Introduction to Machine Learning - Final Exam

Landon Buell

Spring 2020

Problem 1

Assume that you have a data set with a predictor (feature) X and the target Y. You run simple linear regression ($Y\ X$) and get the best fit with RSS=20 and TSS=120.

1. What is the covariance between X and Y if the variances of X and Y are Var(X)=15 and Var(Y)=20?

The *covariance* between variables X and Y is defined by:

$$Cov(X,Y) = E[XY] + E[X]E[Y] \tag{1}$$

2. Repeat if RSS=50 and TSS=40. Discuss the result.

Problem 2

Suppose that we estimate the regression coefficients in a LASSO model by solving:

$$\min_{\beta_0,\ldots,\beta_p} \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \frac{1}{\lambda}\sum_{j=1}^{p}|\beta_j| \tag{2}$$

for a particular value of λ. For parts (1) through (5), indicate which of i. through v. is correct. Justify your answer.

1. As we increase λ from 0.1 to ∞, the training RSS will typically:

As λ increases, the coefficient in front of the sum decreases. This decreasing value imposes a *smaller and smaller* constraint on the lasso regression, thus the constraint region (figure 6.7 of ISL) gets smaller and smaller, $\sum|\beta_j| \leq \frac{1}{\lambda} \to 0$ as well. As the region shrinks in sizes, it is likely to get farther and farther from the $\hat{\beta}$, which is at RSS predicted coefficients. Thus the training RSS will (ii.) slowly decrease.

2. As we increase λ from 0.1 to ∞, the test MSE will typically:

Increasing λ causes $1/\lambda$ to decrease to zero. Somewhere along the way, the appropriate value for λ will be chosen such that MSE is reduced as much as possible. (this would be an example of cross-validation) Thus, as the constrain decreases, the resulting MSE will (v.) decrease initially, and then start increasing in a 'U' shape, as indicated by figure 6.8 in ISL.

3. As we increase λ from 0.1 to ∞, the (squared bias) will typically:

Again, increasing λ causes $1/\lambda$ to decrease to zero. Thus decreasing the volume of the constrain region, which will cause the squared bias to (ii.) steadily decrease, as also indicated by figure 6.8 in ISL. (In their cases, $\lambda \to 0$)

4. As we increase λ from 0.1 to ∞, the variance will typically:

Again, increasing λ causes $1/\lambda$ to decrease to zero. Thus decreasing the volume of the constrain region, which will cause the variance to (iii.) steadily increase, as also indicated by figure 6.8 in ISL. (In their cases, $\lambda \to 0$)

5. As we increase λ from 0.1 to ∞, the irreducible error will typically:

As λ is adjusted, the irreducible error will be untouched, thus (i.) remain constant.

6. How would I choose the best value for λ?

There is no single rule for what λ works best for any given problem, thus I would have to implement some sort of *cross-validation* method. I would essentially choose a subset of values to test, construct a LASSO model with each value, and then choose the λ that results in the lowest *testing* RSS error.

Problem 3

1. Do you expect random forests to achieve a smaller or larger training error than bagged trees with the same number of trees? Justify your answer.

Bagging trees is a method used to create N different decision trees based on N different subsets of a training data set. Each subset is then used to produce N different boot strapped models, $\hat{f}^{*1}, \hat{f}^{*2}, ..., \hat{f}^{*N}$. The *bagged tree* model is then a uniform sum of each of the boot-strapped models. On the other hand, a random forest only generate new leaf splits based on a subset of all possible features, as opposed to even nagged tree use all possible features for splits. This subset of predictors lets forest have wildly different trees to account for different feature strengths, where as boosted tress will all have roughly similar leaf structures. With this, I would expect random forests to have a *smaller* training error than bagged trees, with the same number of trees.

2. How would you expect the training error of bagged trees to compare with boosted trees? Justify your answer.

Bagged trees as described above, contrast with Boosted trees. Boosted trees involve taking a subset of the design matrix, and using residuals, rather than the start target. In each iteration, some small decision tree is produced based on that feature or sample subset and target and then that small tree is added to the larger model (along with a shrinkage parameters). This process repeats for B trees as outlined in ISL Algorithm 8.2. By modifying these parameters as needed, the tress can grow into all different shapes and make splits and leafs based on different parameters. In general, this would cause of the training error of boosted trees to be lower than that of bagged trees.

3. How would you decide how many trees to use for any particular data set in a random forest? It is best to choose the number that minimizes the training error? Why or why not?

As always, there is no set rule to determine the number of Trees to use in any problem, it varies between data sets. It is not wise to choose a number of trees that reduces the *training error* because this indicates a model that will over fit that particular training set and thus not do well on a testing set. Thus, to choose the appropriate number of trees for a given data set or problem, I would again apply some sort of *cross-validation* procedure to find what number of decision trees in a random forest reduces the *testing error* when trained on a similar data set. This would allow for the determining of the correct number of trees, along with a a roughtly minimized testing error.

Problem 4

This problem examines the differences between SVC (linear SVM), SVM with a polynomial kernel, and other linear classifiers.

1. For an arbitrary training set, would you expect for SVC (linear) or SVM (polynomial kernel) to work better on the training set? Why?

For any given data set, I would predict that a polynomial kernel SVM would work better on a training set. By design, a linear Support Vector Classifier, as the name implies, uses linear support vectors to create it's decision boundary. This model will only have a low training and testing error is the decision boundary between features is linear or can be approximated as linear. In most real-world data sets do not have linear decision boundaries.

On the other hand, a polynomial kernel Support Vector Machine, is built off of polynomial decision boundaries. Adding more terms and features is guaranteed to decrease the training error, but not necessarily the test error. Furthermore, In some cases, features with almost-linear decision boundary can be approximated with a polynomial where higher degrees have extremely small coefficients (Almost like Taylor-Series). In this cases, a polynomial kernel would still perform well on a linearly separable data set.

2. If the Bayes decision boundary between the two classes is linear, would you expect SVC or SVM to work better on the training set?

If we know the decision boundary is linear, then a SVC, which assumes a linear decision boundary, would be the optimal choice. However, an SVM, which often produces a non-linear boundary might perform better on a *training* set because it has the advantage of a greater number of features, which will always reduce the training error. Thus a linear SVC would so better on a test set and a polynomial SVM would still do better on a training set due to it's flexibility to fit that set.

3. True or False: There is no need to use slack variables in SVMs with polynomial kernels because the decision boundary can be non-linear. Justify your answer.

4. LDA, Logistic regression, and SVC (linear) all fit a linear decision boundary. Will their fits be the same? What would be some reasons for you to prefer LDA over SVC?

The fits between LDA, Log Reg and SVC are all based on linear decision boundaries, but each one is constructed differently and comes with different purposes - thus the fits are not guaranteed to be the same, and in most cases, will not be the same. For example, Logistic Regression a probability concept based on the logistic function. The value of the function is then used as a percentage probability that a sample existent within a certain class given a feature space, split along the 50/50 line, which creates the decision boundary. Support Vector Classifiers also build decision boundaries but also included maximum margin ranges, which are the support vectors. This does not really include a formal probability metrics, but does add a tolerance band for class features.

In the case of LDA, it also attempts to build a class prediction based on a set of predictors, but allows for allows for a few other ideas that may make it favorable to SVC's in a few contexts: As stated in ISL, pg. 138, LDA performs will with well-defined decision boundaries, and offers a probability for it's sample to be in a class, given it's predictors, whereas a SVC mostly gives a decision boundary and a maximum margin tolerance. Additionally, if the predictors in a data set are vaguely Gaussian, LDA will perform well because this distribution is one of it's core assumptions.

Problem 5