Elements of Statistical Learning

Chapter 1 - Introduction

- Supervised learning is when the outcome variable is present and is used to guide the learning process.
- Unsupervised learning is where we observe the features but have no measurements of the outcome.
    - The task is to describe how the data is clustered or organized.

Chapter 2 - Overview of Supervised Learning

- Quantitative outputs have a continuous range, while qualitative measurements assume values from a finite set.
- Classification and regression can be seen as types of function approximation.
- Dummy variables are created when you have a K-level qualitative variable, and it is represented by a set of one hot vectors with K variables.
    - Categories of red, green, and blue can be [1, 0, 0], [0, 1, 0], [0, 0, 1].
- 2 very simple approaches to prediction are linear model with least squares and KNN.
- Linear Model basically multiplies the input vector with a weight vector and adds a bias term to get the prediction.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j.$$

    - Least squares is the method by which you fit the linear model to the training data. In other words, it's what determines the weights. The goal is to pick coefficients beta to minimize the residual sum of squares.

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2.$$

    - RSS is a function that represents our error and is a function that we can minimize. Minimizing this loss can either come through solving the normal equations. This is a pretty much solved problem in math. The unique solution is

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y},$$

    - However, you can also use gradient descent to get to that optimal value. Both are equivalent, but you might want to use gradient descent when the above operation

is too computationally expensive or when X transpose X is not invertible, which happens when you have less training examples than features.

- KNN uses the observations in the training set closest in the input space to the current input x to form the y hat prediction.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

  - N(x) is the neighborhood of x defined by the k closest points to x.
  - Basically, we find the k closest points and take the average of their labels.
  - The train error for KNN will always be 0 for K=1, and will always increase as K increases. This begs the question why don't we just keep K=1? The answer lies in the problem of overfitting, because although train error is minimized, the resulting decision boundary is not generalizable.
  - Remember that KNN is a non parametric classifier because you don't really have any coefficients you want to optimize (like you did with Lin Reg). The number of parameters grows with the training set, and is not fixed.
- KNN can be thought of as having a high variance since the decision boundary is dependent on just a handful of inputs that are close by. Lin Reg, however, can be thought of as low variance because it's a pretty simple model (?)
- Lin Reg assumes f(x) is well approximated by a globally linear function, while KNN assumes f(x) is well approximated by a locally constant function.
- Curse of dimensionality refers to problems that only show up when analyzing/organizing data in high dimensions.
  - One of the problems is that the volume of the data increases so fast that the available data becomes sparse.
  - Something my friend was telling me is that it would take 100 million training examples to be able to perfectly fit a input with 10 variables, and each taking on discrete values from a set of 10 options.
  - In theory, the curse means that as our dimensionality increases, our amount of training data should increase exponentially to make up for it.
- Maximum likelihood estimation is basically a more general principle for estimation and loss functions. Basically, it's calculating the log probability of the label y, subject to some parameters theta.

$$L(\theta) = \sum_{i=1}^{N} \log \Pr_\theta(y_i).$$

  - The principle of maximum likelihood assumes that the most reasonable values for theta are those for which the probability of the observed sample is the largest.
  - This is where cross entropy loss is based of.

- As model complexity increases, the variance also increases, and the biases decreases.

Chapter 3 - Linear Methods for Regression

- Simple linear models can sometimes outperform nonlinear ones when there is a small number of training examples, or sparse data, or a low signal to noise ratio.
- A lot of this chapter went over my head mathematically :(
- Ridge regression shrinks the regression coefficients by imposing a penalty on their size.

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

  - The last term in the argmin basically shows that the loss function penalizes large weight terms. The larger the value of lambda, the larger the penalty.
- The idea of penalizing by the sum of squares of the parameters is used in neural nets and is called weight decay.
- The largest principal component is the direction that maximizes the variance of the projected data, and the smallest principal component minimizes that variance.
  - Ridge regression projects y onto these components and then shrinks the coefficients of the low variance components more than those of the high variance components.
- Lasso regression is the same as ridge except you don't square the weight term.

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

- Ridge regression has an L2 ridge penalty while lasso regression has an L1 lasso penalty.
- The above two models are methods of restricting the linear regression model.

