Problem Set #2

Kevin McAlister

January 26th, 2022

This is the second problem set for QTM 385 - Intro to Statistical Learning. This homework will cover applied exercises related to validation sets, loss functions, and linear regression.

Please use the intro to RMarkdown posted in the Intro module and my .Rmd file as a guide for writing up your answers. You can use any language you want, but I think that a number of the computational problems are easier in R. Please post any questions about the content of this problem set or RMarkdown questions to the corresponding discussion board.

Your final deliverable should be a .zip archive that includes a .Rmd/.ipynb file and either a rendered HTML file or a PDF. Students can complete this assignment in groups of up to 3. Please identify your collaborators at the top of your document. All students should turn in a copy of the solutions, but your solutions can be identical to those of your collaborators.

This assignment is due by February 4th, 2022 at 11:59 PM EST.

`taiwanPricing_train.csv`, `taiwanPricing_test1.csv`, and `taiwanPricing_test2.csv` are data sets that include information about real estate valuation in Sindian District, New Taipei City, Taiwan. The training data has 314 instances of property value (`Price`) in New Taiwan Dollars per 3.3 meters squared, observation IDs (`ID`), and 7 predictors:

1. `Year` and `Month` of transaction
2. `Age` of property sold
3. `DisttoMRT`: The distance from the property to the nearest Taiwan Metro Transit station
4. `NumConvStore`: The number of convenience stores in the living circle
5. `Lat` and `Long`: Latitude and Longitude of the property

Your goal for this problem set is to build a model that does a good job of predicting the price of properties that we have not yet seen!

An important note: The two test data sets, with 50 observations each, are our holdout data for assessing the quality of our predictions on data that have no part in training the model. Do not touch these when training your model! You can forget about these two data sets until the final parts of this assignment.

Problem 1

Part 1

For the training data, create a plot that shows the relationship between the latitude and longitude of the property and its price. There are many ways to accomplish this: you can try creating a 3D scatter plot, but I think there are more clever ways to accomplish this in 2 dimensions using color, point size, or smoothers to create a contour plot.

What relationships do you see? Do you think that this relationship is easily captured using a linear model? Why or why not?

Part 2

One approach to capturing this kind of relationship is to move to a **nonparametric model** that gives us an opportunity to capture the weird non-standard dependence structures that come from geographic data. We haven't discussed many of these yet, but we have briefly discussed K-nearest neighbors regression. For this kind of problem where *distance* is actually distance, this simple model makes a lot of sense.

As a reminder, K-nearest neighbors regression is a voting-style model that maps outcomes from the training data to predictions using the following algorithm:

1. Given a vector $\boldsymbol{x}_{Test}$ in the P-dimensional covariate space, compute the distance (however you may want to define distance) from $\boldsymbol{x}_{Test}$ to each $\boldsymbol{x}_{i,Train}$ in the training set.
2. Select the K smallest distances and store the set of k outcomes, $y_{i,Train}$.
3. Compute y_{Test} as a distance-weighted combination of the k selected outcomes:

$$y_{Test} = \sum_{k=1}^{K} \alpha_k y_{k,Train}$$

where α_k is a weight that is a function of the distance between the test point and the respective training point (smaller distance goes with larger weight). Sometimes, this weight is just set to $\frac{1}{K}$.

At its core K-nearest neighbors is a pretty simple algorithm, but the search for nearest neighbors can be quite intensive with hacky approaches (like the kinds we would write if we were to naively write the search algorithm). For this reason, we'll use prebuilt implementations for our analyses. In R, I recommend using the function `knn.reg` in the `FNN` package or `knnreg` in the `caret` package. In Python, the `KNeighborsRegressor` function in `sklearn` will do the same thing.

Using the latitude, longitude, and price included in the training data set, compute the mean squared error of the predictions made by the model **for the in-sample data** for K-nearest neighbors for 1 through 20 nearest neighbors. Plot the mean squared error (MSE) against the number of nearest neighbors.

Recall that the in-sample MSE always decreases as the flexibility and complexity of the model increases. What is the least complex value of k for the K-nearest neighbors algorithm? In a sentence or two, explain why this is the case.

