Homework 2 Questions

1 Programming exercise

Using scikit-learn, train an SVM to do classification on the iris data set and evaluate the significance of the obtained classification accuracy. As a hint, consider the following steps:

1. Consider two possible models - one linear and one with an RBF kernel.
2. Determine which model is better for the classification task.
3. Using the better of the two models, obtain a test accuracy.
4. Evaluate whether the test classification accuracy is significant using a permutation test.
5. Report the p-value you obtained.

Question 1: In class, we considered one way of doing feature selection through quantifying the mutual information between features and class labels. Which regularization penalty can be used as another way to discard irrelevant features?

a. ℓ_1

b. ℓ_2

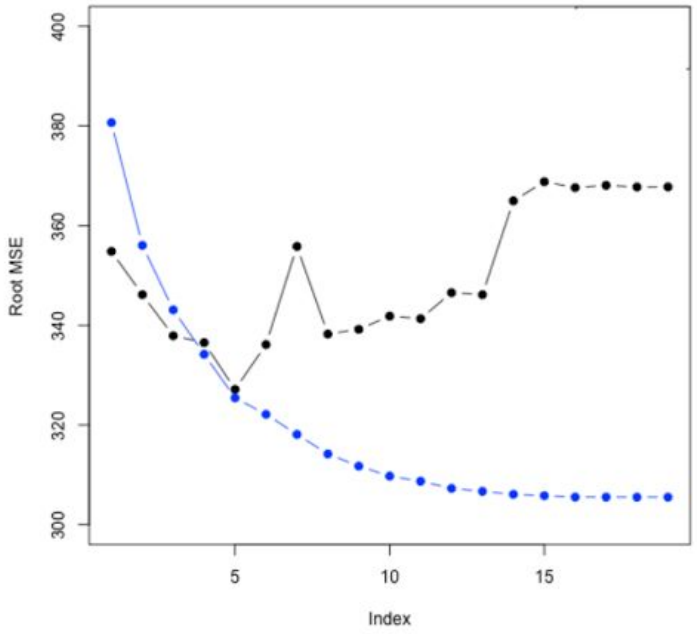
c. none of the above

Question 2: Consider a data set with 19 features per data instance. You decide to do feature selection in a brute force way by selecting subsets of features and performing cross validation to determine whether the selected subset of features is a good one for some task. You form 19 subsets of features with sizes 1, 2,...,19. Note that you decide not to investigate all possible subsets (that would be 2^{19} possible subsets). You have better things to do with your life.

You have performed 19 complete cross validations, and plot the training and validation errors in order of increasing subset size. Which of the following lines is the training error? (note that "root MSE" (mean squared error) is one way to quantify error, and "Index" is the subset index in the list, which is equivalent to the size of the subset)

a. blue

b. black



Question 3: Consider the setting of question 2. Based on the training and validation error plots, which size subset would you choose for the final evaluation of your model?

a. 5

b. 7

c. 14

d. 19

Question 4: Consider performing leave-2-out cross validation on a data set with 150 total data instances. If you decide to leave 10% of the data for testing, how many folds of cross validation will you need to run?

a. 2

b. 9045

c. 11175

d. none of the above

Question 5: Consider performing 2-fold cross validation on the same data set as in question 4. Now, if you decide to leave 20% of the data for testing, how many folds of cross validation will you need to run?

a. 2

b. 9045

c. 11175

d. none of the above

Question 6: Which of the following methods will be best to use to determine the best number of nearest neighbors k in kNN for a certain data set?

a. feature selection

b. cross validation

c. regularization

Dimensionality reduction and clustering

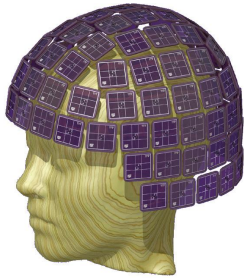
06/17/2016

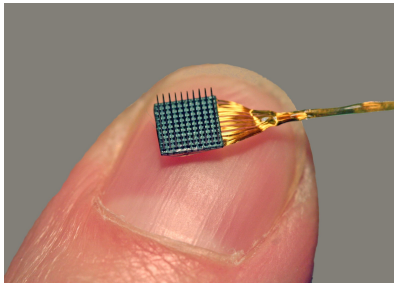
Mariya Toneva
mariya@cmu.edu

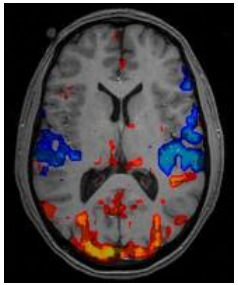
Some figures derived from slides by Alona Fyshe, Aarti Singh, Barnabás Póczos, Tom Mitchell

How can ML help neuroscientists?

- **Deal with large number of sensors/recording sites**







  - investigate high-dimensional representations
    - classification (what does this high-dimensional data represent?)
    - regression (how does it represent it? can we predict a different representation?)
    - model selection (what model would best describe this high dimensional data?)
    - **clustering (which high-dimensional representations are similar to each other?)**
  - **uncover few underlying processes that interact in complex ways**
    - **dimensionality reduction techniques**

Supervised vs unsupervised learning

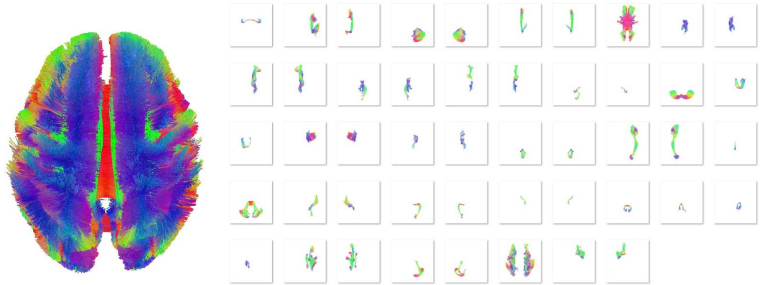
- So far we've only looked at supervised learning methods

Supervised vs unsupervised learning

- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression

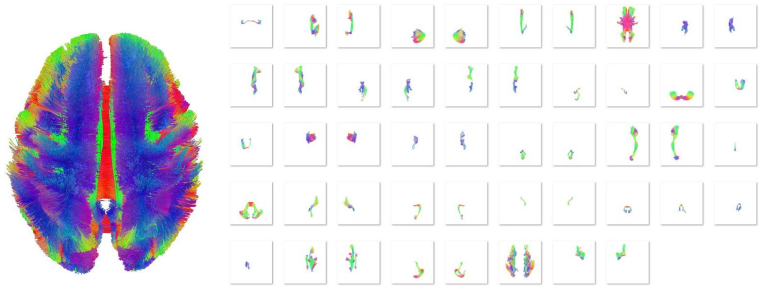
Supervised vs unsupervised learning

- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles



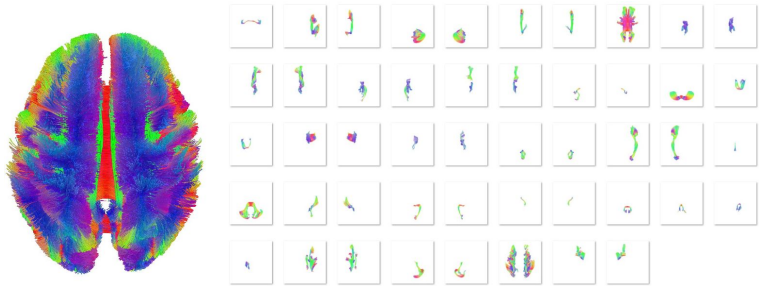
Supervised vs unsupervised learning

- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles
- But what if we have a lot of unlabeled data and acquiring labels is expensive, or not even possible?



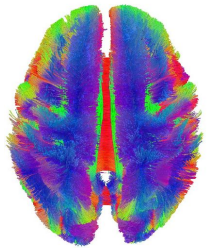
Supervised vs unsupervised learning

- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles
- But what if we have a lot of unlabeled data and acquiring labels is expensive, or not even possible?
  - expensive labels: DTI fibers assignments to anatomical bundles



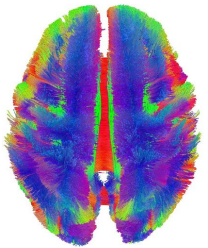
Supervised vs unsupervised learning

- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles
- But what if we have a lot of unlabeled data and acquiring labels is expensive, or not even possible?
  - expensive labels: DTI fibers assignments to anatomical bundles
  - unknown labels: "brain states" assignments of resting state fMRI



Supervised vs unsupervised learning

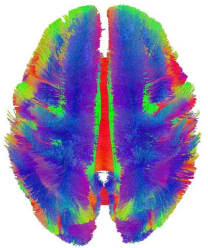
- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles
- But what if we have a lot of unlabeled data and acquiring labels is expensive, or not even possible?
  - expensive labels: DTI fibers assignments to anatomical bundles
  - unknown labels: "brain states" assignments of resting state fMRI



- Unsupervised learning! We'll discuss two such methods today

Supervised vs unsupervised learning

- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles
- But what if we have a lot of unlabeled data and acquiring labels is expensive, or not even possible?
  - expensive labels: DTI fibers assignments to anatomical bundles
  - unknown labels: "brain states" assignments of resting state fMRI



- Unsupervised learning! We'll discuss two such methods today
  - dimensionality reduction
  - clustering

Today: dimensionality reduction & clustering

- Dimensionality reduction techniques
  - Principal component analysis (PCA)
  - Independent component analysis (ICA)
  - Canonical correlation analysis (CCA)
  - Laplacian eigenmaps

Today: dimensionality reduction & clustering

- Dimensionality reduction techniques
  - Principal component analysis (PCA)
  - Independent component analysis (ICA)
  - Canonical correlation analysis (CCA)
  - Laplacian eigenmaps
- Clustering
  - Partitional algorithms
    - K-means
    - Spectral clustering
  - Hierarchical algorithms
    - Divisive
    - Agglomerative

Today: dimensionality reduction & clustering

- Dimensionality reduction techniques
  - Principal component analysis (PCA)
  - Independent component analysis (ICA)
  - Canonical correlation analysis (CCA)
  - Laplacian eigenmaps
- Clustering
  - Partitional algorithms
    - K-means
    - Spectral clustering
  - Hierarchical algorithms
    - Divisive
    - Agglomerative

Dimensionality of the relevant information is often lower than dimensionality of the data

- Data dimension: high
  - 100s of fMRI voxels in an ROI
  - 100s of sensors or sources in MEG recordings

Dimensionality of the relevant information is often lower than dimensionality of the data

- Data dimension: high
  - 100s of fMRI voxels in an ROI
  - 100s of sensors or sources in MEG recordings
- Relevant information dimension: low

Dimensionality of the relevant information is often lower than dimensionality of the data

- Data dimension: high
  - 100s of fMRI voxels in an ROI
  - 100s of sensors or sources in MEG recordings
- Relevant information dimension: low
  - Redundant features can add more noise than signal

Dimensionality of the relevant information is often lower than dimensionality of the data

- Data dimension: high
  - 100s of fMRI voxels in an ROI
  - 100s of sensors or sources in MEG recordings
- Relevant information dimension: low
  - Redundant features can add more noise than signal
  - Dimension of relevant information depends on the number of free parameters describing the probability densities

Dimensionality of the relevant information is often lower than dimensionality of the data

- Data dimension: high
  - 100s of fMRI voxels in an ROI
  - 100s of sensors or sources in MEG recordings
- Relevant information dimension: low
  - Redundant features can add more noise than signal
  - Dimension of relevant information depends on the number of free parameters describing the probability densities
    - For supervised methods, want to learn P(labels|data)

Dimensionality of the relevant information is often lower than dimensionality of the data

- Data dimension: high
  - 100s of fMRI voxels in an ROI
  - 100s of sensors or sources in MEG recordings
- Relevant information dimension: low
  - Redundant features can add more noise than signal
  - Dimension of relevant information depends on the number of free parameters describing the probability densities
    - For supervised methods, want to learn P(labels|data)
    - For unsupervised methods, want to learn P(data)

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features

Goal: construct features that better account for the variance of the data

- ❏ Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- ❏ We've seen one way to do this already -- what is it?

