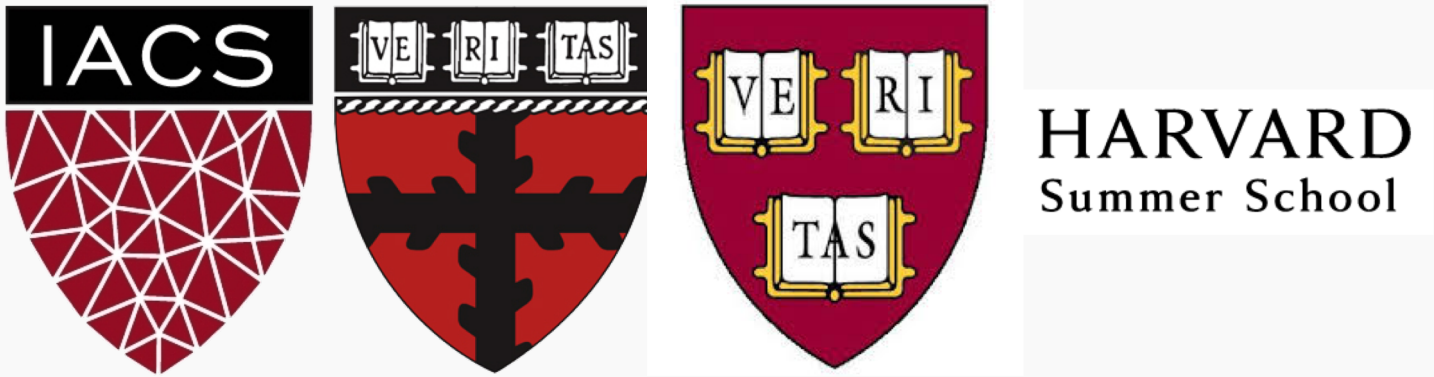
Lecture #11: Clustering, Missingness, and Wrap-Up

CS-S109A: Introduction to Data Science

Kevin Rader



Outline

- Unsupervised Clustering
    - K-means Clustering
    - Hierarchical Clustering (Agglomerate)
- Dealing with Missingness
    - Types of Missingness
    - Imputation Methods
- Course Wrap-Up

Supervised Models

The many *statistical learning* methods we have learned have been **supervised**: there is a clear response variable *Y* that we are trying to model (to estimate *f*).

They have been divded into two main tasks:

- Numeric *Y*: regression methods
- Categorical *Y*: classification methods

But what if the response variable *Y* is unmeasured and should be inferred from data? When can this arise or be useful in practice?

Unsupervised Models

When the response variable, *Y*, is unknown or is ignored, a method is called **unsupervised** (note: PCA can be thought of this way).

If the task is to generate a new numeric response variable, our task is impossible. But if our task is to generate a categorical response variable, our task is feasible.

This task of trying to determine new groups (aka, clusters) of your data based on the predictors is called **unsupervised clustering**.

This is a common task. For example, in medical domains: this might be way to generate new subclasses of disease, which may lead to very different prognoses or even different treatments.

K-Means Clustering

Determining Clusters Algorithmically

In order to generate novel groups/clusters when they are unknown, a systematic approach needs to be taken.

Generally speaking the goal of creating clusters is to find homogeneous subgroups among the observations. That is, the observations within the clusters should be similar, while observations in different clusters should be dissimilar (as measured in the predictor space).

Generally speaking, we want the clusters (C_1, C_2, …, C_k) to be non-overlapping and partition the observations. That is, each observations belongs in one and exactly one cluster.

Measuring the Clusters

Step #1: a loss function needs to be chosen to measure similarity among observations. The most natural choice (assuming all predictors are numeric): Euclidean distance.

Thus we want to minimize within-cluster metric of variability (think squared distance):

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

where:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

K-Means Clustering: the algorithm

The *K*-Means algorithm is a simple approach that will find a solution:

0. Set *K*.

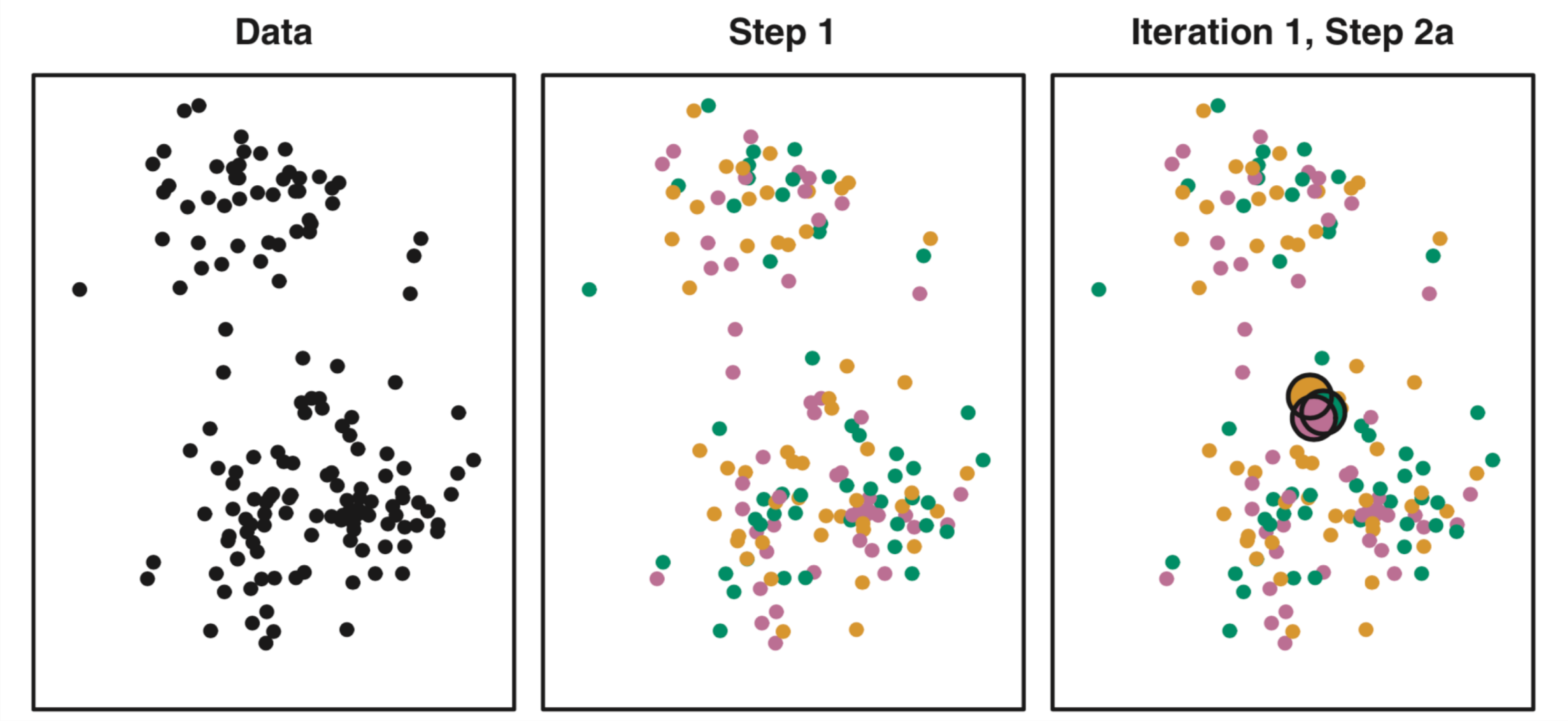
1. Perform an initial Clustering: randomly assign each observation to a cluster, 1 through *K*.

