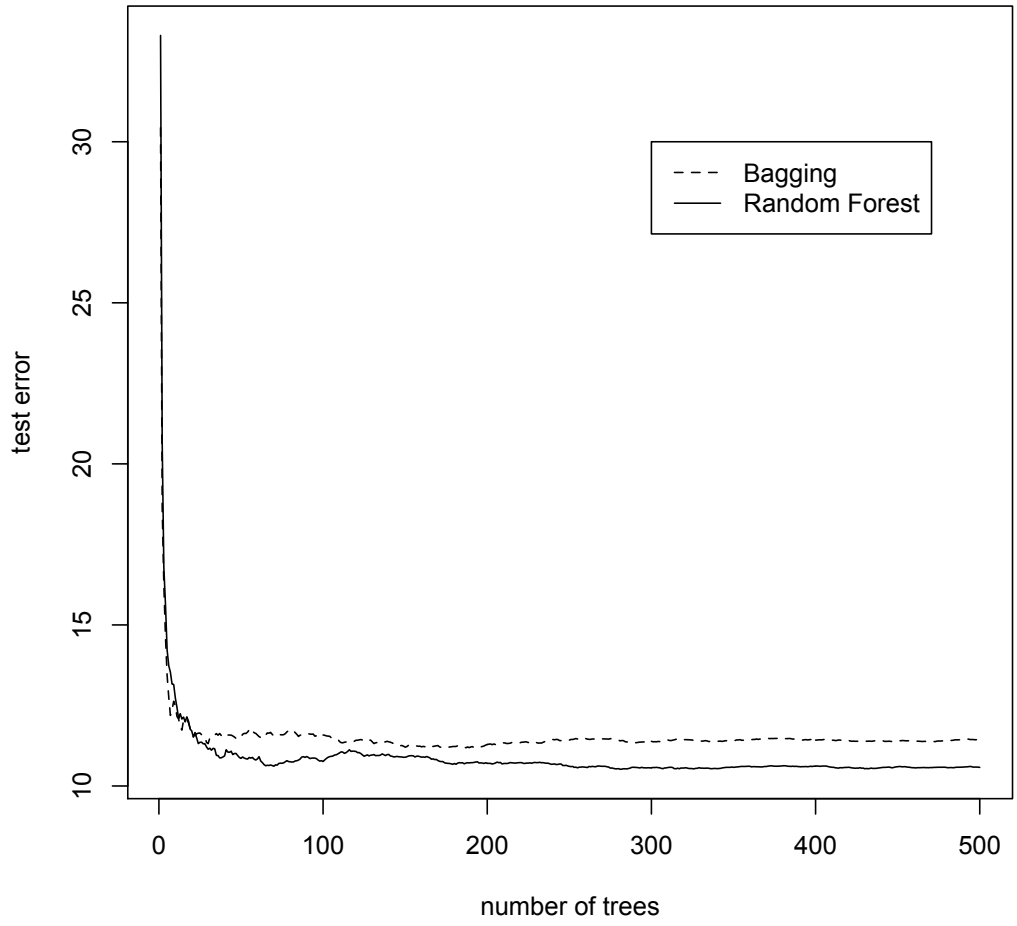
Statistics 216
Homework 4 Solutions, 55 total points

1. *10 points: (a) 4 points (b) 2 points (c) 2 points (d) 2 points*

Grading notes for part (a): Award full credit for any figure that looks somewhat like the figure below. Student answers will vary due to randomness in test set selection. Award half credit for a figure that does not look anything like the one below.

(a)



Grading notes for part (b): Again, student answers will vary, due to randomness in test set selection and also due to choice of importance metric. One point is for identifying the important variables (as long as at least two of the variables mentioned in the solution below are mentioned in the student's answer, and all variables mentioned are some type of girth). The other point is for observing that bagging and the random forest identify similar variables as important.

(b) The four most important variables, as identified by the random forest, are waist girth, chest girth, bicep girth and forearm girth. The four most important variables, as identified by bagging, are waist girth, bicep girth, chest girth and forearm girth. So they are the same as bagging.

