Statistical Learning Notes

1. Ch. 2
   1. Reading
      1. In essence, statistical learning refers to a set of approaches for estimating f -> Prediction vs. Inference
         1. In financial services, interprability is important because they have to provide turn down reasons when someone is rejected
      2. The accuracy of Yˆ as a prediction for Y depends on two quantities, which we will call the reducible error and the irreducible error
         1. Y is also a function of , which, by definition, cannot be predicted using X
         2. No matter how well we estimate f, we cannot reduce the error introduced by
         3. The company is not interested in obtaining a deep understanding of the relationships between each individual predictor and the response; instead, the company simply wants an accurate model to predict the response using the predictors. (Like at Electrolux)
            1. For instance, what effect will changing the price of a product have on sales? This is an example of modeling for inference (But this also seems like it could have been Electrolux even though it’s inference)
      3. In other words, we want to find a function ˆf such that Y ≈ ˆf(X) for any observation (X, Y ). Broadly speaking, most statistical learning methods for this task can be characterized as either parametric or non-parametric
         1. The model-based approach just described is referred to as parametric; it reduces the problem of estimating f down to one of estimating a set of parameters
         2. Non-parametric methods do not make explicit assumptions about the functional form of f. Instead they seek an estimate of f that gets as close to the data points as possible without being too rough or wiggly
      4. For instance, when inference is the goal, the linear model may be a good choice since it will be quite easy to understand the relationship between Y and X1, X2,...,X
         1. The lasso, discussed in Chapter 6, relies upon the lasso linear model (2.4) but uses an alternative fitting procedure for estimating the coefficients β0, β1,...,βp.
            1. The new procedure is more restrictive in estimating the coefficients, and sets a number of them to exactly zero
         2. Generalized additive models (GAMs), discussed in Chapter 7, instead extend the lin- generalized additive model ear model (2.4) to allow for certain non-linear relationships
      5. Most statistical learning problems fall into one of two categories: supervised supervised or unsupervised
         1. Supervised: For each observation of the predictor measurement(s) xi, i = 1,...,n there is an associated response measurement yi.
            1. I have a target variable I want to predict from something else
         2. In contrast, unsupervised learning describes the somewhat more challenging situation in which for every observation i = 1,...,n, we observe a vector of measurements xi but no associated response y
            1. One statistical learning tool that we may use in this setting is cluster analysis, or clustering. The goal of cluster analysis is to ascertain, on the basis of x1,...,xn, whether the observations fall into relatively distinct groups
            2. Big in marketing analytics
            3. Anomaly and outlier detection
            4. Unsupervised learning helps with hypothesis generation
      6. We tend to refer to problems with a quantitative response as regression problems, while those involving a qualitative response are often referred to as classification problems (Logistic is exception)
      7. Measuring accuracy
         1. the most commonly-used measure is the mean squared error (MSE), given by MSE = 1 n n i=1 (yi − ˆf(xi))2, (2.5) 30 2. Statistical Learning where ˆf(xi) is the prediction that ˆf gives for the ith observation
            1. The MSE will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially
            2. A more restricted and hence smoother curve has fewer degrees of freedom than a wiggly curve—note that in Figure 2.9, linear regression is at the most restrictive end, with two degrees of freedom.

The training MSE declines monotonically as flexibility increases.

As model flexibility increases, training MSE will decrease, but the test MSE may not

When a given method yields a small training MSE but a large test MSE, we are said to be overfitting the data

* + - * 1. cross-validation (Chapter 5), which is a method for estimating test MSE using the training data.
    1. MSE: the variance of ˆf(x0), the squared bias of ˆf(x0) and the variance of the error variance bias terms
       1. In order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias
       2. However, if a method has high variance then small changes in the training data can result in large changes in ˆf
       3. As a general rule, as we use more flexible methods, the variance will increase and the bias will decrease
          1. If something is more flexible it can become more accurate
          2. Deep learning is much more accurate but harder to interpret
          3. Bias and variance is the same as underfitting and overfitting

Overfitting is when the model does well for some of the data but poorly for other data

Underfit means high bias

Fitting the noise and not the signal

You will have a low bias (Low SSE) in a flexible situation because it’s easy to get there when you’re fitting all points

If there are no quadratics or other polynomials, you’ll have a higher SSE, meaning higher bias

You’re not overfitting though

* + - 1. Good test set performance of a statistical learning method requires low variance as well as low squared bias
    1. It is possible to show (though the proof is outside of the scope of this book) that the test error rate given in (2.9) is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values
       1. Note that (2.10) is a conditional probability: it is the probability conditional that Y = j, given the observed predictor vector x0.
       2. This very simple clas- probability sifier is called the Bayes classifier
          1. The Bayes classifier produces the lowest possible test error rate, called the Bayes error rate
    2. In theory we would always like to predict qualitative responses using the Bayes classifier.
       1. But for real data, we do not know the conditional distribution of Y given X, and so computing the Bayes classifier is impossible
       2. Many approaches attempt to estimate the conditional distribution of Y given X, and then classify a given observation to the class with highest estimated probability.
          1. One such method is the K-nearest neighbors (KNN) classifier

When K = 1, the decision boundary is overly flexible and finds patterns in the data that don’t correspond to the Bayes decision boundary.

This corresponds to a classifier that has low bias but very high variance.

* 1. Curse of dimensionality
     1. The volume grows and the same amount of data covers less space
     2. As you go to higher and higher dimensions, the same amount of data takes up less space
     3. Why does distance to the nearest neighbor matter?
        1. It’s easier to make predictions
     4. When do we run into the problem of high dimensional data?
        1. Text mining
        2. Time series -> If you’re predicting a number of sets back to pick up the signal, there will be high dimensionality
        3. Image classification -> How do you define the dimensionality? By pixel
        4. Natural language processing
           1. Bag of words -> For each text document, I might care about which words follow other words
           2. We’ll have a word vector and every article will have a 0 or 1 for whether the word is in there or not
        5. Genetics
     5. High dimensional space is fine if you have lots of data to fill up the space
     6. How do we fight this curse?
        1. Subset features -> Maybe only some of the pixels carry meaningful information
        2. Feature learning models -> They learn not only prediction but also how to search through the space of high dimensional and low dimensional and come up with which ones are best for prediction

