BUSN 20800: Big Data

Lecture 3: Model Selection

Dacheng Xiu

University of Chicago Booth School of Business

Model Decisions and Model Building

Out-of-Sample vs In-Sample performance

Screening

Stepwise strategies

Regularization paths and the lasso

OOS experiments and Cross Validation

Information Criteria

- AIC
- BIC and Bayesian model selection

Variable Selection as Model Selection

Suppose we observe y, an outcome of interest, and $x = (x_1, ...x_p)'$, a set of potential explanatory variables or predictors.

The variable selection problem:

To find the "best" subset of x's for predicting y

Particularly of interest when p is large and $x_1, ...x_p$ is thought to contain many redundant or irrelevant variables.

Given a set of covariates x, the model is always $\mathbb{E}[y|\mathbf{x}] = f(\mathbf{x}\boldsymbol{\beta})$.

- Gaussian (linear): $y \sim \mathrm{N}(\mathbf{x}\boldsymbol{\beta}, \sigma^2)$.
- Binomial (logistic): $\mathrm{p}(y = 1) = e^{\mathbf{x}\boldsymbol{\beta}}/(1 + e^{\mathbf{x}\boldsymbol{\beta}})$.

Some basic facts about linear models

Estimation of β: Maximize Likelihood $\Leftrightarrow$ Minimize Deviance

Likelihood LHD($\boldsymbol{\beta}$|data) is

$$\mathrm{p}(\mathit{data}|\boldsymbol{\beta}) = \mathrm{p}(y_1|\mathbf{x}_1, \boldsymbol{\beta}) \times \mathrm{p}(y_2|\mathbf{x}_2, \boldsymbol{\beta}) \cdots \times \mathrm{p}(y_n|\mathbf{x}_n, \boldsymbol{\beta}).$$

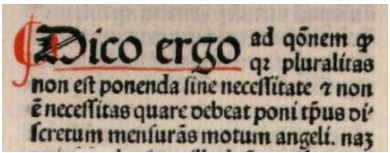
Deviance dev($\boldsymbol{\beta}$) is proportional to -2 log LHD($\boldsymbol{\beta}$)

$\hat{\boldsymbol{\beta}}$ denotes the maximum likelihood estimator of $\boldsymbol{\beta}$ (the most likely value of $\boldsymbol{\beta}$ to have generated the data)

Goodness-of-fit (GOF) is summarized by

$$R^2 = 1 - \frac{\mathrm{dev}(\hat{\boldsymbol{\beta}})}{\mathrm{dev}(\hat{y} = \bar{y})}.$$

Guiding Principle for Variable Selection



Ockham's Razor: **Plurality ought not be posited without necessity.**

Brevity is the soul of wit.

Overly complicated models lead to bad forecasts.

The only R^2 we ever really care about is out-of-sample R^2

The difference between *in* and *out* of sample R^2 is what data are used to fit $\hat{\boldsymbol{\beta}}$ and what the deviances are calculated on.

Example: Linear Regression

For in-sample R^2, we have data $[\mathbf{x}_1, y_1] \dots [\mathbf{x}_n, y_n]$ and you use this data to fit $\hat{\boldsymbol{\beta}}$. The deviance is then, say for linear regression,

$$\text{dev}(\hat{\boldsymbol{\beta}}) = \sum_{i=1}^{n} (y_i - \mathbf{x}_i' \hat{\boldsymbol{\beta}})^2.$$

The denominator is

$$\text{dev}(\hat{y} = \bar{y}) = \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

For out-of-sample R^2, observations $1 \ldots n$ are still used to fit $\hat{\boldsymbol{\beta}}$ but the deviance is now caluclated over $n$ew observations, say

$$\text{dev}(\hat{\boldsymbol{\beta}}) = \sum_{i=n+1}^{n+m} (y_i - \mathbf{x}_i' \hat{\boldsymbol{\beta}})^2$$

The denominator choice for OOS R^2 has certain freedom.

- A common choice is based on $\bar{y}$ from in sample data.
  Why not out of sample $\bar{y}$?

$$\text{dev}(\hat{y} = \bar{y}) = \sum_{i=n+1}^{n+m} (y_i - \bar{y})^2.$$

- Or one may use 0 or other estimators of β based on in sample data.