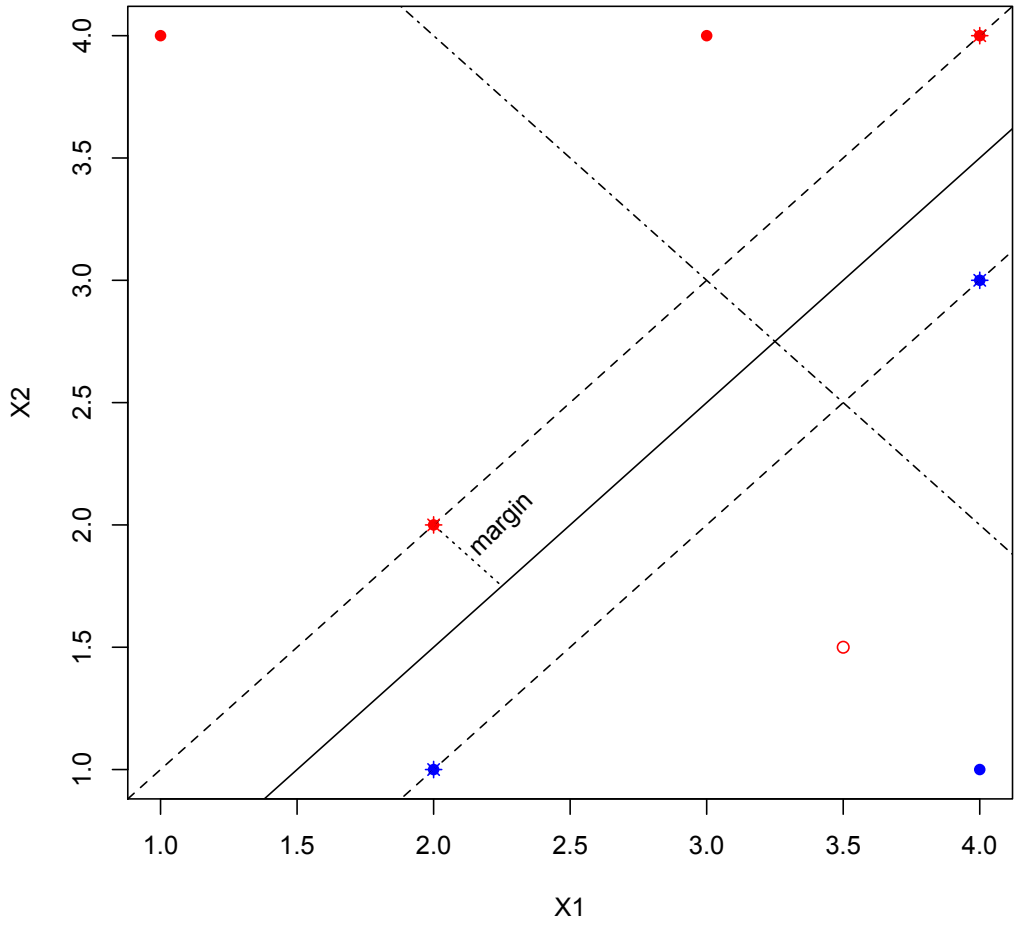
Grading notes for part (c): Any test MSE between 7 and 13 could be valid for the random forest. Award one point for reporting a valid test MSE and one point for making a comparison against the test MSEs from Homework 3.

(c) The test MSE of the random forest is 10.58. This is larger than the test MSE for partial least squares regression (8.65), principal components regression (9.27) or forward stepwise regression (8.63) from the solutions to Homework 3.

(d) Yes, in this case 500 is enough trees because the test error has leveled off, as we observe in the figure in part (a). It seems as though 300 trees would have been sufficient in this case.

2. *10 points: (a) 1 point (b) 2 points (c) 1 point (d) 2 points (e) 1 point (f) 1 point (g) 1 point (h) 1 point*

(a) The seven observations are plotted below, along with everything that is asked to be plotted later on in the problem. The seven filled-in points are the observations.



