BUS41100 Applied Regression Analysis

Week 7: Model Building

Variable Selection, BIC, AIC, LASSO

Max H. Farrell
The University of Chicago Booth School of Business

Model Selection

Our job now is to pick which X variables belong in our model.

- A good prediction model summarizes the data but does not overfit.

What if the goal isn't just prediction?

- A good model answers the question at issue.
  - Better predictions don't matter if the model doesn't answer the question.
- A good regression model obeys our assumptions.
  - Especially important when the goal is inference/relationships.
  - A causal model is only good when it meets even more assumptions.

What is the goal?

1. Relationship-type questions and inference?
   - Are women paid differently than men on average?
     ```
     > lm(log.WR ~ sex)
     ```
   - Does age/experience differently affect men and women?
     ```
     > lm(log.WR ~ age*sex - sex)
     ```
   - No other models matter
2. Data summarization?
   - Matched the dynamics/trends
   - Describe a past phenomenon
3. Prediction?
   - Need a fair, objective criterion that matches the idea of predicting the future. Avoid overfitting.

Overfitting

We have already seen overfitting twice:

1. Week 4: $R^2 \uparrow$ as more variables went into MLR

   ```
   > c(summary(trucklm1)$r.square, summary(trucklm3)$r.square,
   +      summary(trucklm6)$r.square)
   [1]  0.021  0.511  0.693
   ```

2. Week 6: Classification error $\downarrow$ as more variables into logit

   ```
     empty history    full
     0.300   0.283   0.214
   ```

Fitting the data at hand better and better

. . . but getting worse at predicting the next observation.

How can we use the data to pick the model without relying on the data too much?

Out-of-sample prediction

How do we evaluate a forecasting model?

- Make predictions!
- Out-of-sample prediction error is the Gold Standard for comparing models. (If what you care about is prediction.)

Basic Idea: We want to use the model to forecast outcomes for observations we have not seen before.

- Use the data to create a prediction problem.
- See how our candidate models perform.

We'll use most of the data for training the model, and the left over part for validating/testing it.

In a validation scheme, you

- fit a bunch of models to most of the data (training set)
- choose the one performing best on the rest (testing set).

For each model:

- Obtain $b_0, \dots, b_d$ on the training data.
- Use the model to obtain fitted values for the n_{test} testing data points: $\hat{Y}_j = \mathbf{x}_j'\mathbf{b}$ or $\hat{Y}_j = \mathbb{1}\{\hat{\mathbb{P}}[Y=1|\mathbf{x}_j] > 0.5\}$
- Calculate the Mean Square Error for these predictions.

$$\boxed{MSE = \frac{1}{n_{\text{test}}} \sum_{j=1}^{n_{\text{test}}} (Y_j - \hat{Y}_j)^2}$$

Out of sample validation steps:

1) Split the data into testing/training samples.

```
> set.seed(2)
> train.samples <- sample.int(nrow(credit), 0.95*nrow(credit))
> train <- credit[train.samples,]
> test <- credit[-train.samples,]
```

2) Fit models on the *training* data

```
> full <- glm(GoodCredit~., family=binomial, data=train)
> history <- glm(GoodCredit~history3, family=binomial, data=trai
```

3) Predict on the *test* data

```
> predfull <- predict(full, type="response", newdata=test)
> errorfull <- test[,"GoodCredit"] - (predfull >= .5)
```

4) Compute MSE/MAE

```
> c(empty=mean(errorempty^2), history=mean(errorhistory^2),
+   full=mean(errorfull^2) , too.good=mean(errortoo.good^2) )
   empty  history     full too.good
    0.24     0.20     0.32     0.42
```

This missing piece is in

2) Fit models on the *training* data

Which models?

The **rest of this week** is about tools to help with choosing models. We'll do linear and logistic examples.

- Once we have tools for step 2, it's easy to compute out-of-sample MSE.

There are two pieces to the puzzle:

- Select the "universe of variables"
- Choose the best model(s)

The computer helps only with the 2nd!

The universe of variables is HUGE!

- includes all possible covariates that you think might have a linear effect on the response
- . . . and all squared terms . . . and all interactions . . . .

