CS 412

JAN 30TH – SUBSET SELECTION

HTF – CHAPTER 3.3,4.3

Administrivia

M + T vesday

Office hours, my office (SEO931):

- Today: 5-7

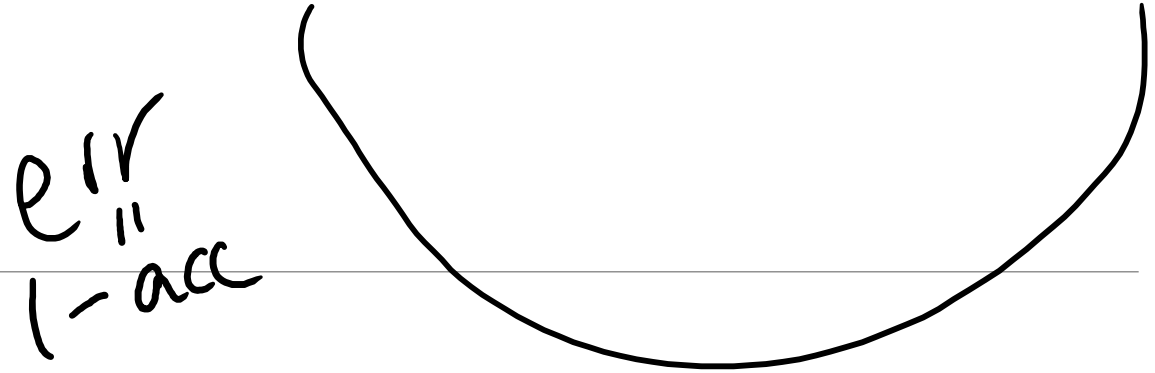
HW1 is due tonight on gradescope

- If you use jupyter notebook, you do not need to submit your code

Lecture capture

- Both sections have it
- If you look in the "tools" section, you'll see Echo360 which is the lecture capture
- The green screen affect was from my other computer while I was doing the code demo

Review



Bias vs. Variance

- Relevant to the cross validation errors
- Bias is the ability of the model to explain the data
  - How far away from the "actual" model is our model
  - Remember, all data has an implicit amount of noise, so you can never fully eliminate bias
  - High bias is an indicator of underfitting
- Variance is how much the bias changes for the model depending on the data
  - With 10-fold cross validation, if some runs are very accurate and some runs are not, then the model has high variance
  - High variance is an indicator of overfitting
- Because you cannot exactly calculate the noise of the data, there is no exact way to balance these two outcomes

metric = "

