Lecture 5.1: Cross-Validation

STAT 1361/2360: Statistical Learning and Data Science

University of Pittsburgh
Prof. Lucas Mentch

Course Overview

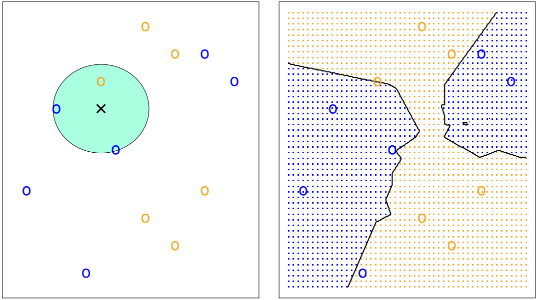
- Let's take a look back at what we've covered thus far
  - **Chapter 2:** Overview of important questions/topics of interest (stats vs. ML, bias vs. variance, flexibility vs. interpretability, prediction vs. inference)
  - **Chapter 3:** Basic regression methods (Linear Regression)
  - **Chapter 4:** Basic classification methods (Logistic Regression, LDA, QDA, kNN)

- In order to get beyond these basic approaches, we need some new ways of thinking about some big issues in statistics

k-Nearest Neighbors Review

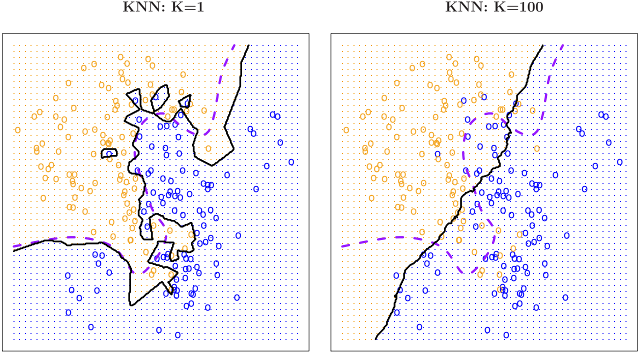
kNN Review

- Given a particular point x^* where we want to make a prediction, we find the k closest points to x^* and take a **majority vote** amongst responses at those k points (for classification) or take the **average** of responses at those k points (for regression)



ISLR Figure 2.14: k-nearest neighbors for classification with $k = 3$. Resulting decision boundary is shown on the right.

Recall that the amount of flexibility of kNN depends entirely on the choice of k:



ISLR Figure 2.16: kNN decision boundaries in black; (true) Bayes decision boundaries in purple. (Left) k-nearest neighbors with $k = 1$ – very flexible. (Right) k-nearest neighbors with $k = 100$ – *much less* flexible.

Tuning Parameters

- Note that the particular value of k chosen for kNN defines the kind of model that will be built
  - k is what we would call a **tuning parameter**: a parameter associated with a particular modeling method (usually a statistical or machine learning method) that governs *how* or *what kind* of model is built.
  - Tuning parameters often serve to control the flexibility (bias-variance tradeoff) of a particular model or method

So going back to the question we left Chapter 4 with ...

- How should we choose k?

So going back to the question we left Chapter 4 with ...

- How should we choose k?
  - If I do kNN twice, once with k=3 and once with k=10, how do I know which is better?

So going back to the question we left Chapter 4 with ...

- How should we choose k?
  - If I do kNN twice, once with k=3 and once with k=10, how do I know which is better?
- More generally, given a particular model with an associated tuning parameter, how do we choose the value of that tuning parameter?

So going back to the question we left Chapter 4 with ...

- How should we choose k?
  - If I do kNN twice, once with k=3 and once with k=10, how do I know which is better?
- More generally, given a particular model with an associated tuning parameter, how do we choose the value of that tuning parameter?
- Even more generally, given models $M_1, M_2, \ldots, M_b$ how should we choose the best model(s)?
  - e.g. Logistic Regression, LDA, QDA, kNN for several choices of k – how should I decide which model to use?

Disclaimer

- This idea of **Model Selection** is a very,very broad area
- When comparing two models or procedures M_i and M_j, there are often many factors to take into account and many tools available
  - E.g. What kinds of scientific/inferential questions do I want to be able to answer?
- Later we'll see that more specific measures of model "appropriateness" are available in specific contexts
- For now, we'll proceed in a general context where we might want to compare across many types of models and **we're looking for the model that best fits the data**

Cross-Validation

Model Evaluation Criteria # 1

- Let's start by considering some "bad" (less than optimal) ideas

Model Evaluation Criteria # 1

- Let's start by considering some "bad" (less than optimal) ideas
- What's the simplest thing we could think to do to compare model fit?