2. Iterate until the clusters converge (assignments stop changing):

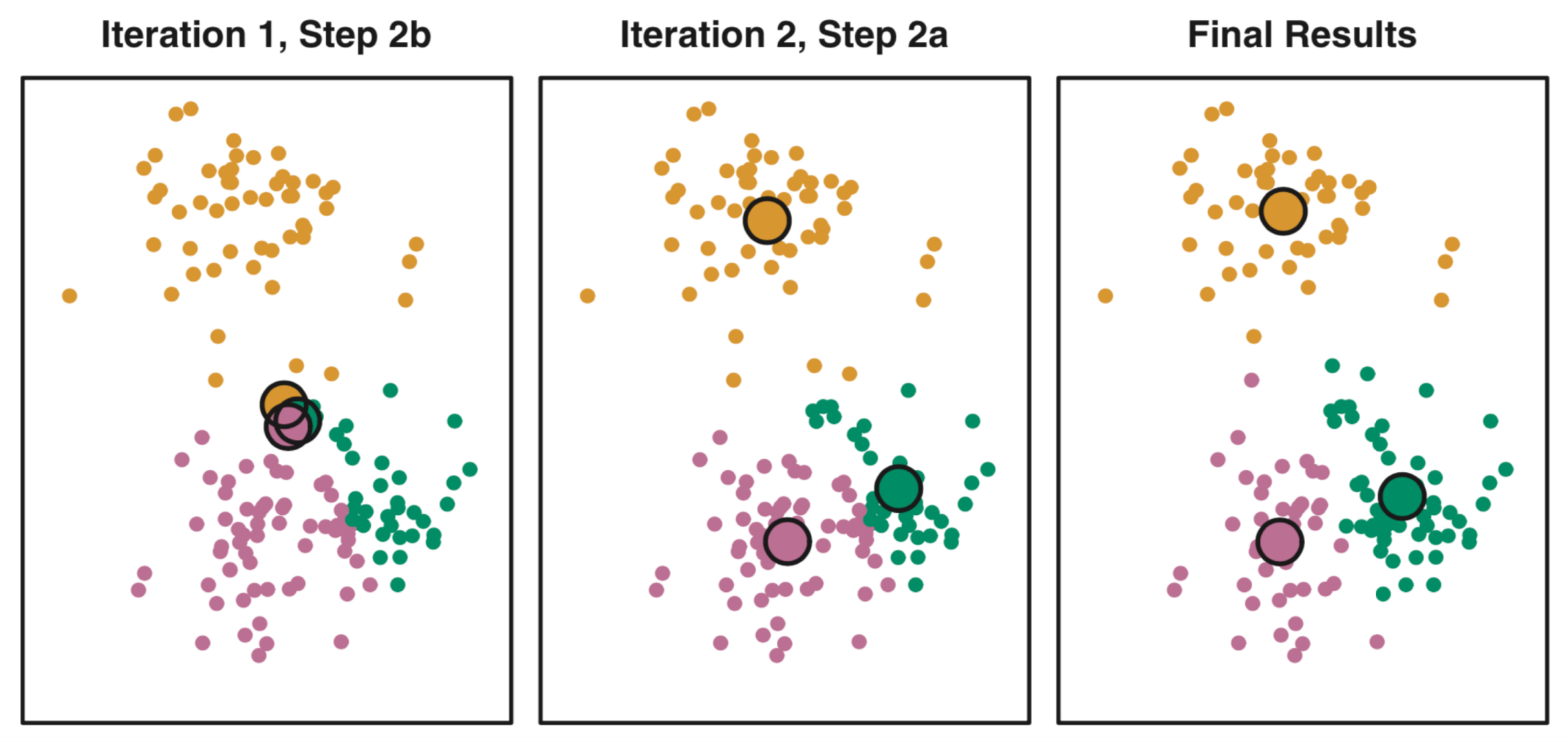
    a) For each of the *K* clusters, compute the cluster **centroid:** the vector of the variable means for the observations in that cluster.

    b) Re-assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).

K-Means Clustering, a picture is worth a thousand words:



K-Means Clustering, a picture is worth a thousand words (cont.):



