Review: 9 weeks in 80 minutes

Lecture 18

Connor Dowd

May 27th, 2021

Today's Class

Today's Class

0. Themes: OOS, Loss, Models as Legos
1. Week 1: Errors, FDR, Loss
2. Week 2: Regressions (linear, logit), Deviance, OOS, Bootstrap
3. Week 3: AIC, Stepwise Regression, LASSO, CV (K-fold)
4. Week 4: KNN, Classification, ROC, Multinomial
5. Week 5: Trees, Bagging, Ensembles, Forests
6. Week 6: Boosting, More Ensembles, Rolling Block CV
7. Week 7: Cleaning Data: pivot, join, aggregate, winsorize
8. Week 8: RCTs, ATEs, CATEs, Targeting
9. Week 9: Neural nets, SGD, Optimization, Review

Themes

OOS

1. Out of sample predictive performance is the one true metric

- Cross validation and its variants are incredible tools
- Avoid ‘contaminating’ training data with test data
- If your OOS predictions are bad, why would your inference be good?

Costs

2. The model we estimate *depends* on the costs of the mistakes we could make

- We often face different risks in one direction or the other. Don't ignore that.

Selection vs Aggregation

3. Choosing between models is hard. Aggregating them is easy

- Why not use them all?
- OOS performance to help downweight useless models
- Lots of dumb models can make for one good model

Uncertainty

4. Multiple sources of uncertainty

- In-side the final model
- Our model choice procedure
- Our data's reliability

Models are Legos

5. Models are like building blocks

- We can build crazy things by combining them
- and we can decide we don't trust a bridge made of them

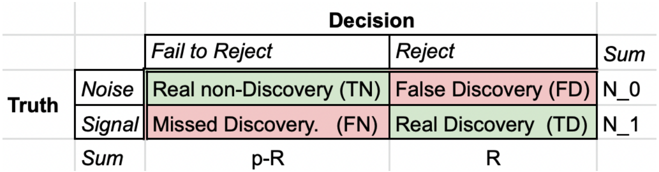
Week 1: Errors, FDR, Loss

Large Scale Testing

We wish to test K simultaneous null hypothesis:

$$H0_1, H0_2, ..., H0_K$$

Out of the K null hypothesis, N_0 are true nulls and $N_1 = K - N_0$ are false – i.e. there is an effect.



		Decision		
		Fail to Reject	*Reject*	*Sum*
Truth	*Noise*	Real non-Discovery (TN)	False Discovery (FD)	N_0
	Signal	Missed Discovery. (FN)	Real Discovery (TD)	N_1
	Sum	p-R	R	

False Discovery Rate

FD Proportion = False positives / #Significant = $\frac{FD}{R}$
We can't know this.

We can control its expectation though: False Discovery Rate, $FDR = E[FDP]$.

If all tests are tested at α level, we have $\alpha = E[FD/N_0]$, whereas $FDR = E[FD/R]$

We can find in-sample analogues (ish) of these things.

FDR Control

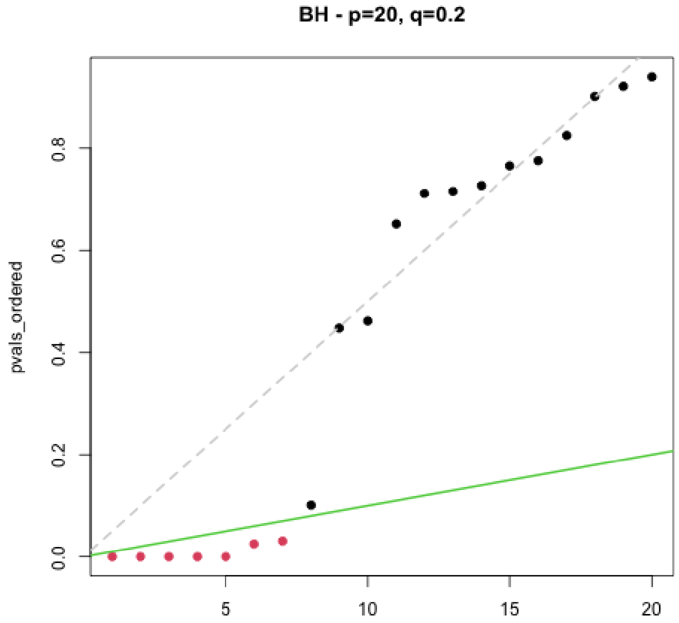
Suppose we want to know that $FDR \leq q = 0.1$.

Benjamini + Hochberg Algorithm

1. Rank your p-values smallest to largest.
2. Set p-value cutoff as $\alpha^* = max\{p_{(k)} : p_{(k)} \leq q\frac{k}{K}\}$

Then $FDR \leq q$ – assuming approximate independence between tests.

