Cross Validation

Objectives

- Describe why cross validation is used
- Describe how to do it (especially k-fold cross validation)
- In the individual assignment, code it from scratch
- Aside: introduce feature selection/elimination
- Contrast cross validation with other statistical model comparison methods

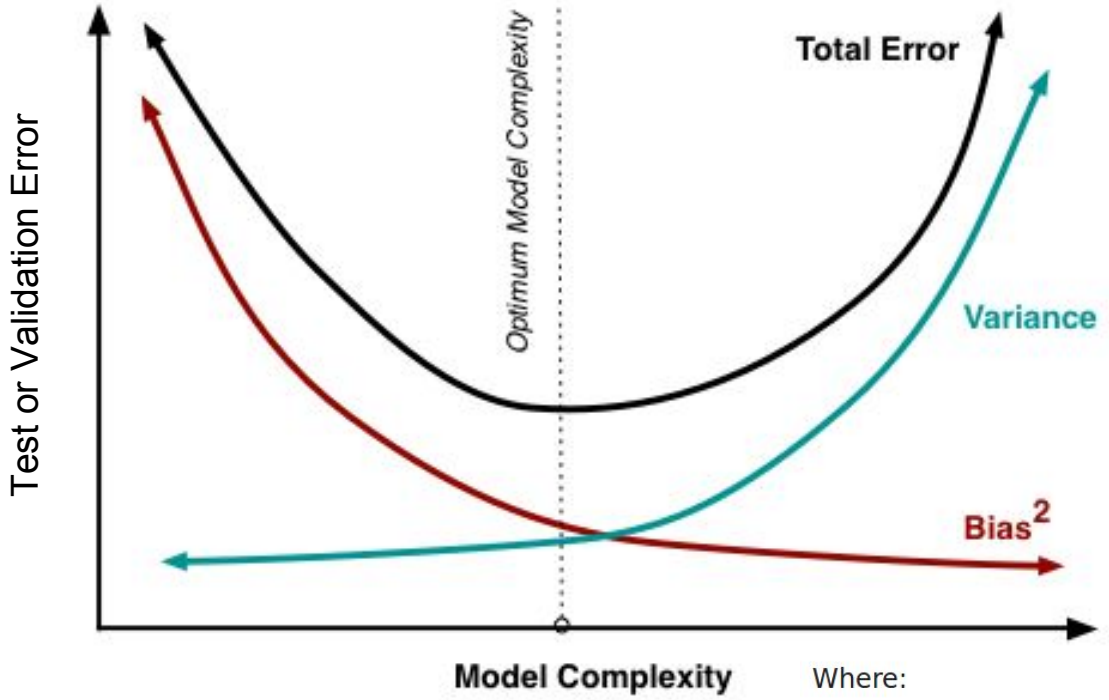
Motivation: We're stuck with the bias-variance tradeoff

So how is the "correct" model complexity chosen?

Model complexity can be the order of the fit, the number of features, interaction of features, number of splits (decision tree), number of neurons/layer in a neural net, number of layers...

We can't do anything to reduce the sampling error from the population, but can we find the model complexity that minimizes the sum of the bias^2 and the variance?

Typical Bias-Variance tradeoff behavior



$$\mathrm{E}\left[\left(y-\hat{f}\left(x\right)\right)^2\right]=\mathrm{Bias}\left[\hat{f}\left(x\right)\right]^2+\mathrm{Var}\left[\hat{f}\left(x\right)\right]+\sigma^2$$