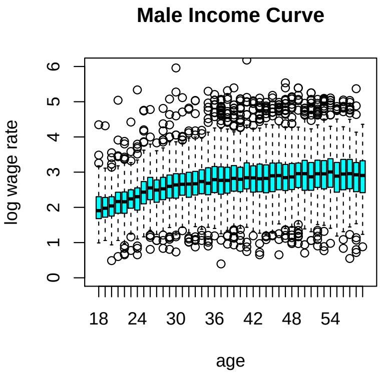
You decide on this universe through your experience and discipline-based knowledge (and data availability).

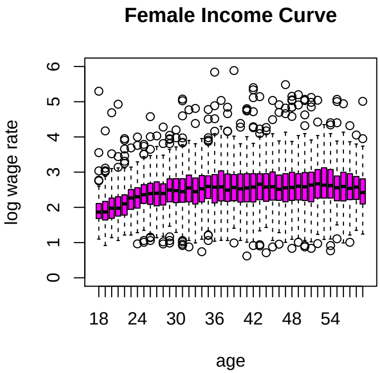
- Consult subject matter research and experts.
- Consider carefully what variables have explanatory power, and how they should be transformed.
- If you can avoid it, don’t just throw everything in.

This step is very important! And also difficult.

. . . and sadly, not much we can do today.

Today's linear model example: Census data on wages





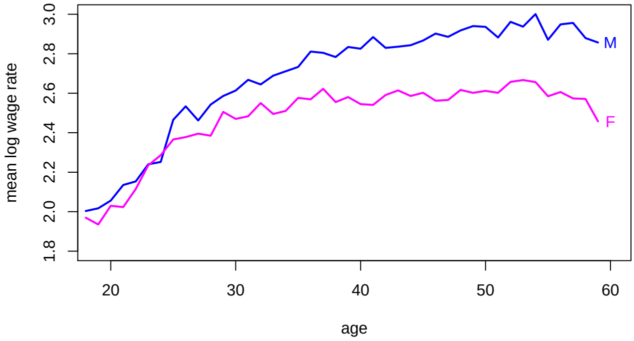
We look at people earning >\$5000, working >500 hrs, and <60 years old.

(Sound familiar?)

A discrepancy between mean $\log(\text{WR})$ for men and women.

- Female wages flatten at about 30, while men's keep rising.

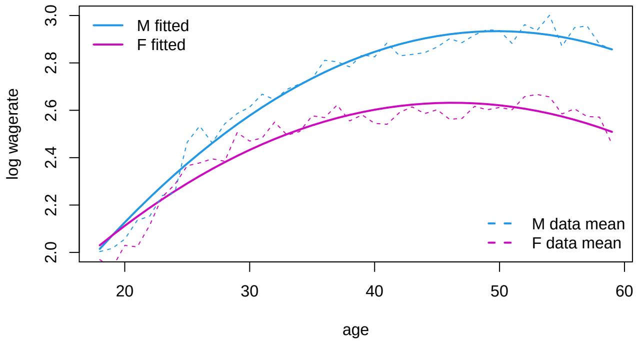
```
> men <- sex=="M"
> malemean <- tapply(log.WR[men], age[men], mean)
> femalemean <- tapply(log.WR[!men], age[!men], mean)
```



If we just cared about capturing the trends ...

$$\mathbb{E}[\log(\mathrm{WR})] = 1 + .07 \cdot \mathrm{age} - .0008 \cdot \mathrm{age}^2 + (.02 \cdot \mathrm{age} - .00015 \cdot \mathrm{age}^2 - .34)\mathbb{1}_{[\mathrm{sex}=M]}.$$

```
> wagereg <- lm(log.WR ~ age*sex + age2*sex)
```



- This model provides a generally decent looking fit
- It is (arguably?) simple and interpretable

How should we model

$$\mathbb{E}\left[\texttt{log.WR} \mid \texttt{age}, \texttt{sex}, \texttt{race}, \texttt{marital}, \texttt{edu}\right] = \textbf{?}$$

- Data visualization suggests `sex*(age + age^2)`

But what else? Polynomials? More interactions? Everything?