BH



Loss

At its simplest, a loss function maps the truth, and your prediction, into how unhappy you are about your error.

$$L(y, \hat{y}) = ????$$

The most common class loss function is the squared error loss.

$$L_2(y, \hat{y}) = l_2(y - \hat{y}) = l_2(e) = e^2$$

This loss only cares about the magnitude of the error, not its location, not its direction. It also penalizes larger loss more than smaller loss. This is not always a reasonable set of choices.

Week 2: Regressions (linear, logit), Deviance, OOS, Bootstrap

Regression

Many problems involve a response or outcome (`y`),
And a bunch of covariates or predictors (`x`) to be used for regression.

A general tactic is to deal in averages and lines.

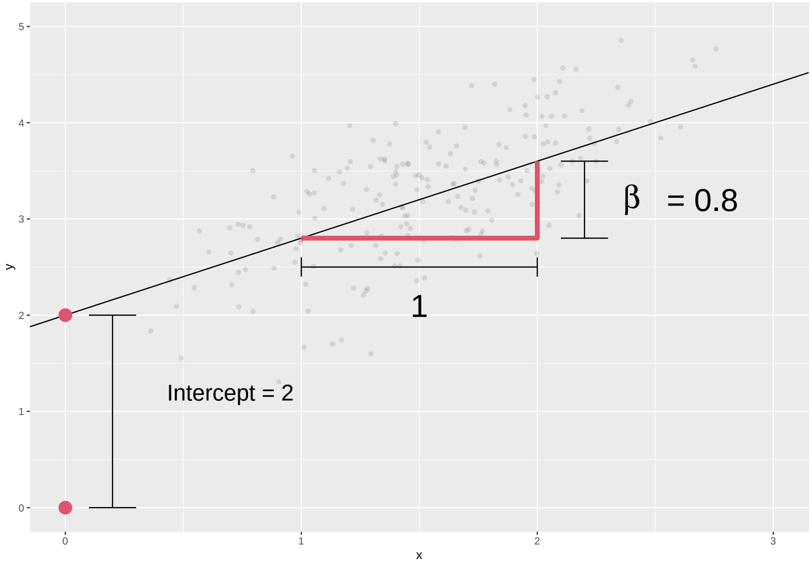
$$E[y|\mathbf{x}] = f(\mathbf{x}'\beta)$$

Where $\mathbf{x} = [1, x_1, x_2, x_3, ..., x_p]$ is our vector of p covariates.
$\beta = [\beta_0, \beta_1, \beta_2, ..., \beta_p]$ are the corresponding coefficients.
The product $x'\beta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$.

For simplicity we denote $x_0 = 1$ to estimate intercepts.

Linear Regression

Linear regression takes this seriously, and adds that $E[Y|X] = X'\beta$.



Logistic Regression

Building a linear model for **binary data**.
Recall our original specification: $E[Y|X] = f(x'\beta)$

The Response y is 0 or 1, leading to a conditional mean:

$$E[y|x] = P[y=1|x] \times 1 + P[y=0|x] \times 0 = P[y=1|x]$$

$\implies$ The expectation is a probability.

The ‘logit’ link is common, for a few reasons. One big reason?

$$log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_K x_K$$

This is a linear model for log odds.

Deviance

Deviance refers to the distance between our fit and the data. You generally want to minimize it.

$$Dev(\beta) = -2log(L(\beta|data)) + C$$

We can ignore C when comparing models estimated on the same data and with similar structures.

Deviance is a measure of fit that is similar to the residual sum of squares, for a broader class of models (logistic regression, etc).

We'll think about deviance as a cost to be minimized.

Minimize Deviance $\iff$ Maximize likelihood

Bootstrap

Each observation in our data was independently sampled from some distribution. What if every time we wanted a new sample, we just put all our observations in an urn, drew our n of them (with replacement), or $n - 10$ (without replacement) and calculated the parameter in that sample?

We could do this ten-thousand times. And have a theory-free distribution of possible values our sample could have taken.

This is the simplest form of the bootstrap. And it is powerful.

- Can be used for model aggregation, or for inference.

OOS

We care about how well our model works **out-of-sample**.

One way to test this is to use a “validation sample”.

1. Randomly split your data into two samples
   - usually named “testing” and “training”.
2. Fit your model using the “training” data
3. Test predictions on the left-out “testing” data.

Week 3: AIC, Stepwise Regression, LASSO, CV (K-fold)

AIC

We can (with some assumptions) estimate OOS deviance using Akaike's Information Criterion. For a model M with k variables estimated to be $\hat{\beta}_M$

$$AIC = 2k + Dev(\hat{\beta}_M) = 2k - 2log(L(\hat{\beta}_M|data))$$

OOS performance

In sample, adding variables always improves predictive accuracy.

- IS R^2 for full model (200 vars): 56%
- IS R^2 for FDR cut model (25 vars): 18%

Out of Sample, we may gain predictive accuracy by dropping variables.

- OOS R^2 for the full model: -5.87.
- OOS R^2 for the cut model: 0.10.