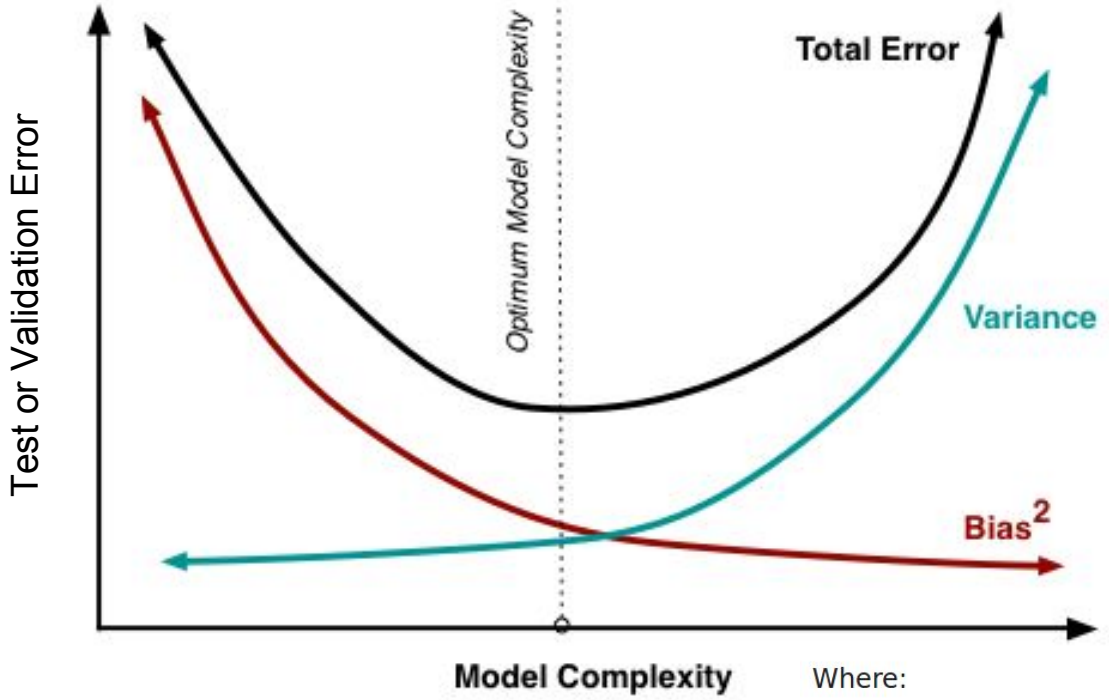
Where:

$$\mathrm{Bias}\left[\hat{f}\left(x\right)\right]=\mathrm{E}\left[\hat{f}\left(x\right)-f(x)\right]$$

and

$$\mathrm{Var}\left[\hat{f}\left(x\right)\right]=\mathrm{E}[\hat{f}\left(x\right)^2]-\mathrm{E}[f(x)]^2$$

Typical Bias-Variance tradeoff behavior



Bias: unknown

Variance: unknown

But, if we have data with targets that the model has **not** trained on, we can plot the expected residual for a given model complexity, and choose the complexity that gives the lowest residual.

$$\mathrm{E}\left[\left(y-\hat{f}\left(x\right)\right)^2\right]=\mathrm{Bias}\left[\hat{f}\left(x\right)\right]^2+\mathrm{Var}\left[\hat{f}\left(x\right)\right]+\sigma^2$$

Where:

$$\mathrm{Bias}\left[\hat{f}\left(x\right)\right]=\mathrm{E}\left[\hat{f}\left(x\right)-f(x)\right]$$

and

$$\mathrm{Var}\left[\hat{f}\left(x\right)\right]=\mathrm{E}[\hat{f}\left(x\right)^2]-\mathrm{E}[f(x)]^2$$