Grading notes for part (b): One point for drawing the hyperplane, one point for giving an equation for the hyperplane. Any equation of the form $0.5c - cX_1 + cX_2 = 0$ *for some* c *is valid.*

(b) The optimal separating hyperplane is sketched as the solid black line in the figure in part (a). The equation for this hyperplane is $0.5 - X_1 + X_2 = 0$.

Grading notes for part (c): Again, any inequality of the form $0.5c - cX_1 + cX_2 > 0$ is valid.

(c) Classify to Red if $0.5 - X_1 + X_2 > 0$, and classify to Blue otherwise.

Grading notes for part (d): One point for drawing the margin on the figure, one point for giving the correct width.

(d) The margin is indicated on the figure in part (a) by dotted lines running parallel to the optimal separating hyperplane. The width of the margin is $\sqrt{2}/4 = 0.354$.

(e) The support vectors are indicated by stars in the figure in part (a).

(f) The seventh point is the one in the bottom right-hand corner of the figure. Because it is not a support vector, a slight movement of this observation would not affect the maximal margin hyperplane. As long as this observation is not close to the margin, it will have no effect.

(g) The hyperplane represented by the dash-dot line (with "negative slope") is not the optimal separating hyperplane. The equation for this hyperplane is $-6 + X_1 + X_2 = 0$.

(h) Consider the additional observation represented by the empty red circle on the plot.
With the addition of this data point, the two classes are no longer separable by a hyperplane.

3. *22 points: (a) 0 pts. (b) 1 pt. (c) 2 pts. (d) 2 pts. (e) 2 pts. (f) 7 pts. (g) 7 pts. (h) 1 pt.*

Grading notes: This problem should actually be very easy to grade because we will grade for completion. Student answers will vary significantly based on test set selection and other issues. Award full credit for any answer that is not obviously wrong. Award half credit (rounded down to the nearest integer number of points) for any part that is answered horribly wrongly.

(a) *(No answer expected here; nothing to grade)*

(b) The number of support vectors is about 600 (or about 75% of training data), with about half coming from each class. This makes sense because of the low cost (0.01) for violating the margin.

(c) The training error rate is 23.8%. The test error rate is 28.1%.

(d) The optimal cost is 2.56.

(e) For optimal cost, the training error rate is 16.4%. The test error rate is 17.4%.

(f) (b) The number of support vectors is about 600 (or about 75% of training data), with about half coming from each class.
(c) The training error rate is 37.6%. The test error rate is 43.0%.
(d) The optimal cost is 0.16.
(e) For optimal cost, the training error rate is 30.5%. The test error rate is 36.7%.

(g) (b) The number of support vectors is about 350 (or about 40-45% of training data), with about half coming from each class.
(c) The training error rate is 16.3%. The test error rate is 16.3%.
(d) The optimal cost is 10.24. The optimal degree is 2.
(e) For optimal cost and degree, the training error rate is 16.9%. The test error rate is 15.2%.

(h) Overall, the best results seem to come from a polynomial kernel with cost 10.24 and degree 2.

4. *13 points: (a) 4 points (b) 4 points (c) 1 point (d) 4 points*

(a) *Grading notes for part (a): There are many ways to show this identity. Award full credit for any way that is correct. Award half credit if the identity is not correctly proved.*

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj}+\bar{x}_{kj}-x_{i'j})^2$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(\bar{x}_{kj}-x_{i'j})^2+\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(\bar{x}_{kj}-x_{i'j})$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{i'j}-\bar{x}_{kj})^2+\frac{2}{|C_k|}\sum_{i\in C_k}\sum_{i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(\bar{x}_{kj}-x_{i'j})$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{2}{|C_k|}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})\sum_{i'\in C_k}(\bar{x}_{kj}-x_{i'j})$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{2}{|C_k|}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(0)$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2+0$$

$$=\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

$$=\frac{2}{|C_k|}\sum_{i\in C_k}\sum_{i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

$$=\frac{2}{|C_k|}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2\sum_{i'\in C_k}(1)$$

$$=\frac{2}{|C_k|}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2|C_k|$$

$$=2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

Grading notes for part (b): There are two steps (i. and ii.) that must be considered. Each is worth 2 points. Award full credit for any valid argument. Award half credit for any invalid argument.