THIS IS NEGATIVE. Out-of-Sample, we would be much *BETTER* off predicting the mean than with the full model. How do we choose variables to use?

Stepwise Regression

There are too many possible subsets of our variables to compare all of them. So we need some other method for coming up with a subset to compare.

Stepwise regression does this. It starts with a simple model, and “greedily” adds the best new variable repeatedly until adding a new variable no longer improves the AIC.

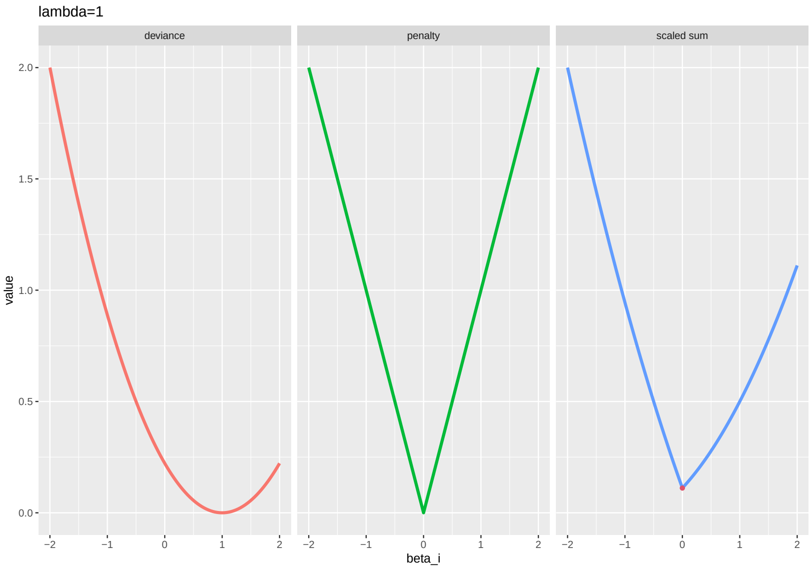
Because each choice depends on the current model parameters, small changes to the data can have big consequences for our model choices. This instability is bad for our OOS prediction.

LASSO

LASSO is the most commonly used regularized (or penalized) regression model. The lasso penalty is the l_1 norm of our parameters: $pen(\beta) = \sum |\beta_j|$, which penalizes larger coefficients and non-zero coefficients. So our estimates are:

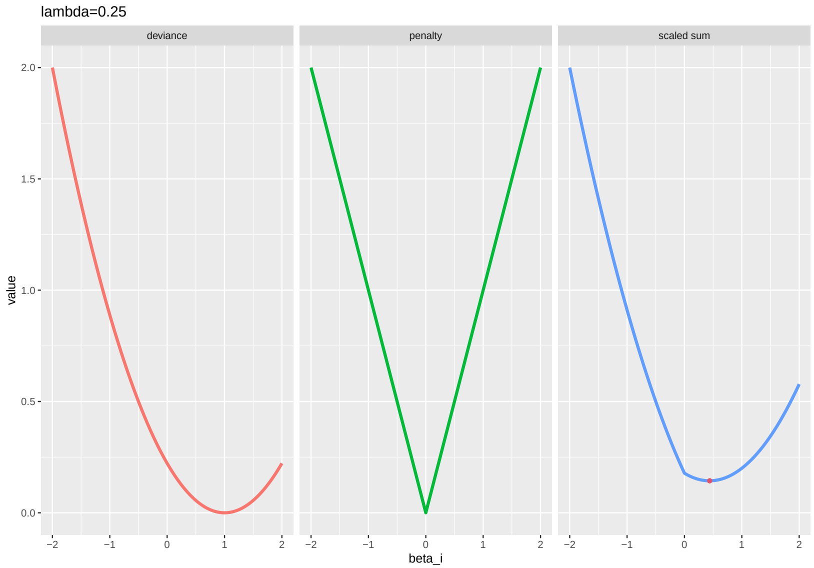
$$\hat{\beta}_\lambda = \underset{\beta}{\operatorname{argmin}} \left(Dev(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

LASSO

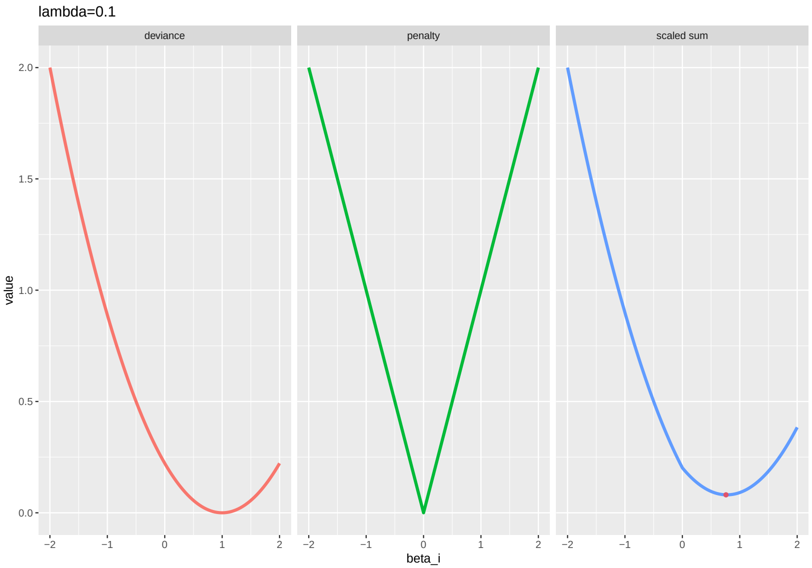


lambda=1
deviance
penalty
scaled sum
value
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
beta_i