$E_{CV} \approx E_{test}$

Review

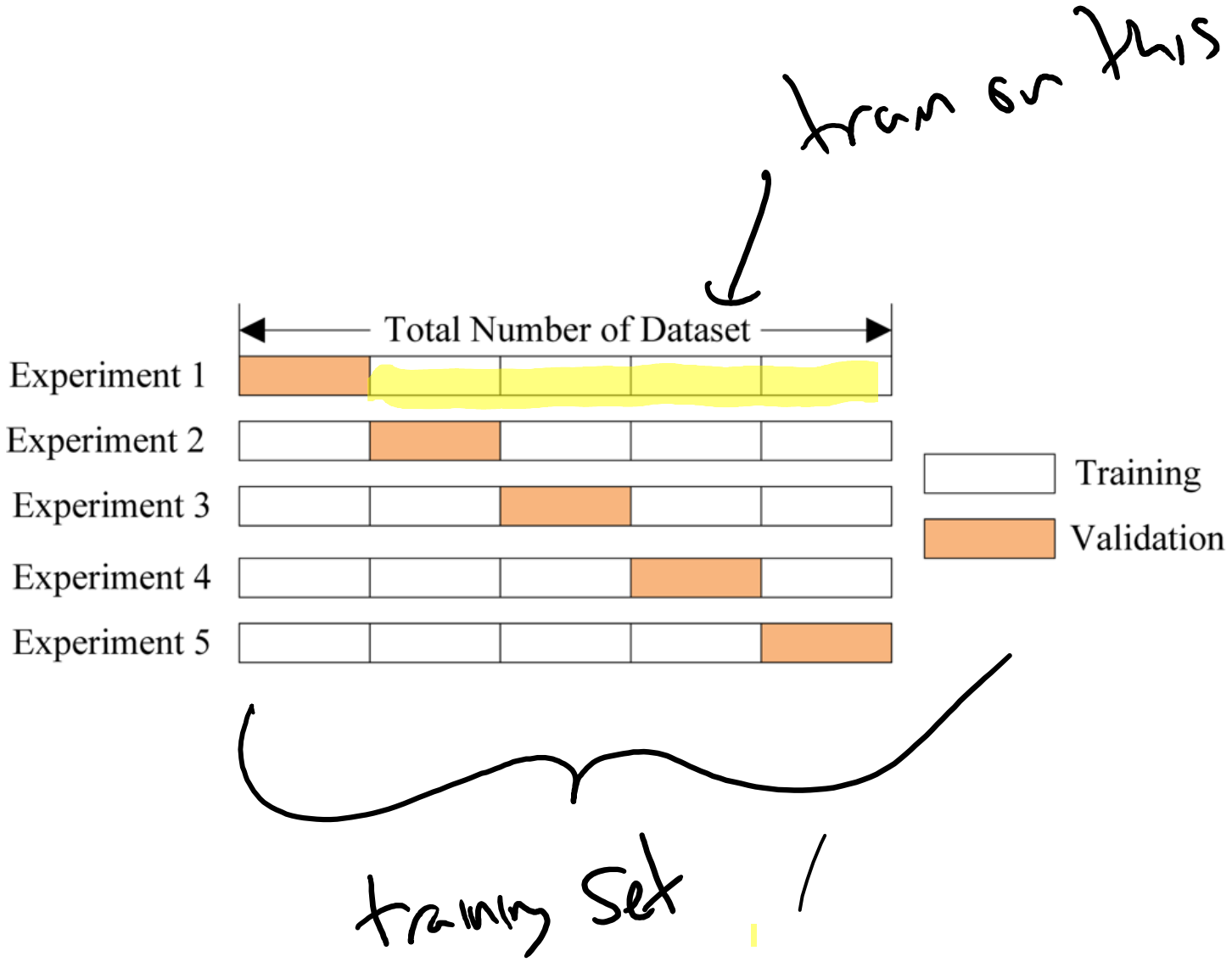
>200 elements in your training set

Training vs. Testing set

- We want to produce the best model we can (which is a reason to use a larger training set)
- However, more importantly, we want to accurately report the expected error of our data
- To do this, we need one final run on a testing set. If we test on the testing set with multiple models, we decrease the certainty of our final error estimate.

Modeling

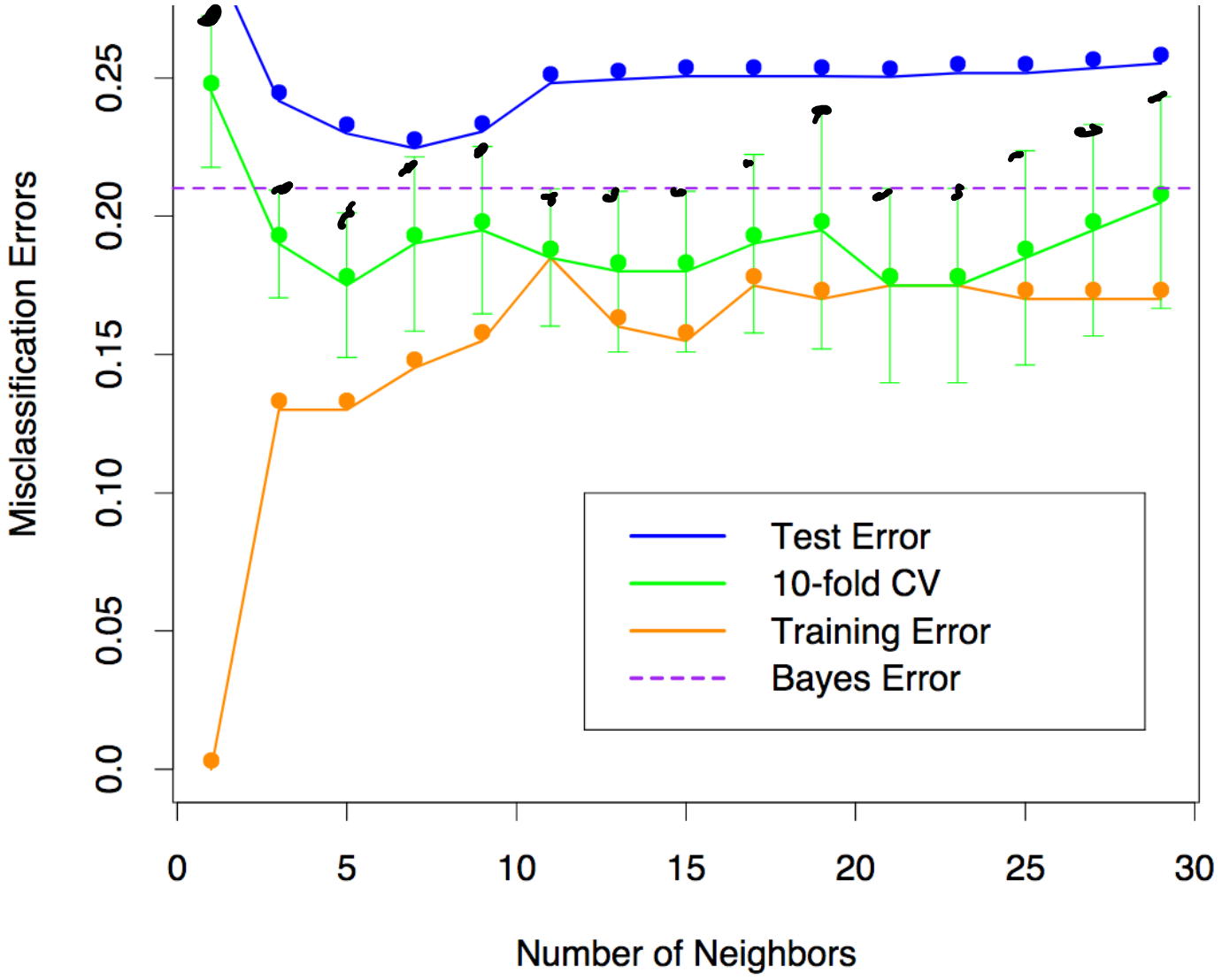
- ML is not about building the model itself, it is more about deciding what model building algorithm is most appropriate.
- You want to test multiple approaches on the training data (using cross-validation) and then pick which algorithm you think is best
- When you go to predict on new data, you will use the model trained on the whole data set



Review

Cross-validation

- This is showing 5-fold cross validation
- The model trains on all of the white partitions of the data and then predicts the validation data.
- This is repeated until all partitions have been used as the validation data
- The average and variance of these runs are used to inform decision making