Ultimately, OOS R^2 is a comparison of deviances between two estimators. It can be negative!

Example: Business News and Business Cycles

An approach to measuring the state of the economy via textual analysis of news

- estimate a topic model from the full text content of 800,000 Wall Street Journal articles for 1984–2017
- use these text-based monthly topic attention to track macroeconomic indicators

Hundreds of topics useful or debilitating?

$\mathbf{x}$ is 179 input signals (topics attention), y is inflation (can also be GDP growth, unemployment rate, S&P 500 returns, etc.)

Source: Bybee, Kelly, Manela, and Xiu, The Structure of Economic News (2021).

THE STRUCTURE OF ECONOMIC NEWS

We propose an approach to measuring the state of the economy via textual analysis of business news. From the full text content of 800,000 Wall Street Journal articles for 1984–2017, we estimate a topic model that summarizes business news as easily interpretable topical themes and quantifies the proportion of news attention allocated to each theme at each point in time. We then use our news attention estimates as inputs into statistical models of numerical economic time series. We demonstrate that these text-based inputs accurately track a wide range of economic activity measures and that they have incremental forecasting power for macroeconomic outcomes, above and beyond standard numerical predictors. Finally, we use our model to retrieve the news-based narratives that underly "shocks" in numerical economic data.

Authors

Leland Bybee

Yale School of Management

Bryan T. Kelly

Yale SOM; AQR Capital Management, LLC; National Bureau of Economic Research (NBER)

Asaf Manela

Washington University in St. Louis - John M. Olin Business School

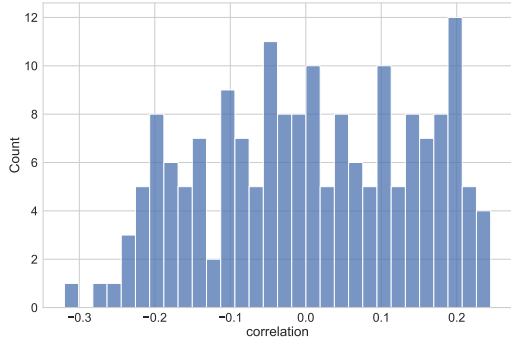
Dacheng Xiu

University of Chicago - Booth School of Business

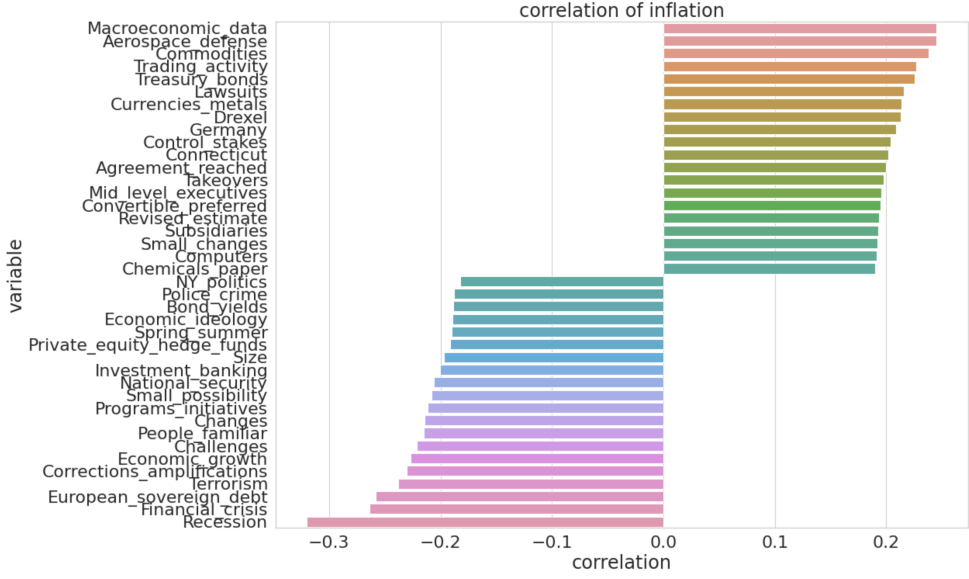
Reference: Bybee, Leland and Kelly, Bryan T. and Manela, Asaf and Xiu, Dacheng, The Structure of Economic News (September 1, 2019). Available at SSRN

