 2.4, page 52 on the course book.

Explain whether each scenario is **classification/regression/cluster analysis,** and indicate whether we are most interested in **prediction or inference.** Finally, provide *n* and *p.*

NOTE: due to the ambigous interpretation of the word "inference" I preferred to use "interpretation" during the lab, saying that "inference" could be used to describe some kind of prediction too (given e.g. a classification model, classes could be "inferred" from the data). The book, however, uses inference referring to interpretation. To avoid misunderstandings, feel free to use the word inference with either acceptation, but explicitly say what you mean.

- we collect movie data from Netflix. From a set of 2000 movies we got genre, year, director, budget, user rating (1 to 5 stars), and we would like to use this information to predict how many stars a new movie will get

- we are analyzing an email mailbox with 10000 messages and would like to understand whether a new incoming message will be spam or not. For each message we have the sender, the subject, the date, and the body of the email, plus a class assignment (SPAM/NOT SPAM)

- we would like to know more about the types of users who visit a given news website. From a set of 200000 users, for each user we get age, gender, and country, plus the time she spent on each of the 10 different sections (e.g. sports, politics, entertainment, local, ...) of the website.

This is a variation of question 4, Section 2.4, page 53 on the course book.

Think about some real-life applications of statistical learning (pick up the exercise at page 53 and discuss together, plus comment the answers provided at https://github.com/asadoughi/stat-learning/blob/master/ch2/answers).

---

(2) Reducible/irreducible error

Start from the following main definitions and formulas:

- (general) relationship between X, our *predictors*, and Y, our *responses*:

$$Y = f(X) + \epsilon.$$

- expected error:

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible}} \end{aligned}$$

- try to answer the following question (feel free to use R to actually test it):

if we generate our predictors X as

X = rnorm(50)

and our responses Y as

Y = 2*x + rnorm(50, 0, .1)

what is the irreducible error? Note that Y above here describes an "ideal" function *plus* some random noise, exactly as in Y = f(X) + e. Suppose the function we fit (Y_hat in the formulas above) perfectly matches f(X) and describe what happens.

---

(3) Bias/Variance tradeoff

Let us start with the definition of Mean Squared Error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

If we decompose the **expected** test MSE in Variance, Bias, and irreducible error we obtain the following:

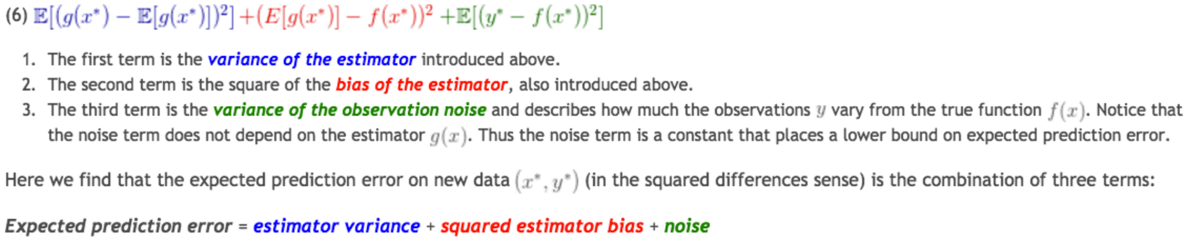
$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon)$$

(What does **expected** mean? Why do we have a mean around some *_0 terms? What are we iterating on?)

Suppose your dataset is large enough so that you can have a **single** test set and **many** different training sets. Now, you try to fit a function from the training data **many times**, each one from a different training set. This means that, for every single x0 in your test set, you will have **many different f_hat(x0)**, and you want to verify how *precise* (i.e. how close to the real function) and how *consistent* your fit is by using the above formula.

To better understand the meaning of this decomposition, check the following material:
https://theclevermachine.wordpress.com/2013/04/21/model-selection-underfitting-overfitting-and-the-bias-variance-tradeoff/

The same function is shown as:



(6) $\mathbb{E}[(g(x^*) - \mathbb{E}[g(x^*)])^2] + (E[g(x^*)] - f(x^*))^2 + \mathbb{E}[(y^* - f(x^*))^2]$

1. The first term is the ***variance of the estimator*** introduced above.
2. The second term is the square of the ***bias of the estimator***, also introduced above.
3. The third term is the ***variance of the observation noise*** and describes how much the observations y vary from the true function $f(x)$. Notice that the noise term does not depend on the estimator $g(x)$. Thus the noise term is a constant that places a lower bound on expected prediction error.

Here we find that the expected prediction error on new data (x^*, y^*) (in the squared differences sense) is the combination of three terms:

Expected prediction error = ***estimator variance*** + ***squared estimator bias*** + ***noise***

To be consistent with our notation, this would become:

$$E\left(\hat{f}(x_0) - E(\hat{f}(x_0))\right)^2 + \left(E(\hat{f}(x_0)) - f(x_0)\right)^2 + E\left(y_0 - f(x_0)\right)^2$$

You can now easily see that the first term is the **variance** of the estimator (i.e. how **consistent** your fit is: the closer your estimators are to their own mean, the smaller the variance is). The second term is the **bias** of the estimator (i.e. how close your fit is to the real function: if the mean of your estimates is close to f(x0), then the bias is small). The third term is the **irreducible error**.