1. R is not part of their day to day ☹
   1. Where does linear regression fall in terms of the bias-variance trade off
      1. Homoskedasticity -> E(x1) = E
      2. Normal = Gaussian
      3. Standard error: Estimate of how much wiggle there is in the coefficients -> We estimate what the variance would be in terms of drawing coefficients (i.e. different samples) out of a hat
      4. Residual standard error -> Our guess of what the standard deviation on the error term is
         1. He says that residual standard error is kind of the absolute value and R2 is the percent
      5. TSS: E(yi – ybar)^2
         1. RSS:Eei2 = E(yi – yhati)
         2. R2 = TSS – RSS / TSS = 1 – RSS/TSS
      6. What does the F Statistic tell us?
         1. F-statistic is your hypothesis test for the model -> p-value that your coefficients are not equal to zero
      7. How to relate the standard error to the p-value?
         1. If confidence interval overlaps with 0, then you might not even need that variable
            1. If estimate was 1, but the standard error was 3, then it’s possible that the true value is 0
            2. What is the probability that the interval contains 0
   2. Bootstrapping estimate
      1. Select from a sample with replacement
      2. Approximate datasets coming from the same noise
   3. Two kinds of uncertainty on the output
      1. Irreducible error -> Error not reduced with more data
         1. RSE: As I get more and more data, will the spread change? No.
      2. Reducible error -> Error reduced with more data
         1. Confidence interval will get smaller
            1. More and more data means uncertainty in what the parameters are goes to 0
2. Generalization of Linear Regression
   1. Idea behind the names is that as we get more data, the error can be reduced (for the reducible error)
   2. What does he mean when he says additive?
      1. He means that he’s added terms for every predictor that he has in the dataset
   3. Multiple linear regression defines a hyperplane as opposed to just a line as in single variable linear regression
      1. So we have an intercept of sorts with multiple regressions it’s just within a hyperplane space so it doesn’t mean the same thing as it does with one variable
      2. With dummy variables, we need to do dummy encoding or one hot encoding
3. 1/30/19 – Other Considerations in the Reg Models
   1. We use the F-statistic to answer the question of whether the model overall has any predictive value -> (TSS - RSS)/p divided by RSS/(n – p -1)
   2. Deciding on important variables
      1. All subsets or best subsets regression -> Goal is to balance training error with model size
         1. There is an automatic way to do this (2 approaches)
            1. Forward selection: Start with a null that has no predictors and just the intercept (i.e. the mean of y); add var that has the lowest RSS to one predictor model; add other vars two that with two predictor model and choose the one that has lowest RSS; and so on
            2. Backward selection: Start from model with all vars in model and remove one at a time by removing the var that does the least damage to the model (remove the one with the least significance) (ME: ISN’T THIS P-HACKING?)
            3. There are other “optimal member” methods: Mallow’s Cp, Akaike Information Criterion (AIC), Bayesian Information Criteria (BIC), adjusted R2, and Cross-validation
   3. Qualitative predictors just describes categorical or factor variables
      1. Choice of baseline/reference category does not affect fit of the model
   4. Interaction effect in statistics is known as synergy effect in marketing
   5. Difference between KNN classified and KNN regression methods?
   6. Notes from in class
      1. When predicting, the coefficients all become hatted
      2. Design Matrix
         1. Separate target variable from the predictor variables
         2. Add a column where all values are 1 to the predictor variables separation
         3. The reason that you do this is so that your model is just Yhat = XBhat where X is a vector of the predictor variables and B is a vector or the different Betas (B0, B1, B2, etc.)
         4. Now it’s just matrix algebra (Well multiply Beta vector with X matrix)
            1. The “dot product” will be equal to one times beta zero plus 25 times beta one plus 60 times beta two
         5. This just really helps for a compact notation
      3. Best summary of how the model is doing is the residual sum or squares
         1. We need to find values of B0 and B1 that minimize loss function
            1. Find optimal beta hat that minimized sum of squares -> The solution of course is OLS
            2. Everything else you use numerical approximation, but with OLS, there is an analytical solution
      4. For an equation GPA = B0 + B1 \* Female + B2Age, might want to reevaluate what B) is
         1. How about for GPA = B0 + B1Age + B2Female + B3Age \* Female
            1. B0 = Average GPA for Male
            2. B1 = Effect on Age for Male
            3. B2 = Difference in Average GPA for Female and Male (i.e. difference in intercepts)
            4. B3 = Difference in effect of Age for Females Compared to Male (i.e. difference in slopes)
      5. Hierarchical principal – Term we use to describe keeping main effects in model when we do interactions
      6. Using P-values to select which variables are important as I increase the number of values that I am putting in the model
         1. What’s the probability of a false-positive (Type 1 error): Increasing the number of hypothesis tests you do increases the likelihood of false-positive (Multiple comparisons)
         2. Interpretation of p = .05 threshold requires caution then. Don’t just interpret p-value of predictors; also look at test statistic of whole model
         3. How to correct?
            1. Bonn-Feroni
            2. False Detection Rate correction
   7. What about outliers?
      1. When you have a data point that is very far away from everything else, we call it a high leverage point
         1. There is a lot of ability to drag your fit down
   8. Variance Inflation Factor -> How much extra variance with all these variables versus just this one
4. Autoregression
   1. I’m taking my time series window and kind of sliding it over – I use x at a and x at 2 to describe x at 3 to describing x at 4 with x at 2 and x at 3…
   2. Xt = B1Xt-1 + B2Xt-2 + B0 + E
   3. We’ve coaxed it into the model by thinking about regressors as being lagged versions of the variable
      1. Autoregression model of order k where it is the number of points you look backwards in time
5. An Aside: Data Science Interview Questions
   * 1. An important metric goes down, how would you dig into the causes?
     2. How do you remove the missing values from a data set. What if it causes bias? What will you do then?
     3. Design a metric that help reduce bias in the data set.
     4. How would you impute missing information in a dataset?
     5. Find the potential causes of an anomaly in web traffic dataset.
     6. Explain Logistic regression.
     7. Why are you interested in Airbnb?
     8. What metrics will you evaluate based on a scenario? — (e.g. Launch in new city)
     9. How can you report the statistical intensive results to a non-statistician group?
     10. Talk about your first interaction with Airbnb.
   1. Practice questions on Leetcode which has both SQL and traditional data structures/algorithm questions
      1. Review Brilliant for math and statistics questions.
      2. SQL Zoo and Mode Analytics both offer various SQL exercises you can solve in your browser.
   2. Before you start coding, read through all the questions. This allows your unconscious mind to start working on problems in the background.
      1. Start with the hardest problem first, when you hit a snag, move to the simpler problem before returning to the harder one.
      2. Focus on passing all the test cases first, then worry about improving complexity and readability.
      3. If you’re done and have a few minutes left, go get a drink and try to clear your head. Read through your solutions one last time, then submit.
      4. It’s okay to not finish a coding challenge. Sometimes companies will create unreasonably tedious coding challenges with one-week time limits that require 5–10 hours to complete. Unless you’re desperate, you can always walk away and spend your time preparing for the next interview.
   3. Flashcards are typically the best way to review machine learning theory which may come up at this stage. You can either make your own or purchase this set for $12. The Machine Learning Cheatsheet is also a good resource to review.
   4. Preparation:
      1. Practice take-home challenges which you can either purchase from datamasked or by looking at the answers without the questions on this Github repo.
      2. Brush up on libraries and tools that may help with your work. For example SpeedML or Tableau for rapid data visualization.
   5. When you’re negotiating, there are various levers you can pull.
      1. The three main ones are your base salary, stock options, and signing/relocation bonus.
      2. Generally speaking, signing/relocation is the easiest to negotiate, followed by stock options and then base salary
6. Ch. 4.7 – Classification and Logistic Regression
   1. Linear regression with a binary outcome can also be called linear discriminant analysis
   2. How to I change the model if we have a categorical Y (Classification)
      1. Problem with LPM with two is that you could get probabilities greater than one or less than zero
      2. There’s no ordering with true categorical variables, but the LPM would make 2 higher than 1 higher than 0, etc.
   3. So what transformation do I need for the Y?
      1. Should be between 0 and 1 always (-Inf, Inf) -> (0,1)
      2. Logistic function =
      3. We’re leveling off the curves at 0 and 1, which would happen when we input x numbers into the above
      4. Passing the output of the linear combination through that function
         1. P(X) = logistic(Xβ) =
            1. We might also here that we want to solve this for xβ
            2. We do log(odds)

Because if we used algebra we’d end up with that

* + - 1. How we’d say this is Bo is log odds of clas s1 vs class 2
      2. Beta 1 is change of log odds per unit change in X (A layman doesn’t know what log odds are, so you’d have to break that down further)
         1. Keegan: A linear change in X results in Beta 1 log odds change in Y
         2. If I exponentiate both sides, I have just odds, so I can say E to the beta1 change to the odds (That is the multiplicative effect)
  1. So how do we actually fit this thing?
     1. With linear, the loss function was the OLS
     2. With probit/logit, we use Maximum Likelihood, which says that if the output of the model is a probability over all possible outcomes, we need to ask which set of parameters give us the model most likely to generate the data
     3. Likelihood of the data depends on the parameters
        1. Iterative Least Squares….?
     4. Math
        1. For a linear regression, there is an analytic solution
        2. For logistic, there is not
        3. Fun aside: If I take the log of a product -> log(ab) = log(a) + log(b)
           1. Logs turn products into sums
           2. What’s the log of e to the something? The log and the e cancel out and it becomes 1
        4. Minimize the difference of the sum of squares to maximize…?
           1. He’s making the point that MLE as a solution is not different from OLS
     5. Y(P =1) = B0 + B1X1 + B2X2 + B3(X1 \* X2)
        1. What is the intercept with probit/logit?
           1. At age=0, this is the baseline probability of passing the class
           2. What does Beta 0 increasing mean, the baseline probability will get larger