Screening

The histogram of 179 correlations between each topic's attention and inflation:



By only considering topics with abs correlations greater than a certain threshold, we can reduce the total number of covariates.


correlation of inflation
Macroeconomic_data
Aerospace_defense
Commodities
Trading_activity
Treasury_bonds
Lawsuits
Currencies_metals
Drexel
Germany
Control_stakes
Connecticut
Agreement_reached
Takeovers
Mid_level_executives
Convertible_preferred
Revised_estimate
Subsidiaries
Small_changes
Computers
Chemicals_paper
NY_politics
Police_crime
Bond_yields
Economic_ideology
Spring_summer
Private_equity_hedge_funds
Size
Investment_banking
National_security
Small_possibility
Programs_initiatives
Changes
People_familiar
Challenges
Economic_growth
Corrections_amplifications
Terrorism
European_sovereign_debt
Financial_crisis
Recession
variable
−0.3
−0.2
−0.1
0.0
0.1
0.2
correlation

IS vs OOS

A *cut* model, using 100 signals (that have correlations ≥ 0.1), has $R^2_{cut} = 0.41$.
This is much smaller than the full model's $R^2_{full} = 0.58$.

In-Sample (IS) R^2 *always* increases with more covariates.
This is exactly what MLE $\hat{\beta}$ is fit to maximize.
But how well does each model predict *new* data?

An out-of-sample (OOS) experiment

- split the data into 5 random subsets ('folds').
- Do 5x: fit model $\hat{\beta}$ using only 4/5 of data, and record R^2 on the left-out subset.

These OOS R^2 give us a sense of how well each model can predict data that it has not already seen.

OOS full model

Full model has mean OOS R^2 of -8.35.

The full model is terrible. It is overfit and worse than $\bar{y}$.
Negative R^2 are more common than you might expect.

In-Sample R^2 can be misleadingly optimistic.

Using OOS experiments to choose the best model is called *cross validation*. It will be a big part of our big data lives.

Selection of 'the best' model is at the core of all big data.

But before getting to selection, we first need strategies to come up with a good set of candidate models to choose from.

Screening as a selection tool?

Maybe we grab models with covariates selected based on different correlation thresholds and compare?

Problems with screening:

It may keep redundant variables and omit variables that are important only when combined.

In post-screening regression, multicollinearity causes problems.

Bringing together variables that are useful individually does not mean that they will be useful collectively.

Forward stepwise regression

With p predictors, there are 2^p models to choose from.
We cannot enumerate them all.

Forward stepwise procedures: start from a simple 'null' model, and incrementally update fit to allow slightly more complexity.

Better than backwards methods

- The 'full' model can be expensive or tough to fit, while the null model is usually available in closed form.
- Jitter the data and the full model can change dramatically (because it is overfit). The null model is always the same.

Stepwise approaches are 'myopic': they find the best solution at each step without thought to global path properties.

Forward stepwise regression algorithm

The `forward_selected()` function executes a common routine:

- Fit all univariate models. Choose that with lowest AIC and put that variable – say $x_{(1)}$ – in your model.
- Fit all bivariate models including $x_{(1)}$ ($y \sim \beta_{(1)}x_{(1)} + \beta_j x_j$), and add x_j from one with lowest AIC to your model.
- Repeat: min AIC by adding one variable to your model.

You stop when some model selection rule (AIC) is lower for the current model than for any of the models that add one variable.
We will talk about AIC later.

A stepwise forward model uses 29 variables, $R^2 = 0.38$.

Subset selection

SS: Enumerate *all* candidate models by applying maximum likelihood estimation for subsets of coefficients, with the rest set to zero.
SS is impossible for p as small as 50!

Forward stepwise is a faster heuristic.
`forward_selected()` is still very slow.

A related subtle (but massively important) issue is stability.

With small n (relative to p), MLEs have high *sampling variability*: they change a lot from one dataset to another. So which MLE model is 'best' changes a lot.