LASSO


lambda=0.25
deviance
penalty
scaled sum
value
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
beta_i

LASSO


lambda=0.1
deviance
penalty
scaled sum
value
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
beta_i

K-fold Cross-validation

LASSO gives us estimates conditional on some penalty lambda. How do we pick lambda? OOS performance.

We will "fold" the data – i.e. we will partition it into K different groups of observations. Then, `for k=1:K`

- Use observations in group k as test data.
- Train the models on the remaining data (for every λ)
- Predict the observations in group k using those models
- Record the prediction errors for each lambda

Choosing K

There are several options:

- Leave-one-out Cross-validation: AKA $K = n$ is great, but much slower (fits every model under consideration n times)
- $K = 5$ corresponds to 5 different 20% leave-out samples.
- $K = 20$ corresponds to 20 different 5% leave-out samples.

Most people set $K \in [5, 20]$. I'll mostly use 10.

$\implies$ Optimizing K is very 3rd order. Not worth worrying about too much beyond time considerations and some preference for larger K.

Week 4: KNN, Classification, ROC, Multinomial

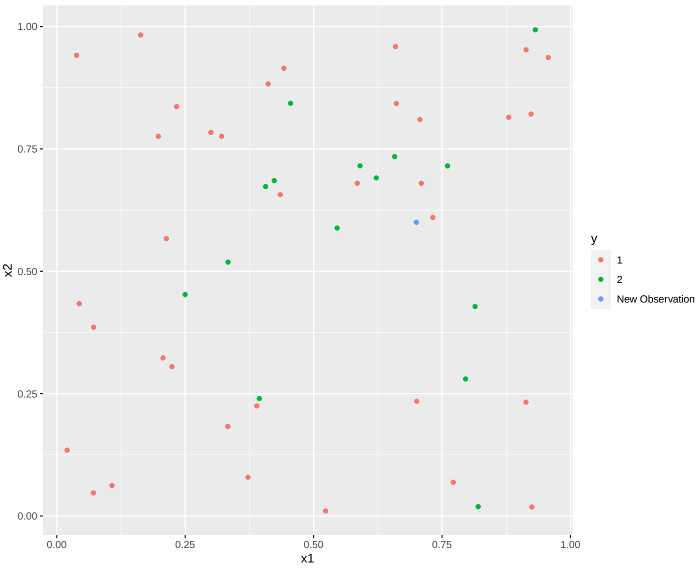
KNN Basics

Basic Idea: Estimate $P[y|x]$ locally using the labels of similar observations in the training data.

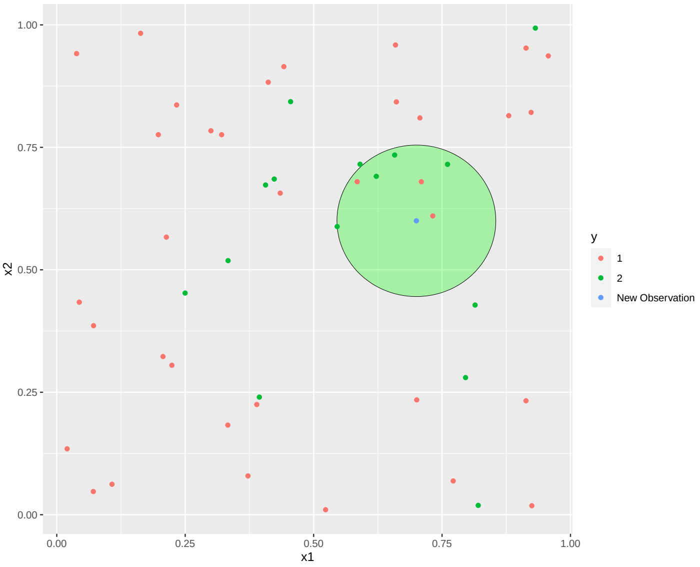
KNN: What is the most common class near x^{new}?

1. Take the K nearest neighbors $x_{i,1}, ..., x_{i,K}$ of x^{new} in the training data
   - Nearness is (usually) Euclidean distance: $\sqrt{\sum_{j=1}^{p}(x_j^{new} - x_{i,k,j})^2}$
2. Estimate $P[y = j|x] = \frac{1}{n}\sum_{i=1}^{K} 1(y_i = j)$
3. Select the class j with the highest probability.