The curve will then shift to the left

As an intercept moves, the curve shifts left and right

* + - 1. If I increase the Beta 1 from a 2 to the 6, what will change? The curve will get steeper and sharp
      2. B2 will change baseline probability for male vs female (the difference)
         1. So curve shifts left as we go down
      3. Interaction term allows you to change both the slope and the intercept
  1. Decision boundary -> What will separate the two classes
     1. What is a decision boundary and what is the model trying to learn?
     2. Shape of the decision boundary can have a lot of impact on the kind of data you use
        1. A decision boundary can be roughly linear

1. Video 1
   1. Logistic regression good enough when things are linearly separable (i.e. Neural Networks)
      1. But what about when we have a nonlinear decision boundary?
         1. Bullseye – Groups don’t overlap but certainly not linearly separable
   2. Biology Connection
      1. For any single neuron, it aggregates all the different signals
         1. All neurons that talk to it impinge upon its dendrites
      2. Similarly, we are aggregating inputs or doing matrix multiplication and addition
         1. Computes output in nonlinear function
         2. The weights between the inputs and the hidden layer signify how much the hidden layer should care about each input
      3. Visual processing stream – higher and higher up the stream you find neurons that codify more interesting things
         1. Jennifer Anniston Cells -> Neurons would only activate when they showed person picture of Jennifer Anniston
   3. Types
      1. Convolutional Neural Networks – Useful in image processing and image classification
         1. What is the probability that any one of these thousands of labels is the right one for this picture
      2. Recurrent Neural Networks – Most probable sequences and subsequences of characters in a text
         1. Helps us sequence fake sequences strikingly similar to real ones
      3. Natural Language Processing – Word2Vec – Learn about correlations in word usage and preserve relationships between words
         1. Can complete SAT style things like king is to prince as queen is to what?
   4. R community is a little bit behind when it comes to deep learning, so not a whole lot of support, but can use the Python tutorials
2. Video 2
   1. A graph with one color dots all in SW and one color dots all in NE can be a classic logistic regression classification problem -> log odds of red = -5. 6.3x + 8y
      1. E^y / 1 + e^y signifies nonlinear in logit analysis
      2. Logistic reg uses hyperplanes
   2. Input layer (x = n-dim input vector) -> Hidden Layer (p nodes) -> Output Layer (z) with weights between connections (B = p \* (n +1) of input to hidden and between connections (w = p + 1-dim) from hidden to output
      1. Deep Learning: More than one hidden layer
      2. More than one output layer: Vector output also possible
      3. Keep weights between layers fixed at 0: Convolutional networks
      4. If there is no hidden layer: Logistic Regression
      5. No hidden layer and play around with weights: Multivariate regression
   3. Feed forward operation
      1. Compute linear combination of each row with the input vector
   4. Universal Approximation Property – Approximate any continuous function with a free forward artificial network
      1. All you need is one hidden layer and a lot of nodes
3. Video 3
   1. Determining network topology
      1. How many parameters do we think we need?
      2. What are your computational capabilities?
   2. Compute feed forward pass and store a few terms
      1. You can also compute to gradients through back propagation
         1. This made computation of optimization terms much easier
   3. Tensorflow is Google’s Thing and I think it is related to neural networks
   4. We look at the training error to analyze how well it works
   5. You can use a network topology that is too small or not small enough, so optimization is hard
4. 2/13/19
   1. No need to read Ch. 5 stuff for 2/20
   2. Midterm through Neural Networks
   3. Clarification on Ch. 26: We need just a single classified
      1. The two vars are X1 and X2
         1. So we’d have a plot with X1 on X and X2 on Y
      2. There might be different ways to design features
         1. Avg. pixel intensity in top left or bottom right or something
         2. If we scatter against, we want small correlation (i.e. not a clear slope correlation)
      3. High variation just means that they scan a large range
      4. We’ll have picked two digits and we’ll want to build a dataframe around that and then fit the model
      5. If you use the same pixel
         1. Y ~ x
         2. Y ~x, + x2 (This model will always do better)
            1. On the training set it has to do better because you have mor information to work with
            2. If on the test set the other does better, then you are probably overfitting on the training set so that it doesn’t generalize well on the test set
            3. Can plot the ROC on the training curve to see that Y ~ x, + x2 will definitely do better
   4. Deep learning is a subset of AI AND machine learning
      1. Google’s recent Go playing algorithm combines interesting deep learning stuff WITH symbolic AI
      2. A subfield of AI IS machine learning (I was wrong)
      3. AI hardcodes the solutions and machine learning recognizes patterns to find solutions
   5. Reinforcement learning – No labels, but you have rewards
5. 2/27/19
   1. Error gets smaller and smaller as the model gets more complex.
   2. How do we choose the right model (i.e. how many hidden nodes) in the neural network?
      1. Pick the point where the U shape is at its lowest (Referencing the training versus test-set performance slide)
   3. Validation process
      1. One option is to split into two halves
         1. Downside is that half of the dataset is not going into training the model
      2. More common is to do 80-20 or 90-10
      3. If you have a chart with the mean of squared error on the Y and a degree of polynomial on the X, you can identify the right degree of the polynomial at the kink of the elbow (i.e. pick 2 because everything after is approximately as good as 2)
      4. What is the downside of the test-train approach?
         1. If you’re choosing rows at random, because of the sampling variability, you might get a quite different model if you do it again and again
            1. Seen and scored against different train and test, respectively
   4. K-fold cross validation
      1. Estimate of a test error that is averaged over ks
      2. As k goes higher, it should in theory be a better and better approximation
      3. We need leave one out cross-validation
      4. What is the downside of leave one out cross validation -> It takes a long time because you have to train a model for as many datapoints as you have
         1. In many cases, you can’t use it because of computational limitations
         2. 10-fold CV is the nice compromise
      5. How does variance of MSE change between test set and cross validation?
   5. Bootstrapping?
      1. How certain should I be of an estimate of something?
      2. Instead of holding out a piece of the data like in cross validation, we resample in this case
         1. If we hold out a small set, the stat might not be same
         2. We generate lots of fake datasets of same size through sampling with replacement
         3. The key is that we want to generate more datasets like our dataset, and we want them to have the same number of elements
            1. If I have N things and I want to sample N other things, and I do that without replacement, I’ll always get the same thing
            2. To make variation but have the same size, we sample without replacement to get other things
         4. Let’s generate a bunch of bootstrap samples and get a statistic on all of those
            1. If we keep getting the same thing, then we should be pretty certain
            2. If I get wildly different answers, then I conclude that the test statistic is a bit uncertain
         5. For example, we can compute a bunch of alpha values over and over again
            1. The fact that the means are different shouldn’t matter, but the spread is what matters
      3. When to do bootstrapping?
         1. We don’t need to do this for linear models because we know the exact answer for linear
         2. Sounds like you can bootstrap for everything though!
            1. For prediction, because you don’t care what your parameters are, maybe you care less
         3. Bootstrapping really only matters if you can’t get a direct SE/Confidence Interval
6. 3/13/19 -> Subset Selection
   1. Ways to do cross validation
      1. K-fold
         1. What happens when you make your k folds in k fold cross validation super small?
            1. LOOCV -> Leave one out cross validation
      2. Error vs. complexity
         1. Chart
            1. Error on Y
            2. Complexity on X
            3. Training error goes down as complexity goes up (Low bias)
            4. Test error will be underfitted with low complexity and overfit with high complex
            5. We have a line of complexity where the slops for the test slopes upward from being underfit to being overfit
         2. What can we do to make it more of less complex?
            1. To make the model less complicated, we can take out some of the predictors (Move complexity line left)
            2. To make the model more complicated, we can add in predictors (Move complexity right)
            3. This is what subset selection means
            4. For each category, you have a choice of including or not including (p = predictor), and so we end up with 2^p total number of options

These are independent

We can end up with a million by the time we get a p of 20

* + - 1. How can we organize all 2^p into the less complex and the more complex?
         1. Best subset selection

For k: 1 -> p

Fit all possible models with k predictor

Binomial coefficient (p choose k)

Select the one with the best residual sum of squares (RSS) on the training data

We’ll call this model k

Pick the Mk with the best RSS (test) as our model

* + - * 1. Why do we use train for inside the loop but test for outside of the loop for this algorithm?