(b) The important thing to observe here is that because of the result of above, instead of considering whether $W(C_k)$ increases or decreases, we can just consider whether the expression on the right-hand side of the equation above (hereafter referred to as "the objective") increases or decreases. Each iteration of the K-means algorithm does two things:

i. Replace each centroid with the mean of the data points assigned to the cluster. This is guaranteed not to increase the objective because for each $j = 1, ..., p$, and each $k = 1, ..., K$, the mean $\bar{x}_{kj}$ is the minimizer of $\sum_{i\in C_k}(x_{ij} - \bar{x}_{kj})^2$.

ii. Reassign each data point to the nearest centroid. This is guaranteed not to increase the objective because by design each data point is assigned to the nearest centroid, so the distance between the point and its centroid cannot increase.

(c) The conclusion of part (b) implies that the sequence $W(C_k)$ must converge because any non-increasing sequence which is bounded below must converge.

Grading notes for part (d): 1 point for presenting a data set. 1 point for presenting initial centroids leading to non optimal convergence. 1 point for presenting the value $W(C_k)$ in the non optimal convergence. 1 point for presenting the optimal value of $W(C_k)$.

(d) Consider the following 6 data points in $\mathbb{R}^2$:

$$(0,0), (0,1), (1,0), (1,1), (3,0), \text{ and } (3,1)$$

If the centers are initialized to be $c_1 = (1,0)$ and $c_2 = (1,1)$, then $(0,0), (1,0)$, and $(3,0)$ will be assigned to the cluster C_1, and $(0,1), (1,1)$, and $(3,1)$ will be assigned to C_2.

The center of C_1 becomes $c_1 = (4/3, 0)$, and the center of C_2 becomes $c_2 = (4/3, 1)$.
After this update, the data points are all assigned to the same clusters as in the previous step, so the algorithm halts, with cost

$$W(C_k) = 2\left(0 - \frac{4}{3}\right)^2 + 2\left(1 - \frac{4}{3}\right)^2 + 2\left(3 - \frac{4}{3}\right)^2 = \frac{28}{3}$$

However, this is not the cost of the optimal clustering. The optimal clustering is with centers $c_1 = (\frac{1}{2}, \frac{1}{2})$ and $c_2 = (3, \frac{1}{2})$ with cost

$$W(C_k) = 4\left(\frac{1}{2}^2 + \frac{1}{2}^2\right) + 2\left(\frac{1}{2}^2\right) = 4\left(\frac{1}{4} + \frac{1}{4}\right) + 2\left(\frac{1}{4}\right) = 4\left(\frac{1}{2}\right) + 2\left(\frac{1}{4}\right) = 2 + \frac{1}{2} = \frac{5}{2}$$