Model Evaluation Criteria # 1

- Let's start by considering some "bad" (less than optimal) ideas
- What's the simplest thing we could think to do to compare model fit?
  - Use entire sample to construct each model, then evaluate MSE (or misclassification error) on the entire sample

Model Evaluation Criteria # 1

- Let's start by considering some "bad" (less than optimal) ideas
- What's the simplest thing we could think to do to compare model fit?
  - Use entire sample to construct each model, then evaluate MSE (or misclassification error) on the entire sample

Downside:

Model Evaluation Criteria # 1

- Let's start by considering some "bad" (less than optimal) ideas
- What's the simplest thing we could think to do to compare model fit?
  - Use entire sample to construct each model, then evaluate MSE (or misclassification error) on the entire sample

Downside: Overfitting: model may fit well to *this* specific dataset, but may not generalize well to new data. More flexible models will always be preferred. (Hint: kNN with $k = 1$ is not always the best model in every possible situation)

Model Evaluation Criteria # 2

- Okay, but we already talked about that downside in Chapter 2

Model Evaluation Criteria # 2

- Okay, but we already talked about that downside in Chapter 2
- What was the solution we talked about?

Model Evaluation Criteria # 2

- Okay, but we already talked about that downside in Chapter 2
- What was the solution we talked about?
  - Divide data into *train* and *test* samples: use *training* sample to construct each model, then evaluate MSE (or misclassification error) on *test* sample

- Okay, but we already talked about that downside in Chapter 2
- What was the solution we talked about?
  - Divide data into *train* and *test* samples: use *training* sample to construct each model, then evaluate MSE (or misclassification error) on *test* sample

Downside:

Model Evaluation Criteria # 2

- Okay, but we already talked about that downside in Chapter 2
- What was the solution we talked about?
    - Divide data into *train* and *test* samples: use *training* sample to construct each model, then evaluate MSE (or misclassification error) on *test* sample

Downside: ???

Hint 1: What is the fundamental purpose of the train-test split approach?

Hints

Hint 1: What is the fundamental purpose of the train-test split approach?

To estimate how well the model fits to new data

Hints

Hint 1: What is the fundamental purpose of the train-test split approach?

To estimate how well the model fits to new data

Hint 2: So we're trying to estimate something – what?

Hints

Hint 1: What is the fundamental purpose of the train-test split approach?

To estimate how well the model fits to new data

Hint 2: So we're trying to estimate something – what?

The test error: $\widehat{Err}_{TEST}$

Hints

Hint 1: What is the fundamental purpose of the train-test split approach?

To estimate how well the model fits to new data

Hint 2: So we're trying to estimate something – what?

The test error: $\widehat{Err}_{TEST}$

Hint 3: What's "wrong" with this estimation procedure?

Hint 1: What is the fundamental purpose of the train-test split approach?

To estimate how well the model fits to new data

Hint 2: So we're trying to estimate something – what?

The test error: $\widehat{Err}_{TEST}$

Hint 3: What's "wrong" with this estimation procedure?

???

Consider the following example:

Example

Consider the following example:

You're interested in determining the average age of Pitt students. You walk out of the classroom, see a student, ask them their age, and determine that they're 22. You thus estimate that the average age of Pitt students is 22.

Example

Consider the following example:

You're interested in determining the average age of Pitt students. You walk out of the classroom, see a student, ask them their age, and determine that they're 22. You thus estimate that the average age of Pitt students is 22.

Problem?

Example

Consider the following example:

You're interested in determining the average age of Pitt students. You walk out of the classroom, see a student, ask them their age, and determine that they're 22. You thus estimate that the average age of Pitt students is 22.

Problem? One sample!

Counterargument

- One could argue that if there are n_{test} samples in the test set, then this isn't *one* estimate, it's an average over n_{test} estimates
- This is a perfectly reasonable perspective – if we were in a setting where we could magically generate a test set as large as we wanted at little to no cost (e.g. simulations), then this is actually a great way of doing things
- In practice though, creating a test set often means splitting up the original sample – this means (i) we're using less data to build (train) the model and (ii) if we'd split the data in a different way, we'd get a different estimate of the test/generalizability error

Model Evaluation Criteria # 3: Cross-Validation

k-fold Cross-Validation

k-fold Cross-Validation

- Begin by splitting the data into k groups

Model Evaluation Criteria # 3: Cross-Validation

k-fold Cross-Validation

- Begin by splitting the data into k groups
- Build (*train*) model with groups $2, \ldots, k$; calculate test error on 1^{st} group $\implies \widehat{Err}_1$