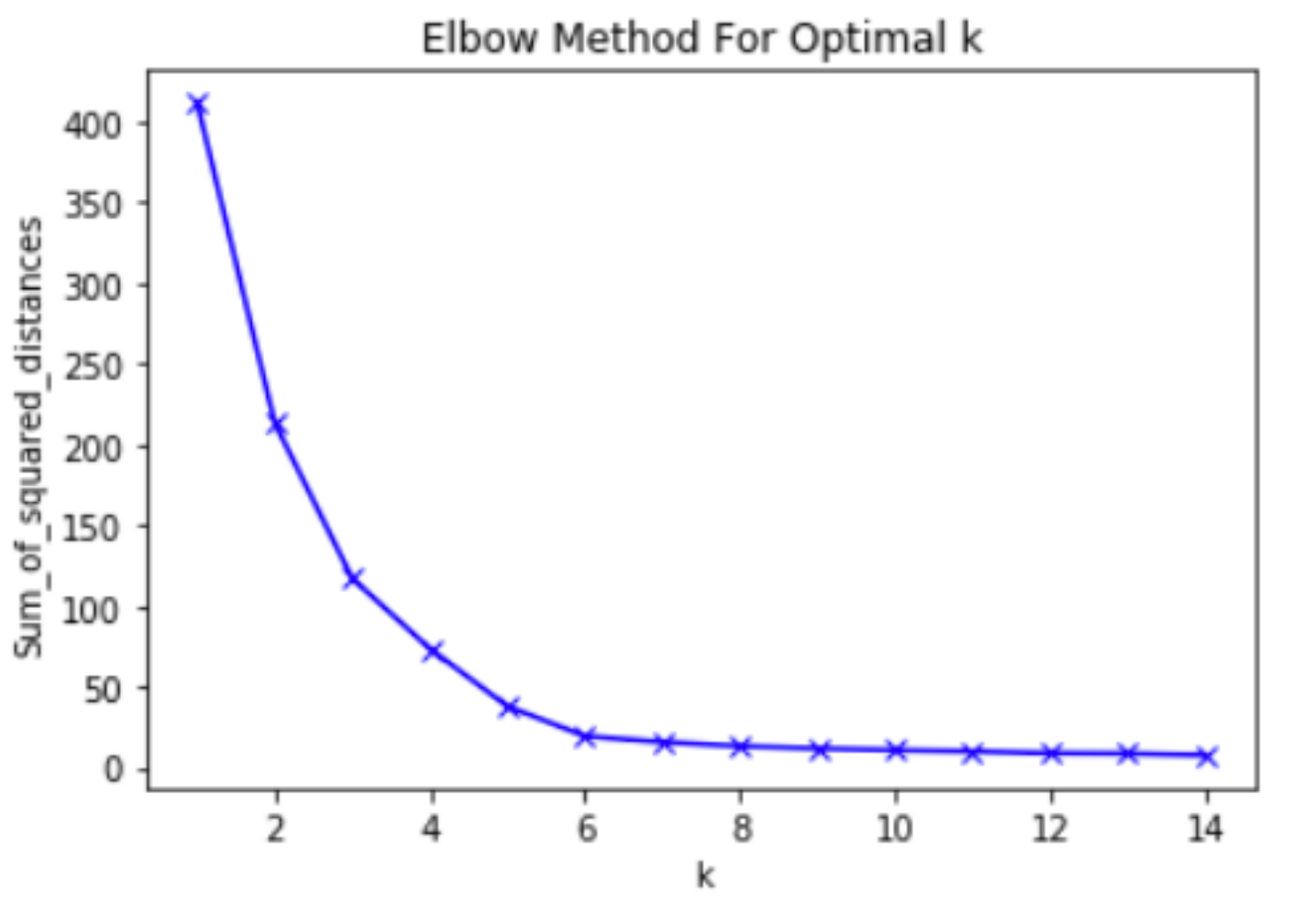
K-Means Clustering: some details

Note #1: this algorithm finds a local minimum (not the global minimum) which depends on the initial assignment. Thus it is recommended to run a few times and to select the best run that leads to the lowest distance metric.

Note #2: closeness depends vary much on scaling of your predictors (aka, standardization), and so the clusters could potentially be very different depending on this choice.

Note #3: the choice of a best *K* is not an easy task. One general technique is called the **elbow method**: the point where *diminishing returns* on improving the loss function are no longer worth the additional cost of a larger *K*.

Elbow Method



Elbow Method #2

Hierarchical Clustering (Agglomerate)

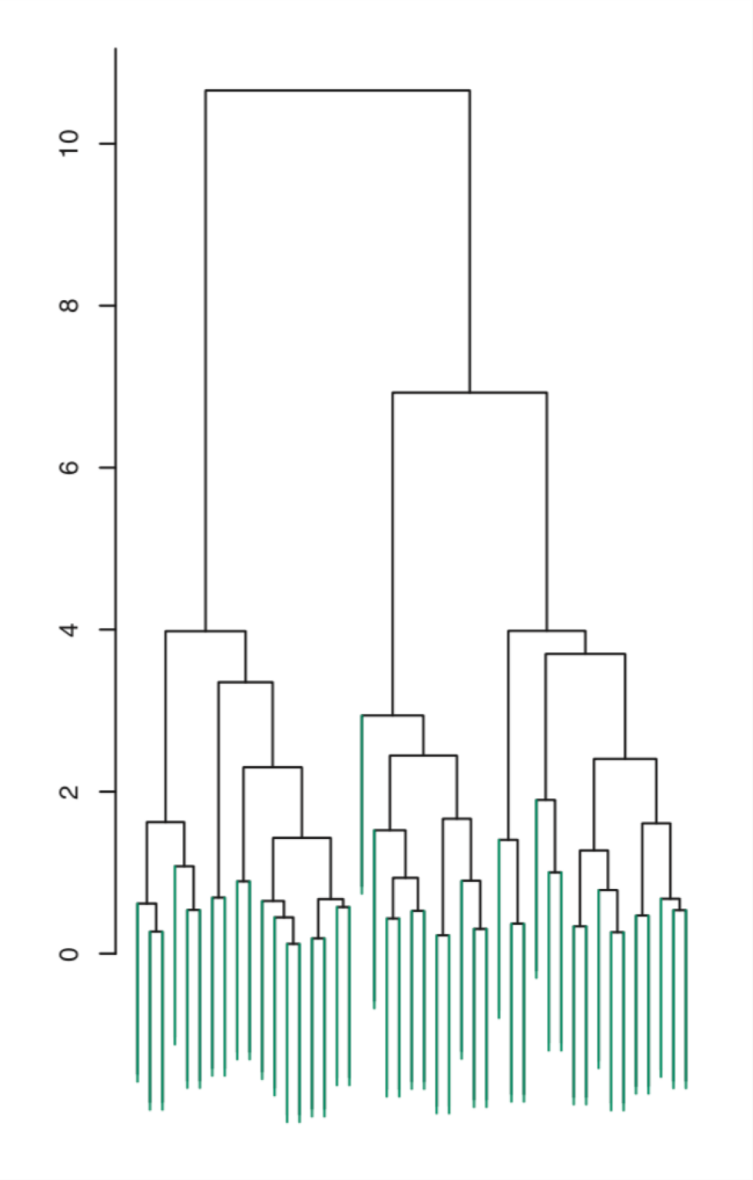
An alternative approach to

There are many other approaches to creating unknown clusters based on a set of predictors (too many to explore here). One other common approach is called **hierarchical clustering.**

Just like *K*-means, hierarchical clustering begins with a choice of distance: called **dissimilarity score** here (more on this later).

Hierarchical clustering is based on the **dendogram**: a tree-based depiction of the clusters and these dissimilarity scores (usually depicted vertically).