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection
    - Directly removes subsets of the observed features

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection
    - Directly removes subsets of the observed features
    - We've seen this in the context of a supervised task -- wanting to maximize the mutual information between selected features and labels

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection
    - Directly removes subsets of the observed features
    - We've seen this in the context of a supervised task -- wanting to maximize the mutual information between selected features and labels
    - What is the alternative in unsupervised tasks?

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection
    - Directly removes subsets of the observed features
    - We've seen this in the context of a supervised task -- wanting to maximize the mutual information between selected features and labels
    - What is the alternative in unsupervised tasks?
      - Non trivial

Goal: construct features that better account for the variance of the data

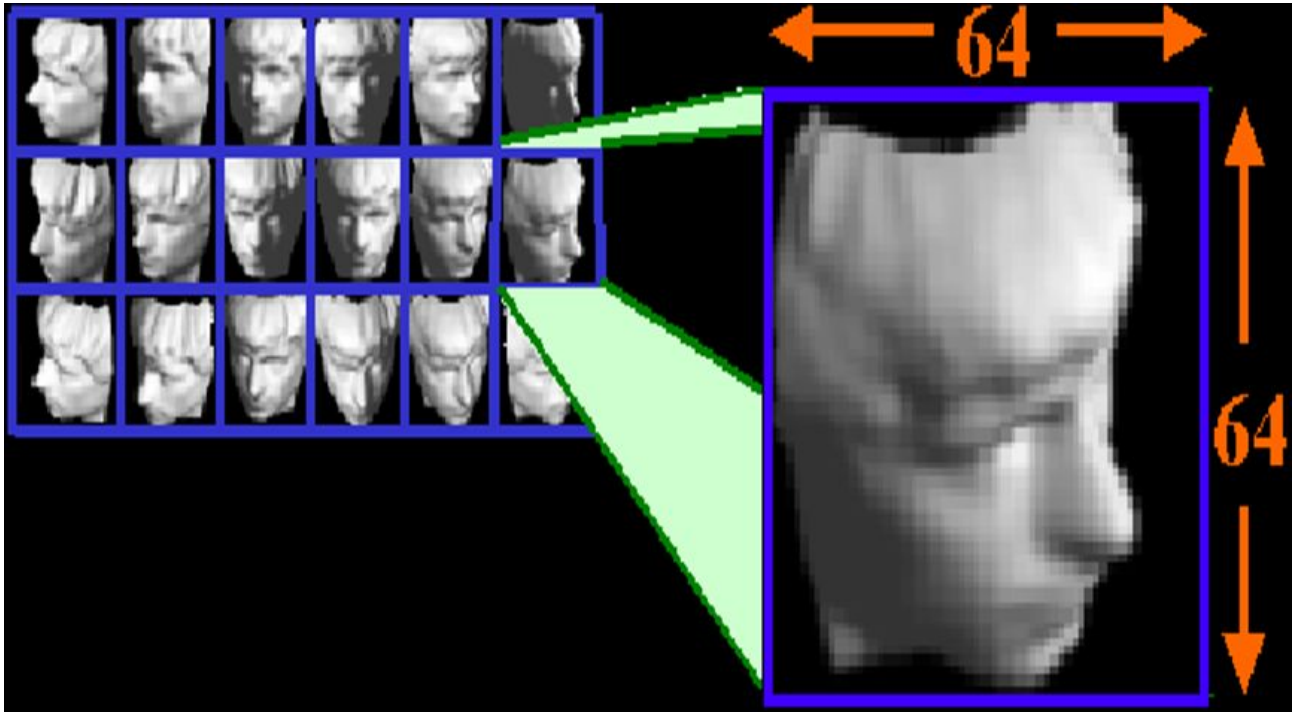
- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection
    - Directly removes subsets of the observed features
    - We've seen this in the context of a supervised task -- wanting to maximize the mutual information between selected features and labels
    - What is the alternative in unsupervised tasks?
      - Non trivial
  - Latent features extraction (usually what people have in mind when they say dimensionality reduction)

Goal: construct features that better account for the variance of the data

- Want to combine highly correlated or dependent features and focus on uncorrelated or independent features
- We've seen one way to do this already -- what is it?
  - Feature selection
    - Directly removes subsets of the observed features
    - We've seen this in the context of a supervised task -- wanting to maximize the mutual information between selected features and labels
    - What is the alternative in unsupervised tasks?
      - Non trivial
  - Latent features extraction (usually what people have in mind when they say dimensionality reduction)
    - Some linear or nonlinear combination of observed features provides more efficient representation for the data than observed features

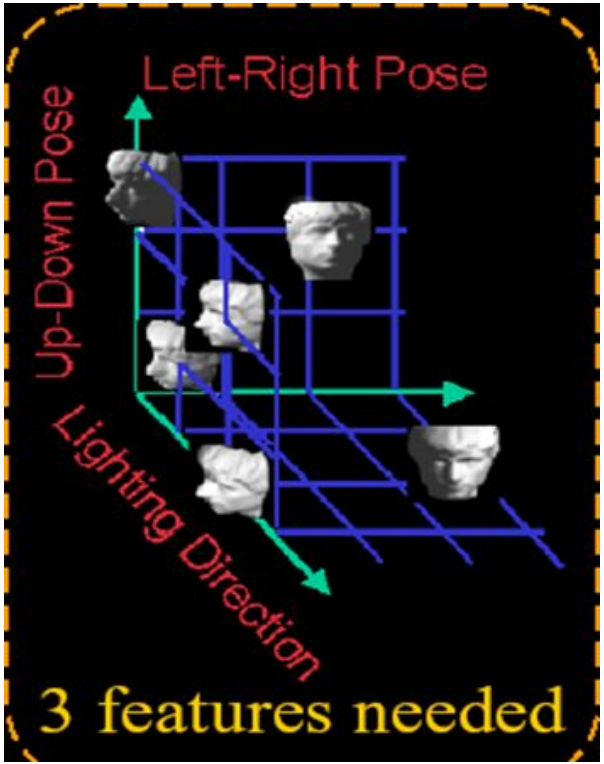
Example: how does the brain store these pictures?

Data: 64x64 dimensions, but are there fewer underlying relevant dimensions?



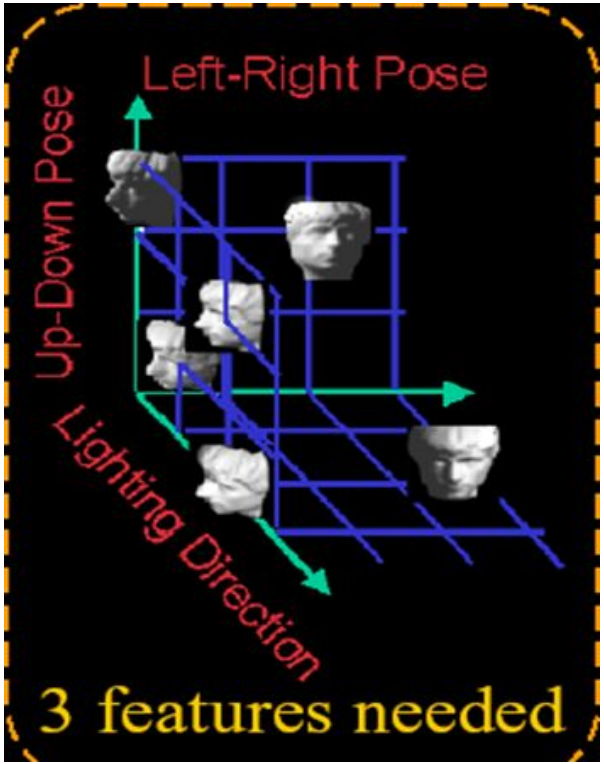
We can condense data to 3 underlying informative dimensions

- Don’t need every pixel (64x64)



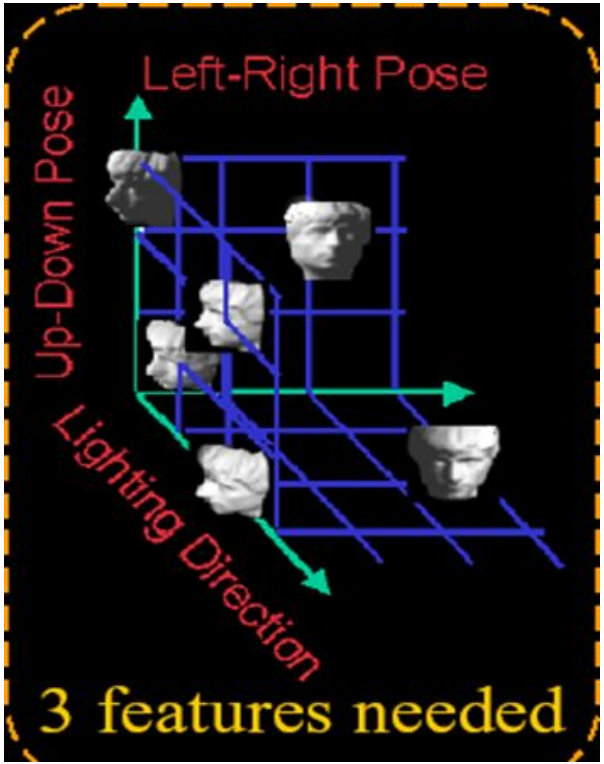
We can condense data to 3 underlying informative dimensions

- Don’t need every pixel (64x64)
- We want to extract perceptually meaningful structure



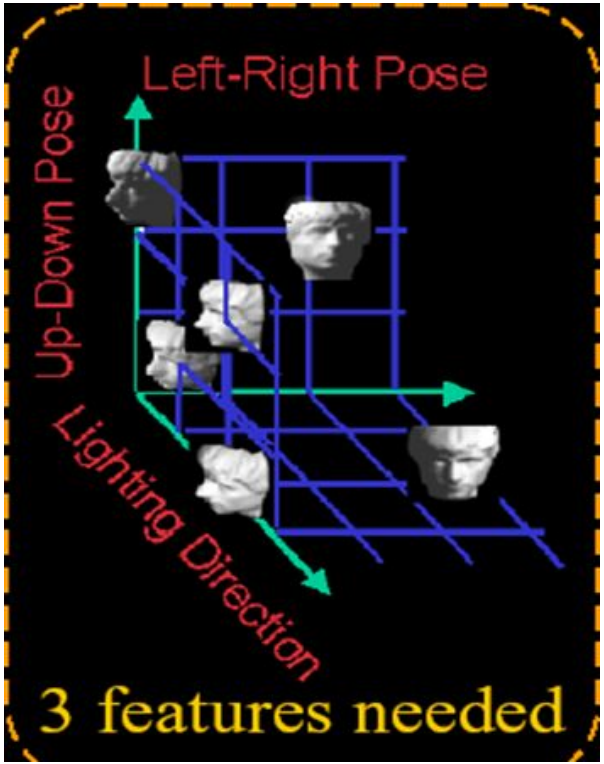
We can condense data to 3 underlying informative dimensions

- Don’t need every pixel (64x64)
- We want to extract perceptually meaningful structure
  - Up-down pose
  - Left-right pose
  - Lighting direction



We can condense data to 3 underlying informative dimensions

- Don’t need every pixel (64x64)
- We want to extract perceptually meaningful structure
  - Up-down pose
  - Left-right pose
  - Lighting direction
- Reduction of high-dimensional inputs to 3-dimensional intrinsic manifold

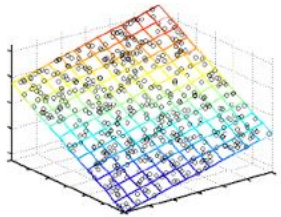


How do we find latent dimensions in the observed data?