Results

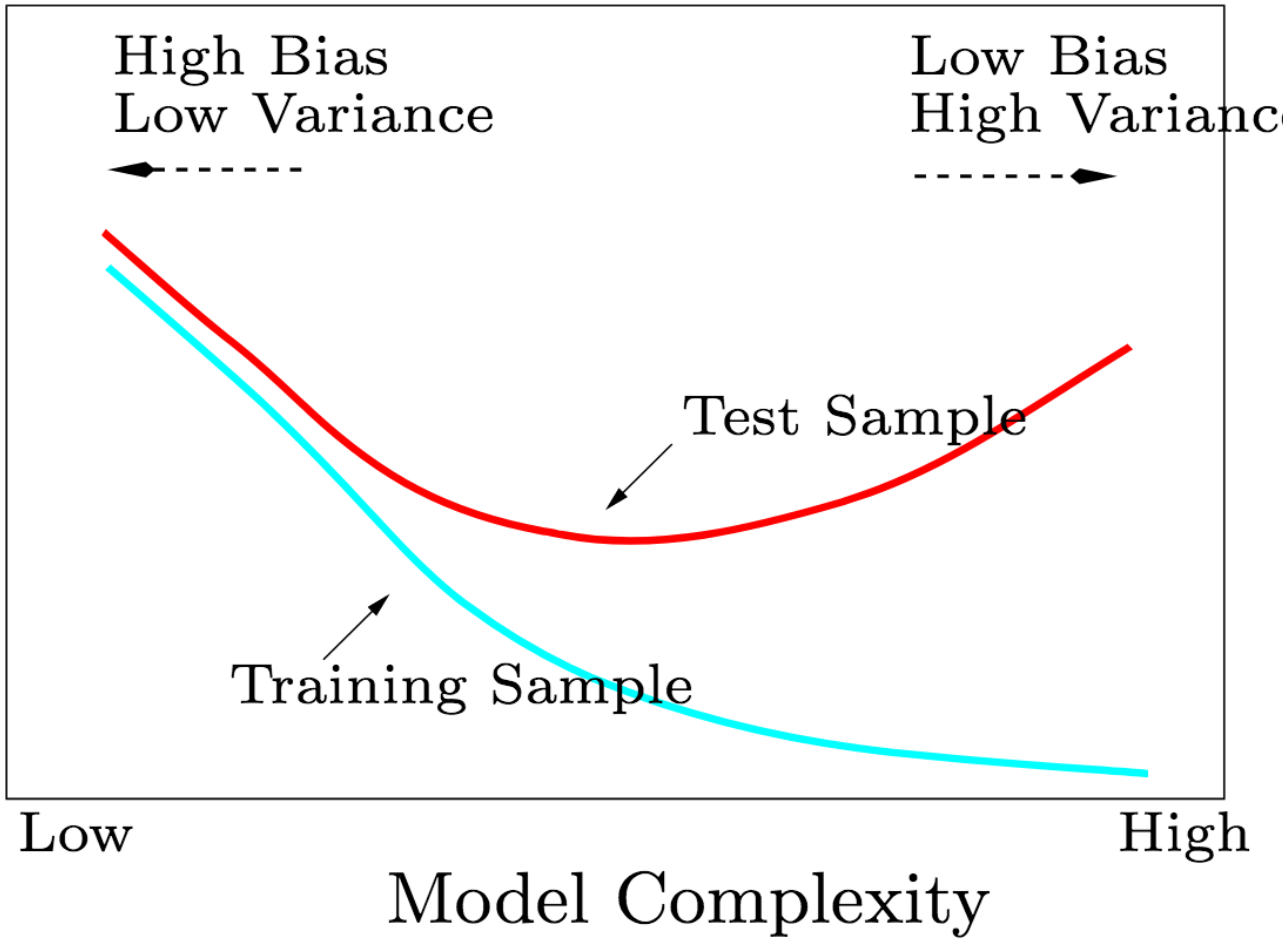
This illustrates a common trade-off

Bias vs. Variance

The more we test (and the more complicated our model), the lower our bias is.

However, we introduce more variance, which is represented in the test data.

