Homework 1

Casey Gibson

Problem 1

a) Unsupervised learning, clustering documents around a similarity metric (kNN).

b)

Supervised learning, even in the case of auto-regression we have labeled training pairs (X, Y), it just so happens X is previous values of Y.

c) Unsupervised learning, clustering

d) Supervised learning, linear regression or spline regression with labelled data.

Problem 2

$$MSE(\hat{\beta}) = E(\hat{\beta} - \beta)^2$$

We know that

$$Var(X) = E[X^2] - [E(X)]^2$$

so we can re-write the above as

$$MSE(\hat{\beta}) = Var(\hat{\beta} - \beta) + [E(\hat{\beta} - \beta)]^2$$

$$= Var(\hat{\beta}) + Var(\beta) + 2Cov(\hat{\beta}, \beta) + [E(\hat{\beta}) - E(\beta)]^2$$

Let's take each of these terms separately.

$$Var(\hat{\beta}) = Var(w\hat{\beta}_1 + (1-w)\hat{\beta}_2)$$

$$= w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2Cov(w\hat{\beta}_1, (1-w)\hat{\beta}_2)$$

$$= w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12}$$

$$Var(\beta) = 0$$

$$2Cov(\hat{\beta}, \beta) = 0$$

Let's next consider

$$[E(\hat{\beta}) - E(\beta)]^2$$

$$[w(b_1 + \beta) + (1-w)(b_2 + \beta) - \beta]^2$$

Putting this altogether we get

$$MSE(\hat{\beta}) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12} + [w(b_1+\beta) + (1-w)(b_2+\beta) - \beta]^2$$
$$MSE(\hat{\beta}) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12} + [wb_1 + w\beta + b_2 + \beta - w_b 2 - w\beta - \beta]^2$$

$$MSE(\hat{\beta}) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12} + [wb_1 + b_2 - wb_2]^2$$
$$MSE(\hat{\beta}) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12} + [w(b_1 - b_2-) + b_2]^2$$

$$MSE(\hat{\beta}) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12} + w^2(b_1 - b_2-)^2 + 2b_2w(b_1-b_2) + b_2^2$$
$$MSE(\hat{\beta}) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\sigma_{12} + w^2(b_1 - b_2)^2 + 2b_2w(b_1-b_2) + b_2^2$$

In order to minimize this function we can take the derivative with respect to w

$$\frac{\delta}{\delta w}MSE(\hat{\beta}) = 2w\sigma_1^2 - 2(1-w)\sigma_2^2 + 2(1-w)\sigma_{12} - 2w\sigma_{12} + 2w(b_1-b_2)^2 + 2b_2(b_1-b_2)$$
$$0 = 2w\sigma_1^2 - 2(1-w)\sigma_2^2 + 2(1-w)\sigma_{12} - 2w\sigma_{12} + 2w(b_1-b_2)^2 + 2b_2(b_1-b_2)$$
$$-2b_2(b_1-b_2) = 2w\sigma_1^2 - 2\sigma_2^2 + 2w\sigma_2^2 + 2\sigma_{12} - 2w\sigma_{12} - 2w\sigma_{12} + 2w(b_1-b_2)^2$$

$$2\sigma_2^2 - 2b_2(b_1-b_2) - 2\sigma_{12} = 2w\sigma_1^2 + 2w\sigma_2^2 - 2w\sigma_{12} - 2w\sigma_{12} + 2w(b_1-b_2)^2$$
$$\sigma_2^2 - b_2(b_1-b_2) - \sigma_{12} = w(\sigma_1^2 + \sigma_2^2 - \sigma_{12} - \sigma_{12} + (b_1-b_2)^2)$$

$$w = \frac{\sigma_2^2 - b_2(b_1-b_2) - \sigma_{12}}{\sigma_1^2 - \sigma_2^2 - \sigma_{12} + \sigma_{12} + (b_1-b_2)^2}$$

$$w = \frac{\sigma_2^2 - b_2(b_1-b_2) - \sigma_{12}}{\sigma_1^2 + \sigma_2^2 - 2\sigma_{12} + (b_1-b_2)^2} I_{0\leq w\leq 1}$$

In order to show that this is truly a minimum we need to find the sign of the second derivative.

$$\frac{\delta^2}{\delta^2 w}MSE(\hat{\beta}) = 2\sigma_1^2 + 2\sigma_2^2 - 2\sigma_{12} - 2\sigma_{12} + 2(b_1-b_2)^2$$

We know by the Cauchy-Schwarz inequality that that

$$\sigma_{12} < \sigma_1\sigma_2$$

$$2\sigma_{12} < 2\sigma_1\sigma_2$$
$$-2\sigma_{12} > -2\sigma_1\sigma_2$$

$$-2\sigma_{12} + \sigma_1^2 + \sigma_2^2 > -2\sigma_1\sigma_2 + \sigma_1^2 + \sigma_2^2$$
$$2\sigma_{12} + \sigma_1^2 + \sigma_2^2 > (\sigma_1 - \sigma_2)^2$$

So we see that the second derivative is ≥ 0 and we have indeed found a minimum. ### Problem 2

a)