Chapter 4 - Linear Methods for Classification

- Fitting of models is achieved by minimizing MSE for regression, and can be achieved by minimizing cross entropy for classification.
- Math too hard :(

Chapter 5 - Basis Expansions and Regularization

- Math wayyyyy too hard :(

Chapter 6 - Kernel Smoothing Methods

- There is another class of regression techniques where you estimate the function f(x), but you fit a different yet simple model separately at the each query point by using only

those observations close to the target point to fit the simple model in such a way that the resulting estimated function f is smooth.
    - The localization is achieved using a kernel.
- KNN methods result in decision boundaries that are mostly pretty bumpy.
    - Rather than giving all of the points in the neighborhood an equal weight, we can assign points that die off smoothly with distance from the target point.

average

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)},$$

with the *Epanechnikov* quadratic kernel

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right),$$

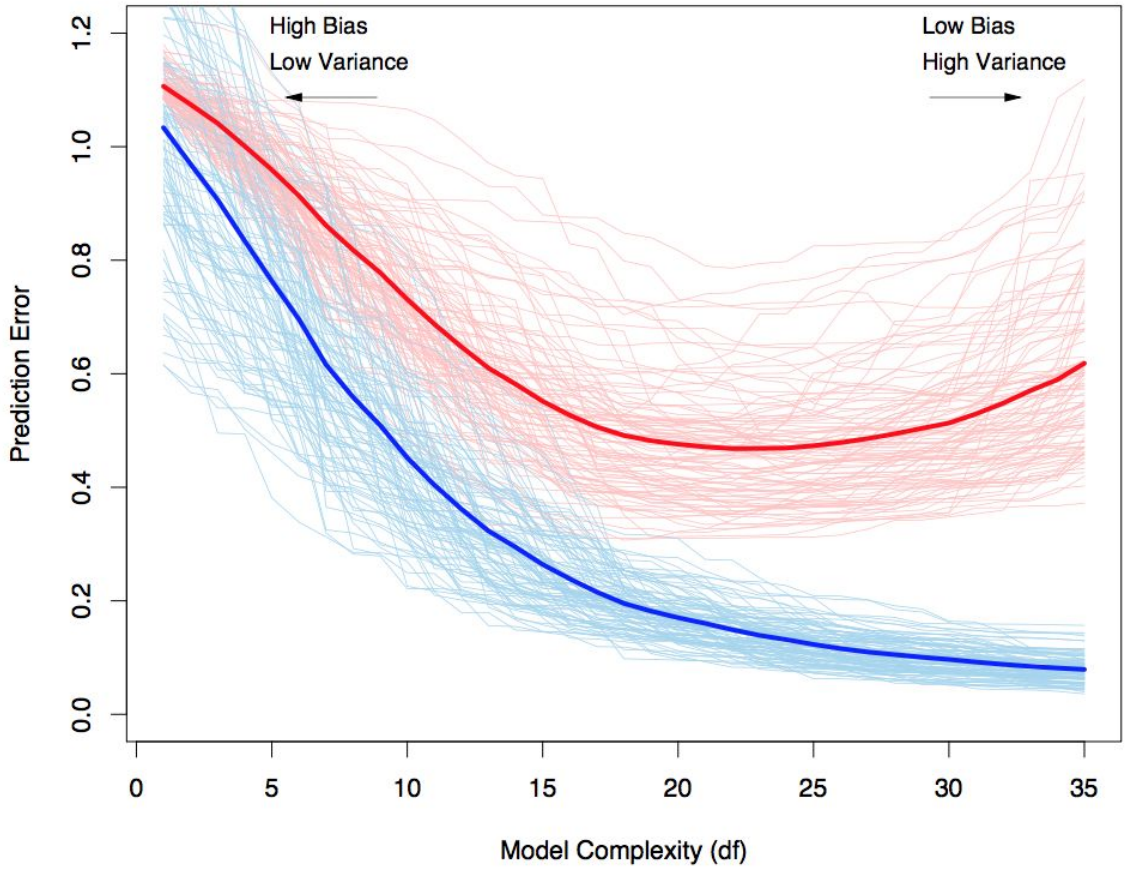
with

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1; \\ 0 & \text{otherwise.} \end{cases}$$

    - The above shows that we can apply a specially defined kernel to the classic method of KNN in order to more desirable properties, such as the smoothness of the decision boundary.
- Worth noting that locally weighted averages have bias problems at or near the boundaries of the domain.

