1 Linear Regression

We assume the model as: $y = XB + \epsilon$ (matrix form) - We see this holds for a single case, as the ith observation of $XB + \epsilon = x_1\beta_1 + x_2\beta_2 + \ldots + \epsilon_i$ - Assume ϵ distributed $MVN(0, \sigma^2 I)$ - Errors are centered around 0, or the expected value of the errors is 0 - They all have the same variance (identially distributed) - There is no covariance between the errors (independently distributed)

Linear: We model the response as linear combination of the betas. Does *not* mean we can't have curves. We can have predictors like x^2, they don't need to be linear

Finds the line / hyperplane of best fit, where best is defined as minimizing the sum of squared residuals (SSR), defined as: $\sum(y_i - y_{pred})^2$ - Squared so as to treat negative mistakes equally as positive mistakes - Can also minimize the average quantity (divide by n). These are equivalent - positive scalars do not affect argmin

Typically low variance, high bias compared to other algorithms - linearity can be a very unrealistic

Not possible in high-dimensional setting ($p > n$) - reduce predictors, get more data, or introduce regularization

1.1 OLS Estimates

The standard solution is given by $\hat{\beta}_{OLS} = (X^T X)^{-1} X^T y$. This are the **ordinary least squares** estimators

X is the **design matrix**- rows denote observations, columns denote features

Obtained via:

1. Rewriting SSR in matrix form: $(y - X\beta)^T (y - X\beta)$
2. Taking partial wrt β (matrix differentiations work similarly to scalars - we can work out the individual betas and see we get equivalent results)
3. Set equal to 0 $\rightarrow$ solve for β

Unbiased ($E[\hat{\beta}] = \beta$)

1.2 Model Comparison

The most immediate metric for a model is R^2: the percentage of variance explained by the model. However, has a few problems

- Monotonically increases with more predictors. Therefore under this metric, there is no penalty to adding as many predictors as we want, even if they're barely / not at all related with the response

- With above, cannot be used to compare models with different number of predictors, as the more parsimonious one is disadvantaged
- Doesn't really tell you if the model is performing well - in some domains with a lot of variability (social sciences), low R^2 values are common, even though the model itself is agreed to be "good"

Therefore some alternative criterion are introduced to account for additional predictors:

- Adjusted R^2: Like R^2, but "penalizes" on adding more predictors. Can compare models with different number of predictors
- AIC / BIC

2 Logistic Regression

Designed for binary response ($y \in \{0, 1\}$)

Instantiation of generalized linear model:

1. Linear predictor η (as before, linear combination of the predictors): $X\beta$ (no epsilon)
2. Link function: Relates the conditional expectation ($\mu = E[Y|X]$) to linear predictor via $g(mu) = X\beta$. - For logistic regression, the link function is the logit (log odds)
3. Family of probability distribution in exponential family: Bernouili

No closed form solution

No residuals. Why? - Bernouili distribution does not have constant variance: parameters closer to 0.5 have larger variance

3 Lasso Regression

3.1 Why?

Classic linear regression does not work in high-dimensional setting ($X^T X$ is singular, and therefore its inverse is not defined). Lasso is

Hope: Better performance on unseen data by utilizing the bias-variance tradeoff. We introduce more *bias* to the model (doesn't fit training data as well), in hopes that the subsequent decrease in variance will be worth

Depending on magnitude of penalization, will perform feature selection

3.2 Feature Selection: Geometrically

Geometrically, the L1 penalty can be thought of as a diamond (2d), tetrahedron (3d), etc. shape around the origin. With the penalty term, we're saying the beta solutions need to live in this shape. So we radiate out in equal SSR contours from the OLS estimate until we hit this shape, and we call that our solution. As the dimensions go up, it's a virtual certainty that we will hit some sharp "corner" of this shape. This is because the curse of dimensionality pushes more and more volume into the corners in high dimensions, very spindly. And by definition, the corners of this shape are axes, and axes are defined such that some beta values are 0.

2 extremes:

- $\lambda = 0$: No penalty on beta coefficients. Recover OLS estimates
- $\lambda \leftarrow \infty$: Extreme penalty on beta coefficients. Fits an intercept only hyperplane for the model

Will not penalize the intercept

Adds L1 penaliation term to beta coefficients

Can be framed either as minimizing a loss function, or a constrained optimization problem (equivalent specifications)

4 Ridge Regression

4.1 Why?

- See Lasso. Introduce bias for reduced variance (hopefully) L2 penalty on non-intercept beta coefficients

Will *not* perform feature selection (although beta values can get arbitrarily small)

Extremes as above: Can either recover OLS estimates with no penalty, or intercept only model with extreme penalty

5 Decision Tree

Split based on entropy or gini impurity

5.1 ISLR

5.1.1 Regression Trees

Goal: *Partition* predictor space $\in R^p$ (assuming real-valued predictors for simplicity) into J regions. Minimize the RSS for these J regions, or written mathematically:

$$\underset{R_1,\ldots,R_J}{\arg\min} \sum_j \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

(Looping over all observations for their respective regions)

Problem: This isn't even close to tractable.