This model corresponds to

$$Y \sim N(5 - x + 2 * x^2, 1)$$

where $n = 100, p = 2$

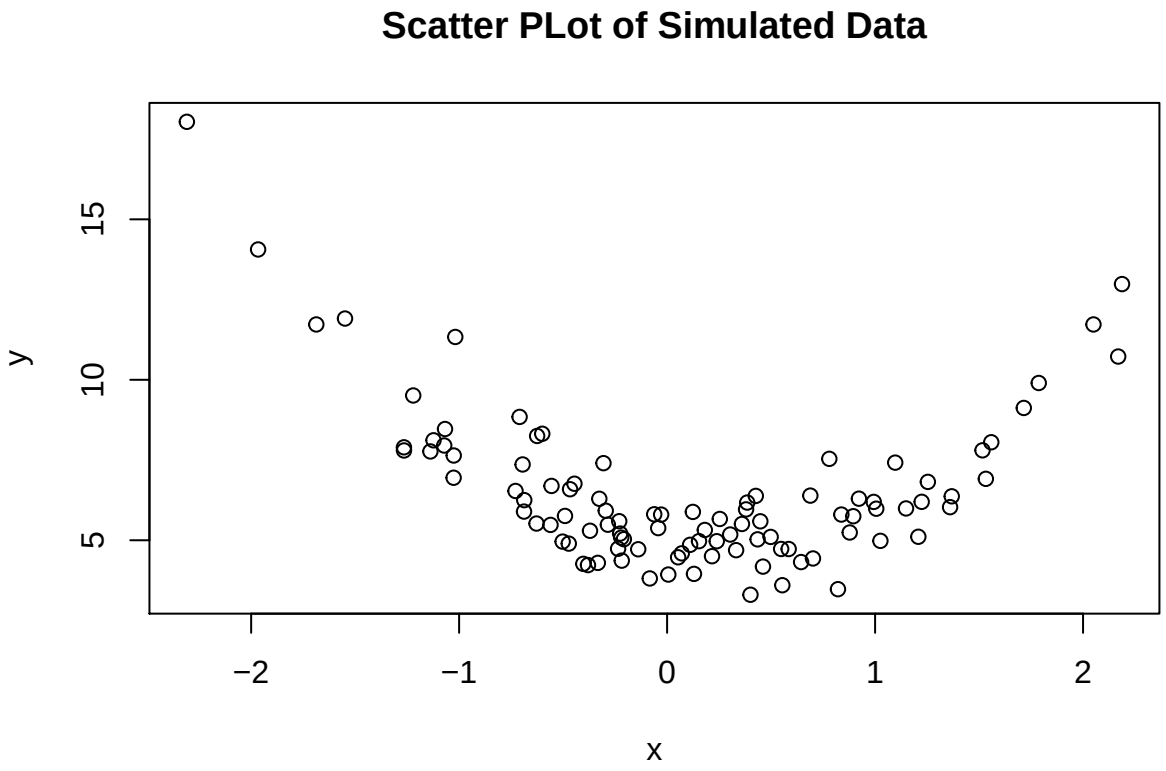
b)

```
set.seed(123)
x=rnorm(100)
e0=rnorm(100)
y=5-x+2*x^2+e0

data = as.data.frame(cbind(x,y))



plot(data,main="Scatter PLot of Simulated Data")
```



According to the scatterplot there seems to be a quadratic relationship between X and Y which would not be captured in linear regression. **c)**

```
fit = lm(y ~ poly(x,2),data=data)
print (fit)
```

```
##
## Call:
## lm(formula = y ~ poly(x, 2), data = data)
##
## Coefficients:
## (Intercept)  poly(x, 2)1  poly(x, 2)2
```

```
##       6.468       -5.276       21.353
```

d)

```
#equal size split means 50/50
train_subset_indexes <- sample(100,50)

train_fit = lm(y ~ poly(x,2),data=data[train_subset_indexes,])

print (mean((predict(train_fit,newdata=data.frame(x=data$x[-train_subset_indexes]))-data$y[-train_subse
```

```
## [1] 0.8545172
```

e)

```
mse <- c()
#train_sizes = c(10, 40, 80, 90, 30)
split_index = 1:5
for (i in seq(1,5)){
  train_subset_indexes <- sample(100,50)
  train_fit = lm(y ~ poly(x,2),data=data[train_subset_indexes,])

  mse<-c(mse,mean((predict(train_fit,newdata=data.frame(x=data$x[-train_subset_indexes]))-data$y[-train_
}

print (cbind(split_index,mse))
```

```
##      split_index       mse
## [1,]           1 1.1499209
## [2,]           2 0.6487054
## [3,]           3 0.9188413
## [4,]           4 1.2799498
## [5,]           5 1.1027397
```

We see that for each split we get a slightly different MSE. This is because we have introduced randomness into the MSE by selecting a random subset of 50 data points to select as the training set.

f)

```
library(boot)
set.seed(1000)
LOOCV <- c()
model_num <- seq(1,4)

glm.fit = glm(y ~ x, data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

glm.fit = glm(y ~ poly(x,2), data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

glm.fit = glm(y ~ poly(x,3), data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

glm.fit = glm(y ~ poly(x,4), data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

print (cbind(model_num,LOOCV))
```

```
##      model_num LOOCV
## [1,]         1 6.024
## [2,]         2 0.966
## [3,]         3 1.000
## [4,]         4 0.999
```

g)

```
set.seed(1)
LOOCV <- c()
model_num <- seq(1,4)

glm.fit = glm(y ~ x, data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

glm.fit = glm(y ~ poly(x,2), data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

glm.fit = glm(y ~ poly(x,3), data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

glm.fit = glm(y ~ poly(x,4), data=data)
LOOCV <- c(LOOCV,round(cv.glm(data, glm.fit)$delta[1],3))

print (cbind(model_num,LOOCV))
```

```
##      model_num LOOCV
## [1,]         1 6.024
## [2,]         2 0.966
## [3,]         3 1.000
## [4,]         4 0.999
```

The LOOCV scores do not change at all, which makes sense because there is no sampling method, we leave out each observation deterministically.

h)

The second model had the best score because it is the true model.