Appendix: Relevant R code

```
# 1
library(randomForest)
load('body.RData')
set.seed(1)
test = sample(1:nrow(X), 200)
train = setdiff(1:nrow(X), test)
#(a)
bagging = randomForest(x = X[train, ], y = Y[train, 'Weight'], xtest = X[test, ], ytest = Y[test, 'Weight'], mtry =
forest = randomForest(x = X[train, ], y = Y[train, 'Weight'], xtest = X[test, ], ytest = Y[test, 'Weight'])
plot(rep(1:500, 2), c(bagging$test$mse, forest$test$mse), type = 'n', xlab = 'number of trees', ylab = 'test error'
points(bagging$test$mse, type = 'l', lty = 2)
points(forest$test$mse, type = 'l')
legend(300, 30, c('Bagging', 'Random Forest'), lty = c(2, 1))
#(b)
forest$importance[order(-forest$importance), ]
bagging$importance[order(-bagging$importance), ]
#(c)
forest$test$mse[500]

# 2
#(a)
X1 = c(3, 2, 4, 1, 2, 4, 4)
X2 = c(4, 2, 4, 4, 1, 3, 1)
plot(X1, X2, col = rep(c('Red', 'Blue'), c(4, 3)), pch = 16)    # 2.pdf
#(b)
abline(-0.5, 1)
```

```
#(d)
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
points(c(2, 2.25), c(2, 1.75), type = 'l', lty = 3)
text(2.25, 2, 'margin', srt = 45)
#(e)
points(c(2, 4, 2, 4), c(2, 4, 1, 3), col = rep(c('Red', 'Blue'), c(2, 2)), pch = 8)
#(g)
abline(6, -1, lty = 4)
#(h)
points(3.5, 1.5, col = 'red')

# 3
library(ISLR)
library(e1071)
#(a)
set.seed(2196)
train = sample(1:nrow(OJ), 800)
test = setdiff(1:nrow(OJ), train)
#(b)
svmLinear = svm(Purchase ~ ., data = OJ, kernel = 'linear', cost = 0.01, scale = FALSE, subset = train)
summary(svmLinear)
#(c)
mean(predict(svmLinear, newdata = OJ[train, ]) != OJ[train, 'Purchase'])        # training error
mean(predict(svmLinear, newdata = OJ[test, ]) != OJ[test, 'Purchase'])          # test error
#(d)
tuneLinear = tune(svm, Purchase ~ ., data = OJ[train, ], kernel = 'linear', ranges = list(cost = 0.01*2^(1:10)))
summary(tuneLinear)
optimalCost = tuneLinear$best.parameters
#(e)
svmLinearOptimal = svm(Purchase ~ ., data = OJ, kernel = 'linear', cost = optimalCost, scale = FALSE, subset = trai
mean(predict(svmLinearOptimal, newdata = OJ[train, ]) != OJ[train, 'Purchase']) # training error
mean(predict(svmLinearOptimal, newdata = OJ[test, ]) != OJ[test, 'Purchase'])   # test error
#(f)
svmRadial = svm(Purchase ~ ., data = OJ, kernel = 'radial', cost = 0.01, scale = FALSE, subset = train)
summary(svmRadial)
mean(predict(svmRadial, newdata = OJ[train, ]) != OJ[train, 'Purchase'])        # training error
mean(predict(svmRadial, newdata = OJ[test, ]) != OJ[test, 'Purchase'])          # test error
tuneRadial = tune(svm, Purchase ~ ., data = OJ[train, ], kernel = 'radial', ranges = list(cost = 0.01*2^(1:10)))
summary(tuneRadial)
optimalCost = tuneRadial$best.parameters
svmRadialOptimal = svm(Purchase ~ ., data = OJ, kernel = 'radial', cost = optimalCost, scale = FALSE, subset = trai
mean(predict(svmRadialOptimal, newdata = OJ[train, ]) != OJ[train, 'Purchase']) # training error
mean(predict(svmRadialOptimal, newdata = OJ[test, ]) != OJ[test, 'Purchase'])   # test error
#(g)
svmPolynomial = svm(Purchase ~ ., data = OJ, kernel = 'polynomial', cost = 0.01, degree = 2, scale = FALSE, subset
summary(svmPolynomial)
mean(predict(svmPolynomial, newdata = OJ[train, ]) != OJ[train, 'Purchase'])    # training error
mean(predict(svmPolynomial, newdata = OJ[test, ]) != OJ[test, 'Purchase'])              # test error
tunePolynomial = tune(svm, Purchase ~ ., data = OJ[train, ], kernel = 'polynomial', ranges = list(cost = 0.01*2^(1:
summary(tunePolynomial)
optimalCost = tunePolynomial$best.parameters$cost
optimalDegree = tunePolynomial$best.parameters$degree
svmPolynomialOptimal = svm(Purchase ~ ., data = OJ, kernel = 'polynomial', cost = optimalCost, degree = optimalDegr
mean(predict(svmPolynomialOptimal, newdata = OJ[train, ]) != OJ[train, 'Purchase'])     # training error
mean(predict(svmPolynomialOptimal, newdata = OJ[test, ]) != OJ[test, 'Purchase'])       # test error
```