Model Evaluation Criteria # 3: Cross-Validation

k-fold Cross-Validation

- Begin by splitting the data into k groups
- Build (*train*) model with groups $2, \ldots, k$; calculate test error on 1^{st} group $\implies \widehat{Err}_1$
- Repeat k times; on the j^{th} iteration, use group j as the test set to calculate $\widehat{Err}_j$

k-fold Cross-Validation

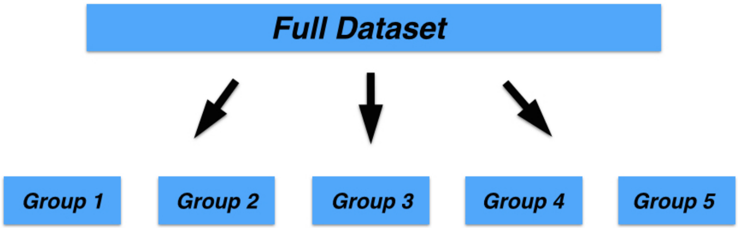
- Begin by splitting the data into k groups
- Build (*train*) model with groups $2, \ldots, k$; calculate test error on 1st group $\implies \widehat{Err}_1$
- Repeat k times; on the j^{th} iteration, use group j as the test set to calculate $\widehat{Err}_j$
- Calculate average test error: $\widehat{Err}_{\text{CV}} = \frac{1}{k}\sum_{j=1}^{k}\widehat{Err}_j$

k-fold Cross-Validation

Example: 5-fold Cross-Validation:

k-fold Cross-Validation

Example: 5-fold Cross-Validation:



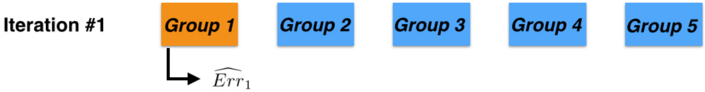
Example: 5-fold Cross-Validation

Iteration #1	***Group 1***	***Group 2***	***Group 3***	***Group 4***	***Group 5***

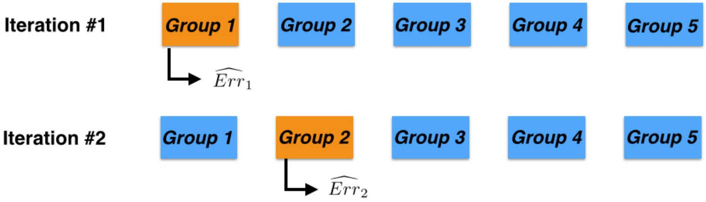
Example: 5-fold Cross-Validation

Iteration #1	***Group 1***	***Group 2***	***Group 3***	***Group 4***	***Group 5***

