Nicholas Orriols

Statistical Learning

Midterm

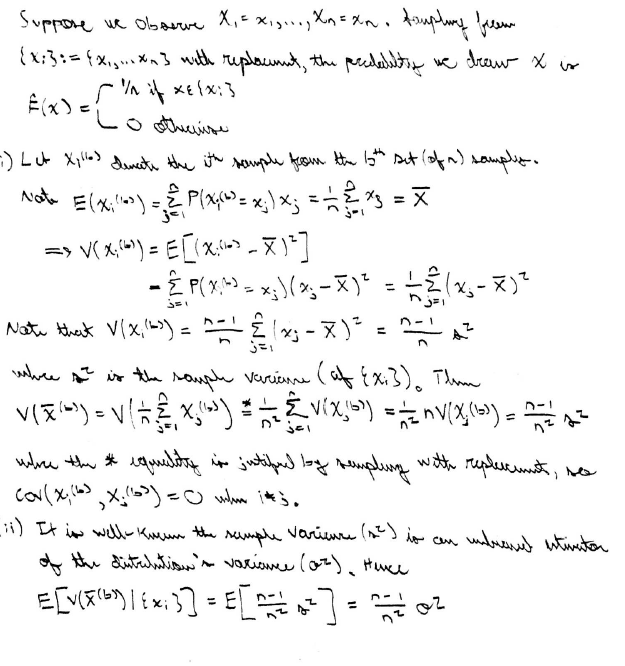
1. (i) The radial SVM prediction for a given input is a linear combination of its kernels with each support vector, and each kernel is proportional to the ‘similarity’ between the vectors (kernels take values in (0,1]). As the feature space increases in dimensionality, for an input vector to be similar to a support vector, it must be closer in distance to the support vector with respect to more features. Then the kernel function between 2 vectors is decreasing with respect to the dimensionality of the feature space. Therefore, support vectors being similar to the input in a lower dimensional space have less of an effect on the prediction (the kernel goes to 0) in a higher dimensional space. Hence, I do think radial SVMs suffer from the curse of dimensionality.

(ii) In LDA, we assume the input vectors of each potential classification are normally distributed with their own mean and a *shared* covariance matrix. In QDA, we assume the input vectors of each potential classification are normally distributed with their own mean and *their own* covariance matrix. QDA is more flexible than LDA because it does not have the restriction of assuming the input vectors for every class have a shared covariance matrix.

(iii) Overfitting is when a model models the training data extremely well while poorly modelling the relationship between the predictors and response. Overfitting is often caused by a model having so much flexibility (e.g. parameters) that it can fit the training data almost perfectly. Consequently, overfitting is commonly detected by the occurrence of both an extremely low training MSE and a large testing MSE. The best way to avoid overfitting is to use models which are not so flexible that they can fit the training data perfectly (although this might introduce some bias into the model).

(iv) Suppose the response is highly correlated with a small set of uncorrelated covariates which account for a small percentage of the total variability in the data. In PCR, we transform the *p* predictors into *M* new predictors (*M < p*) which maximize the amount of explained variability (from the original data) and do least squares regression on those transformed *M* predictors. Since the covariates accounting for most of the data’s variability are not correlated with the response, we would expect PCR to perform poorly in this case. Conversely, in PLS, we transform the predictors to maximize their correlation with the response. If the response were highly correlated with a small set of covariates, PLS would likely be predictive of the response with only a few transformed variables. Hence, if the response is highly correlated with a small set of uncorrelated covariates which account for a small percentage of the total variability in the data, we would expect PLS to outperform PCR.

(v) Since penalized regression minimizes a function of the form where , the penalty generally does not allow for estimated coefficients with large absolute values unless they significantly reduce the RSS. As a result of the penalty, each of the estimated coefficients is biased toward 0 (they approach 0 as ). This can be beneficial because it reduces the variability in estimating each of the coefficients (since they are closer to 0) while minimizing the increase in RSS.

1. 

(iii) We know that the true variance of (since we know ) is

So, the answer in (ii) is not the same as the true variance of .

(iv) Since the result in (ii) is smaller than the true variance of , it follows that the average estimated standard error of via the bootstrap is smaller than the standard error of . Consequently, on average, confidence intervals for using the estimated standard error from the bootstrap of will be smaller than confidence intervals for by using the known standard error of directly.

1. (i) Data set 1 is where lasso outperforms PCR, and data set 2 is where PCR outperform lasso.

For data set 1, the first 10 covariates for each observation are randomly sampled from a normal distribution with variance 3 and mean equal to the covariate number (e.g. the 7th covariate is sampled from ). The rest of the covariates for each observation in data set 1 are sampled from . The response for each observation is the sum of the predictors (so the true coefficients are all 1s). Training data sets had 100 observations with 100 predictors, while the testing data set had 10,000 observations which were all generated in the same way as the training data.

For data set 2, the first 5 covariates of each observation were sampled from , while the rest of the covariates for each observation were sampled from . The response for each observation in data set 2 is the mean of the first 5 covariates (so and for ). The testing data set had 10,000 observations which were all generated in the same way as the training data.

(ii) In data set 1, since the response is the sum of covariates, only the first 10 are significant since they are likely to be substantially different from 0. Since lasso automatically performs variable selection because of its penalty, it should shrink the coefficients of the 90 covariates with no effect on the response to 0 (which is characteristic of the predictors’ relationship to the response). Conversely, although they are sampled with smaller variance, the non-predictive covariates are responsible for a large part of the model’s variability; as a result, the variability that PCR explains (with only 5 principal components) is not related to the response, leading to PCR’s poor performance.