Cross validation

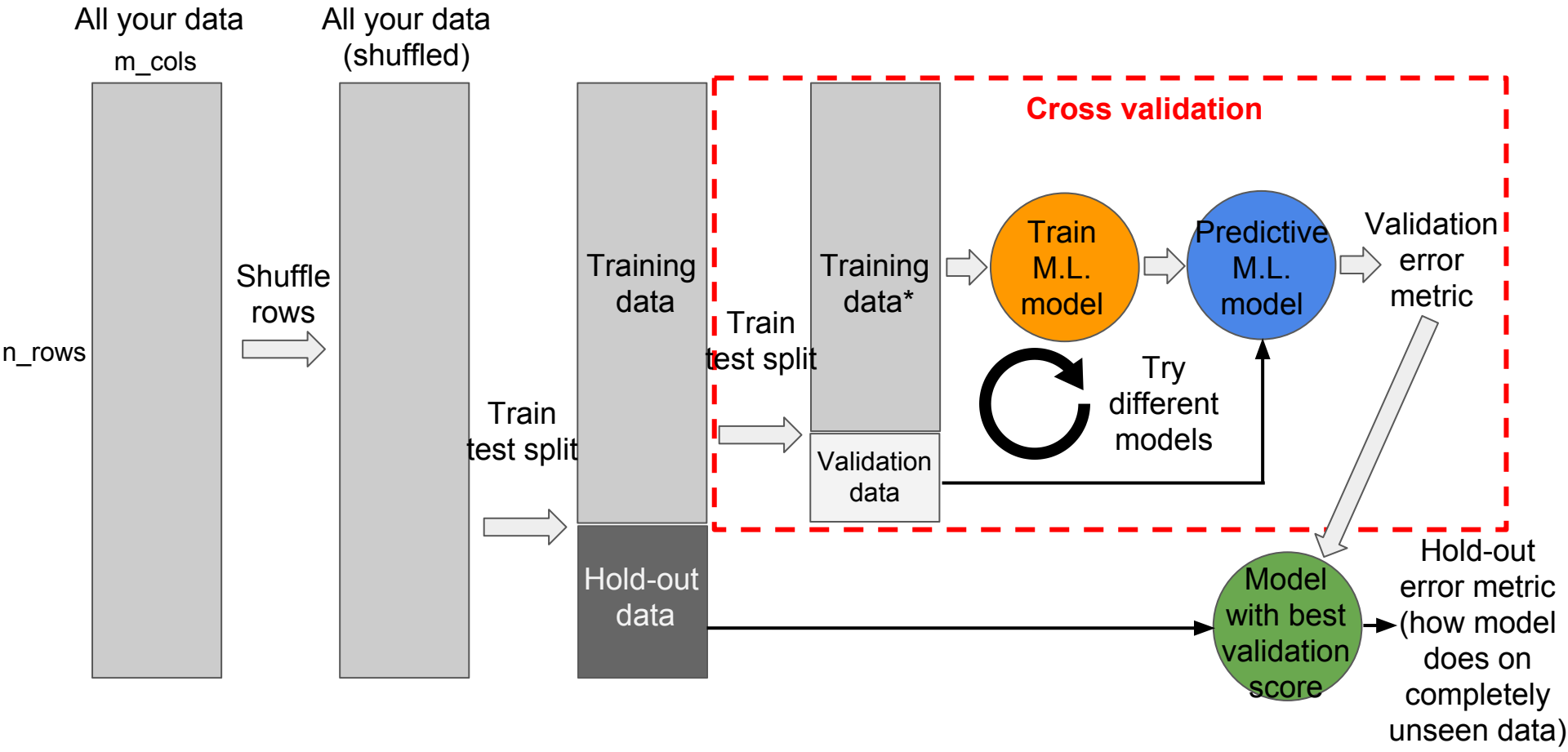
“Cross-validation ... is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.”
- Wikipedia

We use cross-validation for two things:

1) Attempting to quantify how well a model (of some given complexity) will predict on an unseen data set
2) Tuning hyperparameters of models to get best predictions.

Scikit-learn tangent (hyperparameters?)