- What kind of manifold does our observed data lie on?

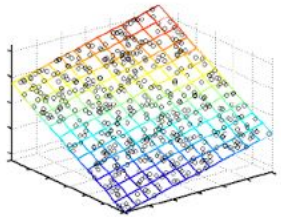
How do we find latent dimensions in the observed data?

- What kind of manifold does our observed data lie on?
  - Linear



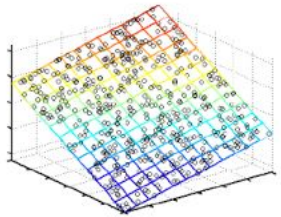
How do we find latent dimensions in the observed data?

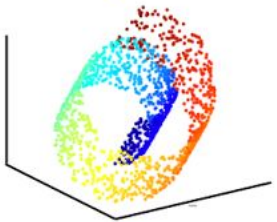
- What kind of manifold does our observed data lie on?
  - Linear
    - Principal component analysis (PCA)
    - Independent component analysis (ICA)
    - Canonical correlation analysis (CCA)



How do we find latent dimensions in the observed data?

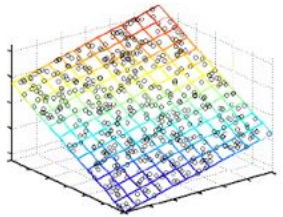
- What kind of manifold does our observed data lie on?
  - Linear
    - Principal component analysis (PCA)
    - Independent component analysis (ICA)
    - Canonical correlation analysis (CCA)
  - Nonlinear

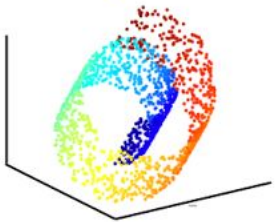




How do we find latent dimensions in the observed data?

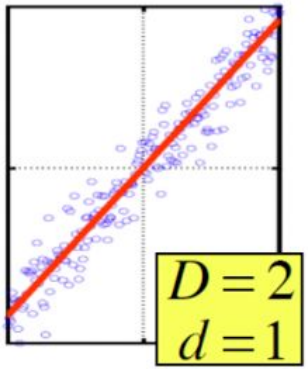
- What kind of manifold does our observed data lie on?
  - Linear
    - Principal component analysis (PCA)
    - Independent component analysis (ICA)
    - Canonical correlation analysis (CCA)
  - Nonlinear
    - Laplacian eigenmaps

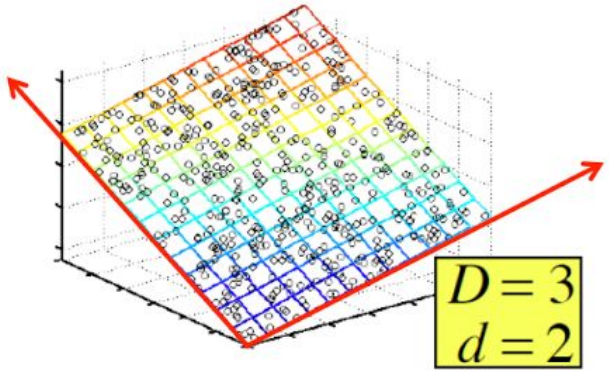




Principal component analysis (PCA) identifies axes of latent low-dimensional linear subspace of features

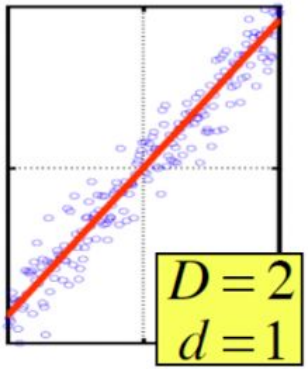
- Assumption: observed **D**-dimensional data lies on or near a low **d**-dimensional linear subspace

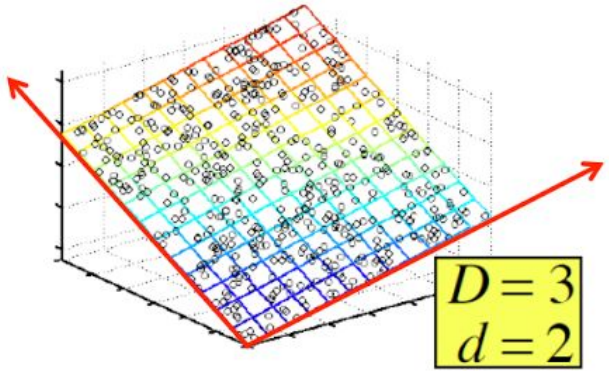




Principal component analysis (PCA) identifies axes of latent low-dimensional linear subspace of features

- Assumption: observed **D**-dimensional data lies on or near a low **d**-dimensional linear subspace

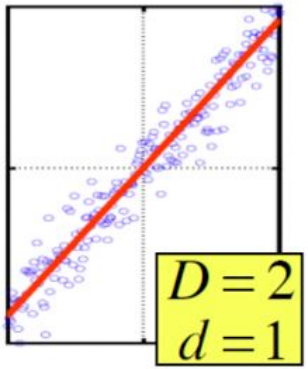


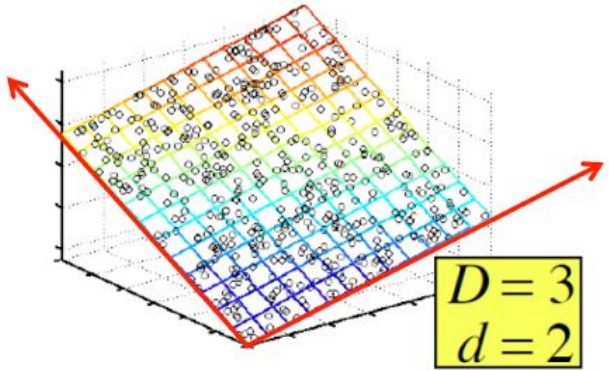


- Axes of this subspace are an effective representation of the data

Principal component analysis (PCA) identifies axes of latent low-dimensional linear subspace of features

- Assumption: observed **D**-dimensional data lies on or near a low **d**-dimensional linear subspace

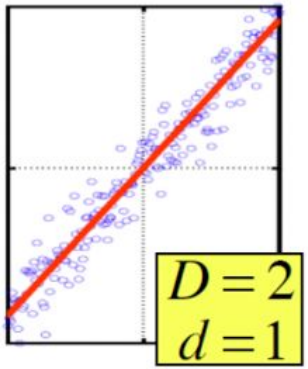


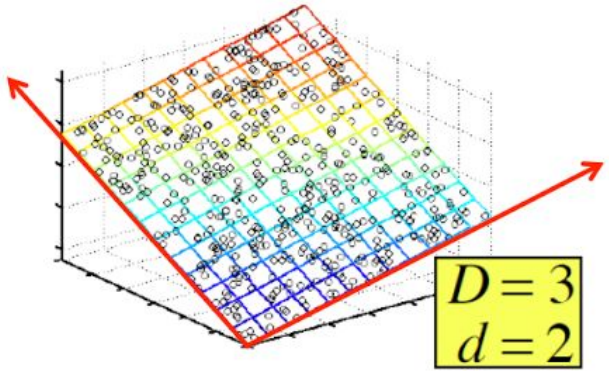


- Axes of this subspace are an effective representation of the data
  - What does an “effective” representation mean?

Principal component analysis (PCA) identifies axes of latent low-dimensional linear subspace of features

- Assumption: observed **D**-dimensional data lies on or near a low **d**-dimensional linear subspace

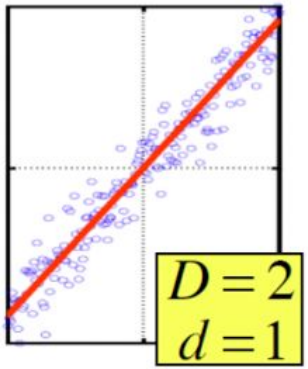


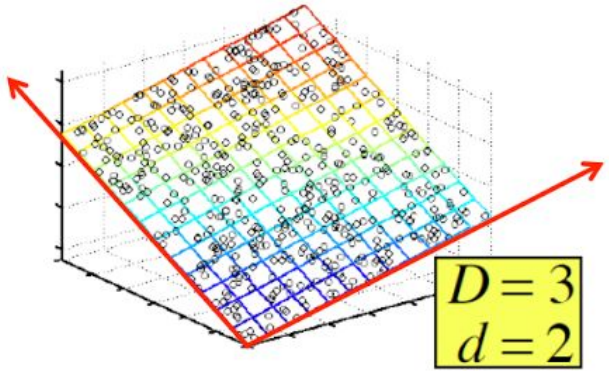


- Axes of this subspace are an effective representation of the data
  - What does an “effective” representation mean?
    - Able to distinguish data instances that are truly different

Principal component analysis (PCA) identifies axes of latent low-dimensional linear subspace of features

- Assumption: observed **D**-dimensional data lies on or near a low **d**-dimensional linear subspace





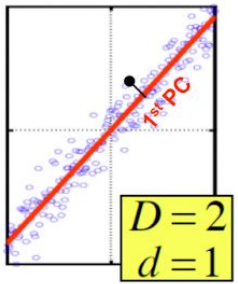
- Axes of this subspace are an effective representation of the data
  - What does an “effective” representation mean?
    - Able to distinguish data instances that are truly different
- Goal: identify these axes = also known as the principal components (PCs)