Solution: Recursive Binary Splitting

- Greedy: Base decision for optimal split made without looking ahead
- Top-down: Start from top of tree

Consider regions $R1, R2$ created by splitting predictor j at a threshold s. Want to find best predictor / cutoff combination, or:

$$\underset{j,s}{\arg\min} \sum_{i \in R1} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R2} (y_i - \hat{y}_{R_2})^2$$

Problem: This likely builds a tree that overfits the training data (tree too complex)

Solution?: Grow a large tree T_0 under this approach, and select a subtree from it with best cross-validation error. But there are too many subtrees to consider!

Solution 2: *Cost complexity pruning* is one solution, providing a tradeoff between model fit and complexity. This tuning is controlled by α - as it increases, we penalize the candidate tree T for having more leaves. Thus, we are interested in the best tree as a function of α:

$$\underset{T}{\arg\min} \sum_{m=1}^{T} RSS(y_i, y_{R_m}) + \alpha|T|$$

As α increases from 0, branches get pruned off predictably in a nested fashion. Choose α with cross-validation

5.1.2 Classification Trees

Recursive binary splitting also used to grow classification trees

Problem: Cannot use RSS for splits

Solution? Use classification error rate to measure how well a node is doing:

$$1 - max_k p_{mk}$$

(Proportion in mth region belonging to kth class). But this isn't sensitive enough to "node purity"

Solution: Use one of **Gini Impurity** or **entropy**. Both reward nodes for having observations close to 0 or 1 proportion.

Note: Still use classification error rate for predictive accuracy

6 Random Forest

2 key ideas

- Bagging ("forest"): Reduce the high variance of a single decision tree by taking many bootstrapped samples and averaging bootstrapped samples obtained by taking samples from the original dataset *with replacement* until we have one of the same size
- Random feature subsets ("random"): If we have one really important feature, then all bagged trees will use it at the top level $\rightarrow$ trees will be correlated. Averaging correlated quantities does not reduce variance as much - Decorrelate trees by only allowing the trees to choose from a random subset of variables at each split

Commonly consider splits of $\sqrt{p}$ or $log_2(p)$

Pro: Random forests are built using bagging, and thus each tree only makes use of about $\frac{2}{3}$ of the data. Thus, we can obtain an immediate estimate of the test error by predicting a labeled observation using all the trees that didn't fit with it. This is called the **out of bag** error, which approximates LOOCV as the number of trees grows large

Feature importance: We can average the drop in Gini index / entropy over all trees that used a given feature to estimate the importance of a variable.

7 Naive Bayes

7.1 Model Formulation

Goal: Calculate the class-dependent probabilities of an observation $\mathbf{x} = [x_1, \ldots, x_n]$, and assign to class with highest probability. Thus,

$$\hat{y} = \arg\max_k p(C_k|x_1, \ldots, x_n)$$

Using Bayes rule, we may re-express the probability of a given class as:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

When considering argmax problems, positive scalars do not affect the solution. Thus, we are interested in

$$\arg\max_k p(C_k)p(x|C_k) = \arg\max_k p(C_k)p(x_1, \ldots, x_n|C_k)$$

From here, we apply the "naive" assumption of Naive Bayes. We assume each of the features are conditionally independent *given* the class label. And thus, we have:

$$\arg\max_k p(C_k)p(x_1, \ldots, x_n|C_k) = \arg\max_k p(C_k) * \prod p(x_i|C_k)$$

Priors on the class labels ($p(C_k)$) are often assumed to be the relative frequency of how often they show up in the training data. The sampling model is more complicated, leading to:

7.2 Event Model

From here, we need to specify an assumption about the distribution $x_i|C_k$. There are a few options (which are primarily dictated by the type of our predictor(s))

- **Gaussian:** For a continuous x_i, assume $x_i|C_k \sim N(\mu_k, \sigma_k^2)$, where the parameters are obtained via ML from the training set (sample mean + variance)
- **Multinomial:** For discrete $\mathbf{x} \in R^n$, assume $x|C_k \sim Mult(\theta_{C_k})$, where the class specific theta parameters are given by smoothed ML estimates (to prevent multiplying by 0)
- **Bernoulli:** For binary-valued predictors

8 Support Vector Machine

9 XG Boost

10 KNN