Cross validation - illustrated

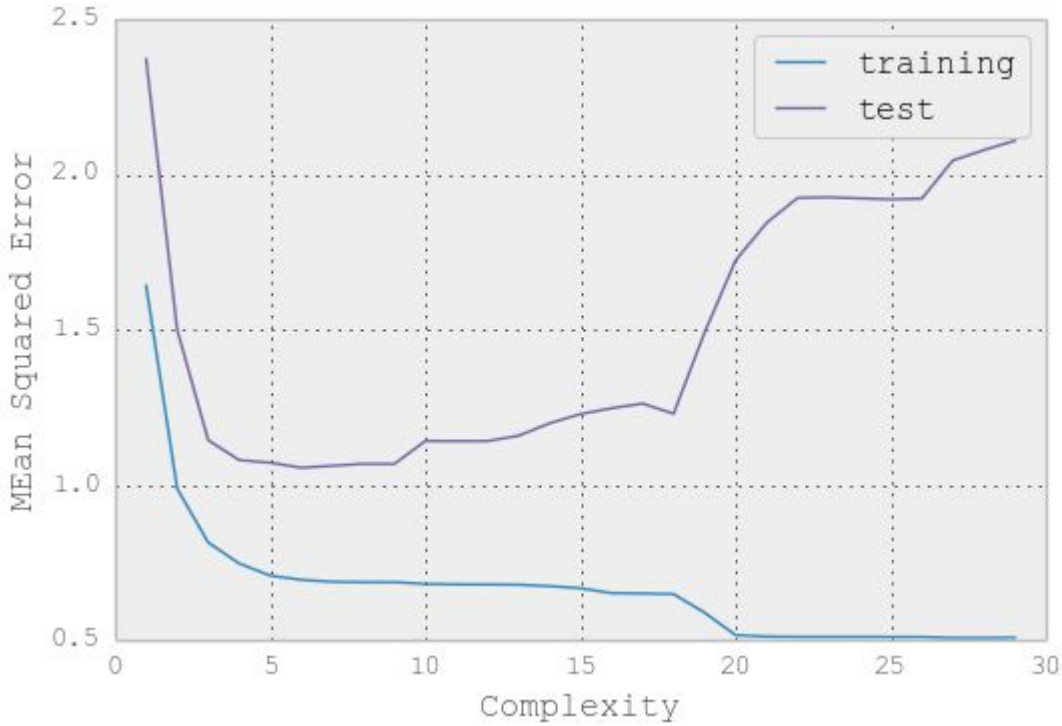


Cross validation - in words

1. Split your data (after splitting out hold-out set) into training/validation sets.
   70/30, 80/20 or 90/10 splits are commonly used
2. Use the training set to train several models of varying complexity.
   e.g. linear regression (w/ and w/out interaction features), neural nets, decision trees, etc.
3. Evaluate each model using the validation set.
   calculate R^2, MSE, accuracy, or whatever you think is best
4. Keep the model that performs best over the **validation** set.

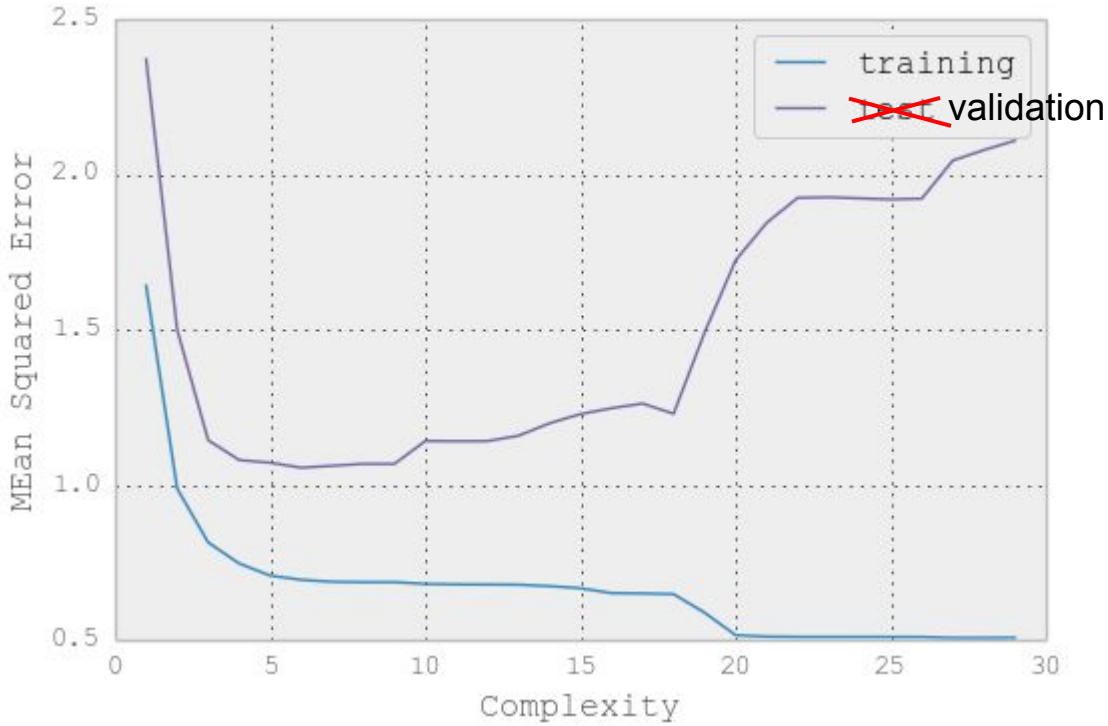
People use different terms for the splits.
All data -> Train, Test, Hold-out
All data -> Train, Validate, Test
All data -> Train, Validate, Hold-out
All the same idea.