$\bar{N} + 1.96\sigma$



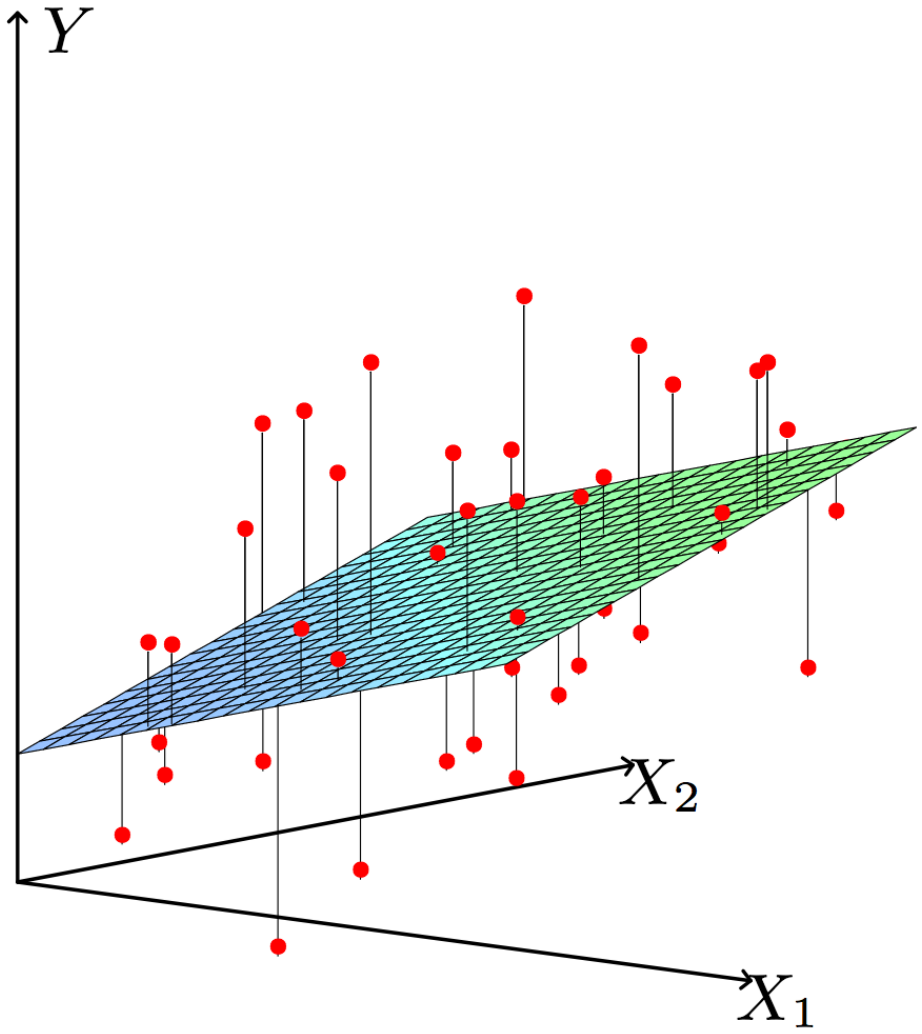
Results

This illustrates a common trade-off

Bias vs. Variance

The more we test (and the more complicated our model), the lower our bias is.

However, we introduce more variance, which is represented in the test data.



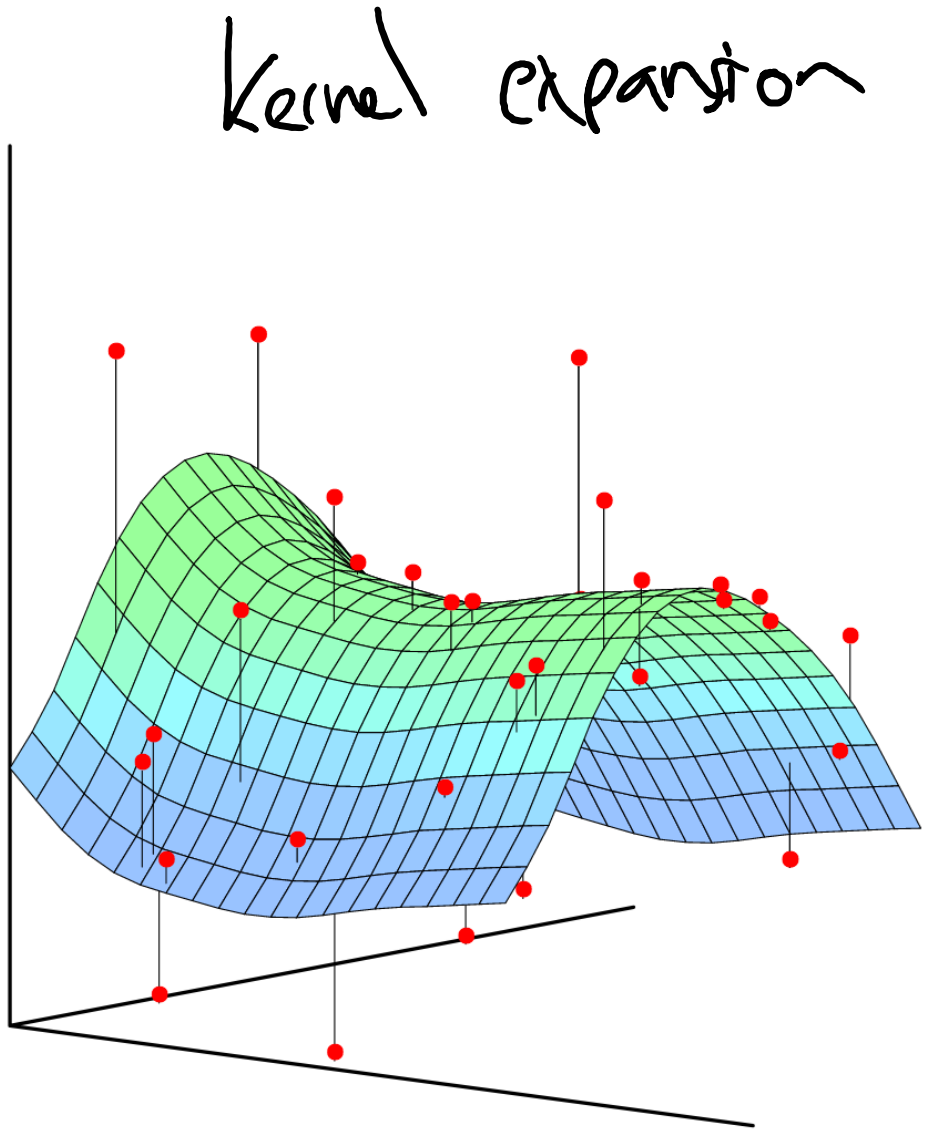
Linear Regression

Regression problems are trying to predict some output value ($Y \in \mathbb{R}$) which is a function of the input variables ($X \in \mathbb{R}^2$)

This is a different problem than *classification*

- kNN can be used for both
- So can linear models

Start with linear regression and move to logistic regression



Linear Regression

The more variables we include:

- higher our risk for overfitting
- higher our expected error
- more complex data can be modeled

Since this is a statistical approach, we can directly bound the error of the model

This is an easy approach for a more robust statistical (and interpretable) result

Linear Regression

$HP = 10{,}000 + 1{,}000 \text{ bedrooms} + 100 \text{ ft}^2$

$\beta_{HB} = [10^4, 10^3, 10^2]$

So what is the linear regression problem?