Four models to use as examples:

```
> wagereg1 <- lm(log.WR ~ age*sex + age2*sex + ., data=train)
> wagereg2 <- lm(log.WR ~ age*sex + age2*sex + marital +
+              (hs+assoc+coll+grad)*age + race*age , data=train)
> wagereg3 <- lm(log.WR ~ race*age*sex + age2*sex + marital +
+                        (hs+assoc+coll+grad)*age, data=train)
> wagereg4 <- lm(log.WR ~ race*age*sex - race + age2*sex +
+              marital + (hs+assoc+coll+grad)*age, data=train)
```

Variable Selection

More variables *always* means higher R^2, but ...

- we don't want the full model
- we can't use hypothesis testing
- we need to be rigorous/transparent

We will study a few variable selection methods and talk about the general framework of

Penalized Regression

Usual disclaimer:

- there's lots more out there, remember those other classes?

Penalized Regression

A systematic way to choose variables is through **penalization**. This leads to a family of methods (that we will only sample).

Remember that we choose $b_0, b_1, b_2, \dots, b_d$ to

$$\min \frac{1}{n} \sum (Y_i - \hat{Y}_i)^2 \;\Leftrightarrow\; \max R^2$$

We want to maximize fit but minimize complexity.
Add a penalty that increases with complexity of the model:

$$\min \left\{ \frac{1}{n} \sum (Y_i - \hat{Y}_i)^2 + \text{penalty}(\text{dim}) \right\}$$

- Different penalties give different models.
- Replace SSE with other loses, e.g. logit.

Information criteria

Information criteria penalties attempt to quantify how well our model would have predicted the data, regardless of what you've estimated for the β_j's.

The best of these is the BIC: Bayes information criterion, which is based on a "Bayesian" philosophy of statistics.

$$BIC = n\log(SSE/n) + p\log(n)$$

$p = \#$ variables, $n =$ sample size, but what sample?

- Choose the model that minimizes BIC.

Remember: $SSE = \sum(Y_i - \hat{Y}_i)^2$, $\min SSE \Leftrightarrow \min n\log(SSE/n)$.

Another popular metric is the Akaike information criterion:

$$AIC = n\log(SSE/n) + 2p$$

A general form for these criterion is $n\log(SSE/n) + kp$, where $k = 2$ for AIC and $k = log(n)$ for BIC.

In R, we can use the `extractAIC()` function to get the BIC.

- `extractAIC(reg)` $\Rightarrow$ AIC
- `extractAIC(reg, k=log(n))` $\Rightarrow$ BIC

AIC prefers more complicated models than BIC, and it is not as easily interpretable.

Back to the Census wage data...

AIC

```
> extractAIC(wagereg1)
[1]      18.00 -24360.83
> extractAIC(wagereg2)
[1]      26.0 -24403.9
> extractAIC(wagereg3)
[1]      34.00 -24455.15
> extractAIC(wagereg4)
[1]      30.00 -24462.91
```

BIC

```
> extractAIC(wagereg1, k=log(n))
[1]      18.00 -24219.45
> extractAIC(wagereg2, k=log(n))
[1]      26.00 -24199.67
> extractAIC(wagereg3, k=log(n))
[1]      34.00 -24188.09
> extractAIC(wagereg4, k=log(n))
[1]      30.00 -24227.26
```

(remember n is training sample size.)

Model probabilities

One (very!) nice thing about the BIC is that you can interpret it in terms of model probabilities.

Given a list (what list?) of possible models $\{M_1, M_2, \ldots, M_R\}$, the probability that model i is correct is

$$P(M_i) \approx \frac{e^{-\frac{1}{2}BIC(M_i)}}{\sum_{r=1}^{R} e^{-\frac{1}{2}BIC(M_r)}} = \frac{e^{-\frac{1}{2}[BIC(M_i)-BIC_{\min}]}}{\sum_{r=1}^{R} e^{-\frac{1}{2}[BIC(M_r)-BIC_{\min}]}}$$

—

Subtract $\min\{BIC(M_1)\ldots BIC(M_R)\}$ for numerical stability.

```
> eBIC <- exp(-0.5*(BIC-min(BIC)))
> eBIC
    wagereg1     wagereg2     wagereg3     wagereg4
2.011842e-02 1.023583e-06 3.120305e-09 1.000000e+00
> probs <- eBIC/sum(eBIC)
> round(probs, 5)
wagereg1 wagereg2 wagereg3 wagereg4
 0.01972  0.00000  0.00000  0.98028
```

BIC indicates that we are 98% sure `wagereg4` is best.