Cross validation - what you should get



Number of features, interaction between features, order of features

Cross validation - what you should get



Which error is most important to minimize? Why?

What model complexity is best? How did you decide?

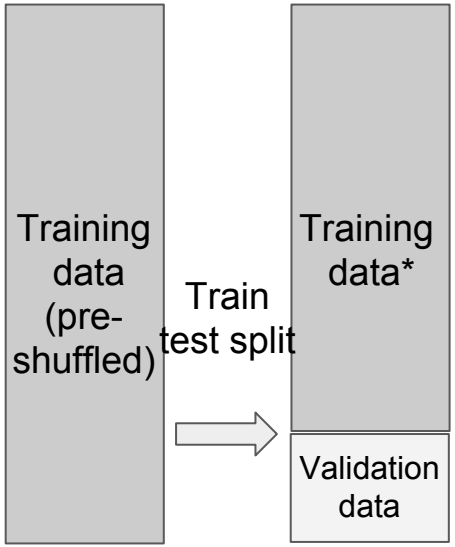
At the optimum model complexity, what is the bias? What is the variance?

Is it possible for the test error to be lower than the training error?

Number of features, interaction between features, order of features

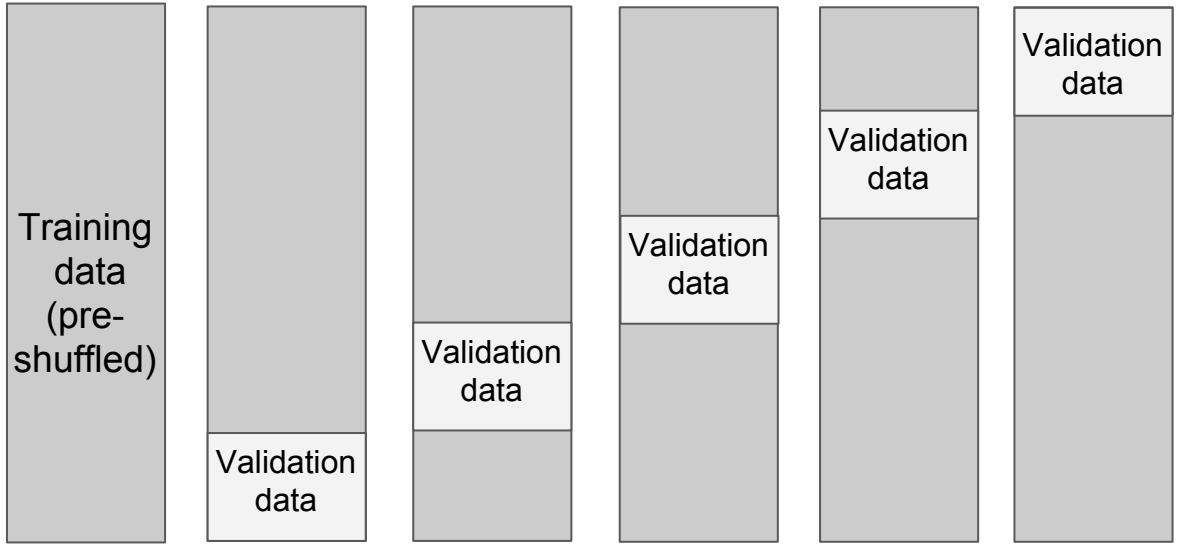
Types of Cross Validation

A single train-test split



After training get only 1 estimate of validation error (what if validation data very different from training data by chance?!?)

k-fold (showing k=5)



After training get k estimates of validation error from the same model complexity, so calculate the mean validation error from those five estimates. This gives a more robust, less variable estimate.