- For each of our p variables in X
- Apply some constant (not dependent on any X_i) multiple β

f(X) is our approximation of the output

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j.$$

Bounding the expected error

This leads to our first way of quantifying error statistically!

- *Notice that it depends on N-p-1. What is this? Degrees of freedom!*

- *Why do we need the constant* $\frac{1}{N-p-1}$*?*
  - This is to make our estimator unbiased
  - This is equivalent to degrees of freedom.
  - What if N = 3 and P = 2?
    - There is only one "degree of freedom" only one point can vary from the line as drawn

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

number of features

Bounding the expected error

This leads to our first way of quantifying error statistically!

ε is the error between the value of y (from the model) and it's reported value.

- We usually assume that this random error is normally distributed.
- What's an example of when it wouldn't be?

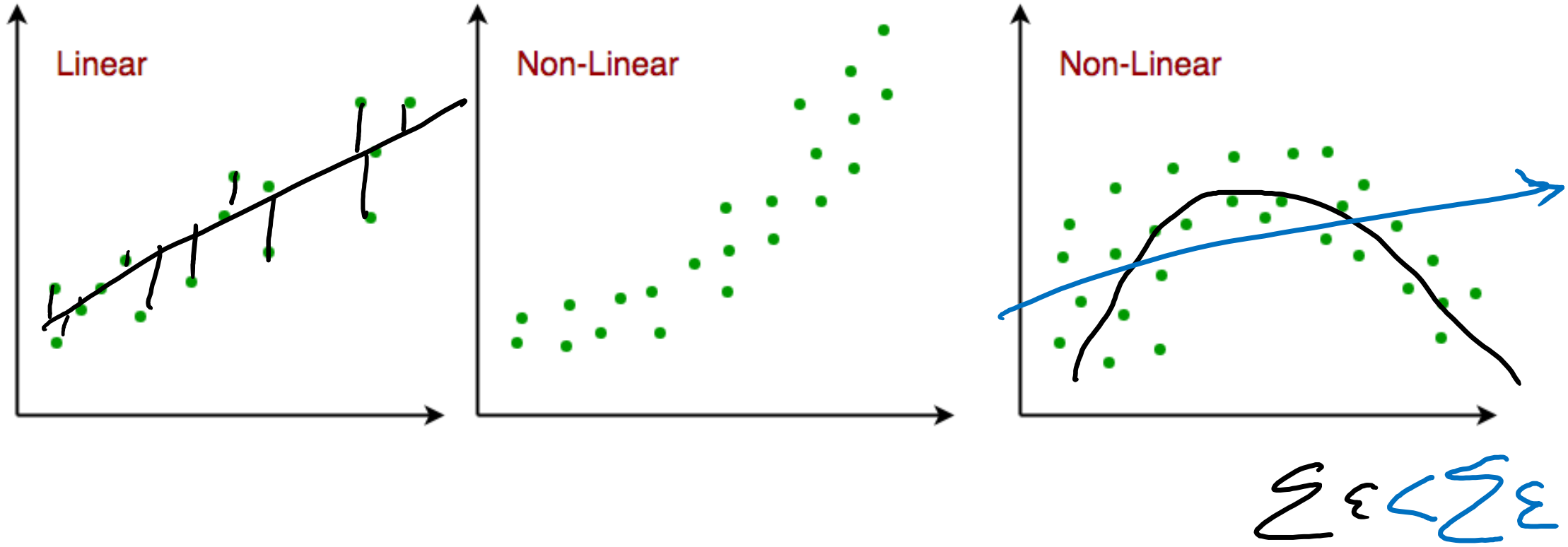
true for any model

$$Y = \mathrm{E}(Y|X_1,\ldots,X_p) + \varepsilon$$

$$= \beta_0 + \sum_{j=1}^{p} X_j\beta_j + \varepsilon,$$

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

Some portion of explainability


Linear
Non-Linear
Non-Linear

Subset selection

Since we can find the statistical impact of each variable
(or combination of variables using the F-statistic)

AND, since we pay a penalty (through degrees of freedom) for having more variables

We want to build our model on the most explanatory subset of variables

Subset selection

Since we can find the statistical impact of each variable
(or combination of variables using the F-statistic)

AND, since we pay a penalty (through degrees of freedom) for having more variables

We want to build our model on the most explanatory value for our variables

- *This leaves us with two options*
  - *Best Subset*
  - *Shrinkage methods*
    - LASSO

Why Reduce Dimensionality?

- Reduces time complexity: Less computation
- Reduces space complexity: Fewer parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

reduction of variance

Feature Selection vs Extraction

Feature selection:

- Choosing $k<d$ important features, ignoring the remaining $d-k$
- Subset selection algorithms

Feature extraction:

- Project the original x_i, $i=1,...,d$ dimensions to new $k<d$ dimensions, z_j, $j=1,...,k$