Part 3

Write a function called `knn_reg_kfold` that takes five arguments: 1) `x` - a $N \times P$ matrix of predictors, 2) `y` - a N vector of outcomes, 3) `k` - an integer value for the number of neighbors to take, 4) `folds` - an integer value for the number of folds in the data, and 5) `seed` - an integer value that will define the seed that dictates how the folds are divided. Your function should return the K-fold cross validation estimate of the expected mean squared prediction error for a new observation.

The implementation of K-nearest neighbors that you choose may have a built in K-fold cross validation method. For this problem, please construct the folds and estimates yourself!

Using `knn_reg_kfold`, set the numer of folds to 2,5,10,20, and N (LOOCV) and estimate the cross validation estimate of the mean squared prediction error. Plot your estimate of the expected prediction error against the number of nearest neighbors for each number of folds on the same graph. Does the number of neighbors with the lowest K-fold prediction error remain the same/similar across all the number of folds?

Note: Leave one out cross validation should be relatively quick given this data size and the speed of the KNN algorithms in the recommended packages. If you find that your function is really dragging, check that your implementation is using a compiled sort and search. If that doesn't work, reach out and I can try to provide some advice.

Part 4

Create the plot in Part 2 again using different seeds for your validation set splits. Do the results change? What does this say about the relationship between the estimated expected prediction error and your choice of splits?

Part 5

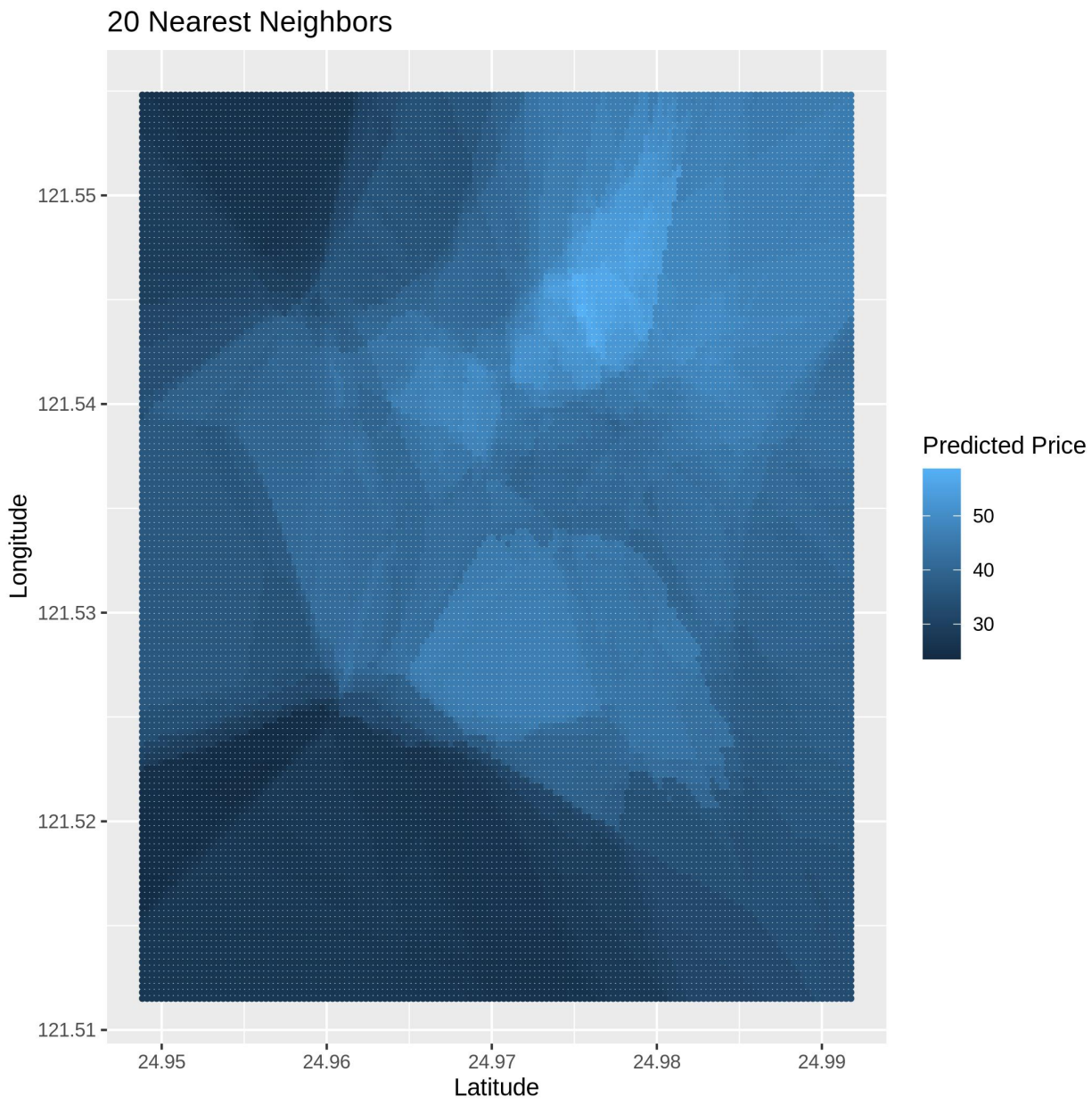
Using the above information, choose a value of K as the "optimal" choice. In a few sentences, defend your choice. You can use any heuristic you'd like to make this choice.