Dendogram



Clustering based on a dendogram

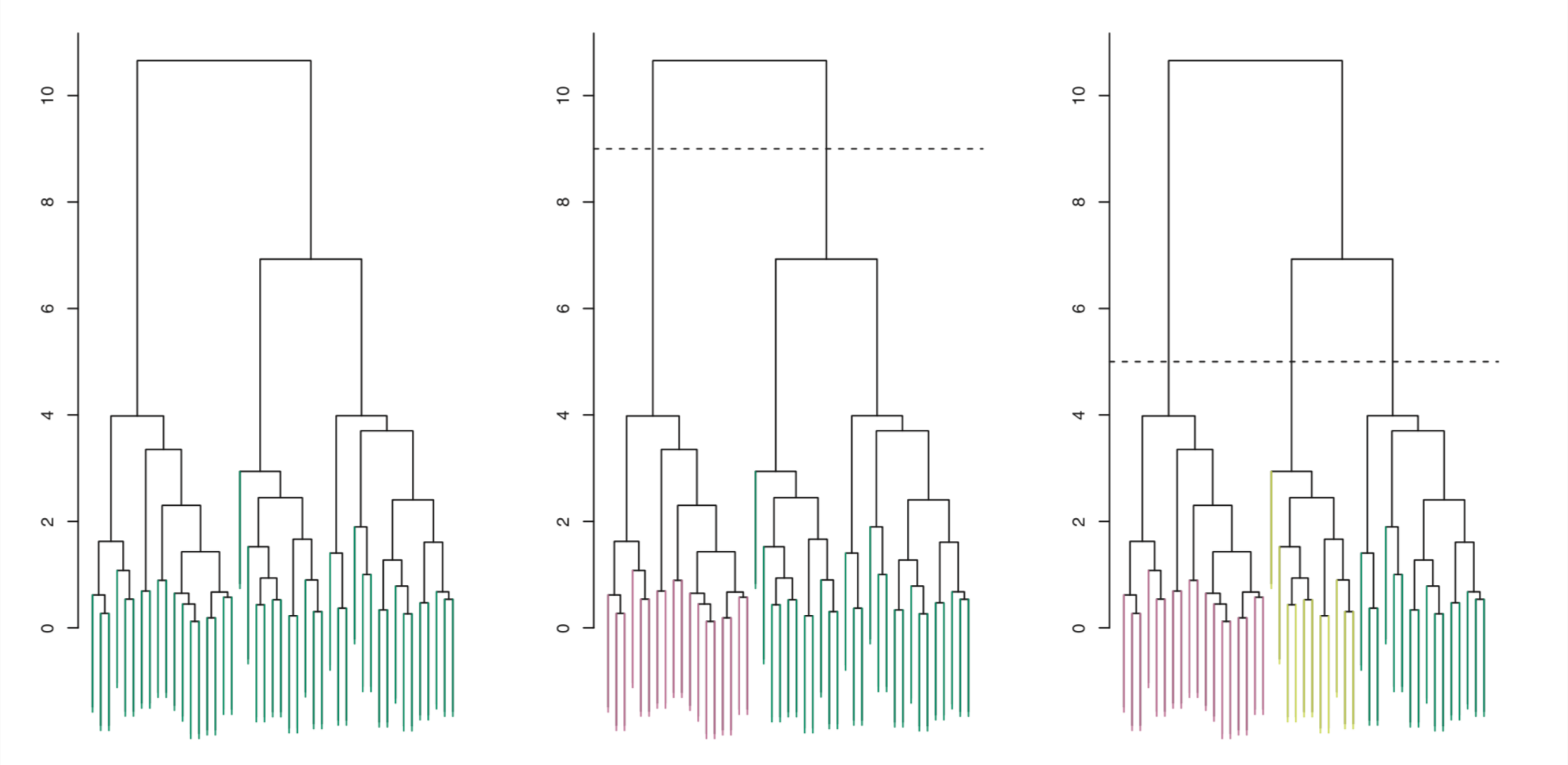
Given this hierarchy (or ordering) of clusters as seen on a dendogram, the clusters can then be derived by moving a horizontal line up and down the dendogram.

As the horizontal line moves up, every time a branching point is passed, those clusters are then fused together to form a larger cluster.

Start at the bottom with n separate clusters, move the horizontal line up, and fuse together clusters, in order, until you are left with just two clusters. You just need to decide where to draw the line ☺

The next slides defines the **agglomerate** (aka, bottom-up) approach to creating the dendogram:

Forming Clusters from a Dendogram



Agglomerate Hierarchical Clustering: the algorithm

0. Choose a metric of dissimilarity (like Euclidean distance).

1. Treat each of the *n* observations as its own cluster. Measure the $\binom{n}{2} = \frac{n(n-1)}{2}$ pairwise dissimilarity measures between clusters.

2. For *i* = *n*, *n*−1,...,2:
   a) Examine all pairwise inter-cluster dissimilarities among the *i* clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
   b) Compute the new pairwise inter-cluster dissimilarities among the *i* − 1 remaining clusters.

Hierarchical Clustering: some details

Note #1: The order of observations along the *x*-axis should not be interpreted…each branching can be spun without affecting the result.

Note #2: the choice of dissimilarity metric is unclear. Correlation-based metric can be used instead: think, not clustering observations with similar magnitude of the predictors, but clustering observations with similar correlations among predictors (maybe an observation is overly-expressed).

Note #3: measuring dissimilarity between two individual points to form a cluster is easy. But how does this generalize to a cluster of many points? Measuring dissimilarity between clusters is called **linkage**, and there is not clear choice. This is summarized in the table on the next slide (directly from the text, page 395).

Common choices of **linkage** in hierarchical clustering

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Word of caution: do not overinterpret

Unsupervised clustering gives nice interpretable results: which observations are similar and can be clustered together.

But do not overinterpret these as true subgroups. Think of unsupervised clustering as a *hypothesis-generating* approach: the resulting clusters could be indicative of real clusters or subgroups, and further analysis is needed (substantiated with a follow-up dataset).

General suggestion: investigate the resulting clusters carefully. Domain knowledge of the data (and familiarity with the specific observations involved) is necessary!

Note: in most situations, there is no true *Y* to compare the results to!

Outline

- Unsupervised Clustering
  - K-means Clustering
  - Hierarchical Clustering (Agglomerate)
- Dealing with Missingness
  - Types of Missingness
  - Imputation Methods
- Course Wrap-Up

Dealing with Missingness

What is missing data? k-Nearest Neighbors

Often times when data is collected, there are some missing values apparent in the dataset. This leads to a few questions to consider:

- How does this show up in pandas?
- How does sklearn handle these NaNs?
- How does this effect our modeling?

What are the simplest ways to handle missing data?

Naively handling missingness

- Drop the observations that have any missing values.
  - Use pd.DataFrame.dropna(axis=0)
- Impute the mean/median (if quantitative) or most common class (if categorical) for all missing values.
  - Use pd.DataFrame.fillna(value=x.mean())

How do statsmodels and sklearn handle these NaNs?

What are some consequences in handling missingness in these fashions?