$\Rightarrow$ predictions based upon the 'best' model will have high variance. And big variance leads to big expected errors.

Regularization

The general idea behind regularization:

impose restrictions on the MLE estimates such that they are suitably disciplined (stable and purposeful).

What do we mean by 'suitable restrictions'?

- We would like our estimates $\hat{\beta}$ to be less variable,
- We would like our estimates $\hat{\beta}$ to yield better prediction,
- We would like our estimates $\hat{\beta}$ to have exact zeros.

We will **penalize** solutions β that are not desirable. Ultimately, we are departing from optimality to stabilize the system (bias-variance tradeoff).

Using all predictors $x_1, ..., x_p$, the *maximum likelihood estimator* $\hat{\beta}_{MLE}$ minimizes deviance:

$$\hat{\beta}_{MLE} = \underset{\beta}{argmin} -\frac{2}{n} \log LHD(\beta)$$

$\hat{\beta}_{MLE}$ can only be obtained when $p < n$, it can be unstable and it cannot achieve exact zeros.

The penalized maximum likelihood estimator $\hat{\beta}_{PMLE}$ minimizes deviance **plus a penalty**:

$$\hat{\beta}_{PMLE} = \underset{\beta}{argmin} -\frac{2}{n} \log LHD(\beta) + pen(\beta)$$

Penalty incurs **cost** when choosing solutions that are far from our preferences.

Decision theory: Cost in Estimation

Decision theory is about how to behave optimally under uncertainty. It is based on the idea that choices have costs.
Estimation and hypothesis testing: what are the costs?

Estimation:

Deviance is the cost of distance between data and the model.
Recall: $\sum_i (y_i - \hat{y}_i)^2$ or $-\sum_i y_i \log(\hat{p}_i) - (1 - y_i) \log(1 - \hat{p}_i)$.

Testing:

Since $\hat{\beta}_j = 0$ is *safe*, it should cost us to decide otherwise.

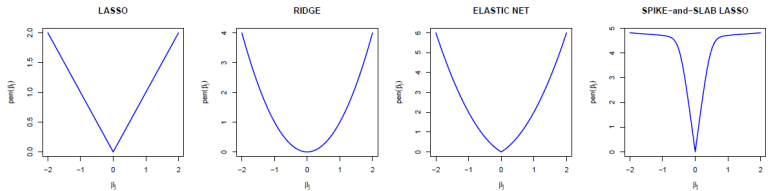
$\Rightarrow$ The cost of $\hat{\beta}$ is deviance plus a penalty away from zero.

Regularized Regression

Sparsity-inducing penalization:

$$\hat{\beta_{PMLE}} = \min \left\{ -\frac{2}{n} \log \mathrm{LHD}(\boldsymbol{\beta}) + \lambda \sum_j pen(\beta_j) \right\}$$

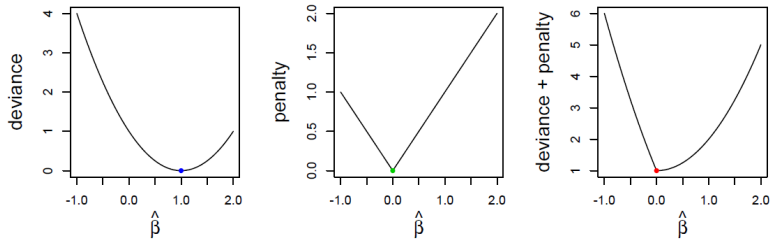
$\lambda > 0$ is the penalty weight, *pen* is a cost (penalty) function.
pen(β) will be lowest at $\beta = 0$ and we pay more for $|\beta| > 0$.



Options: ridge β^2, lasso $|\beta|$, elastic net $\alpha\beta^2 + (1-\alpha)|\beta|$, ...

Penalization can yield automatic variable selection

The minimum of a smooth + spiky function can be at the spike.



The PMLE estimator is **shrunk to zero** $\Rightarrow$ automatic variable selection

For instance, LASSO has this property.