↳ 2D HW Digits dataset

Subset Selection

There are 2^d subsets of d features

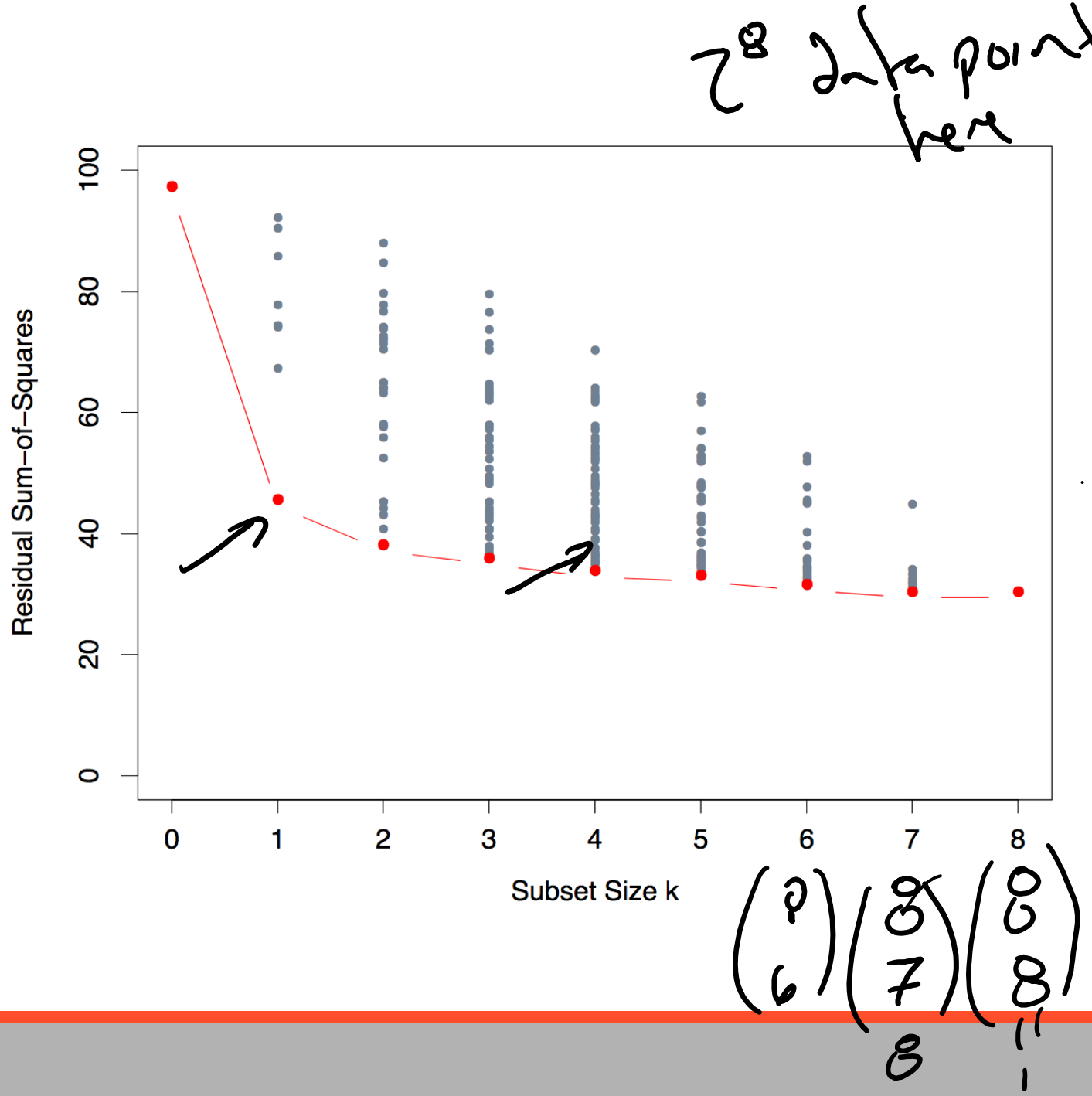
Forward search: Add the best feature at each step

- Set of features F initially $\emptyset$.
- At each iteration, find the best new feature
  $j = \text{argmin}_i E(F \cup x_i)$
- Add x_j to F if $E(F \cup x_j) < E(F)$

Hill-climbing $O(d^2)$ algorithm

Backward search: Start with all features and remove one at a time, if possible.

Floating search (Add k, remove l)



Subset Selection

For each subset size, there are multiple possible subsets

- It is non-trivial to find the optimal subset for each situation

Forward stepwise

Start with no features and then add the feature that explains the largest amount of the deviation . Continue until the new feature does not improve enough

- What are upsides and downsides?

↑ Speed

↓ Greedy

At each step retrain the model and find new weights

Forward stepwise

Start with no features and then add the feature that explains the largest amount of the deviation .

- What are upsides and downsides?
  - Greedy search, may not find the best possible subset
  - Not as computationally intensive, plus it will likely have lower variance, why?

Forward stepwise

Start with no features and then add the feature that explains the largest amount of the deviation .

- What are upsides and downsides?
  - Greedy search, may not find the best possible subset
  - Not as computationally intensive, plus it will likely have lower variance, why?
  - Higher bias