PCA: algorithm intuition

- PCs are the axes of the subspace so they’re orthogonal to each other

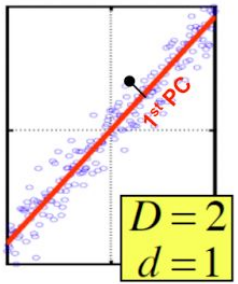
PCA: algorithm intuition

- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data



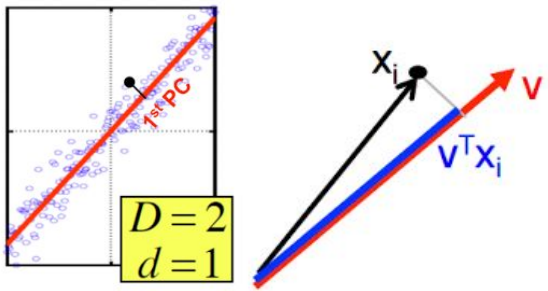
PCA: algorithm intuition

- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on



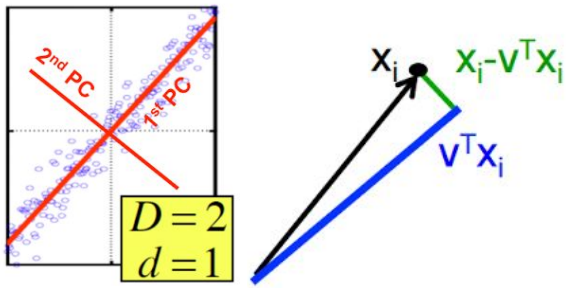
PCA: algorithm intuition

- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on
  - projection of data points along 1st PC discriminate the data most along any direction
    - let data instance x_i be a 2-dimensional vector
    - let 1st PC be vector v
    - projection of x_i onto v is $v^T x_i$



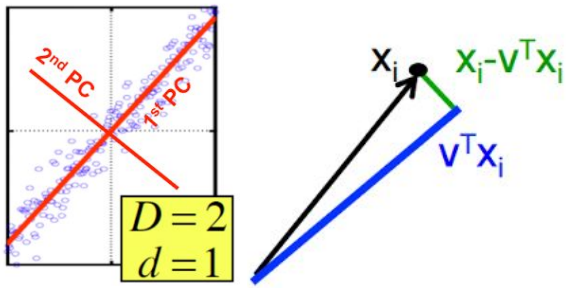
PCA: algorithm intuition

- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on
  - projection of data points along 1st PC discriminate the data most along any direction
    - let data instance x_i be a 2-dimensional vector
    - let 1st PC be vector v
    - projection of x_i onto v is $v^T x_i$
  - 2nd PC = next orthogonal direction of greatest variability



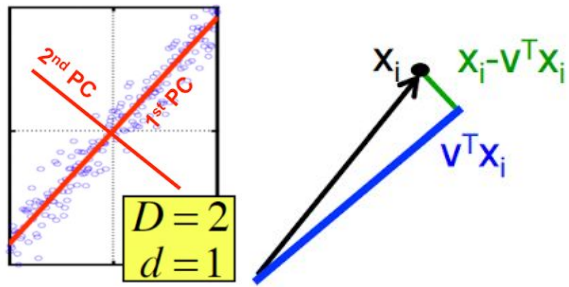
PCA: algorithm intuition

- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on
  - projection of data points along 1st PC discriminate the data most along any direction
    - let data instance x_i be a 2-dimensional vector
    - let 1st PC be vector v
    - projection of x_i onto v is $v^T x_i$
  - 2nd PC = next orthogonal direction of greatest variability
    - Remove all variability in first direction, then find next PC



PCA: algorithm intuition

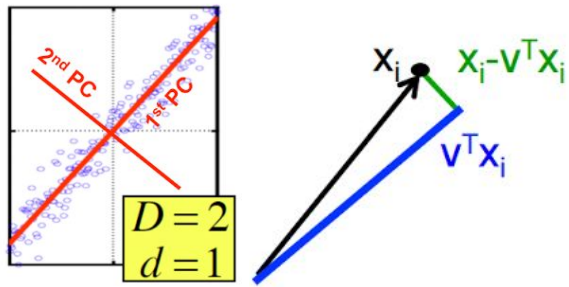
- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on
  - projection of data points along 1st PC discriminate the data most along any direction
    - let data instance x_i be a 2-dimensional vector
    - let 1st PC be vector v
    - projection of x_i onto v is $v^T x_i$
  - 2nd PC = next orthogonal direction of greatest variability
    - Remove all variability in first direction, then find next PC



- Once the PCs are computed, we can reduce dimensionality of data

PCA: algorithm intuition

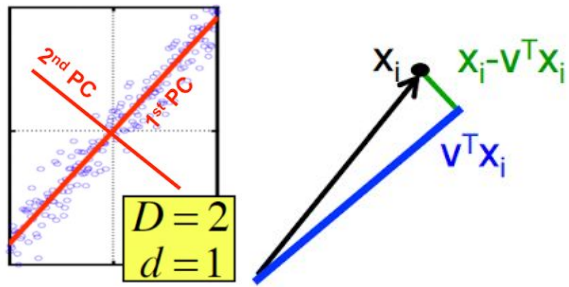
- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on
  - projection of data points along 1st PC discriminate the data most along any direction
    - let data instance x_i be a 2-dimensional vector
    - let 1st PC be vector v
    - projection of x_i onto v is $v^T x_i$
  - 2nd PC = next orthogonal direction of greatest variability
    - Remove all variability in first direction, then find next PC



- Once the PCs are computed, we can reduce dimensionality of data
  - Original D-dimensional data instance $x_i = \langle x_i^1, \ldots, x_i^D \rangle$

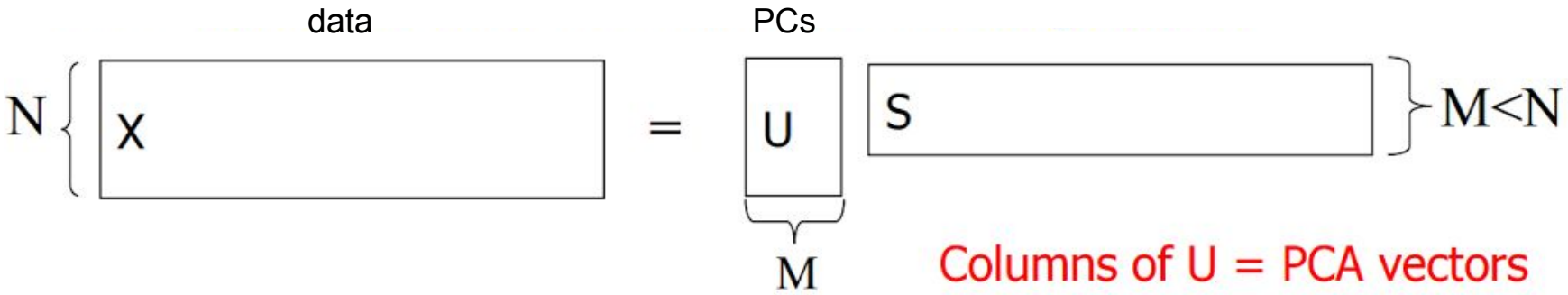
PCA: algorithm intuition

- PCs are the axes of the subspace so they're orthogonal to each other
- Intuitively, PCs are the orthogonal directions that capture most of the variance in the data
  - ordered so that 1st PC is direction of greatest variability in data, and so on
  - projection of data points along 1st PC discriminate the data most along any direction
    - let data instance x_i be a 2-dimensional vector
    - let 1st PC be vector v
    - projection of x_i onto v is $v^T x_i$
  - 2nd PC = next orthogonal direction of greatest variability
    - Remove all variability in first direction, then find next PC



- Once the PCs are computed, we can reduce dimensionality of data
  - Original D-dimensional data instance $x_i = \langle x_i^1, \ldots, x_i^D \rangle$
  - Reduced d-dimensional transformations: transformed $x_i = \langle v_1^T x_i, \ldots, v_d^T x_i \rangle$

PCA: low rank matrix factorization for compression



How do we know how many PCs we need?

- Plot how much variance in the data is explained by each PC

How do we know how many PCs we need?

- Plot how much variance in the data is explained by each PC
- Because PCs are ordered (1st PC explains the most variance), we can do a cumulative plot of variance explained

How do we know how many PCs we need?

- Plot how much variance in the data is explained by each PC
- Because PCs are ordered (1st PC explains the most variance), we can do a cumulative plot of variance explained
  - Cut off when certain % of variance explained is reached or when you see a sharp decrease in % variance explained

PCA takeaways

- PCA is a linear dimensionality reduction technique
  - Limited to linear projections of data

PCA takeaways

- PCA is a linear dimensionality reduction technique
  - Limited to linear projects of data
- Exact solution (non-iterative)

PCA takeaways

- PCA is a linear dimensionality reduction technique
  - Limited to linear projects of data
- Exact solution (non-iterative)
- No local optima

PCA takeaways

- PCA is a linear dimensionality reduction technique
  - Limited to linear projects of data
- Exact solution (non-iterative)
- No local optima
- No tuning parameters

PCA takeaways

- PCA is a linear dimensionality reduction technique
  - Limited to linear projects of data
- Exact solution (non-iterative)
- No local optima
- No tuning parameters
- Note that PCA assumes that data is centered (mean is subtracted)

PCA takeaways

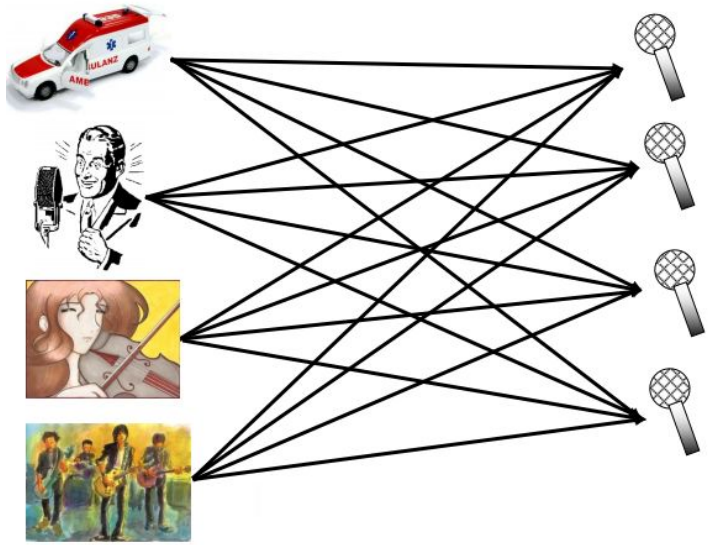
- PCA is a linear dimensionality reduction technique
  - Limited to linear projects of data
- Exact solution (non-iterative)
- No local optima
- No tuning parameters
- Note that PCA assumes that data is centered (mean is subtracted)
- PCs are orthogonal

Do we need the latent dimensions to be orthogonal?

Do we need the latent dimensions to be orthogonal?