(of these 4!).

Another Example: NBA regressions from last class

Our "efficient Vegas" model:

```
> extractAIC(glm(favwin ~ spread-1, family=binomial), k=log(553))
[1]   1.000 534.287
```

A model that includes non-zero intercept:

```
> extractAIC(glm(favwin ~ spread, family=binomial), k=log(553))
[1]   2.0000 540.4333
```

What if we throw in home-court advantage?

```
> extractAIC(glm(favwin ~ spread + favhome, family=binomial), k=log(553))
[1]   3.0000 545.637
```

The simplest/efficient model is best
(The model probabilities are 0.953, 0.044, and 0.003.)

Thus BIC is an alternative to testing for comparing models.

- It is easy to calculate.
- You are able to evaluate model probabilities.
- There are no "multiple testing" type worries.
- It generally leads to more simple models than F-tests, and the models need not be nested.

But which models should we compare?

- 10 X variables means 1,024 models.
- 20 variables means 1,048,576!

As with testing, you need to narrow down your options before comparing models.

Use your knowledge and/or the data

Forward stepwise regression

One approach is to build your regression model step-by-step, adding one variable at a time:

- Run $Y \sim X_j$ for each covariate, then choose the one leading to the smallest BIC to include in your model.
- Given you chose covariate $X^\star$, now run $Y \sim X^\star + X_j$ for each j and again select the model with smallest BIC.
- Repeat this process until none of the expanded models lead to smaller BIC than the previous model.

This is called "forward stepwise regression".

- There is a backwards version, but is often less useful
  - $\hookrightarrow$ Not always! see `week7-Rcode.R`

R has a `step()` function to do forward stepwise regression.

- This is nice, since doing the steps is time intensive
- and would be a bear to code by hand.

The way to use this function is to first run base and full regressions. For example:

```
base <- lm(log.WR ~ 1, data=train)
full <- lm(log.WR ~ . + .^2, data=train)
```

- "`~ . + .^2`" says "everything, and all interactions".

This is one reason that making a `data.frame` is a good idea.

Given `base` (most simple) and `full` (most complicated) models, a search is instantiated as follows.

```
fwd <- step(base, scope=formula(full),
            direction="forward", k=log(n))
```

- `scope` is the largest possible model that we will consider.
- `scope=formula(full)` makes this our “full” model
- `k=log(n)` uses the BIC metric, $n = n_{\text{train}}$

Example: again, look at our wage regression.

The base model has age, age2, and their interaction with sex

...i.e. our final descriptive model

Our scope is all other variables and their possible interaction.

```
> base <- lm(log.WR ~ age*sex + age2*sex, data=train)
> full <- lm(log.WR ~ . + .^2, data=train)
```

And then set it running ...

```
> fwdBIC <- step(base, scope=formula(full),
+             direction="forward", k=log(n))
```

It prints a ton ...

The algorithm stopped because none of the 1-step expanded models led to a lower BIC.

- You can't be absolutely sure you've found the best model.
- Forward stepwise regression is going to miss groups of covariates that are only influential together.
- Use out-of-sample prediction to evaluate the model.

It's not perfect, but it is pretty handy

How did we do?