Missingness Indicator Variable

One simple way to handle missingness in a variable, X_j, is to impute a value (like 0 or $\overline{X}_j$), then create a new variable, $X_{j,miss}$, that indicates this observation had a missing value. If X_j is categorical then just impute 0.

Then include both $X_{j,miss}$ and X_j as predictors in any model.

Illustration is to the right.

X_1	X_2	X_1^*	X_2^*	$X_{1,miss}$	$X_{2,miss}$
10	.	10	0	0	1
5	1	5	1	0	0
21	0	21	0	0	0
15	0	15	0	0	0
16	.	16	0	0	1
.	.	0	0	1	1
21	1	21	1	0	0
12	0	12	0	0	0
.	1	0	1	1	0

Why use a Missingness Indicator Variable?

How does this missingness indicator variable improve the model?

Because the group of individuals with a missing entry may be systematically different than those with that variable measured. Treating them equivalently could lead to bias in quantifying relationships (the β's) and underperform in prediction.

For example: imagine a survey that asks whether or not someone has ever recreationally used opioids, and some people chose not to respond. Does the fact that they did not respond provide extra information? Should we treat them equivalently as never-users?

This approach essentially creates a third group for this predictor: the "did not respond" group.

Types of Missingness

Sources of Missingness

Missing data can arise from various places in data:

- A survey was conducted and values were just randomly missed when being entered in the computer.
- A respondent chooses not to respond to a question like `Have you ever recreationally used opioids?'.
- You decide to start collecting a new variable (due to new actions: like a pandemic) partway through the data collection of a study.
- You want to measure the speed of meteors, and some observations are just 'too quick' to be measured properly.

The source of missing values in data can lead to the major types of missingness:

Types of Missingness

There are 3 major types of missingness to be concerned about:

1. **Missing Completely at Random (MCAR)** - the probability of missingness in a variable is the same for all units. Like randomly poking holes in a data set.
2. **Missing at Random (MAR)** - the probability of missingness in a variable depends only on available information (in other predictors).
3. **Missing Not at Random (MNAR)** - the probability of missingness depends on information that has not been recorded and this information also predicts the missing values.

What are examples of each these 3 types?

Missing completely at random (MCAR)

Missing Completely at Random is the best case scenario, and the easiest to handle:

- Examples: a coin is flipped to determine whether an entry is removed. Or when values were just randomly missed when being entered in the computer.
- Effect if you ignore: there is no effect on inferences (estimates of β).
- How to handle: lots of options, but best to impute (more on next slide).

Missing at random (MAR)

Missing at Random is still a case that can be handled.

- Example(s): men and women respond to the question "have you ever felt harassed at work?" at different rates (and may be harassed at different rates).
- Effect if you ignore: inferences are biased (estimates of β's) and predictions are usually worsened.
- How to handle: use the information in the other predictors to build a model and **impute** a value for the missing entry.

Key: we can fix any biases by modeling and imputing the missing values based on what is observed!

Missing Not at Random (MNAR)

Missing Not at Random is the worst case scenario, and impossible to handle properly:

- Example(s): patients drop out of a study because they experience some really bad side effect that was not measured. Or cheaters are less likely to respond when asked if you've ever cheated.
- Effect if you ignore: there are major effects on inferences (estimates of beta) or predictions.
- How to handle: you can 'improve' things by dealing with it like it is MAR, but you [likely] may never completely fix the bias. And incorporating a **missingness indicator variable** may actually be the best approach (if it is in a predictor).

What type of missingness is present?

Can you ever tell based on your data what type of missingness is actually present?

Since we asked the question, the answer must be no.

It generally cannot be determined whether data really are missing at random, or whether the missingness depends on unobserved predictors or the missing data themselves. The problem is that these potential “lurking variables” are unobserved (by definition) and so can never be completely ruled out.

In practice, a model with as many predictors as possible is used so that the `missing at random' assumption is reasonable.

Imputation Methods

Handling Missing Data

When encountering missing data, the approach to handling it depends on:

1. whether the missing values are in the response or in the predictors. Generally speaking, it is much easier to handle missingness in predictors.
2. whether the variable is quantitative or categorical.
3. how much missingness is present in the variable. If there is too much missingness, you may be doing more damage than good.

Generally speaking, it is a good idea to attempt to **impute** (or 'fill in') entries for missing values in a variable (assuming your method of imputation is a good one).

Imputation Methods

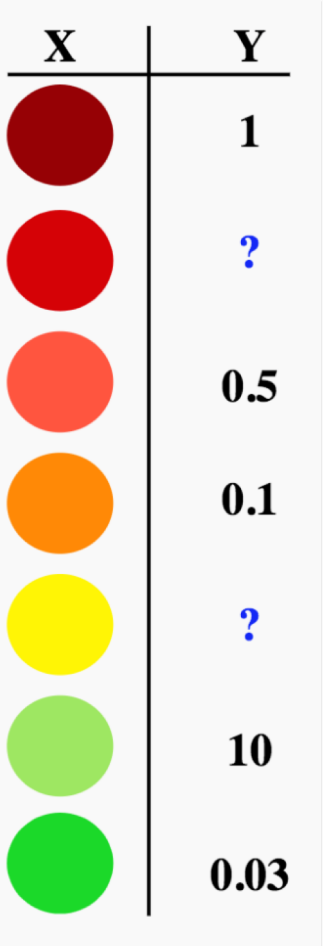
There are several different approaches to imputing missing values:

1. **Impute the mean or median** (quantitative) or most common class (categorical) for all missing values in a variable.
2. Create a new variable that is an **indicator of missingness**, and include it in any model to predict the response (also plug in zero or the mean in the actual variable).
3. **Hot deck imputation**: for each missing entry, randomly select an observed entry in the variable and plug it in.
4. **Model the imputation**: plug in predicted values ($\hat{y}$) from a model based on the other observed predictors.
5. **Model the imputation with uncertainty**: plug in predicted values plus randomness ($\hat{y} + \epsilon$) from a model based on the other observed predictors.

What are the advantages and disadvantages of each approach?