We’re trying to make sure that we can overcome under or over fitting

So we always want the test data error to be on data we’ve never seen before

We can’t use it in the loop and outside of the loop because then it wouldn’t be true validation data because the models had seen it

We kind of let them overfit for all values of k and now we can get better idea of what complexity is

We should see the training error to go down and down and down as we add more predictors

RSS goes down as you add more linear predictors to a regression model (interesting)

* + - 1. Will be very computationally large for anything but a small p, so how do we get around this?
         1. Greedy algorithms -> I will assume that I am right even though I know that I am not

If I found the best for fitting with just one for a one parameter model, what would I do when I move to the second?

I would only consider the combinations that included that one

If X3 was the best in a one parameter regression, you could theoretically have X2 and X4 as a pair be better than X3 and something ->. But we assume this isn’t the case, and hence we are being geedy

He calls this forward subset selection (I assume this is the same thing as forward stepwise regression)

* + - * 1. For k:1 -> p

Fit all models with predictions from model Mk and one additional predictor

Select Mk as best RSS (train)

Select model Mk with best RSS (test)

As p gets bigger, the number of models that I have to fit: p, p – 1, p -2, … , 1

* + - 1. This is equivalent to
    1. Even though best subset selection is a good way to go, from a practical standpoint forward subset selection is a lot more practical
    2. Backward subset selection
       1. Start with all the predictors in the model and remove each one to see which makes the smallest when we remove each one
       2. Model complexity will be the same as forward subset selection
       3. But the two will differ if the number of data points we have is less than the number of predictors we have
          1. Causes a problem for both but in one case we can fix it

Have to have at least as many data points as you do predictors to get a well defined OLS solution

It is OK with forward selection, but we’d just have to stop when we’ve added in n parameters (i.e. in the case of n < p where n is the number of observations and p is the number of observations)

So we stop at k = n

* 1. We have to in some way hold out some data, so that once we find the best model for each k, we can give them a fair trial to see which do best
     1. What is the drawback to holding out a validation data set?
        1. High variance in your estimate of the test error
           1. One 20% will give you a little bit above the right answer but another 20% will give you above, and you have no way of knowing because you only got one shot
  2. Getting a good estimate of what the test error is
     1. Mallow’s C
        1. Take the RSS from your training fit and predict what the standard deviation of the Gaussian distribution is
        2. More and more parameters gives you a penalty you pay on top of a training error
        3. How much more error would I have on a hold out data set
        4. You are trying to estimate the gap between the train and test on chart from earlier
     2. Akaike Information Criterion
     3. Bayesian Information Criterion
        1. If you have more than a certain point, BIC will always be larger than Akaike
        2. What this means is that BIC will always pick a model with less parameters than Akaike
        3. Choose BIC if you are worried about overfitting, but can choose the other two if you’ve got a lot of data
     4. Adjusted R2 is more of a heuristic
  3. One rule of thumb is that you should go and find the data point with the lowest test error but also estimate some confidence intervals on it
     1. If you are doing k fold cross val, you can take mean of k errors or standard deviations and use that as error bar
     2. Use first from left where you are inside of one standard deviation of the lowest SD
     3. Give me an idea of what it would mean for these two things to be pretty close and let’s be conservative and choose the model that is the simplest and roughly equivalent to the one with the lowest error
     4. This is called the one standard error rule

1. 3-20-19
   1. Stepwise regression improves on the really complex and exhausting best subset (i.e. we enumerate every possible combination and look at error)
   2. Secret to every data science interview is 1) LASSO, but it seems Deep Learning is very important these days too
   3. We end up with larger coefficients in overfitting because the fit has to make larger changes to fit each data point as you move towards increasing complexity
   4. LASSO
      1. With OLS, the loss function is RSS, which is equal to , which boils down to f(X,Bvec)
      2. Now we add L = f(X, Bvec) + γg(Bvec) [Regularized regression or Penalized regression]
         1. **We’re trading off fitting the data and putting a penalty on the Beta vector that will be the final model**
         2. I add a new term to the loss function so that we have two functions:
            1. Minimize RSS
            2. Make sure coefficient vector isn’t crazy
         3. So find families of functions g that penalize very large coefficients
         4. If g is big, I incur a big penalty
      3. The difference between Lasso and Ridge Regression
         1. We need functions that tell us something about the magnitude of the whole vector, so a good class of vectors there is the lp norm of a vector
         2. Let’s suppose I have a vector v, I write the p norm of a vector v is equal to the sum over the elements of the vector
            1. Element j of the vector raised to the p power and the sum over j and the whole thing to 1/p power
         3. L1 and L2 Pnorms are what we’ll talk about today -> These are what measure how big your penalty is
            1. L2 Norm -> Square root of element of the vector squared

We essentially have the Pythagorean theorem -> a^2 + b^2 = c^2

C = sqrt(a^2 + b^2)

Euclidean distance of a vector is what we’re dealing with here

So if we have Vector = (3 2), then we get an L2 norm of sqrt(3^2 + 2^2) or (10 15) is sqrt(100 + 115^2)

So if the coefficient is larger, then we penalize the model based on that

* + - * 1. L1 Norm
      1. RSS + γ \* L2Norm
         1. So if we choose a lambda, or a scalar, of 5, for Vector = (10 15), we end up with 5 \* sqrt(100 + 225)
    1. Adjusted R^2 is an after the fact accounting for penalization, whereas Lasso and Ridge Regression is accounting for ahead of time as a means of feature selection

1. Primary and Dual forms -> All Constrained Optimizations can be stated in one or the other
   1. Let’s optimize subject to the constraint that all the coefficients have to live in a circle
2. With Ridge Regression, it’s about the L2 form of beta
   1. B0^2 + B1^2 = s2
   2. The above is just the equation for the area of a disk
   3. We are adding a new constraint with this. We’ve essentially got a red disk centered somewhere, and we ask what is the lowest place in the loss surface that is also in that red risk
   4. As lambda changes, the size of the disk shrinks
   5. Go to the red disk, and it will probably live on the edge of the red disk, and likely the edge of that risk that is closest to where the OLS optimal was
      1. So if the red disk overlaps the OLS answer, then the OLS is the answer
         1. Note: Edge of the Red Disk is another awesome statistics themed band name or at least album name for other possibilities previously considered (Margin of Terror, Tyranny of the Stars, Triple Diff, etc.)
      2. Mathematically, this just means knocking off the right side of the equation with lambda g vector (The penalty has been made so small that the red disk is large)
   6. As lambda goes to infinity, the disk gets smaller, and the estimate of the coefficients
      1. It’ll go towards the origin (Center of our constraint region)
      2. So even though the optimal will be yearning for the OLS point, as it’s getting smaller, it’s being pulled towards the constraint region center origin point
      3. As the constraint region gets smaller, the beta vector goes to the origin
      4. So the model fit will be one where we have a lot of really small coefficients
      5. This is why it is called shrinkage
         1. Trajectory plot shows this in action
         2. As lambda gets large, all coefficients go to 0
   7. As lambda goes to zero, the optimization becomes the OLS Loss function as we’re only solving RSS
   8. Position of disk never changes -> It is always at 0 and OLS point is always at minimal loss function for OLS
   9. For feature selection, ridge regression DOES NOT DO ANYTHING
      1. This is important
      2. The coefficients are never exactly 0 and so we’re regularizing the solution
      3. Helps with overfitting but not with feature selection
      4. At every point along the way, all features are still in the model; they just have different coefficients
   10. Ridge regression is useful when you might have more predictors than observations
       1. Allows you to underfit
3. With Lasso, it’s about the L1 loss function
   1. Lasso is the same game as ridge except that the constrain shape is different
   2. For feature selection, lasso DOES SOMETHING
      1. The sharp corners of the “L1 ball” matter because as it shrinks down it has an impact on what the best estimate of the coefficient is
      2. It’s much more likely that the contours of the loss surface intersect with the constrain region in a place that’s a corner (i.e. on the axis of B2 but not B1 and B1 is becoming 0 here exactly)
      3. As soon as it hits a corner and one of the coefficients becomes 0, it never comes back
   3. When penalty is very small, the solution is also the OLS (Like in Ridge)
   4. When the penalty is very big, the coefficients shrink to 0 (In Ridge they just shrink to being really small)
      1. So some of the features drop out briefly and the ones left are those most powerful for predicting the target variable
      2. As it reaches towards infinity, we have an empty model
      3. As it reaches towards 0, we have the OLS
   5. So we know which features to keep now