BIC:

```
> BIC <-  c(BIC, fwdBIC = extractAIC(fwdBIC, k=log(n))[2])
 wagereg1  wagereg2  wagereg3  wagereg4    fwdBIC
-24219.45 -24199.67 -24188.09 -24227.26 -24279.26
```

Model probabilities:

```
> round(probs <- eBIC/sum(eBIC), 5)
wagereg1 wagereg2 wagereg3 wagereg4   fwdBIC
       0        0        0        0        1
```

What about out of sample predictions?

```
> c(error1=mean(error1^2), error2=mean(error2^2),
+   error3=mean(error3^2), error4=mean(error4^2),
+   errorBIC=mean(errorBIC^2))
   error1    error2    error3    error4  errorBIC
0.2982959 0.2972645 0.2975347 0.2974996 0.2971517
```

LASSO

The **LASSO** does selection and comparison simultaneously.

We're going to skip most details here. The short version is:

$$\min \left\{ \frac{1}{n} \sum (Y_i - \boldsymbol{X}_i' \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \left| \beta_j \right| \right\}$$

The penalty has two pieces:

1. $\sum_{j=1}^{p} \left| \beta_j \right|$ measures the model's complexity
   - Idea: minimize penalty $\Rightarrow$ lots of $\beta_j = 0$, excluded
   - Final model is variables with $\beta_j \neq 0$
2. λ determines how important the penalty is
   - Choose by cross-validation (R does all the work)

The LASSO (and its variants) are **very** popular right now, both in academia and industry. Why?

Suppose we want to model $Y \mid X_1, X_2, \ldots, X_{10}$ but we have no idea what X variables to use, or even how many.

- There are $\binom{10}{1} = 10$ 1-variable models, $\binom{10}{2} = 45$ 2-variables models, . . .
  $$\sum_{k=0}^{10} \binom{10}{k} = 1,024 \text{ total!}$$
- For 20 X variables: over 1 million models.
- For 100 variables: 1,267,650,600,228,229,401,496,703,205,376
  Drops of water on Earth: 26,640,000,000,000,000,000,000,000

  (Thank you Wolfram Alpha®)

Stepwise is a specific path through these models, but LASSO "searches" all combinations at once.

- Easy to run: it's fast, scalable, reliable, ...
- Theoretical guarantees.

A little clumsy in R:

1. Set up the X's

```
> library(glmnet)
> X <- model.matrix(~(age + age2 + sex + race + marital)
                      *(sex + race + marital + hs +
                           assoc + coll + grad), Wages)
> X <- X[,-1]
> ncol(X)
[1] 101
```

2. LASSO command:

```
> lasso.fit <- cv.glmnet(x = X[training.samples,],
+                  y = train$log.WR, family="gaussian",
+                  alpha=1, standardize=FALSE)
```

3. Always refit! (LASSO introduces bias)

```
> betas <- coef(cvfit, s = "lambda.1se")
> model <- which(betas[2:length(betas)]!=0)
> post.lasso <- lm(train$log.WR ~ X[training.samples,model])
```

Trees

Also very popular in machine learning.

- Easy to do and easy to interpret
- Fit nonlinearities and selection automatically
  - No need to specify X_j^2 or $X_j \times X_k$ ahead of time
  - "Adaptive" learning of the universe (kind of)

Build a tree iteratively, based on sub-group **averages**:

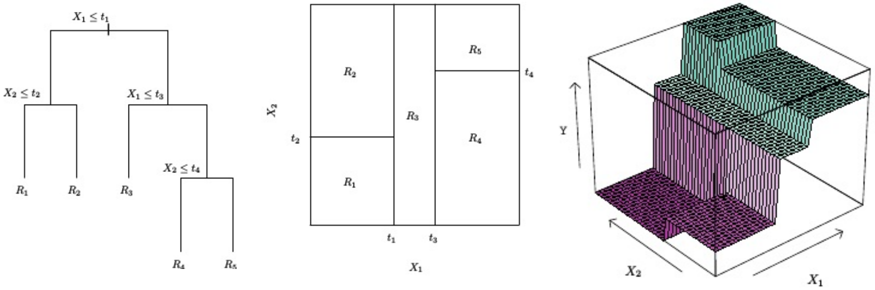
1. Initialize $\hat{Y} = \bar{Y}$
2. For each X_j **one at a time**, find cutoff t_j that **most** improves the predictions:

$$\hat{Y} = \begin{cases} \bar{Y}_{X_j \leq t_j} & \text{if } X_j \leq t_j \\ \bar{Y}_{X_j > t_j} & \text{if } X_j > t_j \end{cases} \qquad \bar{Y}_{X_j \leq t_j} = \frac{\sum_{i=1}^{n} Y_i \mathbb{1}\{X_j \leq t_j\}}{\#\{X_j \leq t_j\}}$$

