Exam 1 Review

CS-6570: Data Science Algorithms I

This review covers the major concepts we've learned so far is CS-6570 through gradient descent.

1 The Bias-Variance Tradeoff

The fundamental assumption behind the models we create is that there is some unknown distribution of our data, we've taken an IID (independently and identically distributed) sample of our data, and our model is trying to guess at what this distribution is. The model is *unbiased* if the expected values of its parameters are the "true" values of those parameters, that is to say the parameters that provide the best approximation to the actual distribution given our chosen class of models. The variance of the model is how much the predicted values of the parameters tend to depart from the true values as a result of the randomness inherent in the sampled data.

Generally speaking, there is a tradeoff between bias and variance, and variance can be decreased at the expense of biasing the model. Traditional least-squares regression is unbiased, which means it has a relatively high variance and can be prone to overfitting. Modifications like ridge or lasso regression introduce some bias in the model (both are biased towards the coefficients being closer to 0) in order to decrease the variance.

2 Simple and Multiple Regression

With simple linear regression, a model of the form $Y = \beta_0 + \beta_1 X$ is used to predict the output Y values based on the input X values. The coefficients β_0 and β_1 are chosen so as to minimize the sum of squares error, which is the sum of the squares of the differences between the actual and predicted output values $\sum_i (y_i - \hat{y}_i)^2$.

We can extent this model in multiple ways - one way is by increasing the degree of the polynomial from a linear (degree 1) model to something higher, producing a model of the form $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots \beta_n X^n$, where the coefficients are again chosen so as to minimize the sum of squares error. The model can also be extended to include multiple input variables $X_1, X_2, \ldots, X_n$, producing a model of the form $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$.

These regression models are unbiased, and adding more variables will always decrease the sum of squares error. However, adding more variables also increases the model variance, which can lead to overfitting (essentially, the model doing well on the training data, but poorly on other data).

3 Ridge and Lasso Regression, Variable Selection, and Cross-Validation

One way to deal with overfitting is to introduce a penalty term for the coefficients. This can decrease the variance of a model at the expense of introducing some bias. In the case of ridge and lasso regression, both models bias towards coefficients closer to 0. In ridge regression the model attempts to minimize the sum of squares error plus a squared penalty term for the coefficients:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

With lasso regression the model attempts to minimize the sum of squares error plus an absolute value term for the coefficients:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

Both models approach a constant model as λ gets large, and approach standard (unbiased) regression as λ gets small. Lasso regression tends to set the coefficients of unimportant variables to 0, while ridge tends to make them small but still non-zero.

Sometimes there are too many input variables and the desire is to create a model with a subset of them, eliminating variables that don't add much predictive value to the model, but might significantly increase the model's variance. To choose a subset of variables there are various (ha!) techniques. One way is best subset selection, in which literally every possible subset it explored. This is exhaustive and optimal when it's possible, but the number of possible subsets increases exponentially with the number of variables, and so this might not always be possible. Forward stepwise selection first finds the best model with 1 variable, the first step, and then finds the best model with 2 variables that includes the variable found on the first step. At each step a new variable is chosen, and at step k one additional variable is added to the $k-1$ found in the previous steps. This method doesn't try every combination of variables, but usually does a good job of finding the optimal subset or getting close to the optimal subset. Backward stepwise selection is the same idea, just starting with every variable, and removing one at each step.

A model will always have a lower RSS than a model built with a subset of its parameters (because there's always the option to set a parameter's coefficient to 0). Consequently, RSS and related measures cannot be used to differentiate among these models. In class we learned a few ways to differentiate among them, the most notable being k-fold cross-validation. k-fold cross-validation is a method for breaking the modeling data into k disjoint segments. One segment is left out, and the model is trained on the $k-1$ remaining segments, and then tested on the one left out. This is repeated leaving each segment out once, so a total of k times. The success of the model is its average performance on the data that is left out.

This approach can be used to determine the optimal value of a *hyperparameter*, which is a parameter that is set before modeling takes place, and not as a part of modeling. The term λ in ridge and lasso regression is a hyperparameter, as is the subset of variables left out in variable selection, and the number of knots to use for a regression spline.

4 Bootstrapping

Bootstrapping is a technique that looks like cheating, but it's not. The basic idea is to start with a set of sample data, and build other samples from this original data using sampling with replacement. So, some of the observations from the original data might show up multiple times in a derived sample, while others won't show up at all. In bootstrapping, these derived samples are used for modeling, and the variation in the modeling parameters are recorded for the various derived samples. This can give us a very good idea of how the modeling parameters vary among samples from the original dataset, and this in turn can give us usually pretty good estimate of how the modeling parameters vary among samples from the actual population.

5 Regression vs. Classification

The fundamental distinction between regression and classification is that with regression the model is attempting to predict a number, while with classification the model is attempting to predict a category. There may or may not (usually not) be any natural notion of a distance between the categories.

A given prediction is either right or wrong, but a more reasonable approach is to predict probabilities instead of specific predictions. The specific prediction is then just the category to which the model assigns the highest probability.

6 Logistic Regression

For binary prediction (yes/no, true/false, 1/0, etc...) one of the more popular methods is *logistic regression*. With logistic regression we start with a linear combination of our input variables as we have in traditional regression:

$$c_1X_1 + c_2X_2 + \cdots + c_nX_n.$$

We then plug this linear combination into the *sigmoid function*, which is a function that takes any real number and predicts an increasing number between 0 and 1 (which can be interpreted as a probability).

$$\frac{1}{1 + e^{-c_1X_1+c_2X_2+\cdots+c_nX_n}}$$

To figure out the coefficients for a given set of training data, we don't minimize the sum of square error like we do with regression. Instead, we calculate the maximum likelihood. The maximum likelihood is the value of the coefficients such that, if the sigmoid is a probability, it would make the observed data most likely. For example, if you have a coin that comes up heads with probability p, and you flip it 10 times, and it comes up heads 7 times, the maximum likelihood estimate for p would be 0.7.

In standard regression calculating the values of the coefficients that minimize the residual sum of squares can be done exactly, in logistic regression it's almost never possible to calculate the maximum likelihood estimate exactly, and so it needs to be approximated numerically using *gradient descent*.

7 Gradient Descent

A situation that comes up all the time with machine learning is that you've got a predictive function, and you have some measurement of how good it is (this measurement is usually called a *loss function*), and you want to optimize this predictive function relative to the loss function. In logistic regression, the predictive function is the sigmoid, and the measure you want to optimize is the likelihood. Finding the global optimal value is generally impossible, however it's typically straightforward to calculate the gradient of your measurement function with respect to the parameters of your model. This gradient tells you the direction you should move if, locally, you want to increase your measurement function the most. If you want to decrease your measurement function, you'd move in the direction opposite the gradient. Gradient descent is, essentially, taking a bunch of small steps in the direction of the gradient at each step in search of an optimal value. There are a few things about which you need to be concerned when using this process:

- If your step size is too great, you could step over and miss your optimal value.
- If your step size is too small, it could take you a very, very long time to find your optimal value.
- You could get stuck in a local extrema, which would mean it's the largest (or smallest) value close to where you are, but it's not the globally optimal value.

Also, typically gradient descent requires you to calculate your measurement function over the entire training dataset at each step. Depending on the size of your training data, this can take a long time. One way to get around this is with *stochastic gradient descent*, in which case the gradient is calculated with respect to just a single randomly chosen observation at each step. It's called stochastic gradient descent because the observation is chosen at random. This can significantly speed up the process of gradient descent, but entails some loss of accuracy.