Chapter 7 - Model Assessment and Selection

- The generalization performance of a learning model relates to its prediction capability on independent test data.
- Variance, bias, and model complexity are all related to the model's generalization error.



- Given how the test and train datasets are created, the training error may not be a good estimate of the test error.
- Most used method for estimating prediction error is cross validation.
- Bootstrapping is a training technique that you might consider using. Basically, let's say you have a model fit to a set of training data. The training set has N examples. The idea behind bootstrapping is that we randomly sample of dataset of N examples *with replacement* (that part is key) from that training set, and do this B times. Now, we have B training sets, each with N examples. Then, we refit the model to each of the bootstrap datasets, and examine the behaviour of the fits.
    - You can kind of estimate prediction error by summing up the loss of all the training examples across all of the B datasets. But the problem of overlapping data comes up, so idk why anyone would want to use this over K-fold CV.

Chapter 8 - Model Inference and Averaging

- Bagging is the process of averaging the predictions over a collection of bootstrap samples, which reduces variance.
    - For each bootstrap training set, we fit a model for each, and then take the average of the predictions.

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

- Committee methods train a bunch of different models and take a simple unweighted average of the predictions from each model.
    - You can also weight each of the models by looking at how well it fits the data and how many parameters it uses.
- Bumping is similar to bagging, where you draw bootstrap samples, and fit a model to each. Instead of averaging the predictions, we choose the model estimated from a sample that best fits the *original* training data.

$$\hat{b} = \arg\min_{b}\sum_{i=1}^{N}[y_i - \hat{f}^{*b}(x_i)]^2.$$

    - The thinking behind this approach is that we try to move the fitting procedure to good areas of the model space.

Chapter 9 - Additive Models, Trees, and Related Methods

- Tree based methods partition the feature space into a set of rectangles and then fit a simple model in each one.
- For regression trees, the algorithm needs to decide on the splitting variables, the split points, and the shape of the trees.
- A large tree might overfit the data, while a small tree might not capture the important structure. A major problem with trees is their high variance. Another problem is the lack of smoothness in the prediction surfaces, which especially hurts in the regression setting.
- Sensitivity is the probability of predicting disease given that the true state is disease.
- Specificity is the probability of predicting non disease given that the true state is non disease.
- Hierarchical mixture of experts procedure can be viewed as a variant of tree models where the tree doesn't split on hard decisions, but soft probabilistic ones.

Chapter 10 - Boosting and Additive Trees

- Boosting combines the outputs of many weak classifiers to produce a powerful committee. Boosting sequentially applies the weak classification algorithm to repeatedly modified versions of the data, producing a set of weak classifiers, and the predictions from all of them are combined through a weighted majority vote.

- The data modifications at each boosting step consist of applying weights to each of the training examples. The weights are initially set to 1/N so the first step trains the classifier on the data as usual. For very next iteration, the observation weights are modified and the classification algorithm is reapplied to the weighted observations. The observations misclassified by the previous classifier have their weights increased, and decreased for those that were correct.
- Basically, it's drawing more attention to the harder examples that the classifier is getting wrong.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Chapter 11 - Neural Networks

- Goal is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features.
- Neural networks are basically linear logistic regression models with hidden units and activation functions.
- Batch learning is when the parameter updates are a sum over all of the training cases.
- A training epoch is one sweep through the entire training set.
- Neural nets are hard to train because the model is generally overparameterized, and the optimization problem is nonconvex and unstable.
- Weight decay and early stopping are regularization methods to avoid overfitting.

Chapter 12 - Support Vector Machines and Flexible Discriminants

- SVMs produce nonlinear boundaries by constructing a linear boundary in a large and transformed version of the feature space.

Chapter 13 - Prototype Methods and Nearest-Neighbors

- K Means clustering is a method for finding clusters and cluster centers in a set of unlabeled data.

Didn’t feel like the rest was too helpful