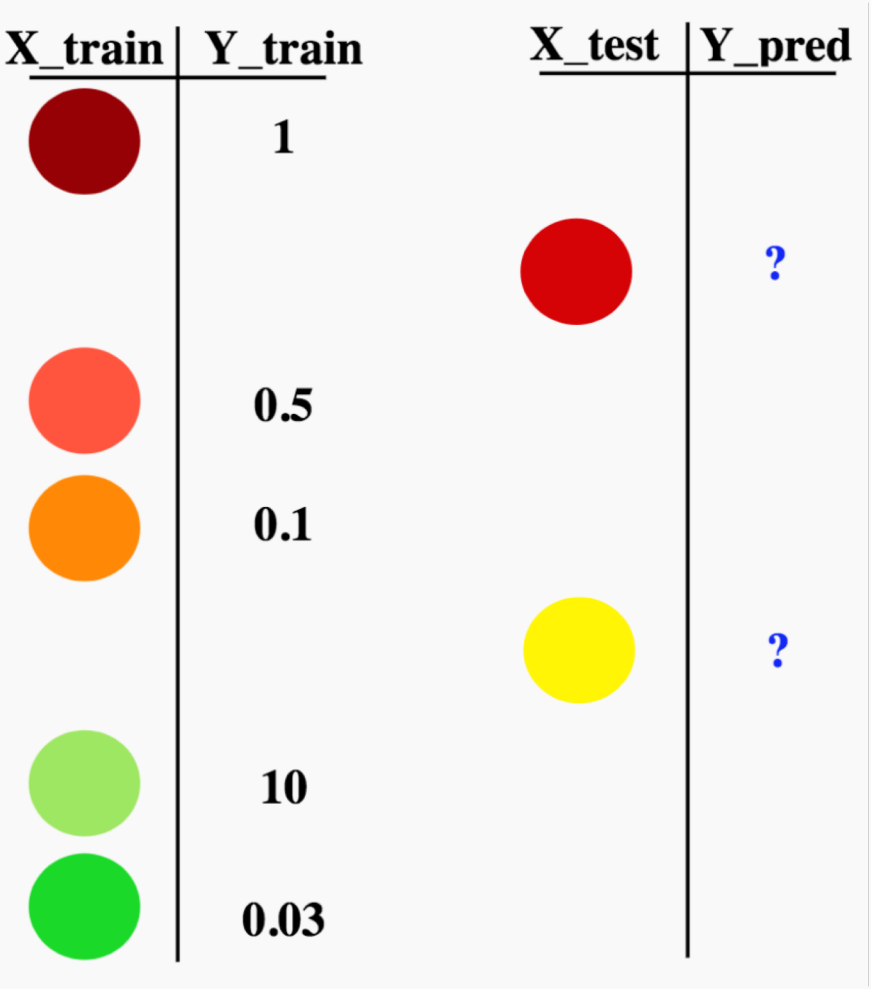
Schematic: imputation through modeling

How do we use models to fill in missing data?



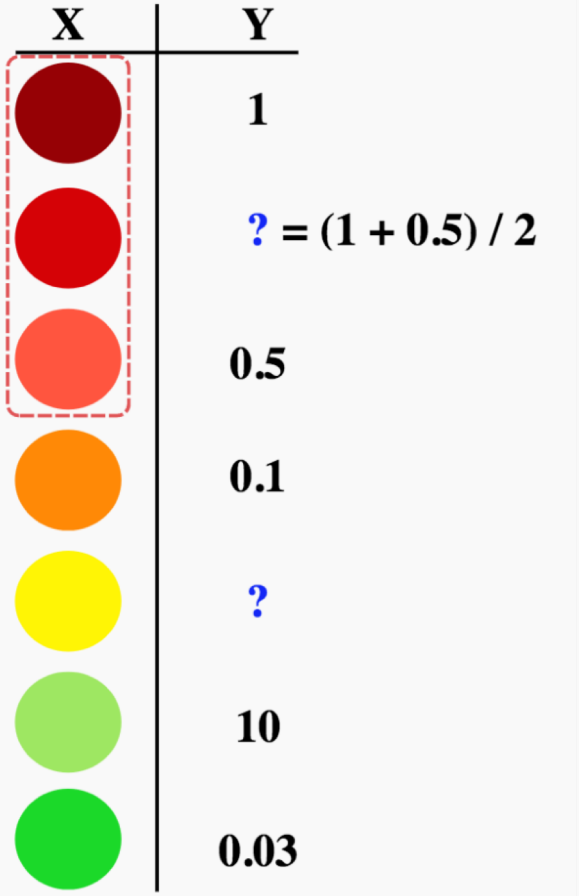
Schematic: imputation through modeling

How do we use models to fill in missing data?



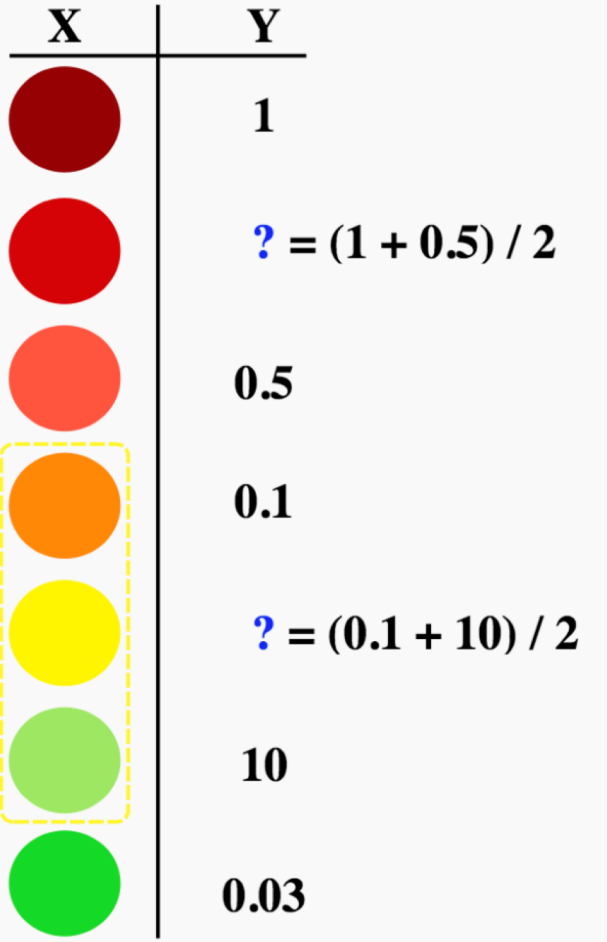
Schematic: imputation through modeling

How do we use models to fill in missing data? Using *k*-NN for *k* = 2?



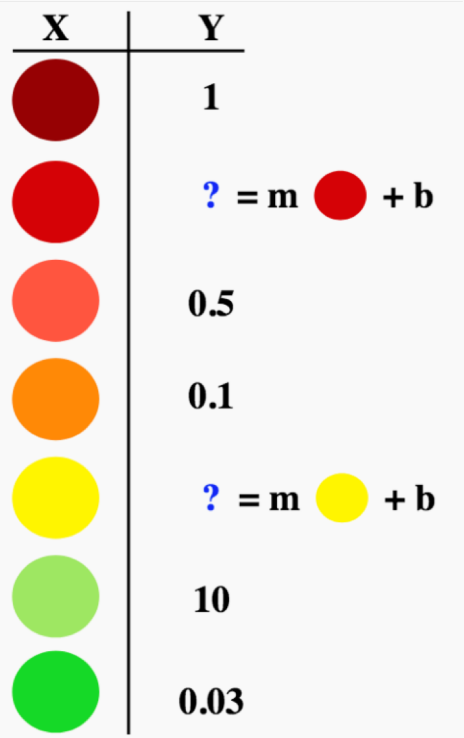
Schematic: imputation through modeling

How do we use models to fill in missing data? Using *k*-NN for *k* = 2?