KNN Example Data

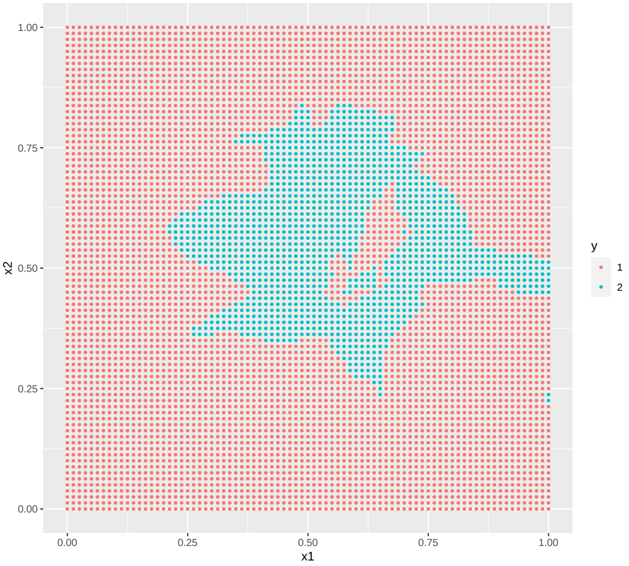


KNN Example $K = 7$



The relative ‘vote counts’ are a very crude estimate of probability.

KNN Example



Binary Classification

Many problems can be reduced to binary classification (as above).

KNNs are a useful non-parametric classification tool.

Logits are a useful parametric classification tool.
- Remember Spam?

- Logits yield parametric decision boundaries. Easy to interpret.
- Logits are global methods. Use *all* the training data to inform predictions.
  - The probability estimates are more useful and more stable.
- Logits can do variable selection.

Sensitivity and Specificity

But we may also want to think about sensitivity and specificity.

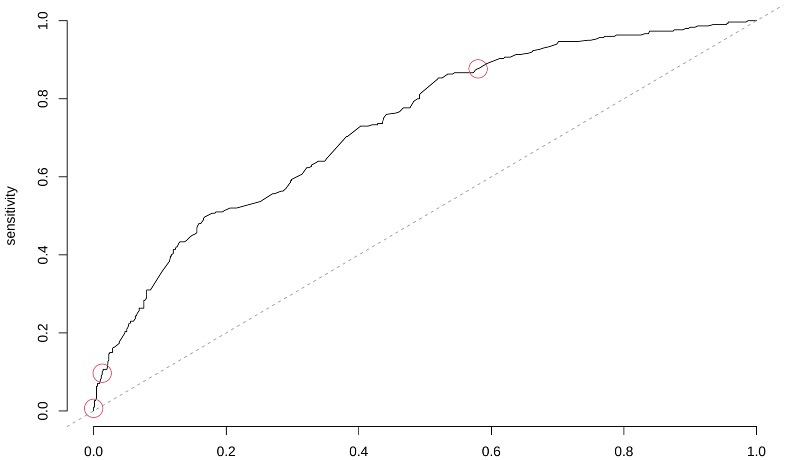
- Sensitivity: proportion of true $y = 1$ classified as such.
- Specificity: proportion of true $y = 0$ classified as such.

A rule is sensitive if it mostly gets the 1s right. A rule is specific if it mostly gets the 0s right.

ROC Curve

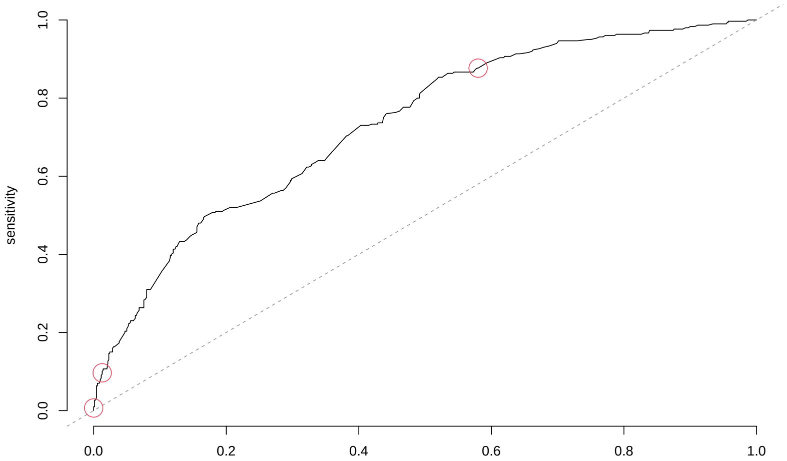
ROC curve measures the tradeoff between sensitivity and specificity.

We can plot the ROC curve for different choices of threshold.



AUC

The AUC is a common metric for choosing between classifiers, which selects the classifier that maximizes the Area Under the Curve – specifically the area under the ROC curve.