Using your chosen value of K, create a latitude-longitude prediction map - using the training data, draw the predictions for new observations within the square created by the minimum and maximum latititudes and longitudes. Create the same drawings for a 1-nearest neighbor regression and a 20-nearest neighbor regression. Compare these pictures to your original plot. How do the different choices compare to the training data?

Hints:

1. There are a number of different ways to create this plot. I'd recommend using a grid-based approach to create a set of predictions and then a smoothed contour plot or colored dot plot to demonstrate the relationship between lat, long, and price.
2. In 2-dimensions, we don't need a huge grid set. 150 equally spaced points per axis should be plenty.
3. Suppose we have two sequences of values and we want a data frame with all possible pairwise combinations of the values. In R, check out the `expand.grid()` function. This way, you can pass this as the test data for a KNN regression and only have to do the search once (as opposed to a double for loop). In Python, check out `meshgrid` in `numpy` - it can quickly be converted to a `numpy` or `Pandas` array.

An example plot:



Problem 2

Part 1

We've used latitude and longitude to build a predictive model. Let's try a different predictor - `Age`. For the training data, plot the age of the house against the price of the house. Is this relationship linear? Polynomial? What degree of polynomial do you think best describes the relationship? In a sentence or two, logically explain why this perceived relationship might hold.

Part 2

Find the degree of polynomial that maximizes the expected prediction error using the training data. That is compare $y_i = \alpha + \epsilon_i$ vs. $y_i = \alpha + \beta_1 x_i + \epsilon_i$ vs. $y_i = \alpha + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$ and so on.

Since the linear model is algebraically "easy"[1], let's use the wealth of non-simulation based estimates of the expected prediction error to make this decision. Write a function called `lm_pred_metrics` that takes in either a fit linear regression model **or** the $N \times P + 1$ matrix of predictors (with the first column being a column of 1s to capture the intercept) and a N-vector of outcomes. The function should compute:

1. The AIC
2. The BIC
3. The leave-one-out cross validation estimate of the expected prediction error
4. The generalized cross validation estimate of the expected prediction error

You should compute the regression coefficients and estimate of σ^2 using a pre-built linear regression function and extract these. For LOOCV, you'll also need the hat/influence matrix. For GCV (under linear regression), you can avoid using the hat matrix if you pick up on the trace trick in the hints. Your chosen regression implementation will also likely return residuals, $\hat{\boldsymbol{y}}$, and many other meaningful metrics of fit.

Some important identities for this exercise:

1. For a linear regression model with N observations of P predictors and outcomes, the log-likelihood of the model under normally distributed idiosyncratic errors (e.g. the standard model you all know) is:

$$\ell\ell(\beta, \sigma^2 | \boldsymbol{X}, \boldsymbol{y}) = -\frac{N}{2}\log 2\pi - \frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i\boldsymbol{\beta})^2$$