\* My question: Why do this when you can just do PCA? (He’ll probably eventually answer)

PCA does not get rid of features

1. Where is the absolute magnitude or scale coming in when we are fitting here?
   1. When doing ridge and lasso, it’s important to standardize all of the predictors!
   2. Rescore everything so that it’s centered and have roughly the same variation
2. Corrections for Midterm
   1. 1d: Parameter confidence intervals for artificial neural network and why using bootstrapping (sampling without replacement) would not be a good approach
      1. If you bootstrap, you’re fitting a neural network over and over again
      2. With bootstrapping, we want a distribution that is essentially Gaussian
      3. Problem with neural network is that you have a degeneracy
         1. I can have exact same by just naming things differently
         2. Now the thing that looks like 311 has become w12 and the connection is somewhere else
      4. Neural network might find a solution but it’s just changing labels
         1. You have no way of knowing how it’s relabeled
   2. 2c: What will happen to RSE when a predictor is dropped
      1. Any time we add variables to a model, RSS goes down no matter what
      2. When we remove the variable, the RSS will go up
      3. But this question was asking about RSE!
         1. This is the thing that estimates the noise term
         2. This is that last e term in our linear regression model
         3. The formula for this is RSE = sqrt(RSS/n – p – 1)
            1. So when we drop a predictor, RSS goes up but p (# of predictors) is going down
            2. We don’t know which goes up faster or slower so we’re not sure here
   3. 3c: Professors didn’t even know what was happening here, so they didn’t take off
   4. 5b: You could have multiple solutions that give you same input output
      1. We’re talking about fitting just one model so even if you choose the same hyper parameters (hidden units, activation, interaction, etc.), you can get two very different models that have same performance
   5. 5e: False: it does not become a neural network model
      1. You want to focus on non-linearity
      2. It is true that one of the hidden units can learn to do an interaction of two of your inputs
      3. But what you’ll never get is the non-linearities on the hidden units so the activation function is the key here
3. 3/27/19
   1. Y = B0 + B1\*X1 + B2\*X2 + …
      1. We can still use all of our fitting tricks when we make the Xs non-lonear and keep the Betas linear
         1. Polynomial regression
            1. Statistical learning all boils down to that bias-variance thing

The test-train difference is kind of what gets you at the variance part

* + - * 1. If I have a polynomial of degree k, what quantity does it relate to in terms of a picture of it?

K-1

So a polynomial of degree 2 is 1 turning point (parabola)

So a polynomial of degree 5 is 4 turning points or wiggles

* + - 1. Step function or piecewise constant
         1. For every X value that falls inside of this range, I’m always going to make the same prediction
         2. Inside of each bin, I’d probably try to find the average

All data points where X is in that bin and then take average Y value across all of them

* + - * 1. We write this in a way that still makes coefficients linear

Which bucket you fall into then ends up being a category and then make dummies from there

Edges for your bins delineate

Y = b0 + b1 \* I(c1 < x < c2) + b2 \* I(c2 < x < c2) … + bk\*I(c(k-1) < x < ck)

I(c1 < x < c2) = 1 (if true), 0 (if false)

* + - * 1. Can fit in same way, but design matrix will look slightly different
      1. What is the complexity setting for piecewise constant? How do I determine how flexible my model is?
         1. In polynomial, we choose how many we add in, what about with piecewise constant?
         2. How do I choose the edges to make the model more complex?

More cs, more edges means the more flexible my model is

* + - * 1. Let’s say I put my edges at 0, 5, 10, 15, and 20 (i..e. 4 bins)

I need to then take my original X and augment my design matrix with indicator variables based on the bin its in (1-4)

X | I1 | I2 | I3 | I4  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
1 1 0 0 0  
7 0 1 0 0  
13 0 0 1 0   
8 0 1 0 0  
2 1 0 1 0

* + - * 1. Then I could essentially delete that first column, unlike in polynomial where I need the Xs
        2. What if the bins were somehow not the same size in terms of edges?

Nothing bad really would happen

But most of the time there really isn’t a reason to do that

Most of the time, the standard is to pick your buckets to be even

You might want uneven buckets if your data is also unevenly distributed

* + - * 1. The idea of computing some function on your original data is called “basis functions”

X | f1(x) | f2(x) | f3(x)

They are the basis for adding non-linear into your model

X | sin(x) | cos (x) | sin (2\*x) | cos(2\*x)

These wiggle even faster

These are called “fourier basis functions”

“Wavelet basis functions”

The idea of basis functions is bigger than data science

I want to stay as close to linearity as possible, but I use basis functions to build the non-linearity in

* 1. Regression splines
     1. Versus piecewise, we can fit polynomials inside of each of the splines if we want
        1. Piecewise just allows horizontal in between the edges
     2. As X goes to infinity, the polynomial goes to infinity
     3. Motivation behind splines is that we want the flexibility of polynomials but the stability of piecewise constant
     4. How to do it
        1. Use the bins from piecewise
        2. Do polynomial fit inside of each bin
     5. What are the problems with regression splines though?
        1. N\*k coefficients can get pretty big pretty fast (We use cross-validation to choose the right number of bins, but in the end we might end up having thousands of parameters to fit)
        2. Discontinuity (Can get this with both piecewise and spline)
     6. To get over some of the problems above, I can add in a constraint
        1. Left: B00 + B01x + B02^2
        2. Break Point
        3. Right: B10 + B11X + B12
        4. To get them to be continuous, I’d want them to have the same Y value at the break point
           1. B00 + B01\*c + B02c^2 = B10 + B11X + B12x^2
        5. To get them to be smooth, I’d want them to have the same derivative as well
           1. If I want the first derivatives to be the same, I can get the smoothing
           2. If I want the rate of change of the rate of change to be the same (second derivative), I can do that too)
           3. But what people usually do is stop once they’ve gotten smoothness

It’s not a big deal at that point if the second derivative changes abruptly or something

* 1. In the spline world, we call the breaks knots
     1. Where do we place the knots? People will often say there are two common ways to do it
        1. Uniformly
        2. Percentiles (of x)
     2. Beta one will tell you how high you should scale the first spline
     3. Final prediction is the sum of all of those scale basis functions
  2. For splines
     1. What would we use to set the complexity of our spline model
        1. More bins (knots) or higher degree of polynomials inside of each bin
        2. In practice, people will fix the degree of the polynomial and change the number of knots just because the cross-validation ends up being hard with two knobs to turn