3. Whatever **single** X_j, t_j is the best, keep that one split
4. Repeat, each time trying to improve on the prior iteration

We're just dividing up the X space into rectangles, and the prediction is just the in-cell average

- It's all just **dummy** variables! But we **learn** which ones.



$\Rightarrow$ Simple model in each cell, but complex/nonlinear overall.

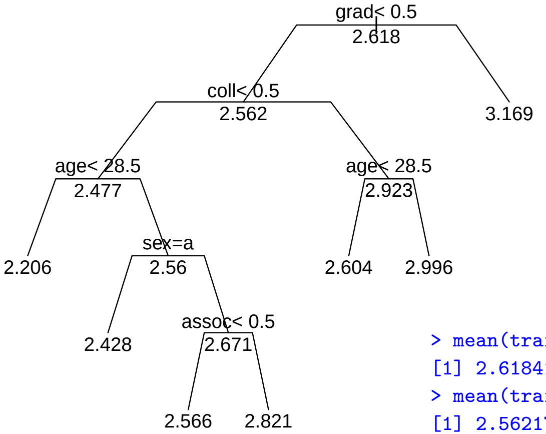
Example: back to the Census data.

```
> library(rpart)
> reg.tree <- rpart(log.WR ~ age*sex + ., data=train)
Error in rpart(log.WR ~ age * sex + ., data = train) :
  Trees cannot handle interaction terms
> reg.tree <- rpart(log.WR ~ ., data=train)

node), split, n, deviance, yval
      * denotes terminal node

 1) root 19052 7447.5600 2.618415
   2) grad< 0.5 17285 6120.0940 2.562171
     4) coll< 0.5 13994 4367.4600 2.477233
       8) age< 28.5 3265  769.0216 2.205987 *
       9) age>=28.5 10729 3285.1150 2.559778
        18) sex=F 4901 1381.2850 2.427549 *
        19) sex=M 5828 1746.0760 2.670975
          38) assoc< 0.5 3429 1009.7000 2.566228 *
          39) assoc>=0.5 2399  644.9775 2.820694 *
     5) coll>=0.5 3291 1222.3860 2.923341
      10) age< 28.5 609  128.0898 2.603981 *
      11) age>=28.5 2682 1018.0800 2.995857 *
   3) grad>=0.5 1767  737.9081 3.168600 *
```

```
>   plot(reg.tree, uniform=TRUE, branch=0.6, margin=0.05)
>   text(reg.tree, use.n=FALSE, all=TRUE, cex=1)
```



```
> mean(train$log.WR)
[1] 2.618415
> mean(train$log.WR[train$grad==0])
[1] 2.562171
> mean(train$log.WR[train$grad==1])
[1] 3.1686
```

Out-of-sample Prediction

Out-of-sample prediction error is the Gold Standard for comparing models. (If what you care about is prediction.)

We've used the training data to select models via

- F-testing
- Stepwise selection
- LASSO

Now we have to see how they perform on the test data

```
> errorBIC <- predict(fwdBIC, newdata=test) - test$log.WR
> mean(errorBIC^2)
> ...
> errorLASSO <- a little more work, see R code
> round(MSE,4)
wagereg1 wagereg2 wagereg3 wagereg4 
  0.2983   0.2973   0.2975   0.2975 
     BIC      AIC    lasso     tree 
  0.2972   0.2967   0.2980   0.3210 
```

So AIC wins this round. But remember train/test samples are random!

- AIC wins by a **tiny** amount
- Different data sets, different results.
- Try `set.seed(5)` and see what happens
  - Challenge: find a seed such that LASSO wins
  - Harder challenge: find a seed such that trees wins

(no cheating by using forests!)

We can use all the same ideas with **logistic regression**.