Schematic: imputation through modeling

How do we use models to fill in missing data? Using linear regression?



Where *m* and *b* are computed from the observations (rows) that do not have missingness (we should call them $b = \beta_0$ and $m = \beta_1$).

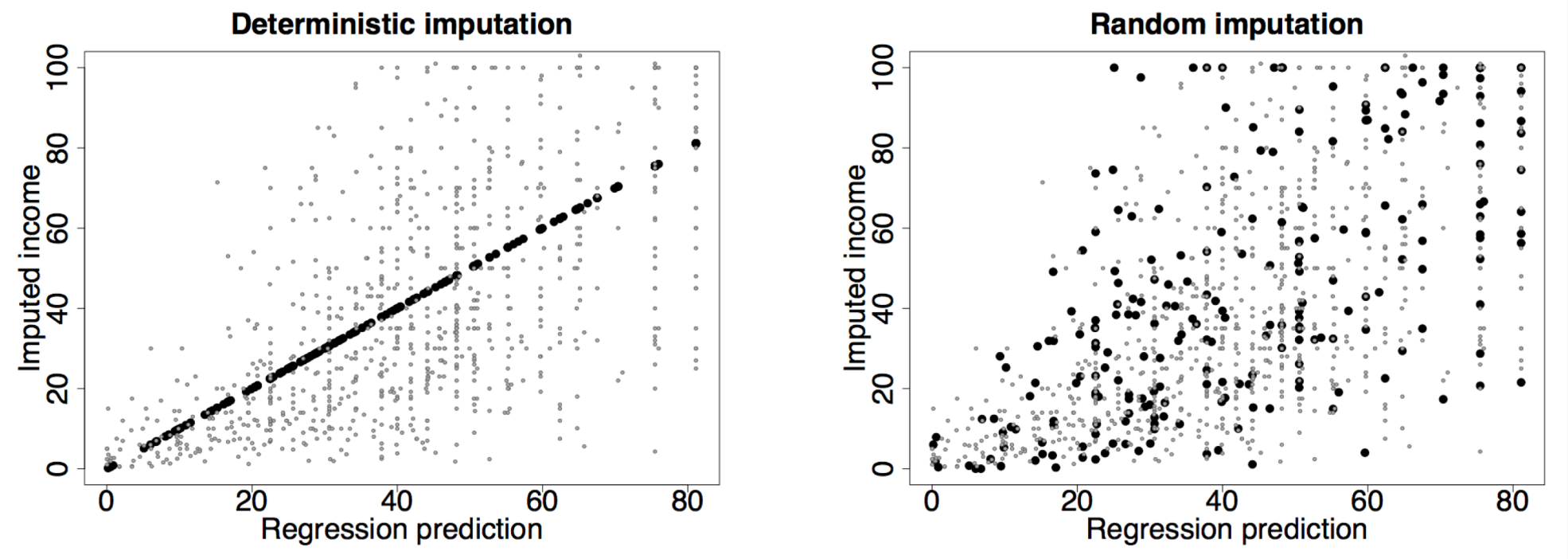
Imputation through modeling with uncertainty

The schematic in the last few slides ignores the fact of imputing with uncertainty. What happens if you ignore this fact and just use the 'best' model to impute values solely on $\hat{y}$?

The distribution of the imputed values will be too narrow and not represent real data (see next slide for illustration). The goal is to impute values that include the uncertainty of the model.

How can this be done in practice in *k*-NN? In linear regression? In logistic regression?

Imputation through modeling with uncertainty: an illustration



Imputation through modeling with uncertainty: an illustration

Recall the probabilistic model in linear regression:

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \varepsilon$$

where $\epsilon \sim N(0, \sigma^2)$. How can we take advantage of this model to impute with uncertainty?

It's a 3 step process:

1. Fit a model to predict the predictor variable with missingness from all the other predictors.
2. Predict the missing values from the model in the previous part.
3. Add in a measure of uncertainty to this prediction by randomly sampling from a $N(0, \sigma^2)$ distribution, where σ^2 is the mean square error (MSE) from the model.

Imputation through modeling with uncertainty: *k*-NN regression

How can we use *k*-NN regression to impute values that mimic the error in our observations?

Two ways:

- Use $k = 1$.
- Use any other *k*, but randomly select from the nearest neighbors in $\mathcal{N}_0$. This can be done with equal probability or with some weighting (inverse to the distance measure used).

Imputation through modeling with uncertainty: classifiers

For classifiers, this imputation with uncertainty/randomness is a little easier process. How can it be implemented?

If a classification model (logistic, *k*-NN, etc...) is used to predict the variable with missingness on the observed predictors, then all you need to do is flip a 'biased coin' (or multi-sided die) with the probabilities of coming up for each class equal to the predicted probabilities from the model.

Warning: do not just classify blindly using the predict command in sklearn!

Imputation across multiple variables

If only one variable has missing entries, life is easy. But what if all the predictor variables have a little bit of missingness (with some observations having multiple entries missing)? How can we handle that?

It's an iterative process. Impute X_1 based on X_2, ..., X_p. Then impute X_2 based on X_1 and X_3, ..., X_p. And continue down the line.

Any issues? Yes, not all of the missing values may be imputed with just one 'run' through the data set. So you will have to repeat these 'runs' until you have a completely filled in data set.

Multiple imputation: beyond this class

What is an issue with treating your now 'complete' data set (a mixture of actually observed values and imputed values) as simply all observed values?

Any inferences or predictions carried out will be tuned and potentially overfit to the random entries imputed for the missing entries. How can we prevent this phenomenon?

By performing **multiple imputation**: rerun the imputation algorithm many times, refit the model on the response many times (one time each), and then 'average' the predictions or estimates of β coefficients to perform inferences (also incorporating the uncertainty involved).

Note: this is beyond what we would expect in this class. But it generally a good thing to be aware of.

sklearn's impute module

Want to do imputation in sklearn? Of course there are functions for that!