In data set 2, the first 5 covariates, which are the only ones predictive of the response, are responsible for most variability in the data. As a result, the first 5 principal components are highly correlated with the response, leading to PCR’s exceptional success. Lasso also performed relatively well since it likely recognized setting the last 95 coefficients to 0 had minimal effect on the RSS, but it did not perform as well as PCR.

(iii) The code corresponding to this question ultimately outputs the following which are the average MSEs for each model in each data set.

> # Lasso (left) should outperform PCR (right)

> round(colMeans(MSEs)[1:2],5)

[1] 0.82309 16.26292

>

> # PCR (right) should outperform Lasso (left)

> round(colMeans(MSEs)[3:4],5)

[1] 1.38288 0.00045

(iv) PCR performed terribly in data set 1, which was expected since the variability in the data was spread evenly amongst uncorrelated predictors: this causes PCR to need many principal components to model the data. Although lasso performed well in data set 2, since the significant covariates were so heavily correlated with the response, PCR still outperformed it substantially.

1. (i) After reading in the data, the following code fits the requested model and outputs its MSE.

> glm=glm(Y~., data=train, family=binomial)

> glm.test = as.numeric(predict(glm, newdata = test[,-1], type="response") > .5)

> mean(glm.test != test[,1])

[1] 0.2858439

(ii) I used bootstrapping to estimate the model parameter. Over 10,000 iterations, I took a random sample from the training data with replacement, fit a GLM to that data, and then calculated the estimate for the parameter based on that model. For the percentile method, the 88% confidence interval limits are the 6th and 94th percentiles of the estimates. A confidence interval which used the standard error of the estimates (and assumed the estimates were centered) is also provided.

> set.seed(9)

>

> B = 10000

> estimates=c()

> for (b in 1:B) {

+

+ index=sort(sample(1:nrow(train), size=nrow(train), replace = T))

+ glm.bs=glm(Y~., data=train[index,], family=binomial)

+ estimates[b]=glm.bs$coefficients[2] + exp(glm.bs$coefficients[3])

+

+ }

>

> estimates=sort(estimates)

> thetaHat=glm$coefficients[2]+exp(glm$coefficients[3])

>

> # Gives percentile method CI

> CI = c(estimates[.06\*B], estimates[.94\*B]); CI

[1] 0.8670051 2.0630557

>

> # Gives standard error estimate CI

> CI = thetaHat + c(-1,1)\*sd(estimates)\*qnorm(.94); CI

[1] 0.6731454 1.9113869

(iii) The following code performs backwards stepwise regression on the (full) model in (i), obtains and outputs the new model’s MSE.

> bsr=step(glm, trace = 0)

>

> # Get MSE of BSR model

> bsr.pred=as.numeric(predict(bsr, newdata=test[,-1], type="response") > .5)

> mean(bsr.pred != test[,1])

[1] 0.2450091

Note that this MSE is lower than the MSE of the full model, indicating backwards stepwise regression improved the model.

(iv) To investigate the inclusion of non-linear terms, I first made a GLM which included each predictor with degrees 1 through 3; so, there were 15\*3 + 1 = 46 terms. The choice of 3 was made because the GLM function would not converge for higher degree polynomials. I then used backwards stepwise regression on this model. The following code does this and outputs the testing MSE of the best model found via backwards stepwise regression.

> V=paste("poly(X.",1:15, sep="")

> for(i in 1:15)

+ V[i]=paste(V[i], ",3)", sep="")

>

> terms=paste(V, collapse = "+")

> f=as.formula(paste("Y~",terms, sep=""))

>

> glm3=glm(f, data=train, family=binomial)

> bsr.3=step(glm3, trace=0)

> bsr.3.pred=as.numeric(

+ predict(bsr.3, newdata = test[,-1], type="response")>.5)

> mean(bsr.3.pred != test[,1])

[1] 0.2912886

Since this testing MSE is larger than the testing MSE for the simpler GLM, I do not think the GLM should include non-linear terms. Note however that the step() function uses AIC (and not testing MSE) to determine which model is best. As a result, the model chosen by step() may not minimize the MSE for GLMs with up to 3rd degree terms.

(v) In this case, there are 131 observations. QDA estimates many more parameters (272) than there are observations (272) whereas LDA only estimates 152 parameters. As a result, QDA would be much more likely to overfit to the data than LDA. Additionally, all the predictors have similar standard deviations (all very close to 1), so it is possible the LDA- assumption that each predictor has a shared variance matrix- is correct while the QDA assumption- a unique variance matrix for each predictor- is unnecessary and may lead to overfitting. Hence, I think LDA would outperform QDA here.

(vii) Since the variability in the training data is relatively evenly spread across the predictors, PCR would probably need a larger number of components just to effectively model the data. Additionally, there is no guarantee those principal components will be predictive of the response. Hence, I think it is unlikely PCR would be helpful in predicting the response.

(vii) I chose the tuning parameters and via 10-fold cross-validation (this is done automatically by the tune function). The following code fits the SVM and outputs the tuning parameters chosen by cross-validation and the out of sample MSE.

> set.seed(10)

> G = 10^(-3:3)

> C = c(.001,.01,.1,1,3,5,10,25,50,75,100)

> rad.svm = tune(svm, Y~., data=train,

+ ranges=list(cost=C, gamma=G))$best.model

> rad.svm$cost; rad.svm$gamma

[1] 3

[1] 0.1

>

>

> rad.svm.pred=predict(rad.svm, newdata=test[,-1])

> mean(rad.svm.pred != test[,1])

[1] 0.2813067

(viii) The last part of code in the R file produces the following plot.