1. Smoothing splines
   1. With regression splines, there is a question about what you do with the ends
      1. Idea of regular splines vs natural splines
   2. The other way to do splines is to not have to pick the number of knots but instead go to the most extreme case (one knot per data point) and connect the dots effectively
      1. If I pick first order polynomial, you connect with straight lines, second with parabolas, and so on
      2. Clearly we’re overfitting like crazy, and so what we do to fix this is impose a penalty
         1. G is the function that represents the spline prediction
         2. If we just used a regression spline, it would always be 0
         3. The penalty looks like the derivative of our spline averaged over all X values (Integral function)
            1. The first derivative tells me the slope (how fast the function is changing)
            2. The second derivative here tells me the curvature (how fast I’m going from changing from going up to down)
   3. You can’t write this is just a basis function expansion
2. Generalized Additive Models
   1. What we’ve been doing is about one variable input and one variable output
   2. What if we do have something where we want to model y = B0 + B1\*X1 + B2\*X2?
      1. The idea is a model called a general additive model (GAM)
      2. The idea is pretty straight forward but also pretty powerful
         1. For each of my predictors, I’m going to choose one of these methods that we’ve talked about and handle non-linearity in that particular variable
         2. So for variable one, I’ll do splines, and maybe for variable two, I’ll do splines again, and then for X3, maybe I choose polynomial
      3. The huge assumption this model makes is that I’ll handle the non-linearities in each and then just sum up the effects
         1. You run into problems of model identifiabaility with this
         2. Big assumption, but makes model fittable and tractable
         3. But they do produce really nice interprable results
3. 4/3/19
   1. Brief talk about jobs
      1. Data Science Challenge -> Not just what you know but how you can talk about it
   2. Tree-based methods
      1. In order to make a prediction for a given
      2. observation, we typically use the mean or the mode of the training observations in the region to which it belongs.
      3. However, they typically are not competitive with the best supervised learning approaches, such as those seen in Chapters 6 and 7, in terms of prediction accuracy. Hence in this chapter we also introduce bagging, random forests, and boosting
   3. Decision Trees
      1. Recursive binary splitting makes decision trees surprisingly simple and quick to learn
      2. Looking at the tree, we see that a split only occurs on one branch and won’t affect another
         1. We see this on the feature space, where a boundary stops in a certain place
      3. Our prediction for salary then will be the average salary for each box (not across)
         1. What is the mean of this many hits and this many years, will it’s the average for whichever box that aligns with
      4. If we look at the decision boundary with logistic regression, it is going to be a straight line
      5. How do we choose where to make the splits?
         1. Look for the right place to make the split by looking at a continuum
         2. We want to quantify how much of an impact by placing the split in a place
            1. RSS is the sum over the difference between actual y for each data point and our prediction y hat for each data point
            2. When we had no split at all, the RSS is just a baseline

When we choose an arbitrary place to make a split, we calculate the rss on two piecewise constant functions

* + - * 1. So we do this at every possible place and calculate the change at each place we’ve used and then we go back and find where the rss is smallest
        2. we first select the predictor Xj and the cutpoint s such that splitting the predictor space into the regions {X|Xj < s} and {X|Xj ≥ s} leads to the greatest possible
        3. reduction in RSS
      1. The leaves are just piecewise constant functions split on these boundaries
         1. The boundaries and regions of x relative to what is happening in y -> When it changes more rapidly, you’ll have smaller ranges (i.e. more splits will be needed and can’t have a single mean over a wide region)
    1. When we have multiple variables, we decide to make the splits based on
       1. The split will always only describe lines, so there are a lot of shapes you simply can’t make
       2. Picking individual variables based on the RSS as comparison to the RSS change of others with the splits
       3. Bakes in feature importance
          1. The one at the top that splits first will be the most important feature
       4. How to handle categorical variables in a decision tree -> Dummy for two levels
          1. Groupings within the category level -> Republicans on the left and everything else on the right (For three levels)
    2. In summary, roughly speaking, there are two steps:
       1. We divide the predictor space—that is, the set of possible values for X1, X2,...,Xp—into J distinct and non-overlapping regions, R1, R2,...,RJ .
       2. For every observation that falls into the region Rj , we make the same prediction, which is simply the mean of the response values for the training observations in Rj .

1. Bagging – Bootstrap Aggregation
   1. First, I bootstrap and get bootstrap replicates
   2. Then I fit a decision tree uniquely on each one
   3. Aggregate final leaf predictions
      1. How do we predict the final category for the leaf? We go down to the leaf node of the tree and find the data that belongs to that leaf and whatever class is the majority will be our prediction for that leaf
      2. I cant take a new data point and run it through each tree and then I average over the ensemble over the trees
      3. The idea is that each tree is individually weak and not a good idea in that it can be overfit
      4. All trees here will be overfit in a slightly different way, so they kind of average out each other’s mistakes
      5. Ensemble of weak learners
      6. Ensemble of simple models can be surprisingly powerful
   4. No one really ever uses it because random forest is good and GBM is even better
2. Random Forest
   1. We have a subset of the features that we choose
   2. I’ll find a subset of k – p predictors that I think about for each node
   3. As I try to grow the tree, I’m only looking at small patches of the data set, which introduces a form of regularization
      1. The overall model ends up more able to pick up on signal and not noise
      2. Not only is this for one tree, but it’s for 10,000 trees
         1. They each go through the same process
      3. Random in two ways – bootstrap replicates and subsetting features that you can split on
3. Boosting (GBM)
   1. Weak learners
   2. Learning on errors
   3. What do we do
      1. Make a simple tree (stump)
         1. I have a y hat for each place on the split
      2. Make predictions on training set
         1. Run training set through the above model and see if you got anything wrong or right
      3. Error set
         1. Grab the set you got wrong and pretend that it’s your only training set
      4. Back to step 1 with error set
      5. T1 + T2 + T3 + … T
      6. Really fast to implement and very simple
      7. GBM will do better than random forest a lot of the time
   4. Ensemble methods boil down to going down through each tree
      1. Some variation happens in all of these different methods – Node purity gets at how do I predict what class should be in a leaf (i.e. what is the majority in that group)
         1. If all are same inn the leaf, then you’ve done a good job
4. 4/10/19
   1. SVM
      1. Find the beta coefficients that create a separating hyperplane
      2. But there are multiple hyperplanes that can solve a problem perfectly in terms of data points that are on the wrong side of the line relative to the class they are in
         1. But you don’t want the one that is too close to the points
         2. We want to separate the classes and be as far as possible from the data points
            1. Why is it bad for the lines to be close to the lines?

Overfitting and bias-variance tradeoff

Overfitting because it is so close that if we happen to see another one, the model might have to shift drastically

Max margin classifier -> Point is to minimize overfitting

* + - 1. What is the black line in between the red and blue?
         1. The decision boundary (i.e. location where we would switch from red to blue, or the center of the max margin classifier)
         2. The distance from the decision boundary out to the margin is defined by maximum distance to points of the data set
         3. What is the point of support vector points?