You can think of the LASSO as the *modern least squares.*

More about the LASSO

The lasso fits $\hat{\boldsymbol{\beta}}$ to minimize $-\frac{2}{n}\log \text{LHD}(\boldsymbol{\beta}) + \lambda \sum_j |\beta_j|$.

How to pick λ?

We'll do this for a *sequence* of penalties $\lambda_1 > \lambda_2 ... > \lambda_T$.

Then we can apply model selection tools to choose best $\hat{\lambda}$.

Path estimation:

Start with big λ_1 so big that $\hat{\boldsymbol{\beta}} = \mathbf{0}$.

For $t = 2 \dots T$: update $\hat{\boldsymbol{\beta}}$ to be optimal under $\lambda_t < \lambda_{t-1}$.

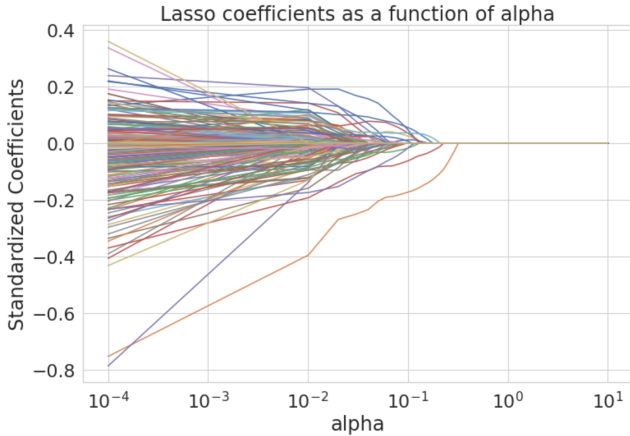
Since estimated $\hat{\boldsymbol{\beta}}$ changes smoothly along this path:

- It's fast! Each update is easy.
- It's stable: optimal λ_t may change a bit from sample to sample, but that won't affect the model much.

It's a better version of forward stepwise selection.

Path plots

The whole enterprise is easiest to understand visually.



The algorithm moves *right to left*.

The y-axis is standardized $\hat{\boldsymbol{\beta}}$ (each line a different $\hat{\beta}_j$) as a function of λ_t.

Running a lasso

Once you have your `x` and `y`, running a lasso is easy.

```
model = LassoCV(cv=5,random_state = 0, max_iter=10000)
model.fit(X, y)
alpha_ = model.alpha_
```

And you can run a lasso based on AIC

```
model_aic = LassoLarsIC(criterion='aic')
model_aic.fit(X, y)
alpha_aic_ = model_aic.alpha_
```

Or based on BIC

```
model_bic = LassoLarsIC(criterion='bic')
model_bic.fit(X, y)
alpha_bic_ = model_bic.alpha_
```

Scale matters

Penalization is sensitive to the scale of $x_1, ..., x_p$. e.g., $x\beta$ has the same effect as $(2x)\beta/2$, but $|\beta|$ is twice as much penalty as $|\beta/2|$.

It would not be 'fair' to penalize $\beta^{'}s$ equally if $x^{'}s$ were not on the same scale.

You can multiply β_j by $\text{sd}(x_j)$ in the cost function to standardize.

That is, minimize $-\frac{2}{n}\log \text{LHD}(\boldsymbol{\beta}) + \lambda \sum_j \text{sd}(x_j)|\beta_j|$.

$\Rightarrow \beta_j$'s penalty is calculated per effect of 1SD change in x_j.

How much regularization?

The lasso minimizes $-\frac{2}{n} \log \mathrm{LHD}(\boldsymbol{\beta}) + \lambda \sum_j |\beta_j|$.

This 'sparse regularization' auto-selects the variables.

Sound too good to be true? You need to choose λ.

Think of $\lambda > 0$ as a signal-to-noise filter: *like squelch on a radio.*

We'll use cross validation or information criteria to choose.

Path algorithms are key to the whole framework:

⋆ They let us quickly enumerate a set of candidate models.

⋆ This set is stable, so selected 'best' is probably pretty good.

Prediction vs Interpretability

Model selection can be used to tackle two tasks:

- learning about the data-generating-mechanism. (what are the contributing variables and can we interpret them?)
- prediction.

None of your models will be 'true' for complicated HD systems.
Instead, just try to get as close to the truth as possible.