- What if instead, they're statistically independent? => Independent component analysis (ICA)

ICA aims to separate the observed data into some underlying signals that have been mixed



underlying sources of signal

mixed observations

Similarities between PCA and ICA

- Both are linear methods (perform linear transformations)

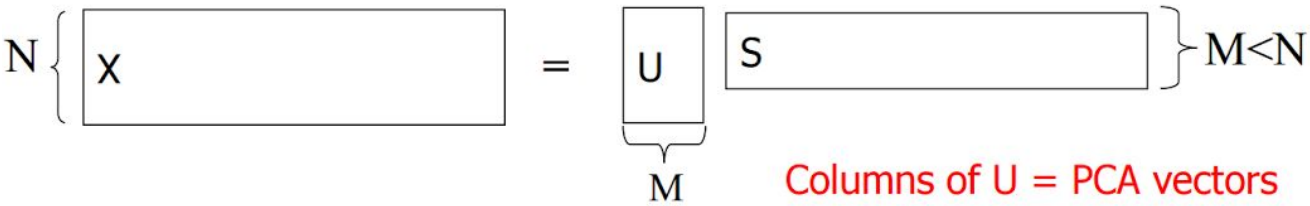
Similarities between PCA and ICA

- Both are linear methods (perform linear transformations)
- Both can be formulated as a matrix factorization problem

Similarities between PCA and ICA

- Both are linear methods (perform linear transformations)
- Both can be formulated as a matrix factorization problem

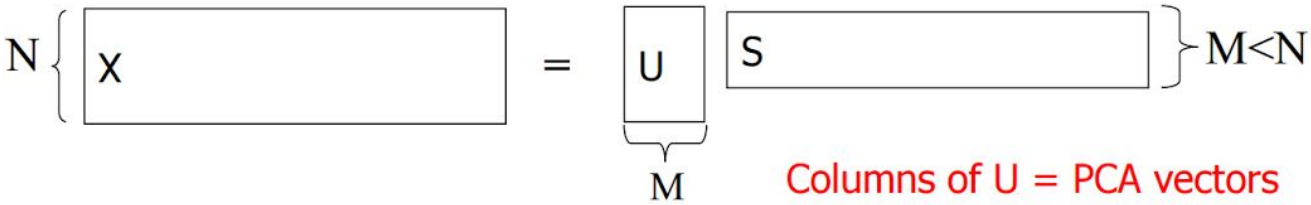
PCA: **low rank** matrix factorization for **compression**



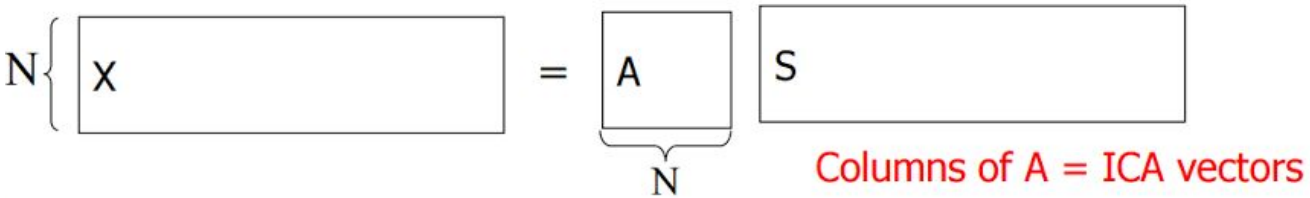
Similarities between PCA and ICA

- Both are linear methods (perform linear transformations)
- Both can be formulated as a matrix factorization problem

PCA: **low rank** matrix factorization for **compression**



ICA: **full rank** matrix factorization to **remove dependencies** among rows



Differences between PCA and ICA

- PCA does compression (fewer latent dimensions than observed dimensions)
- ICA does not do compression (same number of features)

Differences between PCA and ICA

- PCA does compression (fewer latent dimensions than observed dimensions)
- ICA does not do compression (same number of features)

- PCA just removes correlations and not higher order dependence
- ICA removes correlations, and higher order dependence

Differences between PCA and ICA

- PCA does compression (fewer latent dimensions than observed dimensions)
- ICA does not do compression (same number of features)

- PCA just removes correlations and not higher order dependence
- ICA removes correlations, and higher order dependence

- In PCA, some components are more important than others
- In ICA, all components are equally important

Differences between PCA and ICA

- PCA does compression (fewer latent dimensions than observed dimensions)
- ICA does not do compression (same number of features)

- PCA just removes correlations and not higher order dependence
- ICA removes correlations, and higher order dependence

- In PCA, some components are more important than others
- In ICA, all components are equally important

- In PCA, components are orthogonal
- In ICA, components are not orthogonal but statistically independent

ICA takeaways

- ICA is a linear method that aims to separate the observed signal into underlying sources that have been mixed

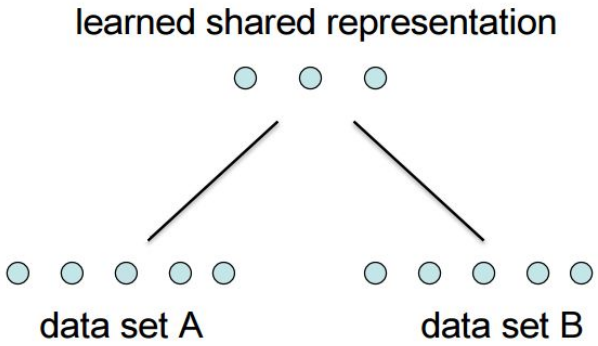
ICA takeaways

- ICA is a linear method that aims to separate the observed signal into underlying sources that have been mixed
- Unlike PCA, which finds orthogonal latent components, ICA finds components that are statistically independent

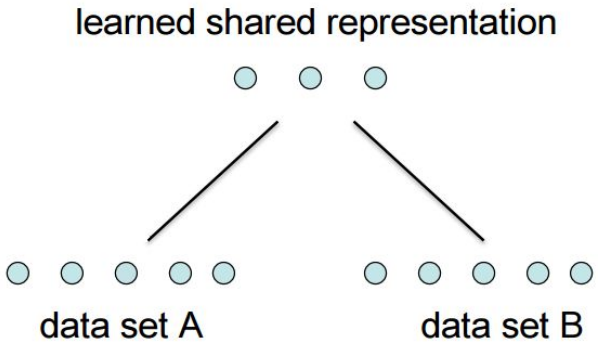
ICA takeaways

- ICA is a linear method that aims to separate the observed signal into underlying sources that have been mixed
- Unlike PCA, which finds orthogonal latent components, ICA finds components that are statistically independent
- Used for denoising and source localization in neuroimaging

What if we're interested in explaining variance in multiple data sets at the same time?

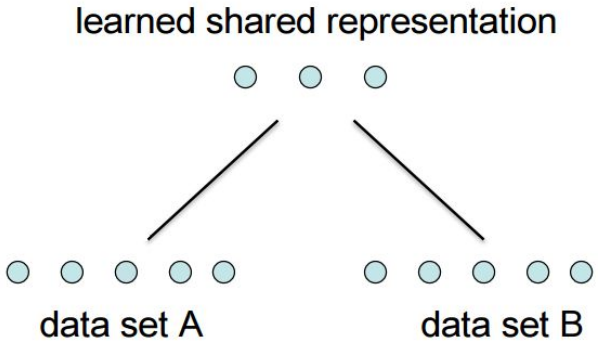


What if we're interested in explaining variance in multiple data sets at the same time?



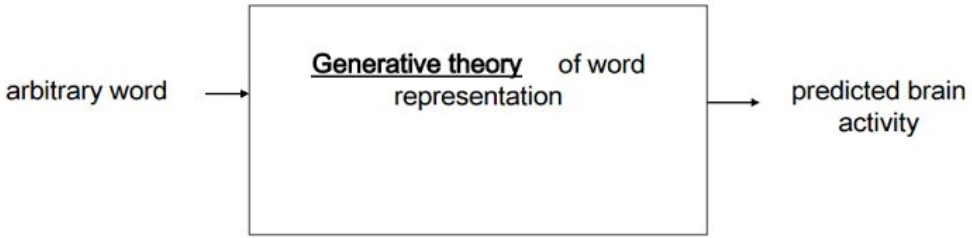
- Example: simultaneous recordings of NIRS (data set A) and fMRI (data set B), can we find the common brain processes?

What if we're interested in explaining variance in multiple data sets at the same time?

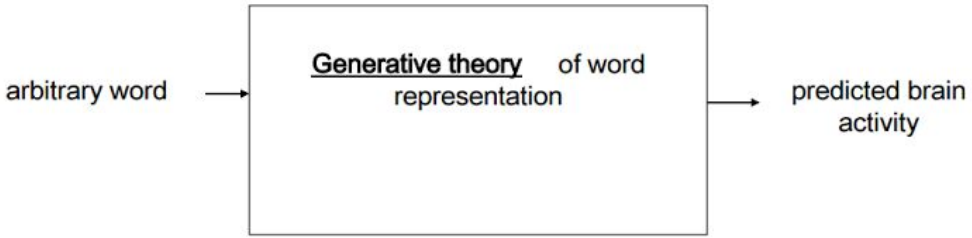


- Example: simultaneous recordings of NIRS (data set A) and fMRI (data set B), can we find the common brain processes?
- Canonical correlation analysis (CCA) maximizes the correlation of the projected data A and data B in the common latent lower-dimensional space

CCA: example use in neuroscience = discovering shared semantic basis between people

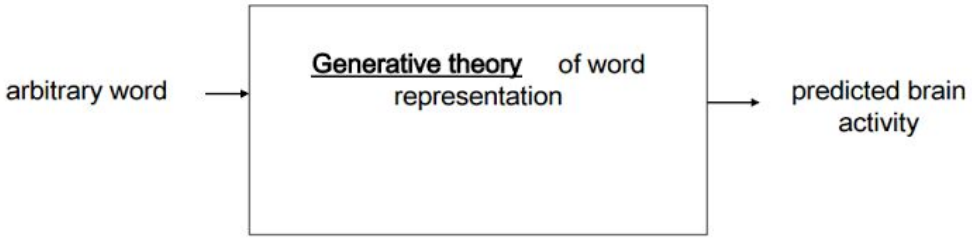


CCA: example use in neuroscience = discovering shared semantic basis between people



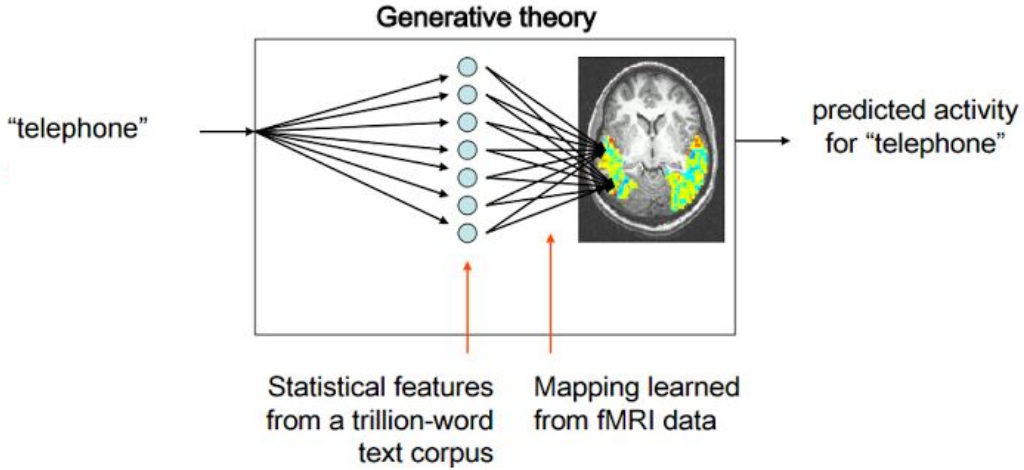
- In an fMRI, we present words to subjects, one at a time