Points in the dataset that define the width of the margin that can be maximized

* + - 1. The way to find the “grey one” is to go from any given line and calculate its margin
         1. Margin is the distance from the potential hyperplane to every point and then it is the smallest one (smallest distance between hyperplane and any point in the training data)
         2. Margin smaller if it is right next to a point
         3. The margin tells us how to find the best hyperplane
         4. You only need three points to define a line and a margin
    1. What is the major difference between the max margin classifier and the support vector classifier?
       1. First, what can it not be used for?
          1. Definitely can’t be used for the bullseye problem
       2. Also not useful when the classes overlap UNLESS we can accommodate for it
          1. We therefore need to allow our classifier to make mistakes

We do this through tampering with cost

Find a separating hyperplane, find the one that maximizes the margin, but also makes the fewest mistakes given the tolerance we’ve set -> This is our optimization

Your budget is 5, so the model is allowed to make 5 mistakes (Not actually this literal of an interpretation though)

If I allow it to make 50 mistakes, then the margin will be much bigger, but might be better in terms of preventing overfitting

Cost = 1 for example with big decision margins -> If it’s on the wrong side but still within the decision boundary, then it is not so bad

If we do cost = 10, we are saying it costs 10 times as much and we end up with a smaller margin compared to cost = 1

A smaller margin means overfitting risk

* + - * 1. How to pick right value of the hyper parameter

We can use cross validation

We hold out some data and pretend we have unseen data and do a sweep over the hyper parameter and then ask which one seems to do the best on data it’s never seen before

By explicitly allowing it to make mistakes, it will have a more generous boundary which hopefully will be more consistent with the signal in the data and not the noise in the data

* 1. What is limiting about all of the decision boundaries drawn over and over again?
     1. A straight line often won’t be the best solution
     2. If we want to be a bit more fancy, we move to support vector machine (versus support vector classifiers that we talked about above)
  2. Fitting the hyperplane
     1. We are still trying to fit betas, but the goal here is to maximize some number M, which is the margin (i.e. how far above and below are we allowed to make mistakes)
     2. We don’t use 0 and 1 but +1 and -1 for classes with SVM
     3. If you’re correct, you’ll always get a positive number (++ or --)
        1. The hope is that the product of the two numbers is bigger than the margin
        2. Cost is how much it violated the margin by
           1. Things on correct side of the boundary get an epsilon of 0
           2. Multiply how bad you’re doing by the margin and this the idea of the cost

You can only have so many of these epsilons

* 1. Kernel trick
     1. When you’re solving the problem, it turns out, instead of writing it as an optimization problem with a bunch of constraints, we can say subject to equalities or inequalities
     2. Whenever you’re given a constrained optimization problem, there is some equivalent unconstrained optimization problem
     3. But you might end up with more variables to optimize over then you started with
        1. The upshot is that when you do this procedure for SVMs and write out the unconstrained loss function, you find that it depends on your data in a really interesting way
        2. In the loss function, your data only ever shows up in what are called inner products
        3. Angle brackets on a vector space -> Inner product
           1. In linear algebra you can play around with all sorts of linear products
           2. A kernel is a function that takes two functions and spits out a scalar value
           3. Instead of dot product, we compute distance between vectors

Inner product of 0 vector has to be 0

* + - * 1. Main idea is that when you do the math for SVMs, you recognize that your data comes in pairs and you do inner products

We do classification in non-linear space and then go back to linear space

* + 1. If we use radial basis function as kernel instead, we end up with different looking kernel and a non-linear boundary
       1. We’ve warped the space
       2. Things are similar to each other when they are close in Euclidian space
          1. If things are red, things nearby will also be red
    2. There is also the polynomial kernel
       1. The implicit new feature space you’re working in is comprised of polynomial functions of your original
       2. Still based on the standard kernel but we’ve added a 1 and raised it to a dimension
    3. What do we do when it is more than 2 classes? (i.e. not binary)
       1. Take one class as -1 and then everything else is +1
          1. Also called 1 vs all
          2. What do I do with the three models that each represent a one vs all for each respective reference category?

What is the output of each model and find the one where it is the highest

* + - 1. How could you build multiple classifiers?
         1. One-vs-one
         2. We’ll have to fit 4 with 6 for example (K choose 2)

Grows second order

* + - * 1. How do I take all my models and figure out what my final prediction is?

We are comparing pairwise, so if I take the one that has the biggest evidence for it won’t work because it will not tell me how it interacts with other classes

Take the average of these three for class 1 and then these three for class 2, etc.

So we take argmax over k of the average output between j and k

* + 1. Complexity vs Simplicity or Bias variance tradeoff
       1. Gamma acts as a complexity tuner
       2. Pumping up gamma makes our model more flexible
       3. We end up predicting red just in the little separate areas of red
       4. Blob gets bigger and more spherical when we go to 1 for gamma
       5. So we can use cost OR gamma to tune model complexity
  1. Are there types of problems where you immediately go to SVM versus RF, Decision Tree, Logistic Regression? Is it just based on accuracy (i.e. AUC or F1) or can you usually decipher after looking at a plot?

1. 4/17/19 – Clustering
   1. In unsupervised learning, we’re missing the labels
      1. We just want to know something about the structure of the data in general
      2. What do we mean when we say “find the structure”
         1. Well all sorts of different algorithms and approaches make some kind of assumption about the structure
         2. If this type of structure is in the data, would you find it for me?
   2. Two types of structures that we’ll look at
      1. Clustering
      2. Dimensionality reduction
   3. What assumption are we making when we talk about clustering?
      1. When we talk about clustering, we are talking about separable, similarity
         1. Separable (clusters) or Similarity within groups
         2. We don’t talk about separable pieces of data, but about the small number of continuous variables that can describe the data
            1. With clusters, there is something discrete (i.e. in these groups)
            2. With DR, we don’t say it’s broken up into chunks, but rather that there are continuous features that could describe the data
   4. K-means algorithm
      1. Assumption:
         1. # of clusters: k
         2. Cluster described by the mean of all data points in that cluster
      2. K means is based on the idea that if we already knew which cluster they belong in, then the mean will likely fall in the middle of the scatter clusters
         1. Each datapoint will be closer to the mean of its cluster than it will be to the mean of any other cluster, in general
            1. This point above is the main intuition behind this algorithm
      3. When we actually have real data though, we don’t know which cluster each of the data points belongs to, and we therefore don’t know where the means are – what do we do then?
         1. Start with a guess and iteratively figure out a way to make my guesses better and better and better
         2. Two step process for k means
            1. 1. Guess the k means randomly
            2. 2. Iterate

Assume the means are correct

Determine cluster membership

Assume the cluster memberships are correct

Determine the means

Stop when minimal change in means

* + - 1. More about the guessing and iterating
         1. Maybe make two random guesses

Essentially split a line down the midpoint of the line segment connecting the two points

With two initial guesses, I would say that all the data points on one side belong to one cluster and then all of the other data points belong to the other cluster (Will usually be wrong at first)

* + - * 1. Every time we iterate, means get closer and closer to true mean that classifies right s
        2. Expectation Maximization (EM) algorithm: I’m going to assume that I know the answer to my problem and then I’ll estimate the parameters (means) and then go back the other way and assume that I know the parameters and estimate the unknowns

Not really a set algorithm as much as it is a set approach

* + - 1. What are the drawbacks to this model?
         1. Shapes

We’d expect it to work well when data comes in spherically symmetric blobs of data

Because the data point needs to be closer to the mean of its cluster than to another one

Will work better on concave

We’d expect it to not work well on the swirl, two half moons, swiss roll for example

Will work less well on convex

If we got big blob with tons of data points and then a small blob off to the side, then k means will get it close to right, but not exactly right

* + - * 1. Outliers

If there is an outlier, K-means will try to stick it in some cluster, thereby distorting the mean of that cluster

But if your data is enough, it won’t be a big deal

* + - * 1. There’s no real good way ahead of time to know what the best value of k is

This is a problem with a lot of unsupervised learning in general in that there is no nice signal to know when we are doing good or when we are doing poorly

* 1. Challenges in Clustering
     1. Outliers
     2. Unbalanced Clusters
     3. Non-spherical clusters
     4. Variable density
     5. How many clusters
  2. DBSCAN Algorithm (More advanced than k-means when you think about it)
     1. Density-Based Spatial Clustering of Applications with Noise
     2. What does it do?
        1. Uses the assumption that within a cluster density of datapoints should be high and fall off between clusters
           1. Already a bit more nuanced than k-means
           2. DBSCAN says OK, well when you’re inside of a cluster
     3. DBSCAN handles the half moon by saying that what really matters is that there is a continuous region of high density data points
     4. DBSCAN
        1. 1. Provide hyperparameters
           1. Epsilon: What is the usual distance between things in the same cluster (“typical” distance between points)

What it means to look around a point

* + - * 1. Minimum number of points: If I look within a distance epsilon of some point, how do I tell if it’s in a cluster