Special case of k-fold: k = n (Leave one out CV).
Models are highly correlated in LOOCV.

What to do if your model is overfitting

Pretty common. If you are starting with 5-10 features that can already be pretty complex. Often assume that if we start with linear regression this is “simple.” This is not necessarily so.

1. **Get more data…** (not usually possible/practical)

2. **Subset Selection:** keep only a subset of your predictors (i.e, dimensions)

3. **Regularization:** restrict your model’s parameter space (this afternoon)

4. **Dimensionality Reduction:** project the data into a lower dimensional space (later in course)

Subset selection

Best subset: Try every model. Every possible combination of *p* predictors

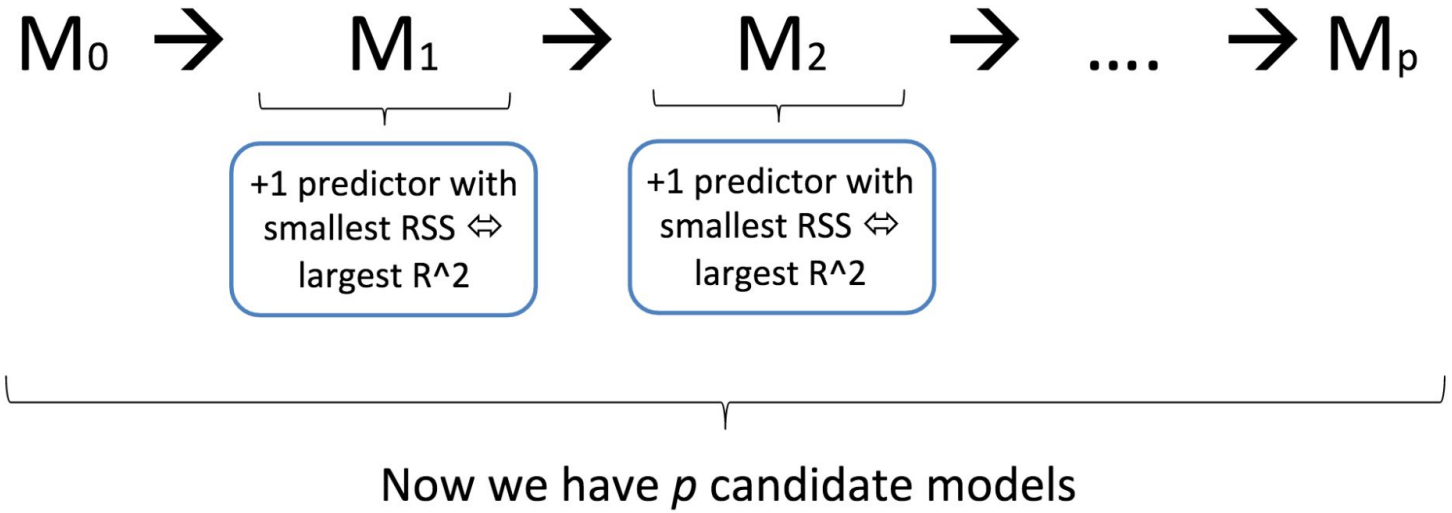
- Computationally intensive. 2^p possible subsets of *p* predictors
- High chance of finding a "good" model by random chance.

  … A sort-of monkeys-Shakespeare situation …

Stepwise: Iteratively pick predictors to be in/out of the final model.