Also check:
http://scott.fortmann-roe.com/docs/BiasVariance.html

---

(4) Error curves (bias, variance, training/test/irreducible error)

Discuss the concepts of flexibility, bias/variance tradeoff, and draw together the 5 curves (as in Exercise 3a) in the flexibility/MSE plane.

For a practical example, check out last year's homework about overfitting:
http://davide.eynard.it/2015/01/05/statistical-learning-with-r-part-1-overfitting/

You can play with the code and see what happens when:

- you increase/decrease the training/test set size
- you change the amount of noise in your data (play with the variance parameter --the third

in norm-- when you define y = f(x) + rnorm(...)

- you change the flexibility of the function you want to fit

---

(5) Bayes classifier and Bayes error rate

How can the previous observations be applied to the *classification* context instead of *regression*?

def. (training) ERROR RATE:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

def (test) ERROR RATE:

$$\text{Ave}\left(I(y_0 \neq \hat{y}_0)\right)$$

It can be shown that the above test error rate can be minimized (on average) by a classifier that assigns each observation *to the most likely class, given its predictor values*. In other words, a classifier that assigns a test observation with predictor vector x0 to the class j for which the following is largest:

$$\Pr(Y = j | X = x_0)$$

Suppose that we only have two classes (e.g. 1 and 2, or 0 and 1): the classifier predicts j = 1 if Pr(Y=1|X=x0)>0.5, and the other class otherwise. This classifier is called the ***Bayes classifier*** and it produces the ***lowest possible test error rate***, called the ***Bayes error rate***: **1 - max_j Pr(Y=j|X=x0)**

The overall (on a set of X observations) Bayes error rate is defined as

$$1 - E\left(\max_j \Pr(Y = j | X)\right)$$

(averaged on the different values taken by X). See the example on the book, page 38:

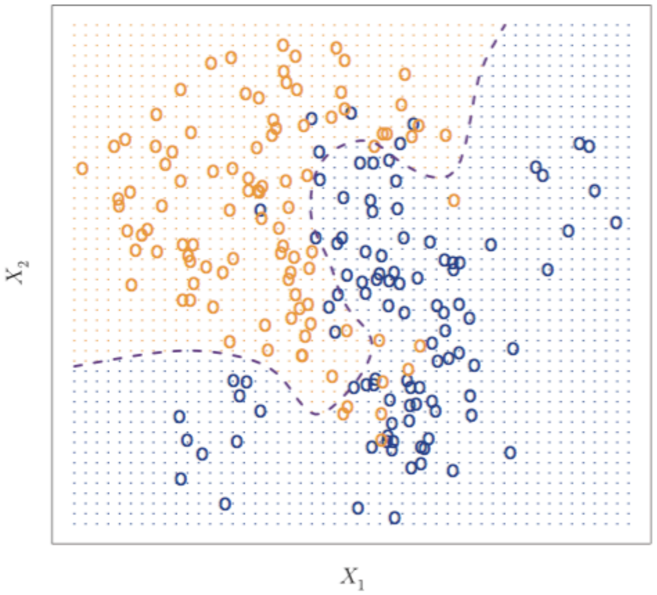


FIGURE 2.13. *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

Typically we don't know the highest Pr(Y=j|X=x0), thus we need to find a way to *estimate* it. KNN (K-nearest neighbors) is a simple but pretty good classifier, estimating the conditional probability for class j as the fraction of points in the neighborhood (of size K) whose response values equal j:

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

KNN Exercises: first, solve Exercise 7 on the book (page 53). Then, try the following ones on R.

```
# install library "animation"
install.packages("animation")
library("animation")

# generate a "training" dataset (12 points, two coordinates, tentatively two concentric circles drawn by hand)
train = matrix (c(2,2,2.5,2.7,3,1,1,2,2,3,4,4,2,3,3.5,2,3,2,3,4,1,1,1.5,4),12,2)

# test, using knn.ani, whether point (2.5,2.5) belongs to class "green" or "red"
x = c(2.5,2.5)
knn.ani(train, x, c(rep("Red",5),rep("Green",7)), k = 3)

# verify by checking the distances
```

```
a = colSums((t(train)-x)^2)
min(a)
which.min(a)
a == min(a)
i = sort(a, index.return=TRUE)

## a more complex example, with randomly generated data
# generate 400 random points and divide them into two columns (to be used as random x,y coords)
train = matrix(rnorm(400,10,10),200,2)

# generate an index identifying points within a circle
idx = sqrt((train[,1]-10)^2+(train[,2]-10)^2)<5

# divide points into two classes (Red and Green)
classes = rep('Red',200)
classes[idx]='Green'

# generate 20 random points to be tested with KNN
test = matrix(rnorm(20,10,5),10,2)

# run the KNN animation using k = 3. TRY OTHER VALUES FOR K AND SEE WHAT HAPPENS!
knn.ani(train, test, classes, k = 3)
```