Multinomial

What if we have more than one class possible? E.g. Not just $\{0, 1\}$ but $\{0, 1, 2, ..., M\}$

KNN just predicts most likely class.

Multinomial Logit builds a logit for the probability of being in each class, then rescales predictions so that the sum of probabilities is 1. (Combining these models)

Week 5: Trees, Bagging, Ensembles, Forests

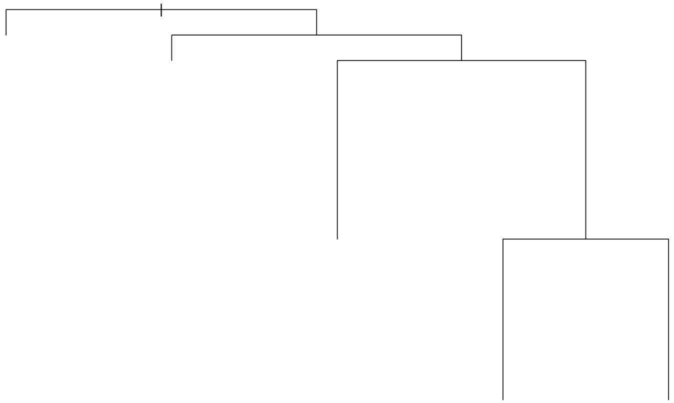
Trees

Trees try to find groupings that are good for making predictions.

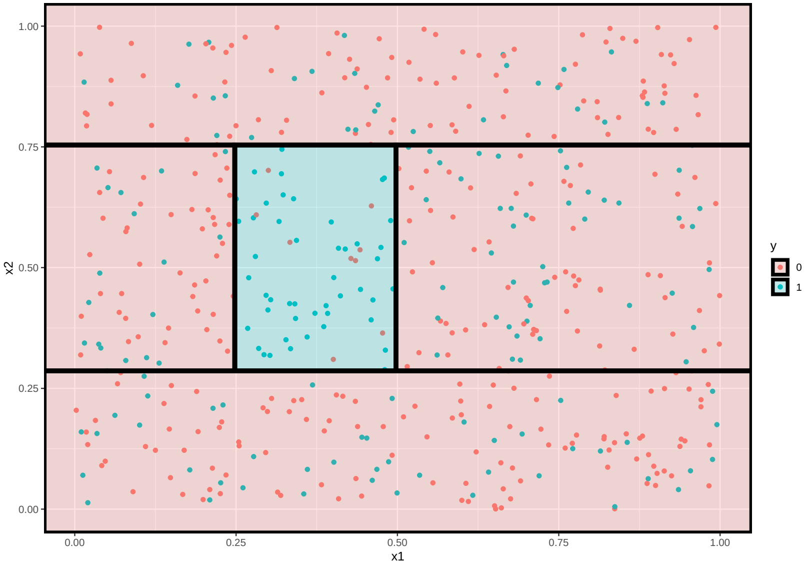
They do this by recursively partitioning groups into two.

At each step, the tree find the partition that best improves its predictions, and makes that partition.

Trees



Trees



Trees Problems

- Overfitting
  - How many leaves? We need to choose.
- Very unstable.
  - **Slight Peturbations** of the data can change the entire tree structure

One solution? Bootstrap-aggregating (“BAgging”)

Bagging: In Brief

We will boostrap the data:

- Recall: this means drawing another similar size sample with replacement and building the model with that sample – repeatedly.

This in essence creates numerous small “peturbations” of the data.

And it will let us create 1000 very similar models. Each of which is potentially very overfit.

Example

```r
resampled_mod = function(x) {
  ind = sample(nrow(df),,replace=T)
  rpart(y~x1+x2,data=df[ind,],cp=0)
}

mod =  rpart(y~x1+x2,data=df,cp=0)
modb1 = resampled_mod(1)
modb2 = resampled_mod(1)
modb3 = resampled_mod(1)
modb4 = resampled_mod(1)
modb5 = resampled_mod(1)
modb6 = resampled_mod(1)
modb7 = resampled_mod(1)
```

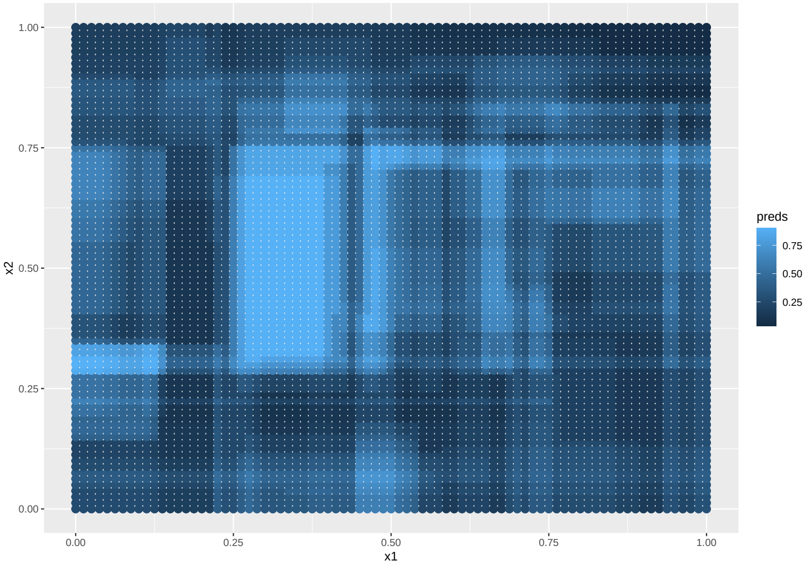
Ensembles – The bare bones ensemble

An ensemble is some combination of smaller models which (hopefully) improves our overall predictions.