CCA: example use in neuroscience = discovering shared semantic basis between people



- In an fMRI, we present words to subjects, one at a time
- We aim to use the fMRI data, along with its corresponding word label to train a general model that can predict brain activity for an arbitrary word (even a word that was never presented to the subject)

Let’s consider semantics within one person first



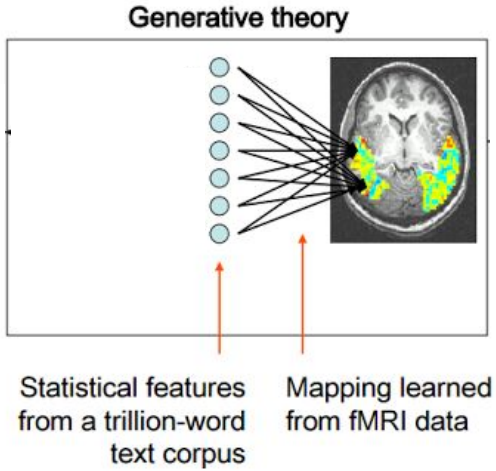
Mitchell et al., Science 2008

Semantic features for 2 presented words

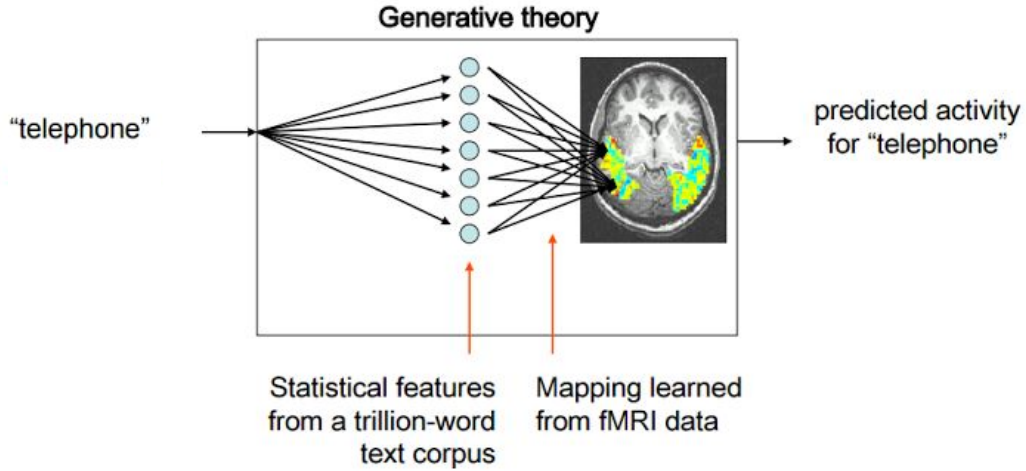
Semantic feature values:
"**celery**"
0.8368, eat
0.3461, taste
0.3153, fill
0.2430, see
0.1145, clean
0.0600, open
0.0586, smell
0.0286, touch
…
…
0.0000, drive
0.0000, wear
0.0000, lift
0.0000, break
0.0000, ride

Semantic feature values:
"**airplane**"
0.8673, ride
0.2891, see
0.2851, say
0.1689, near
0.1228, open
0.0883, hear
0.0771, run
0.0749, lift
…
…
0.0049, smell
0.0010, wear
0.0000, taste
0.0000, rub
0.0000, manipulate

Training the model: learn a regression between semantic vectors and fMRI data for all words in training set



Applying the model: for any given word, look up semantic vector, then apply learned regression weights to generate corresponding fMRI image



How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation

How do we evaluate how well we can predict fMRI images for arbitrary words?

- ❏ We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- ❏ Do leave-2-out cross validation

How do we evaluate how well we can predict fMRI images for arbitrary words?

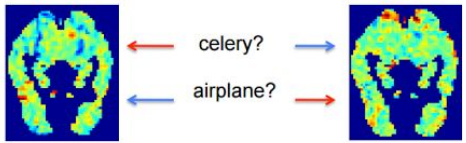
- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded

How do we evaluate how well we can predict fMRI images for arbitrary words?

- ❏ We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- ❏ Do leave-2-out cross validation
  - ❏ Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - ❏ Apply model on the remaining 2 test words to predict 2 fMRI images

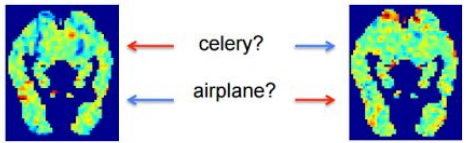
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word



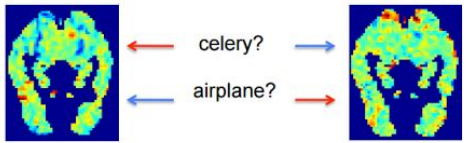
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word
  - 1770 test pairs (60 words choose 2)



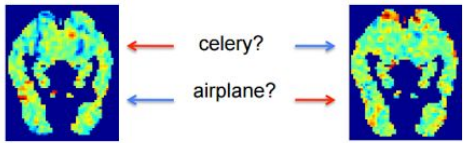
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word
  - 1770 test pairs (60 words choose 2)
    - Random guessing -> ?



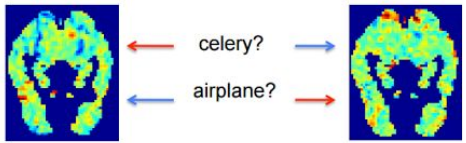
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word
  - 1770 test pairs (60 words choose 2)
    - Random guessing -> 0.50 accuracy



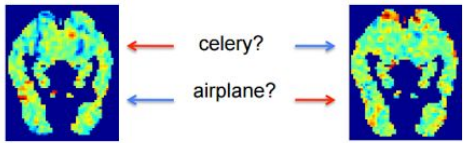
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word
  - 1770 test pairs (60 words choose 2)
    - Random guessing -> 0.50 accuracy
    - Accuracy above 0.61 is significant ($p<0.05$, permutation test)



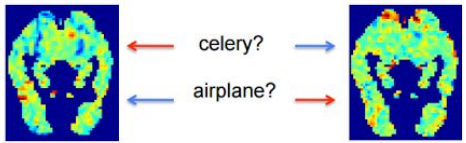
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don’t have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word
  - 1770 test pairs (60 words choose 2)
    - Random guessing -> 0.50 accuracy
    - Accuracy above 0.61 is significant ($p<0.05$, permutation test)
    - Mean accuracy over 9 subjects is 0.79



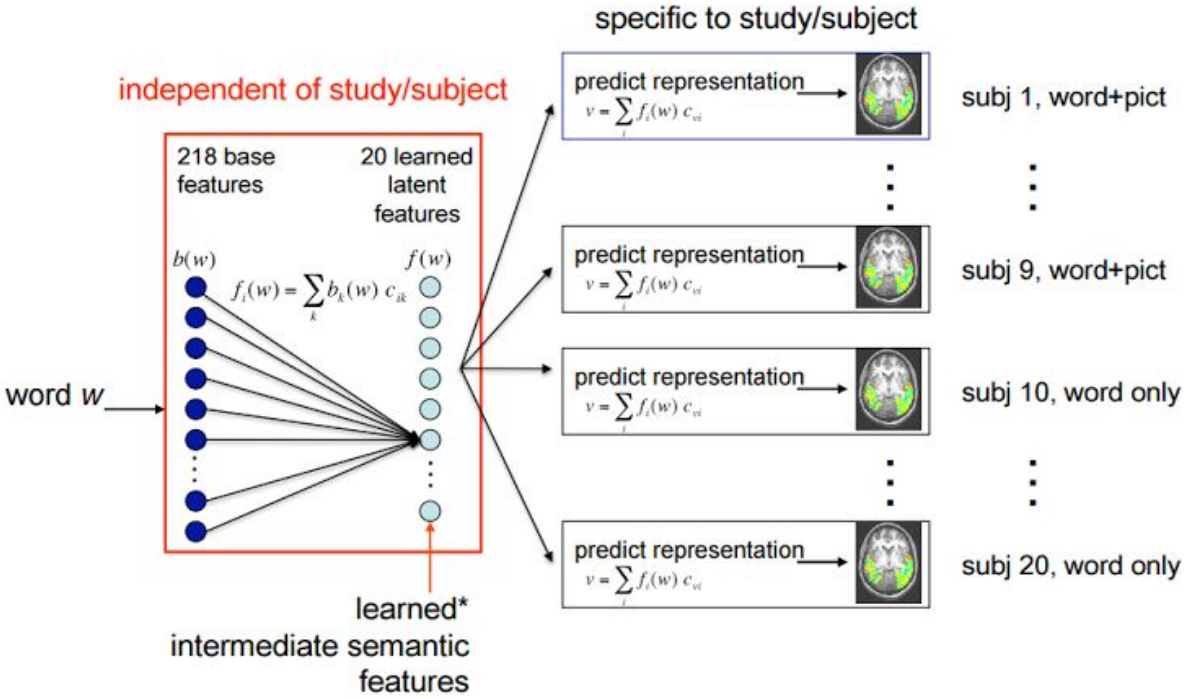
How do we evaluate how well we can predict fMRI images for arbitrary words?