Number of other points to expect when looking around a point in a cluster

What it means to say there are a lot of points

* + 1. Steps for DBSCAN
       1. 1. Go to a data point and look in an area epsilon around that data point and count them up and check to see if there is more than min points or not
          1. Every data point that meets the requirements is called a core data point
          2. The core points end up being the ones on the inside of your cluster
       2. 2. Draw a link between core data points (they’ll be within epsilon of each other)s
       3. 3. Connect to reachable data points (within epsilon of the core data points)
    2. What data might this not do well on?
       1. If you have the half moon situation, but one of the crescents is highly dense and the other is not
          1. HDBSCAN, which is an extension of DBSCAN tries to solve this
       2. DBSCANS a bit more interpretable because you don’t need to know k in the same way
  1. Hierarchical Clustering (Tree-based clustering)
     1. Intuition
        1. If two points belong to the same cluster, they ought to be close to each other
        2. Map of closeness at all scales
     2. What would it take for everything to look similar?
     3. Bottom up clustering (agglomerative clustering)
        1. We use tree representation to map out how similar data points are
        2. We start at bottom of the tree and ask what if every data point were its own cluster?
           1. Then I’ll find two points/clusters that are close to each other and decide they should belong together
           2. OK, I’ll then ask it again
           3. Measuring the distance between single elements and sets or between two sets: This is the choice we have to make

Linkage function

* + - 1. The highest level of the cluster is obviously just all of the data
    1. How to compare sets and know when to join things? We use linkage functions
       1. Complete
          1. What is the largest dissimilarity?

We want to retain clusters that are as different as possible for as long as possible

* + - 1. Single
         1. For any two sets, what’s the smallest possible distance between any two possible members of either set?
         2. In figure 10,11 on bottom right, with green set and blue set, which pair of elements gives smallest single linkage between two sets? 8 and 1
         3. Tends to create generous clusters that are long and odd shaped
         4. Linkage, which defines the dissimilarity between two groups of observations
         5. Minimal intercluster dissimilarity.

Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities.

Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time

* + - 1. Average
      2. Centroid
    1. Challenges in Clustering as it related to hierarchical clustering
       1. Outliers
          1. Seems like better
       2. Unbalanced Clusters
          1. Probably better
       3. Non-spherical clusters
          1. Similar to DBSCAN in that I’m looking about how close I need to go to find a neighbor and I don’t care as much about shape
       4. Variable density
       5. How many clusters?
          1. Cut the tree at the appropriate height if interested in two clusterss
  1. Topic Modeling (Comes from National Language Processing Community)
     1. Latent Dirichlet Allocation
     2. Let’s say I have a bunch of news articles and I want to know what the articles are about (i.e. articles similarity to each other)
     3. Words > Documents > Corpus
        1. Topics
           1. Docs > Topics
     4. Difficulty is that you have unstructured non-numeric thing
        1. Can use wordcount
        2. Steps
           1. Bag of Words
           2. The vector matrix we create is called a document term matrix
     5. The idea of topic modeling is that we have a probability vector over the vocabulary
        1. High probability of words based on the topic
     6. What we need to do
        1. Doc Topic Matrix
        2. Topic Term Matrix
        3. Expectation maximization idea -> Estimate one and pretend you know the other and then flip
     7. How you fit this?

1. 4/24/19 – Principal Component Analysis
   1. With PCA, we are trying to find directions in the feature space that explain the greatest variance overall
   2. If we have a scattering of dots that point upwards kind of at a slope of 1, then a line sloping upwards at a slope of 1 will be the first principal component analysis
      1. Often just by eye we can see this
      2. The second one will be orthogonal to the first (Perpendicular)
      3. With an infinite number, we just iteratively look at what causes the most variance
   3. My question: Is there a natural lower bond of explained variance that would cause us to leave components out of our model?
   4. If I have a feature space of 10,000, and I do a PCA, how many components do I end up with?
      1. 10,000: When you do PCA, you end up with the same number of principal components of the features you measured
         1. All you are doing is a rotation to a better coordination system
         2. You’ll always have the same number as you do features, but they are not all created equally
         3. The point is that if the data lives on a low dimensional subspace, you do not need to leave them all around
      2. Can use PC plot to assess how much variance is explained as we add principal components
         1. We can therefore quantify by deciding how much to keep
         2. So the trajectory always goes downward (or upward if using cumulative at increasingly less steep rates)
      3. Real world examples: Looking at 100,000 cells from an animal and 50,000 genes that the animal can have
         1. Jason said they go until 95% of variance, which got them to 50-100 PCs
         2. But leaving in the component that gets you from 94.9 to 95 and leaving out the 95 to 95.1 seems relatively arbitrary
         3. Is the 95% relatively arbitrary?
         4. The alternative is to retain components above 1 (Kaiser rule)
         5. Do I have this somewhere in Stata or R from Electrolux work?
      4. When we assess what is the first PC, we ask where does X1 project into others
         1. Much more extension into second principal component means more concern with second principal
         2. In contrast, the first principal component seems to be highly correlated with rape, assault, and murder
            1. There is a direction in four dimensional space that is equal weight of the three prime numbers that explains variability from state to state
            2. Second principal captures what is left over, which is urban population in this case
      5. PCA package in R first calculates the mean vector and then subtracts that off of everything
      6. In practice
         1. I first create a covariance matrix that tells me how much each covaries with eachother (X transpose X)
            1. The eigenvectors and eigenvalues of that are the loadings onto the principal components (Orthodonal diagnonal of that square matrix)
         2. Projections of the features into the original components
         3. Whichever extends heavily along the axis of PC1 is the variable that most likely explains the most variance
            1. PC2 is a weighted combination of the two extended into that space (So how do we still have the same amount of PCs then?)
      7. What do you get from R output?
         1. Rotation tells you your vectors (i.e. coordinated of the principal)
      8. The 95% or 1% thing
         1. If your cumulative plot builds up slowly, then that might tell you PCA is not useful anyways
         2. Also look at the downstream algorithm if it can’t handle a lot more in there
   5. Singular Value Decomposition (Similar to PCA but not the exact same thing)
      1. Singular vectors are directions of variation and singular values tell you the importance of each of those vectors
      2. High and low spatial frequency
   6. Need to know PCA, ICA, Penalized matrix factorization, and NMF! These are all dimensionality reductions
   7. Matrix Factorization Methods
      1. In first matrix, we have a tall skinny matrix, or a width equal to number of factors and height features
      2. In second matrix, we have a width equal to the number of features and height factors, so a wider hamburger matrix
      3. Add them together to get the really big matrix with a ton of different a \* c elements in it
      4. X [a \* c] = w [a \* b] \* h [b \*c]
         1. Think about the rows in the h matrix and the columns in the w matrix
   8. We define a loss function which is equal to sum over i j of x minus wh
      1. I need the W and H that reduces the loss function
      2. Like ridge regression and lasso, I add on penalizing terms
      3. I penalize you for picking big values of W and H
      4. It turns out that this is really similar to PCA
   9. Product recommendation: Common use of matrix factorization
      1. Netflix, Spotify
   10. What’s it mean for a vector to be sparse?
       1. You don’t have a lot of non-zero numbers, or a lot of your elements are zeros
   11. Non-linear methods, or manifold learning
       1. Let’s say you had a swiss roll? If you did PCA, there is no way to pick up on fine-grained non-linear patterns going on
          1. The first PCA would just be the mean essentially and there is no good reason to pick one dimension over the other
       2. T-SNE: Start in high dimensional space and go to low
          1. Probabilistic way and use a t test
          2. For every point, look at who is close and come up with way of sticking in low dimensional space
       3. UMAP
       4. LLE
       5. MDS
       6. All of the above are good for pictures
2. Random questions for Jason and Keegan
   1. With k-fold cross validation, do you usually do it on the train and get the accuracy for the train and then still train the model on the entire train set and predict on the test? Is the point to just compare the accuracy rates then?
   2. To remind myself: Important to go back and figure out what exactly expand.grid does
      1. Go back and figure out AIC and BIC and all that (ALA prob best for this)