The simplest ensemble takes an average across each of the submodels predictions.

Any time we want a prediction, we average our predicted probabilities across each model type.

Averaged across 7 models



Random Forests

Forests take the notion of Bagging, and make some minor improvements – mostly in the name of speed.

By introducing variation into the variables under consideration, they create *even more instability* between trees.

But it turns out, because we are averaging across our trees, this leads to improvements in predictive power.

They search across a wider range of models.

Week 6: Boosting, More Ensembles, Rolling Block CV

Ensembles 2

What other types of ensembles can we build?

- More sophisticated averages:
  - inverse variance weights – using OOS MSE to estimate prediction quality
- We could just take the predictions of 10 models and build a model (e.g. linear regression) the output on those predictions.
  - Or build a forest on them. etc.

Weighted – Example

Suppose I have two models: - M_1 always predicts $\hat{y}_1 = M_1(X) = \bar{y}$ - M_2 uses p variables and linear regression to predict $\hat{y}_2 = M_2(X) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p$

These are not equivalently good. M_2 is *likely* better (not always – see overfitting again).

We could plug them both into our naive average, and get a prediction.

Inverse Variance Weights

The basic idea is that a single measure of the prediction errors of each model can help us generate the best weights.

The models with the smallest average errors should get the most weight, and the models with the largest average errors should get the least weight.

If you are minimizing MSE, this will look like weights which are proportional to the “precision" (aka the inverse of the variance).

Boosting

1. Start with a class of models F (e.g. tree, glm, etc). And weights $w_i = \frac{1}{n}$
2. Fit a model M_k in that class using those weights.
3. Find the prediction error for each observation from that model.
4. Increase the weights for observations where predictions were most wrong, decrease the weights for observations where predictions were most correct.
5. Repeat steps 2-4 until you've built K models.

Model M_K will be using weights based on how poorly each previous model did at predicting each observation.

The ensemble model is just the average across models $1 : K$

Rolling Block CV

Standard Cross-validation assumes independence between observations. What if that isn't true?

Suppose you have a model for predicting a week into the future, which uses data from the past month.

We can estimate that model based on days 1-30, and see how wrong its forecast for day 37 is.

And then again for days 2-31 and day 38. And so forth.

This will let us use real OOS predictions for doing model selection and development.

Week 7: Cleaning Data: pivot, join, aggregate, create, winsorizing

Data Cleaning 101

Rules of Thumb:

1. Keep looking at the data
2. Make small changes.
3. Test your changes before you overwrite variables.
4. Don't overwrite actual files unless you're certain.
5. **Don't throw away potentially useful data.**

Winsorizing

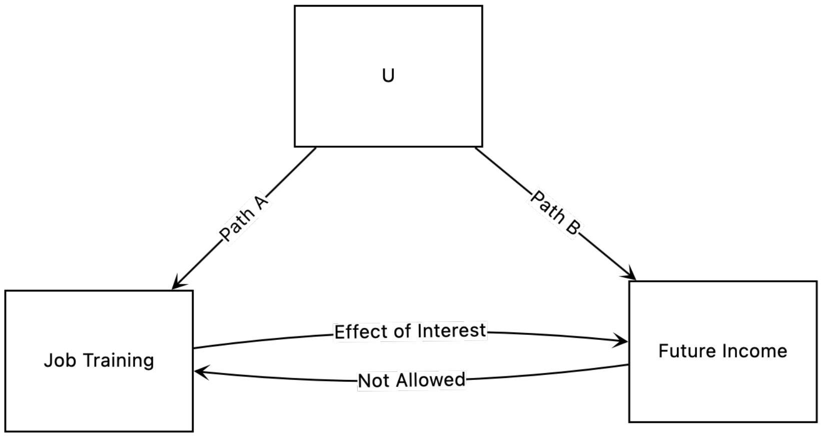
Set some threshold quantile – e.g. 99%.

Set values above that quantile to that quantile – squashing down the big outliers.

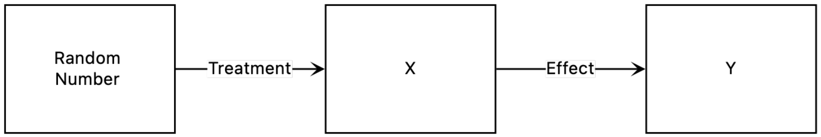
- Strikes a balance between keeping and removing outlier points.