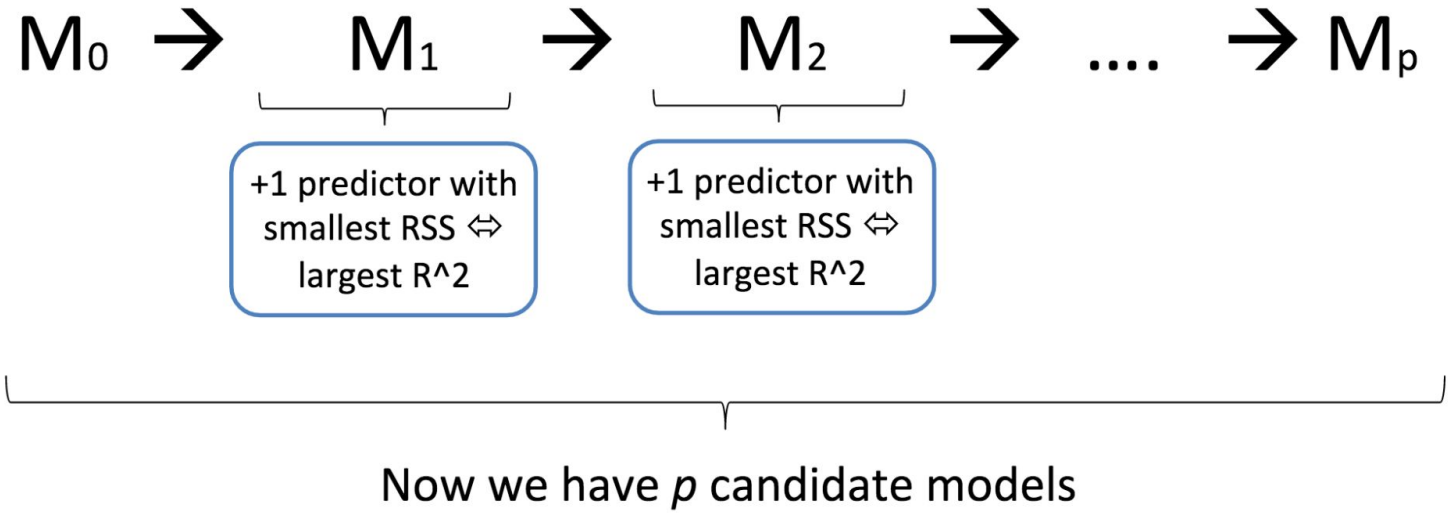
- Forward, backward, forward-backward strategies
  - Forward: starting with just one and adding more features, one-by-one
  - Backward: starting with them all, and removing one-by-one
- Sklearn features only backward recursive elimination.

Forward step-wise selection



Is R^2 a good way to decide amongst the resulting *(p+1)* candidates?

Forward step-wise selection



Is R^2 a good way to decide amongst the resulting *(p+1)* candidates?

R^2 doesn't penalize for model complexity, so no.
Use Mallow's C_p, or AIC, or BIC, or Adjusted R^2.

… or just use cross-validation with any error measurement.

Statistical metrics that penalize model complexity

$$C_p = \frac{1}{n}(RSS + 2p\hat{\sigma}^2)$$

Mallow's C_p
- p is the total # of parameters
- $\hat{\sigma}^2$ is an estimate of the variance of the error, ε

$$AIC = -2logL + 2 \cdot p$$

L is the maximized value of the likelihood function for the model estimated

$$BIC = \frac{1}{n}(RSS + log(n)p\hat{\sigma}^2)$$

This is Cp, except 2 is replaced by log(n). log(n) > 2 for n>7, so BIC generally exacts a heavier penalty for more variables

$$Adjusted\ R^2 = 1 - \frac{RSS/(n-p-1)}{TSS/(n-1)}$$

Similar to R^2, but pays price for more variables

Side Note: Can show AIC and Mallow's Cp are equivalent for linear case

Statistical metrics that penalize model complexity

$$C_p = \frac{1}{n}(RSS + 2p\hat{\sigma}^2)$$

Mallow's Cp
p is the total # of par
$\hat{\sigma}^2$ is an estim
ror, ε

$$AIC = -2logL + 2 \cdot p$$

likelihood
stimated

BIC

is Cp, except 2 is replaced by log(n).
log(n) > 2 for n>7, so BIC generally exacts a heavier penalty for more variables

$$Adj = 1 - \frac{RSS/(n-p-1)}{TSS/(n-1)}$$

Similar to R^2, but pays price for more variables

Yes, you can use these, but why? You have something better ....

Side Note: Can show AIC and Mallow's Cp are equivalent for linear case

Something better

Use cross validation

Objectives

- Describe why cross validation is used
- Describe how to do it (especially k-fold cross validation)
- In the individual assignment, code it from scratch
- Aside: introduce feature selection/elimination
- Contrast cross validation with other statistical model comparison methods