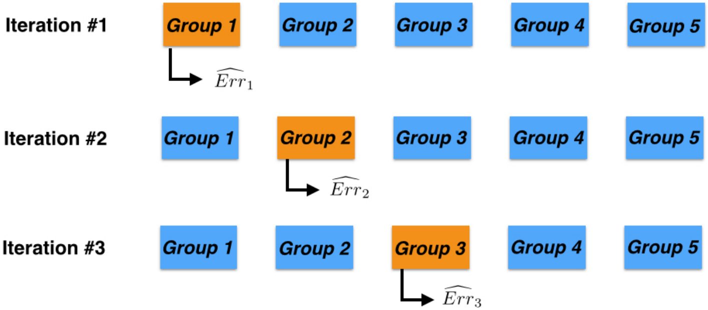
Example: 5-fold Cross-Validation



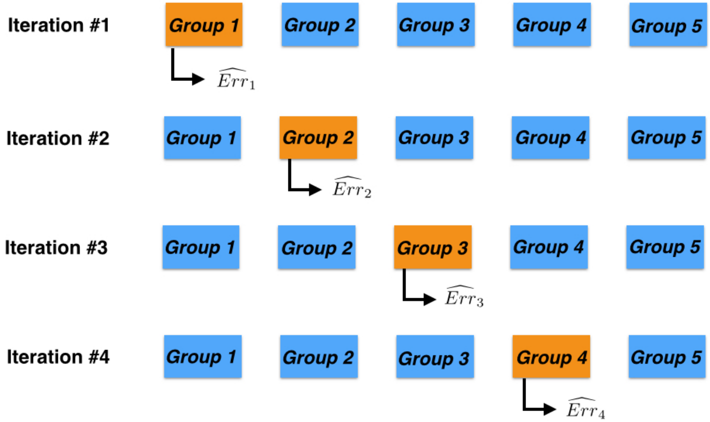
Example: 5-fold Cross-Validation



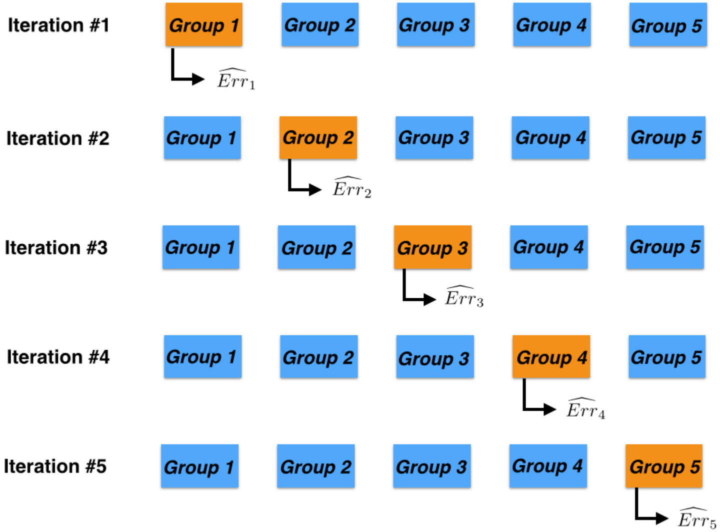
Example: 5-fold Cross-Validation



Example: 5-fold Cross-Validation



Example: 5-fold Cross-Validation



k-fold Cross-Validation Notes:

- For regression, $\widehat{Err}_j$ typically corresponds to the MSE; For classification, $\widehat{Err}_j$ corresponds to the misclassification rate:

- When we break the data into n groups (i.e. $k = n$), each observation gets put in its own group:
  - Use all data except for 1 point to build models; measure error on only that one point that is held out
  - This is more commonly referred to as “Leave-one-out" Cross-Validation (LOOCV)

Choosing the Number of Folds

- **Big Question Still Remaining:** How do we choose the number of folds? Two kinds of issues to consider:

(1) Practical Issues:

- The more groups are chosen (i.e. the larger the k), the more models need constructed
- Big data + complicated model $\implies$ might want to avoid LOOCV

(2) Theoretical Issues:

- There is a very subtle theoretical issue here – why is CV not giving us a “true" (unbiased) estimate of the error?

(2) Theoretical Issues:

- There is a very subtle theoretical issue here – why is CV not giving us a “true" (unbiased) estimate of the error?
    - ► We’re not using all of the data to build the models
    - ► The more data used to build the models, the better they should be

$\Longrightarrow$ Each of our error estimates is actually an *overestimate* of the “true” error for that model

Choosing the Number of Folds

- Same old bias-variance tradeoff issue:

  Many folds/groups: Fewer observations per group
  $\implies$ Errors calculated based on fewer hold-out samples, but models built with more data

  $\implies$ Higher variance; Lower Bias

  Fewer folds/groups: More observations in each group
  $\implies$ Errors calculated based on many hold-out samples, but models built with less data

  $\implies$ Lower variance; Higher Bias

Cross-Validation Final Notes:

- 5-fold and 10-fold cross-validation are overwhelmingly the most popular
  - Generally thought to give a good bias-variance tradeoff
- Cross-validation gives us a way to get a more robust estimate of the "generalizability" error for various methods
  - Avoids the potential issue of one model “accidentally" performing well/poor because of choice of a single test set
- Very general technique: no real restrictions on kinds of models this can be applied to and still arguably the most common way of doing model selection in general settings

Some Final Notes on Language:

- In practice, test sets do not just magically appear – we're given a single dataset and we must use some portion for training and some portion for testing.
  - This is equivalent to saying that some portion of the "training" data needs to be held out and used as a "test" set
- When the "test" set is created in this way, the book refers to this set as a *validation* set
- Realize that this merely a wording choice so that a test set can be seen as something independent from the training set and a validation set can be seen as a subset of the training set – **from this perspective though, in practice, all "test" sets are validation sets**

- The use of a different kind of validation set can still be useful though. What might the issue be with the following approach to choosing the k in k-nearest neighbors?

  1. Randomly divide the initial data into training and test sets
  2. Build a kNN model on the training set
  3. Repeat step (2) for a large range of values of k
  4. Calculate the error of each kNN model on the test set
  5. Choose the model/value of k that gives the lowest test error

Some Final Notes on Language:

- The use of a different kind of validation set can still be useful though. What might the issue be with the following approach to choosing the k in k-nearest neighbors?

  1. Randomly divide the initial data into training and test sets
  2. Build a kNN model on the training set
  3. Repeat step (2) for a large range of values of k
  4. Calculate the error of each kNN model on the test set
  5. Choose the model/value of k that gives the lowest test error

- The value of k that minimizes the error on *this particular* test set may not be the best choice in general or on other test sets – our estimate of the test error for this choice of k **might be optimistic**

Some Final Notes on Language:

- A better approach might be to first split the data into three sets: **training**, **validation**, and **test** and then do the following

  1. Randomly divide the initial data into training, validation, and test sets
  2. Build a kNN model on the training set
  3. Repeat step (2) for a large range of values of k
  4. Calculate the error of each kNN model on the *validation* set
  5. Choose the model/value of k that gives the lowest error on the validation set, but estimate the "true" generalizability error of this model by applying it to the *test* set

- In this class we'll focus mostly on *cross-validation* (still probably the most common approach), though you may sometimes see the approach above used as well