Week 8: RCTs, ATEs, CATEs, Targeting

RCTs


U
Path A
Path B
Job Training
Effect of Interest
Not Allowed
Future Income

RCTs



Similarity

Typically the ATE is:

$$\widehat{ATE} = \bar{y}_1 - \bar{y}_0$$

But we could also define it as the average across individual treatment effects.

$$\widetilde{ATE} = \frac{1}{n} \sum_{i=1}^{n} \widehat{TE}_i$$

With a lot of rewriting of sums – we can show that when our individual treatment effects use a constant mean to predict counterfactuals:

$$\widetilde{ATE} = \widehat{ATE}$$

Review: TEs

An average treatment effect (ATE) is the average difference between the potential outcome under treatment, and the potential outcome under control.

Individual treatment effects are the *individual's* difference between potential outcomes under treatment and control.

Conditional Average Treatment Effects, are the average difference btween potential outcome under treatment and control *for some subgroup*

Review: Targeting

Targeting mostly consists of trying to identify groups with notable (large, negative, etc) CATEs.

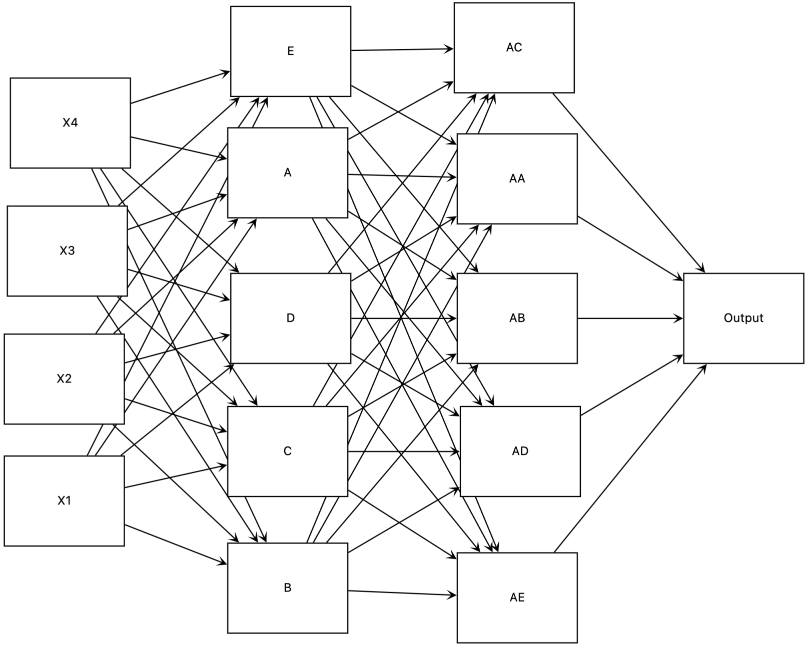
Many methods. We've looked at "T-learners" – which make predictions about individual outcomes under treatment and control, then estimate differences.

With estimated treatment effects for individuals, we can decide who to *target*.

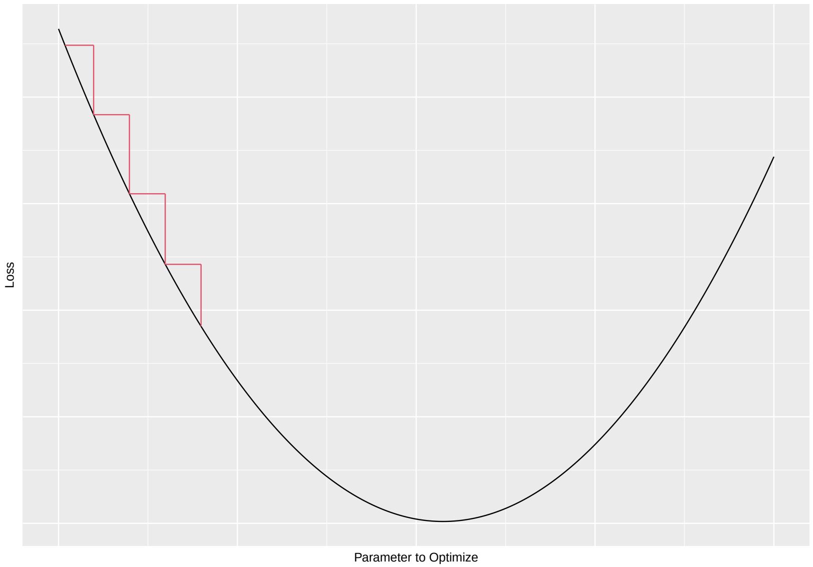
This needs solid out-of-sample performance to be useful.

Week 9: Neural nets, SGD, Optimization, Review

Neural Nets - Essence



Gradient Descent - Essence



Wrap up

Wrap up

Thank you all.