- We don't have fMRI images for any arbitrary word, so there is no ground truth for evaluation
- Do leave-2-out cross validation
  - Train on 58 of 60 words and their corresponding fMRI images we have recorded
  - Apply model on the remaining 2 test words to predict 2 fMRI images
  - Test model by showing it the 2 true fMRI images corresponding to the held out words and asking it to guess which image corresponds to which word
  - 1770 test pairs (60 words choose 2)
    - Random guessing -> 0.50 accuracy
    - Accuracy above 0.61 is significant ($p<0.05$, permutation test)
    - Mean accuracy over 9 subjects is 0.79



- We can predict the fMRI activation corresponding to a word the model has never seen before

CCA: extending the model to multiple subjects and experiments improves accuracy to 87% (by 8%)



CCA takeaways

- CCA learns linear transformations of multiple data sets, such that these transformations are maximally correlated

CCA takeaways

- CCA learns linear transformations of multiple data sets, such that these transformations are maximally correlated
- In neuroscience, it can be used to extend models to multiple subjects or experiments

Beyond linear transformation: laplacian eigenmaps

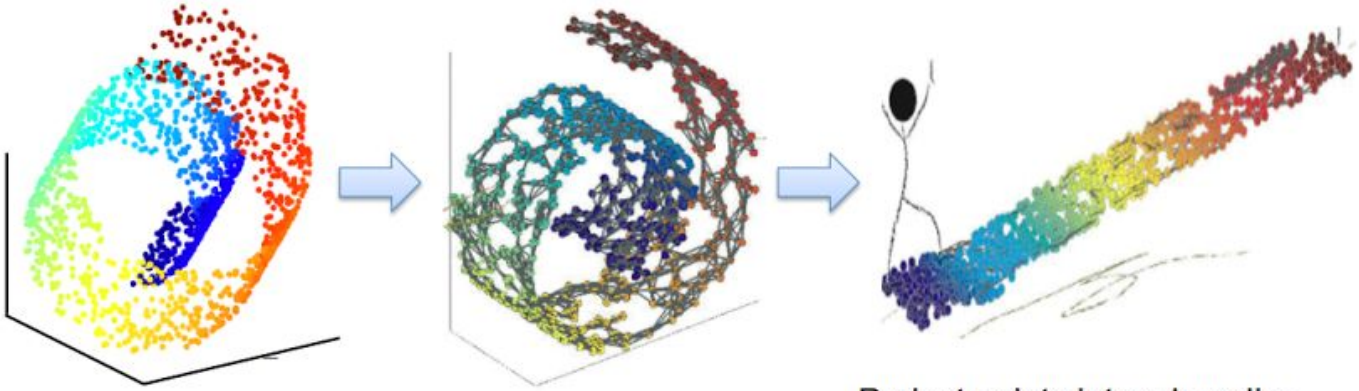
- Linear methods find lower-dimensional linear projects that preserves distances between **all** points

Beyond linear transformation: laplacian eigenmaps

- Linear methods find lower-dimensional linear projects that preserves distances between **all** points
- Laplacian eigenmaps preserves **local information only**

Beyond linear transformation: laplacian eigenmaps

- Linear methods find lower-dimensional linear projects that preserves distances between **all** points
- Laplacian eigenmaps preserves **local information only**



Construct graph from data points (capture local information)

Project points into a low-dim space using “eigenvectors of the graph”

Laplacian eigenmaps: 3 main steps

- ❏ Construct graph

Laplacian eigenmaps: 3 main steps

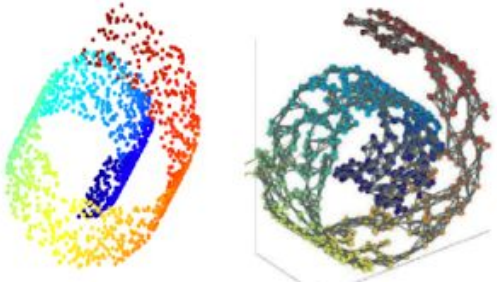
- ❏ Construct graph
- ❏ Compute graph Laplacian

Laplacian eigenmaps: 3 main steps

- ❏ Construct graph
- ❏ Compute graph Laplacian
- ❏ Embed points using graph Laplacian

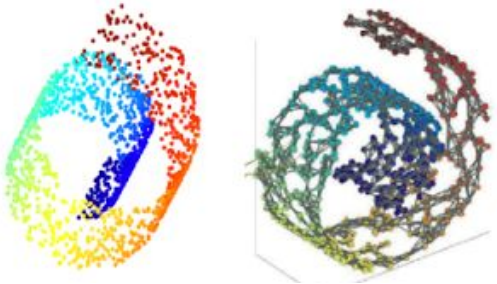
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points



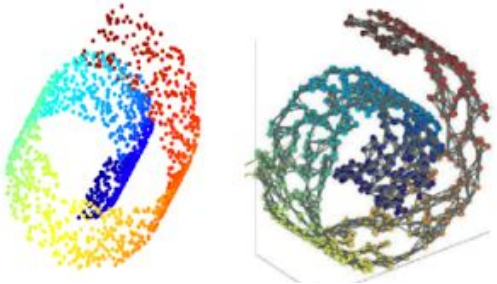
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)



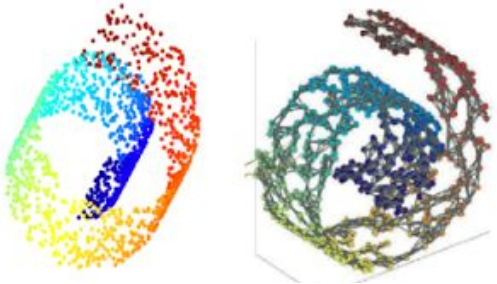
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points



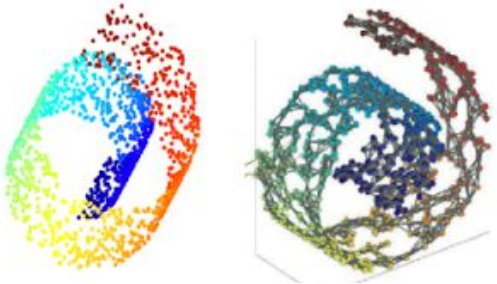
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges



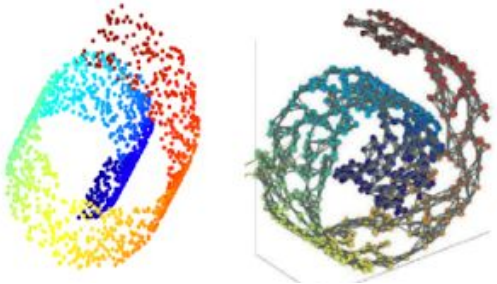
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
      - put edge between 2 data points if they are within ε distance of each other



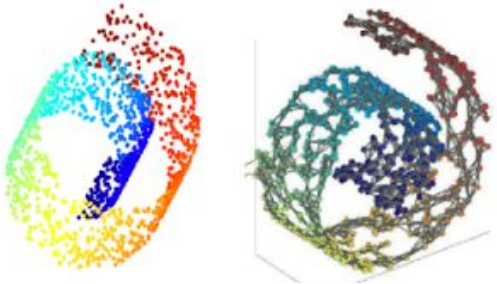
Step 1: constructing a similarity graph

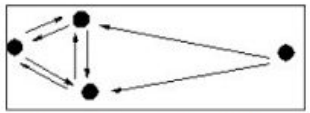
- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
      - put edge between 2 data points if they are within ε distance of each other
      - put edge between 2 data points if one is a k-NN of another
        - can lead to 3 types of graphs:



Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
      - put edge between 2 data points if they are within ε distance of each other
      - put edge between 2 data points if one is a k-NN of another
        - can lead to 3 types of graphs:
          - directed: edge A->B if A is k-NN of B

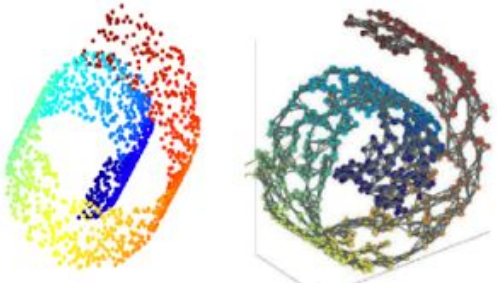


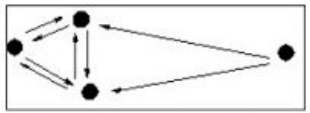


Directed nearest neighbors

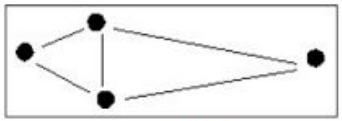
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
      - put edge between 2 data points if they are within ε distance of each other
      - put edge between 2 data points if one is a k-NN of another
        - can lead to 3 types of graphs:
          - directed: edge A->B if A is k-NN of B
          - symmetric: edge A-B if A is k-NN of B OR B is k-NN of A





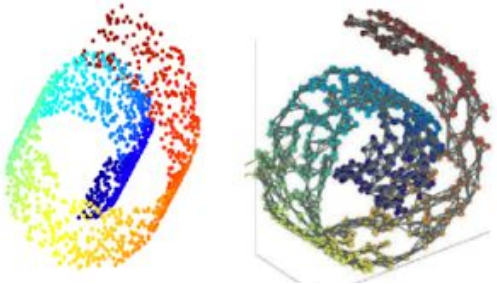
Directed nearest neighbors

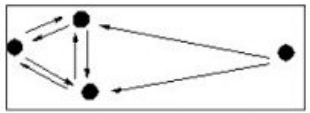


(symmetric) kNN graph

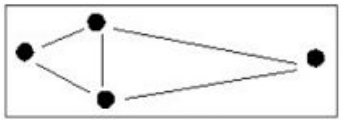
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
      - put edge between 2 data points if they are within ε distance of each other
      - put edge between 2 data points if one is a k-NN of another
        - can lead to 3 types of graphs:
          - directed: edge A->B if A is k-NN of B
          - symmetric: edge A-B if A is k-NN of B OR B is k-NN of A
          - mutual: edge A-B if A is k-NN of B AND B is k-NN of A





Directed nearest neighbors

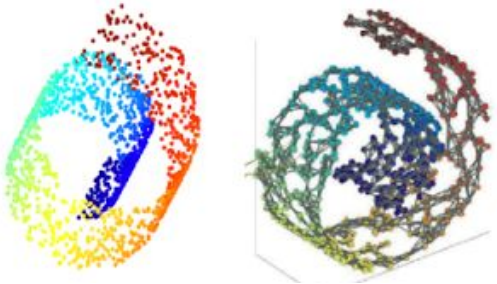


(symmetric) kNN graph

mutual kNN graph

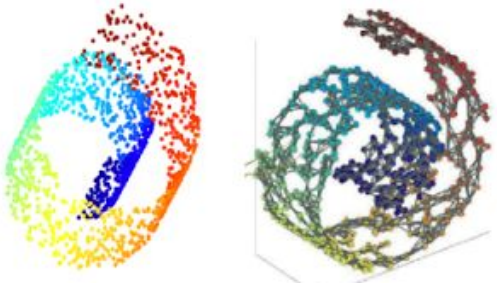
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
  - W = weights



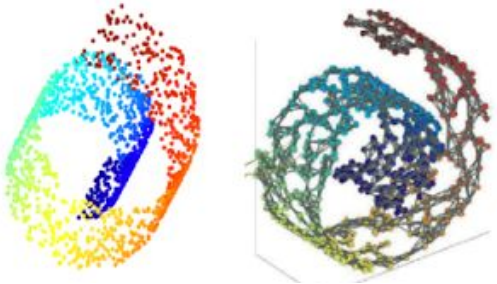
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
  - W = weights
    - 2 ways to make weights:



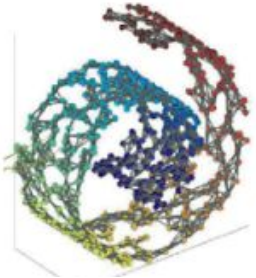
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
  - W = weights
    - 2 ways to make weights:
      - W_{ij}= 1 if edge between nodes i and j is present, 0 otherwise