2. For any linear regression model with an intercept and P predictors, there are $d = P + 2$ parameters: P coefficients, the intercept, and σ^2.
3. Given a matrix of predictors, $\boldsymbol{X}$, and a corresponding vector of coefficients, $\boldsymbol{\beta}$, $\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$
4. The hat/projection matrix for linear regression can be computed as $H = \boldsymbol{X}(\boldsymbol{X'X})^{-1}\boldsymbol{X'}$. In R, we can get the diagonal of the hat matrix very quickly using `hatvalues(model)`. In Python, you will find these values in various regression implementations as the *influence* of a specific point.
5. The trace of H for linear regression can be modified using the trace trick:

$$\text{tr}(H) = \text{tr}\left[\boldsymbol{X}(\boldsymbol{X'X})^{-1}\boldsymbol{X'}\right] = \text{tr}\left[(\boldsymbol{X'X})^{-1}\boldsymbol{X'X}\right]$$

6. The AIC and BIC can be computed as:

$$\text{AIC} = -2\ell\ell + 2d \text{ ; BIC} = -2\ell\ell + d\log n$$

7. The LOOCV and GCV estimates of expected prediction error can be computed as:

$$\text{LOOCV} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{y}_i}{1 - H_{i,i}}\right]^2 \text{ ; GCV} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{y}_i}{1 - \frac{\text{tr}(H)}{N}}\right]^2$$

[1] I say with the most sarcastic quotes of all time. Really I just mean that it's analytically tractible to do a lot of things.

Part 3

Using your function above, compute the various metrics of generalizability for all polynomial models of order 1 through 10 (the third order polynomial model would be $y_i = \alpha + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$, for example). Plot each metric against degree of the basis expansion. Which of the models minimizes the generalization error? Is this conclusion consistent across metrics? Would we expect K-fold cross validation to produce a similar conclusion? What does this say about predictive model selection for standard linear regression models?

Create a plot that shows the predicted values granted by the optimal model for any age that's within the set of ages in the training set and compare it to the observed data in the training data. Does it look right on the entire training data? A little underfit? A little overfit?

Remember the degree of the model as the optimal Age model.

Problem 3

Using the tools you've already developed, come up with a good predictive model for the relationship between distance to nearest metro station (`DisttoMRT`), number of convenience stores in the walking radius (`NumConvStores`), and house price (`Price`). You should use K-nearest neighbors or linear regression as your predictive model. Using the methods we've discussed for quantifying generalization error, argue that your model is optimal in some sense of out-of-sample generalization.

Notes:

1. For linear models, definitely think about **interaction terms** between the two predictors. Maybe you'll need them, maybe you won't.
2. It's infeasible to to test **all** possible linear models for the two predictors (considering interactions and/or basis expansions). Use intuition to restrict the set of models you examine.
3. You may want to use K-fold cross validation for the regression models. You don't have to, but you can implement it yourself **or** use a pre-built implementation.
4. Make sure that you're comparing apples-to-apples when comparing generalization metrics. For example, the comparison between AIC and 5-fold cross validated prediction error does not give us an interpretable comparison.
5. The advantage of K-nearest neighbors is that you can test all possible models. However, it has a tendency to undersmooth some relationships.

Problem 4

Part 1

You've found 3 "optimal" models for predicting the price of a home - one that looks at location, one that looks at age, and one that looks at features surrounding the home. Compare the LOOCV estimate of the expected prediction error across the three models. Which one performs best?

Obviously, we could probably do better if we used all of the predictors in one model. But, each model that we've tested is *cohesive* - we're examining the predictive power of different sets of real estate covariates.

Alas, we've judged out-of-sample predictive power without actually ever using an out-of-sample data set - a.k.a. the "original sin". Fortunately, your gracious teacher has held out some test sets for you. Using the models you declared optimal in the previous steps, assess the mean squared prediction error for each model on the two holdout data sets.

Does the same ranking of models hold for each test set in terms of average predictive accuracy? Are the estimates produced by the various methods close to the average prediction errors in the test sets?

Part 2

A common phenomenon in predictive error quantification is that test sets that are truly randomly pulled from the same data generating process as the training set will overestimate the average prediction error (e.g. be a bit more pessimistic than reality). This is due to the fact that we are optimizing predictive accuracy with respect to smaller surrogate models rather than the full caboodle of training data - even LOOCV is an average over slightly smaller submodels that results in some distance from the truth. On the other hand, test sets that are not plausibly drawn from the same data generating process as the training data will give much higher generalization errors - we can think of this as **extrapolation error**.

Compare the training data to the two test sets. Does this phenomenon help explain the results you just found?

At the end of the day, we're just making good educated guesses. However, we aren't wizards. We can't know what we haven't already seen.