Example: German lending data from last lecture. Select borrower characteristics that predict default.

```
> credit <- read.csv("germancredit.csv")
> train <- sample.int(nrow(credit), 0.75*nrow(credit))
> base <- glm(GoodCredit~history3, family=binomial,
+     data=credit[train,])
> full <- glm(GoodCredit~., family=binomial,
+     data=credit[train,])
> fwdBIC <- step(base, scope=formula(full),
+     direction="forward", k=log(length(train)))
```

The null model has credit history as a variable, since I'd include this regardless, and we've left-out 200 points for validation.

Or we can use LASSO to find a model

```
> cvfit <- cv.glmnet(x=X[train,], y=credit$GoodCredit[train],
+ family="binomial", alpha=1, standardize=TRUE)
> betas <- coef(cvfit, s = "lambda.1se")
> model.1se <- which(betas[2:length(betas)]!=0)
```

Which variables were selected?

```
> colnames(X[,model.1se])
 [1] "checkingstatus1A13" "checkingstatus1A14" "duration2"
 [4] "history3A31"        "history3A34"        "purpose4A41"
 [7] "purpose4A43"        "purpose4A46"        "amount5"
[10] "savings6A64"        "savings6A65"        "employ7A74"
[13] "installment8"       "status9A93"         "others10A103"
[16] "property12A124"     "age13"              "otherplans14A143"
[19] "housing15A152"      "foreign20A202"
```

Comparing with the validation set:

```
> predBIC <- predict(fwdBIC, newdata=credit[-train,],
+     type="response")
> errorBIC <- credit[-train,1] - (predBIC >= .5)
```

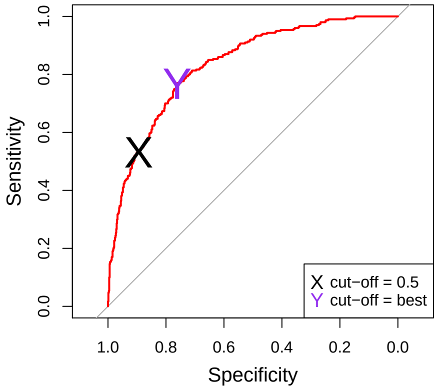
(LASSO takes a few extra lines)

Misclassification rates

```
> c(full=mean(abs(errorfull)), BIC=mean(abs(errorBIC)),
+   lasso=mean(abs(errorLasso.1se))
+   )
 full   BIC lasso
0.292 0.296 0.280
```

- Our LASSO model classifies 72% borrowers correctly
- BIC and full slightly worse

We can also use **ROC curves** to select a model.



Sensitivity

true positive rate

Specificity

true negative rate

Many related names: recall, precision

positive predictive value, . . .

What else can we do?

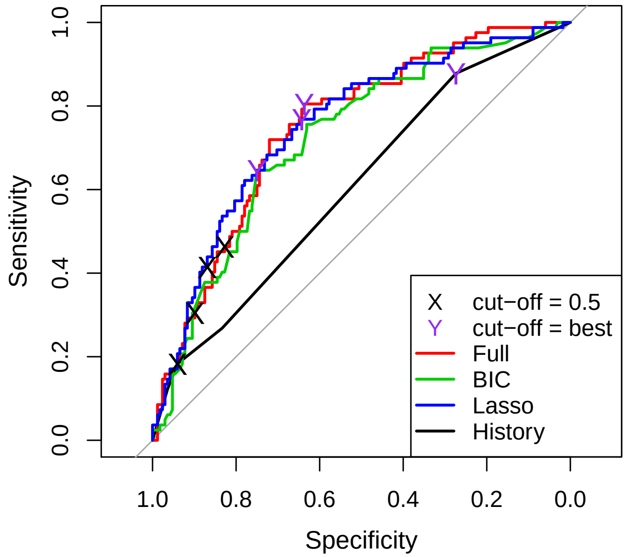
```
> rbind( coords(roc.full, "best"),
+ coords(roc.BIC, "best"), coords(roc.lasso, "best") )
     threshold specificity sensitivity
[1,] 0.3210536   0.7558140   0.6538462
[2,] 0.4065372   0.8313953   0.6153846
[3,] 0.2030214   0.6279070   0.7820513
```