The 4 main functions in this module are:

1. **`sklearn.impute.SimpleImputer`**: perform mean, median, mode, or any specific value to impute
2. **`sklearn.impute.IterativeImputer`**: perform multivariate imputation that estimates each predictor from all the others (user-supplied model/estimator).
3. **`sklearn.impute.KNNImputer`**: perform imputation automatically from a *k*-NN model from all the other predictors.
4. **`sklearn.impute.MissingIndicator`**: to create binary indicator(s) for where the missing values occur.

Outline

- Unsupervised Clustering
  - K-means Clustering
  - Hierarchical Clustering (Agglomerate)
- Dealing with Missingness
  - Types of Missingness
  - Imputation Methods
- Course Wrap-Up

Course Wrap-Up

Modules

The summer has been organized into 4 major ‘modules’:

- Module 0: Intro to Data and Data Science (and Python)
- Module 1: Regression
- Module 2: Classification
- Module 3: Ensemble Methods and miscellaneous

We have learned various approaches to perform both predictions and inferences within each of these frameworks.

Module 1: Regression Methods

When is it appropriate to perform a regression method? What regression models have we learned?

1. Linear Regression (simple, multiple, polynomial, interactions, model selection, Ridge & Lasso, etc...)
2. *k*-NN
3. Regression Trees

What is the main difference between these types of models?

Module 2: Classification Methods

When is it appropriate to perform a classification method? What classification models have we learned?

1. Logistic Regression: same details as linear regression apply
2. *k*-NN
3. Classification Trees

What is the main difference between these types of models (advantages and disadvantages)? When should you use each method?

Module 3: Ensemble Methods

What does it mean for a model to be an ensemble method?

1. Bagging Trees
2. Random Forests
3. Boosting Models
4. Stacking Models

What approach does each model take to improve prediction accuracy?

Choosing between Models

How can we choose between our various methods/models to answer a question at hand? What approaches/measures can we use to make this determination?

1. In-sample: AIC, BIC (barely mentioned)
2. Out-of-sample: Cross-Validation

What measure(s) should we use when we perform cross-validation?

Dealing with Data Issues

What issues have arisen when dealing with real data? How have we handled them?

1. Categorical Predictors: might make sense to one-hot encode
2. Missing Data: might make sense to impute
3. High Dimensionality: might make sense to use a data reduction technique.
4. Too many observations: do preliminary analysis on a subset

How are predictions affected? How are inferences affected?

Dealing with High Dimensionality

What does ‘high dimensionality’ mean? What issues arise when this happens? How can we handle it?

1. Model Selection: subset variable selection
2. Regularization: LASSO and Ridge like approaches (penalize the loss function)
3. PCA: create new predictor variables that encapsulate the ‘essence’ of all your predictor data with a minimal number of variables.

How can we compare methods to determine which approach is best?

Other things we've learned

- Scraping, Data Gathering, Data Wrangling
- EDA: Visualization and Summary Statistics
- *t*-tests and *p*-values: probabilistic/ approaches to perform inferences
- Bootstrapping: empirical approach to perform inferences
- Misclassification Rates, Types of Errors, Confusion Matrices/Tables, and ROC Curves
- Bias-Variance Trade-off
- Train vs. Test vs. Validation
- Standardization vs. Normalization. When should we do it?

Anything lingering questions or thoughts?

Other things we haven't discussed

There are lots of topics we have not covered in one semester…some are covered in 109B in the Spring:

- Deep Learning (Neural Networks)
- Smoothers
- Time Series and Longitudinal Modeling
- Bayesian Data Analysis
- Reinforcement Learning
- Interactive Visualizations
- Database Management (SQL, etc.)
- Cloud Computing and Scaling (AWS)
- And much, much more…

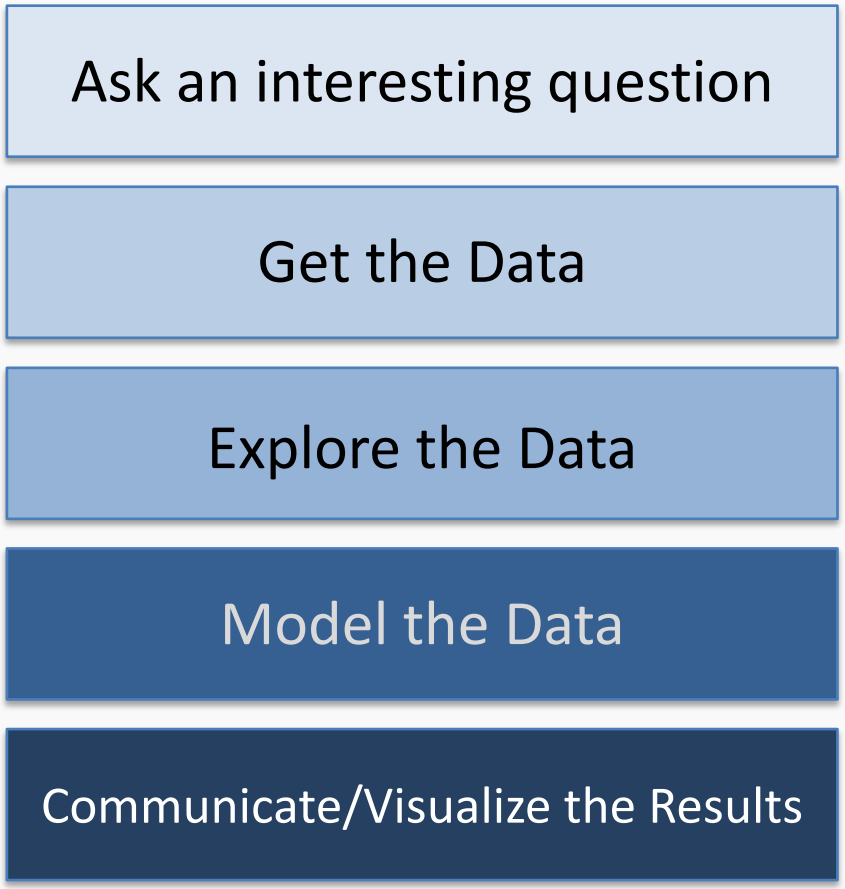
Courses Related to Data Science

- CS 109B: Advanced Topics in Data Science
- CS 171: Visualizations
- CS 181/281: Machine Learning
- CS 182: Artificial Intelligence (AI)
- CS 205: Distributive Computing
- AM207: Advanced Scientific Computing
- Stat 110/210: Probability Theory
- Stat 111/211: Statistical Inference
- Stat 139: Linear Models
- Stat 149: Generalized Linear Models
- Stat 195: Statistical Machine Learning

This list is not exhaustive!

What?

The Data Science Process



Thanks for all your hard work!

It’s been a intense semester for everyone involved. Thank you for your patience, your hard work, and your commitment to data science!

It’s sad to see you go...

But happy to have had the opportunity.