Parsimony principle: If two models do similarly well in predicting the unseen data, choose the simpler one.

- Overly simple models will 'underfit' available patterns.
- Complicated models 'overfit', and make noisy predictors.

The goal is to find the sweet spot in the middle.

Prediction-driven Model Selection: it is all about prediction.

A recipe for model selection.

1. Find a manageable set of candidate models
   (i.e., such that fitting all models is fast).
2. Choose amongst these candidates the one with
   best predictive performance *on unseen data*.

1. is what the lasso paths provide.
2. Seems impossible! But it's not ...

We need to *estimate* the prediction accuracy associated with each model when applied on future unseen data.

Recall that predictive performance can be measured with 'deviance'.

Out-of-sample prediction experiments

We already saw an OOS experiment with topic attention. Implicitly, we were estimating predictive deviance (via R^2).

The procedure of using such experiments to do model selection is called Cross Validation (CV). It follows a basic algorithm:

For $k = 1 \dots K$,

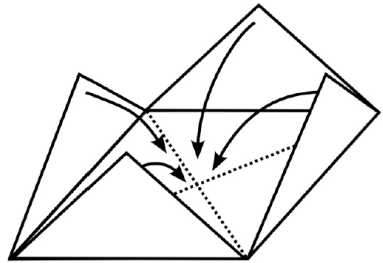
- Use a subset of $n_k < n$ observations to 'train' the model.
- Record the error rate for predictions from this fitted model on the left-out observations.

We'll usually measure 'error rate' as deviance. But alternatives include MSE, misclass rate, integrated ROC, or error quantiles.

You care about both average and spread of OOS error.

K-fold Cross Validation

One option is to just take
repeated random samples.
It is better to 'fold' your data.



- Sample a random ordering of the data (important to avoid order dependence)
- Split the data into K folds: 1st 100/K%, 2nd 100/K%, etc.
- Cycle through K CV iterations with a single fold left-out.

This guarantees each observation is left-out for validation,
and lowers the sampling variance of CV model selection.

Leave-one-out CV, with $K = n$, is nice but takes a long time.
$K = 5$ to 10 is fine in most applications.

CV Lasso

The lasso path algorithm minimizes $-\frac{2}{n}\log \text{LHD}(\boldsymbol{\beta}) + \lambda_t \sum_j |\beta_j|$ over the sequence of penalty weights $\lambda_1 > \lambda_2 \ldots > \lambda_T$.

This gives us a path of T fitted coefficient vectors, $\hat{\boldsymbol{\beta}}_1 \ldots \hat{\boldsymbol{\beta}}_T$, each defining deviance for new data: $-\log \text{p}(\mathbf{y}^{new} \mid \mathbf{X}^{new}\hat{\boldsymbol{\beta}}_t)$.

Set a sequence of penalties $\lambda_1 \ldots \lambda_T$.
Then, for each of $k = 1 \ldots K$ folds,

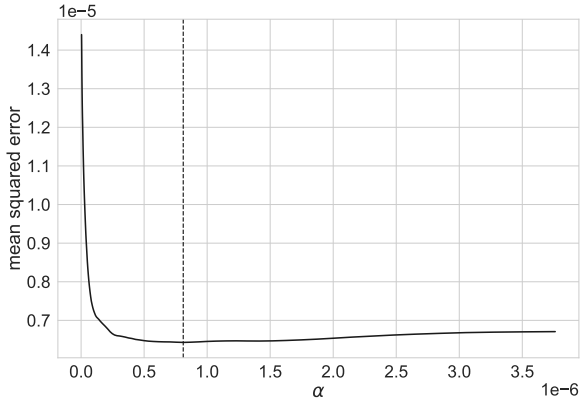
- Fit the path $\hat{\boldsymbol{\beta}}_1^k \ldots \hat{\boldsymbol{\beta}}_T^k$ on all data *except* fold k.
- Get fitted deviance *on left-out data*: $-\log \text{p}(\mathbf{y}^k \mid \mathbf{X}^k\hat{\boldsymbol{\beta}}_t)$.

This gives us K draws of OOS deviance for each λ_t.