Forward stepwise

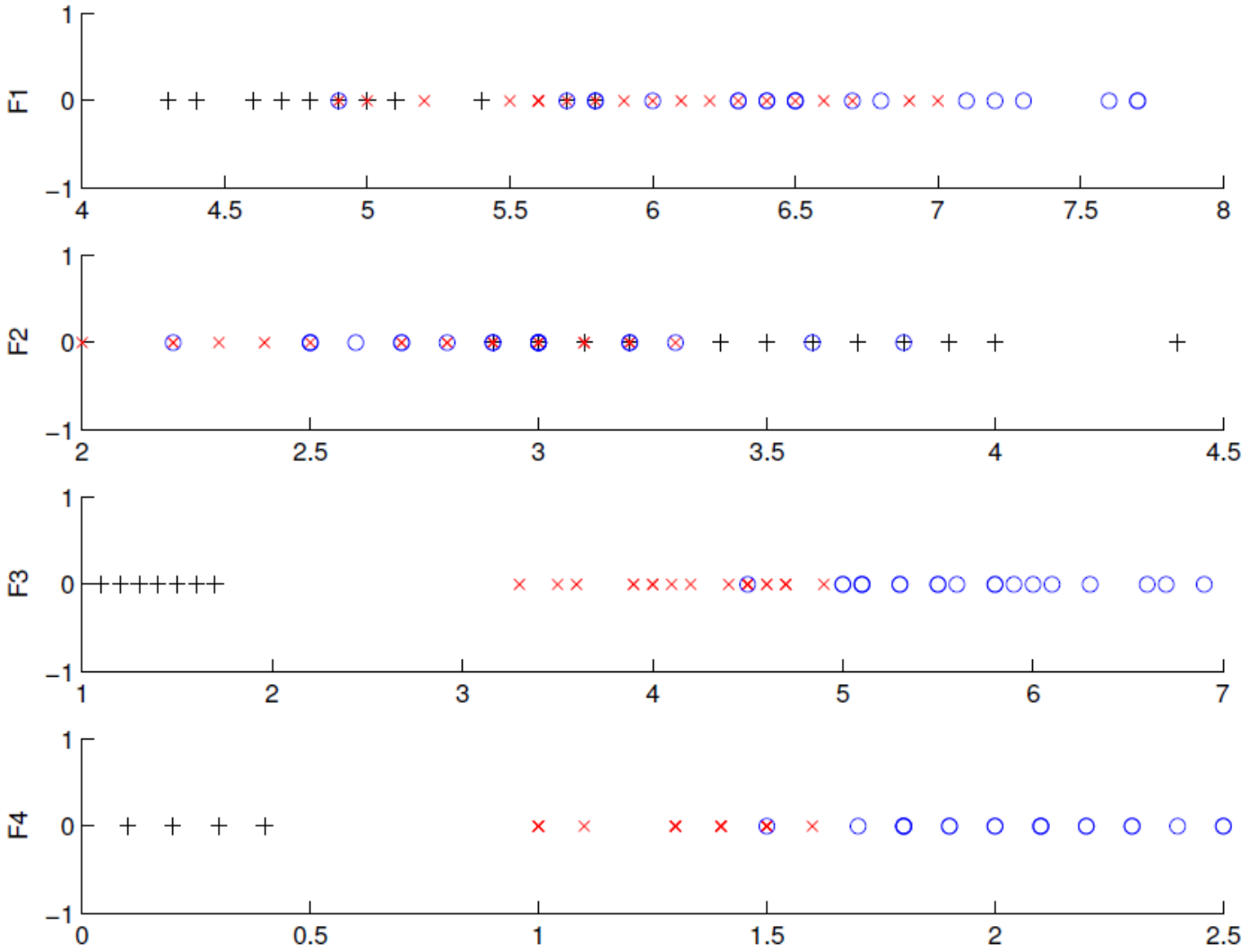
When selecting subsets, it is intractable to calculate the F-scores for all of the 2^d subsets

- Start with no features, and add features one at a time until we reach a certain cutoff
- *Usually, when we reach some proportion of the variance explained*
- Remember, since we don't have an overfitting problem with LR, we can try to minimize noise

Forward stepwise

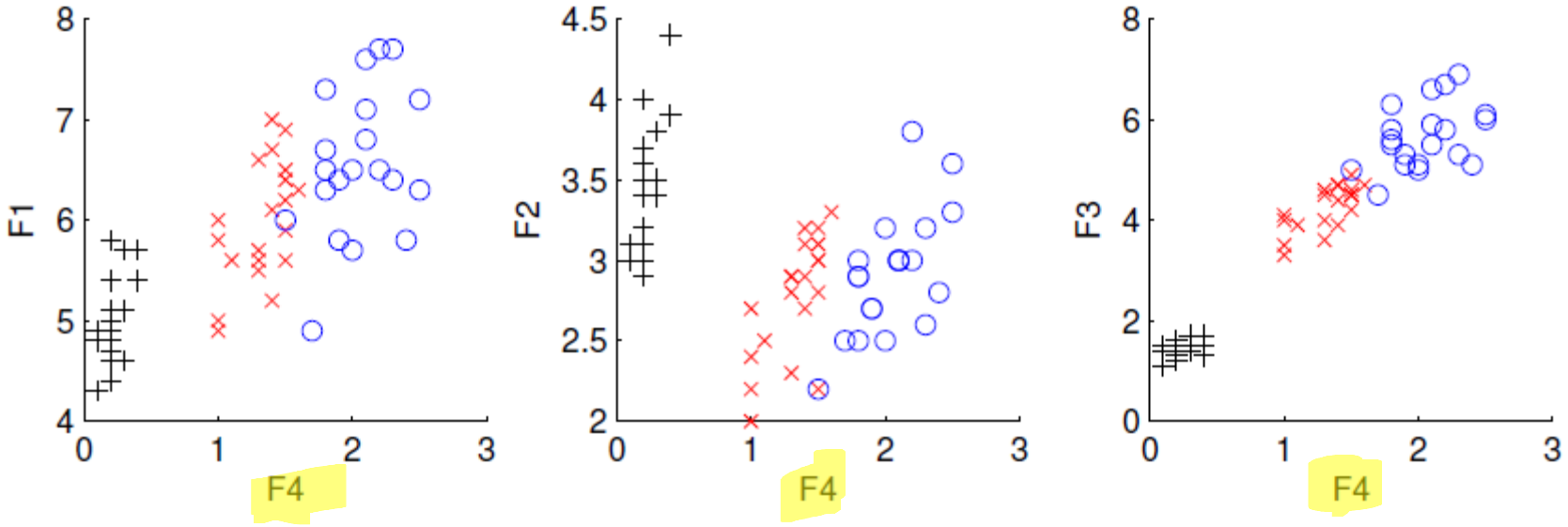
- Add the most significant feature until we reach the next value
- What is the most significant feature? (The one with minimum p-value)
- Notice that this is a *greedy algorithm,* it is not guaranteed to be optimal, but the optimal value is too difficult to calculate. Want to get as close as we can