Is misclassification improved?

```
> errorBIC <- credit[-train,1] - (predBIC >= xy.BIC.best[1])
> errorfull <- credit[-train,1] - (predfull >= xy.full.best[1])
> errorLasso.1se <- credit[-train,1] - (predLasso.1se >= xy.lasso.best[1])
> c(full=mean(abs(errorfull)), BIC=mean(abs(errorBIC)),
+   lasso=mean(abs(errorLasso.1se)))
 full   BIC lasso
0.276 0.236 0.324
```

Different goals, different results.

```
> A bunch of code...
> see week7-Rcode.R
```



Area Under the Curve (AUC)

```
> c(auc(roc.base), auc(roc.full), auc(roc.BIC), auc(roc.lasso))
[1] 0.6077962 0.7498548 0.7233595 0.7521051
```

Not surprising, given the picture.

Formal testing possible, never really useful

```
> roc.test(roc.full,roc.BIC)

        DeLong's test for two correlated ROC curves

data:  roc.full and roc.BIC
Z = -0.72882, p-value = 0.4661
alternative hypothesis: true difference in AUC is not equal to 0
sample estimates:
AUC of roc1 AUC of roc2
  0.7492546   0.7700507
```

Putting it all together ...

Regardless: Remember your discipline based knowledge and don't get lost in fancy regression techniques.

A strategy for building regression models:

1. Decide on the universe of variables.
   - Think about appropriate transformations!
   - Limitation: None of our methods learn nonlinearities automatically. (cf. random forests, deep nets)
2. Reduce the set of X variables with BIC/LASSO/Other.
   - Any variables you *need* to keep?
3. Plot residual diagnostics and take remedies (transformations, etc).
4. Evaluate your model predictions.

Inference After Model Selection

Up until now, we have used the same model for the two different regression questions: prediction and inference.

Is the model we select for prediction good for inference?
Not necessarily!

Then how should we choose variables for “correct” inference?

- We have few answers. This is still an active research area.

One thing we can do: a single coefficient with LASSO selection:

$$Y = \beta_0 + \beta_1 X_1 + \underbrace{\beta_2 X_2 + \beta_3 X_3 + \cdots + \beta_{p-1} X_{p-1} + \beta_p X_p}_{\text{Which variables to “control” for?}} + \varepsilon$$

Inference question: do the returns to age diminish at the same rate for men and women? (just for an example.)

Remember what "X" was . . .

```
> X <- model.matrix(~(age + age2 + sex + race + marital)
                  *(sex + race + marital + hs +
                       assoc + coll + grad), Wages)
> colnames(X)
 [1] "age"
 ...
[29] "age2:sexM"
...
```

So we want inference for β_{29}, "controlling" for all the rest.

Turn to the `hdm` package, made right here at BOOTH

- This is a relationship question, so we use the full data now
- Set the "`index`" argument to the variable of interest

```
> library(hdm)
> hdm.ci <- rlassoEffects(x = X, y = Wages$log.WR, index=c(29))
> summary(hdm.ci)
[1] "Estimates and significance testing of the effect
of target variables"
          Estimate. Std. Error t value Pr(>|t|)
age2:sexM -1.269e-04  6.196e-05  -2.047   0.0406 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As mentioned before, this is a very hard problem:

- Since very few variables are influential, testing is useless.
- You cannot consider all transformations and interactions.
- It is easy to overfit, which leads to bad predictions.

For industrial mining, more powerful tools are needed.

There are two full classes in this area: 41201 & 41204.

Model selection – final words

You have many new tools at your disposal, but don't forget the fundamentals.

- Easy to do, hard to do well
- Never forget your discipline based knowledge
- You think! Not the computer
- You can never consider **everything**
- Always try for the **simple** model
- Prediction is a great equalizer!

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

But what about inference?

- Causal inference?
- Testing several β_j? Prediction intervals?

Coming Up Next

Two more classes

1. Time series data
2. Advanced discrete outcomes

Then final and projects!