Finally, use the results to choose the 'best' $\hat{\lambda}$, then re-fit the model to *all of the data* by minimizing $-\frac{2}{n}\log \text{LHD}(\boldsymbol{\beta}) + \hat{\lambda} \sum_j |\beta_j|$.

CV Lasso

Again, the routine is most easily understood visually.



Both selection rules are good; `1se` has extra bias for simplicity.

Problems with Cross Validation

It is time consuming: When estimation is not instant, fitting K times can become unfeasible even K in 5-10.

It can be unstable: imagine doing CV on many different samples. There can be large variability on the model chosen.

Still, some form of CV is used in most DM applications.

Also, be careful to not cheat: for example, with the screening cut model we've already used the full n observations to select the 67 strongest variables. It is not surprising they do well 'OOS'.
The rules: never use the same data twice (for selection and validation).

'Information Criteria' (IC): measure how much information is lost when we use our model to represent our data. They approximate distance between a model and 'the ideal'.

IC provide an avenue for **in-sample model comparisons**, by trading off *in-sample* deviance and *model complexity.*

You can apply them by choosing the model with minimum IC.

Most common is Akaike's AIC $=$ Deviance $+ 2df$.

$df =$ 'degrees of freedom' used in your model fit.
For lasso and MLE, this is just the # of nonzero $\hat{\beta}_j$.

AIC overfits in high dimensions

The AIC is atually estimating OOS deviance: what your deviance would be on another *independent* sample of size n.

IS deviance is too small, since the model is tuned to this data.
Some deep theory shows that IS - OOS deviance $\approx 2df$.

$$\Rightarrow \text{AIC} \approx \text{OOS deviance.}$$

It's common to claim this approx (i.e., AIC) is good for 'big n'.
Actually, its only good for big n/df.

In Big Data, df (# parameters) can be huge. Often $df \approx n$.
In this case the AIC will be a bad approximation: it overfits!

The BIC is Deviance + $\log(n) \times df$.

This *looks* just like AIC, but comes from a very different place.

BIC $\approx -\log \mathrm{p}(M_b|\mathrm{data})$, the 'probability that model b is true'.

$$\mathrm{p}(M_b|\mathrm{data}) = \frac{\mathrm{p}(\mathrm{data}, M_b)}{\mathrm{p}(\mathrm{data})} \propto \underbrace{\mathrm{p}(\mathrm{data}|M_b)}_{\text{LHD}} \underbrace{\mathrm{p}(M_b)}_{\text{prior}}$$

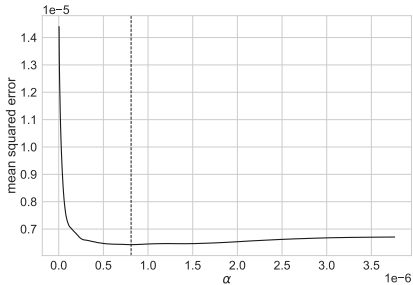
The 'prior' is your probability that a model is true *before* you saw any data. BIC uses a 'unit-info' prior: $\mathrm{N}\left[\hat{\boldsymbol{\beta}}, \frac{2}{n}\mathrm{var}(\hat{\boldsymbol{\beta}})^{-1}\right]$

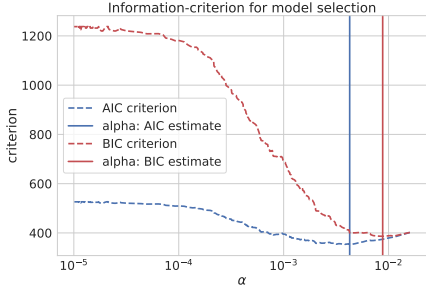
AIC tries to approx OOS deviance.

BIC is trying to get at the 'truth'.

IC and CV on the Topic Model Data

In practice, BIC works more like the 1se CV rule.
But with big n it chooses too simple models (it underfits).





With all of these selection rules, you get a range of answers.
If you have time, do CV. But AIC is fast and stable.

OOS Comparison for Various Lasso Estimators

- LassoCV R^2: 0.21
- LassoAIC R^2: 0.54
- LassoBIC R^2: 0.49