Iris data: Single feature



Which of these two has the Chosen largest (p value) explanatory power

Iris data: Add one more feature to F4



Chosen

first selection

next choose F4, F3, F1 or F4, F3, F2

Backward-stepwise

Start with all the features and eliminate the least impactful feature until we fall below a certain threshold

- Can use the F-statistic (which is the statistical explanatory power of the whole model) but this can fall prey to certain explanatory problems (notably eliminating some necessary variables)
- This incurs a penalty for searching over multiple models, and the F-statistic does not incorporate this
- Many modern packages instead use AIC, which is a comparative standard for model quality
- AIC measures the extent to which the model explains the data
  - **This is not an absolute metric of model quality. AIC values across problems cannot be compared**

alt approach: individual p values for each

Forward Stagewise

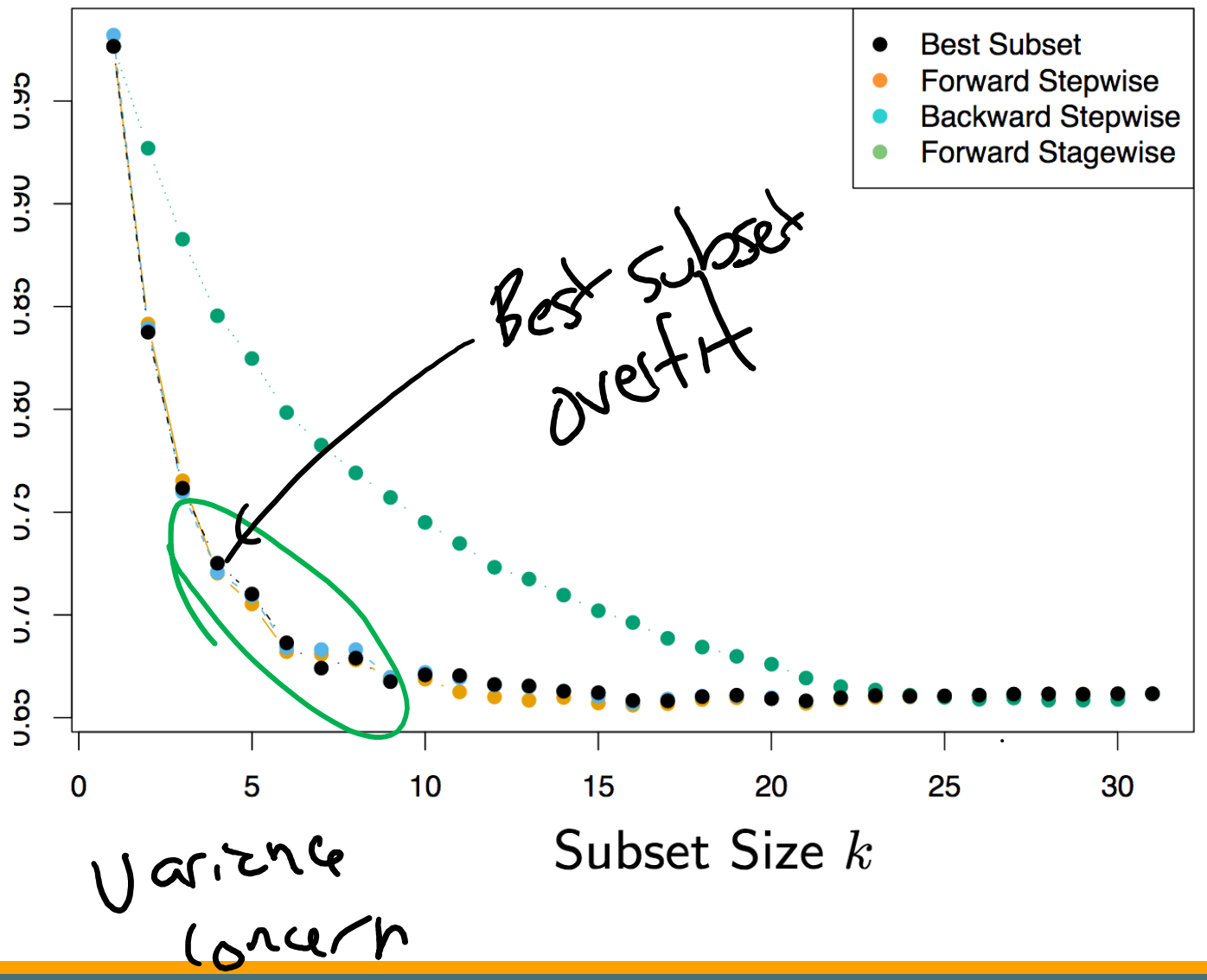
Stagewise regression does not change old coefficients when new variables are added

- This means it will take more iterations before the algorithm converges to a solution, but it is more robust when p is very large.

Start with some fixed intercept and all variables equal to zero

- Then find the variable with the highest correlation
- Find the value for that weight (β) which maximizes correlation
- Continue until none of the remaining variables have correlation with the residuals (ε)

below a threshold



Comparing the options

We see that all of the models, here on prostate cancer data, do eventually converge to near the best subset, without having to search through all of the candidates

Creating candidate features

Remember, that since this is a linear model, the expressive power of the model is limited to the linear combinations of the input features. As a result, we may want to procedurally generate more features before conducting feature elimination

Things to consider

- Interaction terms
- Polynomial terms
- Logarithmic terms

$X_1 X_2$

X_1^2

$\log X_1$

If you have one of these terms, you should always retain the original term underneath.

i.e. if X_1X_2 is significant, you should retain both X_1 and X_2