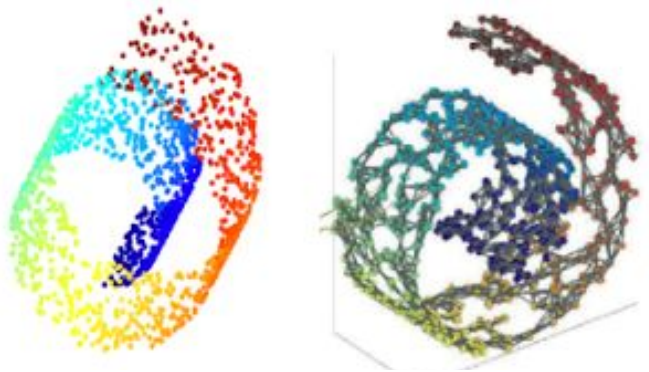
Step 1: constructing a similarity graph

- Similarity graphs model local neighborhood relations between data points
- Graph fully described by vertices, edges, and weights
  - G(V,E,W)
  - V = vertices = all data points
  - E = edges
    - 2 ways to construct edges
  - W = weights
    - 2 ways to make weights:
      - W_{ij}= 1 if edge between nodes i and j is present, 0 otherwise
      - $W_{ij}= e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$, Gaussian kernel similarity function (aka heat kernel)



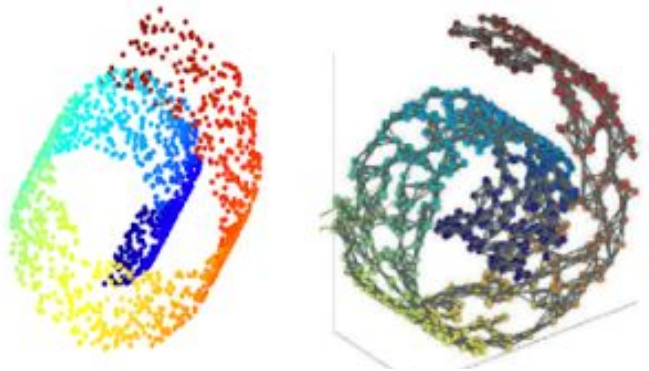
How do we choose k, ε?

- The goal is to preserve local information so we don't want to choose neighborhood sizes that are too large



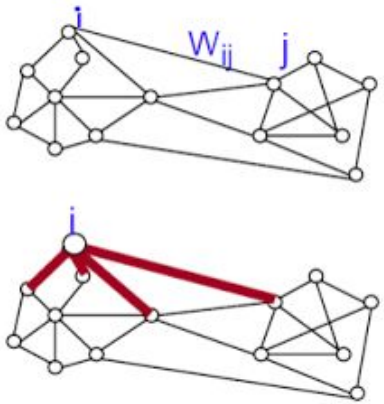
How do we choose k, ε?

- The goal is to preserve local information so we don't want to choose neighborhood sizes that are too large
- Mostly dependent on the data, but want to avoid "shortcuts" that connect different arms of the swiss roll



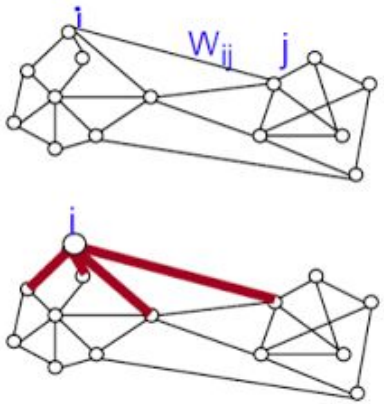
Step 2: compute the graph Laplacian of the constructed graph

- Graph Laplacian = D - W



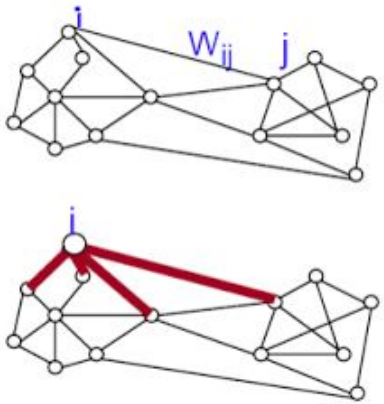
Step 2: compute the graph Laplacian of the constructed graph

- Graph Laplacian = D - W
- W = weight matrix from the constructed graph
  - For n data points, W is size n x n



Step 2: compute the graph Laplacian of the constructed graph

- Graph Laplacian = D - W
- W = weight matrix from the constructed graph
  - For n data points, W is size n x n
- D = degree matrix = diag(d_1,...,d_n)
  - d_i = degree of vertex i = sum of all weights that connect to vertex i
  - For n data points, D is size n x n



Step 3: embed data points using graph Laplacian

Original Representation		Transformed representation
data point		projections
x_i	→	$(f_1(i), \ldots, f_d(i))$
(D-dimensional vector)		(d-dimensional vector)

- Intuition = find vector f such that, if x_i is close to x_j in the graph (i.e. W_{ij} is large), then the projections f(i) and f(j) are also close

Step 3: embed data points using graph Laplacian

Original Representation		Transformed representation
data point		projections
x_i	→	$(f_1(i), \ldots, f_d(i))$
(D-dimensional vector)		(d-dimensional vector)

- Intuition = find vector f such that, if x_i is close to x_j in the graph (i.e. W_{ij} is large), then the projections f(i) and f(j) are also close

- Find eigenvectors of graph Laplacian and corresponding eigenvalues

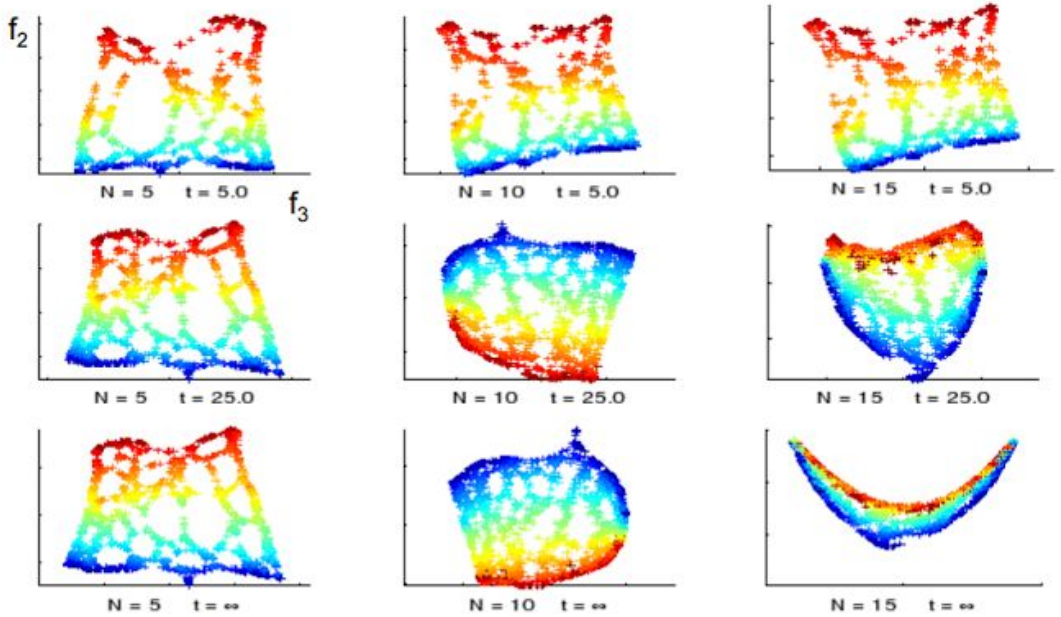
Step 3: embed data points using graph Laplacian

Original Representation		Transformed representation
data point		projections
x_i	→	$(f_1(i), \ldots, f_d(i))$
(D-dimensional vector)		(d-dimensional vector)

- Intuition = find vector f such that, if x_i is close to x_j in the graph (i.e. W_{ij} is large), then the projections f(i) and f(j) are also close

- Find eigenvectors of graph Laplacian and corresponding eigenvalues
- To embed data points in d-dimensional space, we project data into eigenvectors associated with the d smallest eigenvalues

Unrolling the swiss roll with Laplacian eigenmaps



N=number of nearest neighbors, t = the heat kernel parameter (Belkin & Niyogi'03)

Laplacian eigenmaps takeaways

- A way to do nonlinear dimensionality reduction

Laplacian eigenmaps takeaways

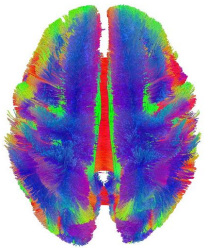
- A way to do nonlinear dimensionality reduction
- Aim to preserve local information

Laplacian eigenmaps takeaways

- A way to do nonlinear dimensionality reduction
- Aim to preserve local information
- Require 3 main steps:
  - Construct a similarity graph between data points
  - Compute the graph Laplacian
  - Use graph Laplacian to embed data points

Supervised vs unsupervised learning

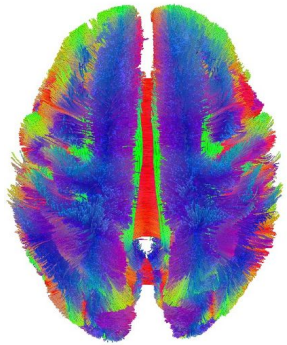
- So far we've only looked at supervised learning methods
  - Supervised methods require labels for each training data instance
    - classification
    - regression
  - Example: kNN classifier for DTI fibers assignments to anatomical bundles
- But what if we have a lot of unlabeled data and acquiring labels is expensive, or not even possible?
  - expensive labels: DTI fibers assignments to anatomical bundles
  - unknown labels: "brain states" assignments of resting state fMRI



- Unsupervised learning! We'll discuss two such methods today
  - dimensionality reduction
  - **clustering**

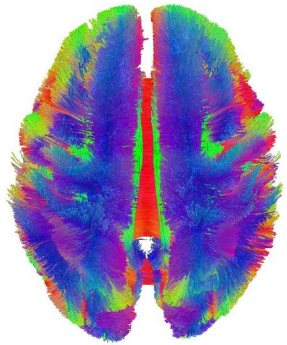
Clustering: motivation

- We talked about using kNN to classify individual DTI fibers into anatomical bundles



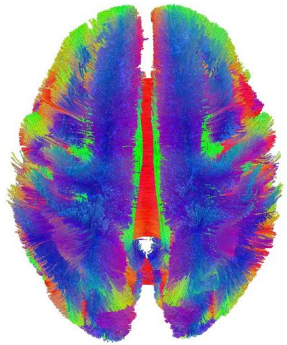
Clustering: motivation

- We talked about using kNN to classify individual DTI fibers into anatomical bundles
- However, this still requires some human effort to label the training fibers with corresponding anatomical bundles



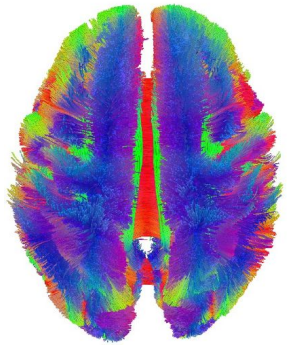
Clustering: motivation

- We talked about using kNN to classify individual DTI fibers into anatomical bundles
- However, this still requires some human effort to label the training fibers with corresponding anatomical bundles
  - error-prone
  - effortful



Clustering: motivation

- We talked about using kNN to classify individual DTI fibers into anatomical bundles
- However, this still requires some human effort to label the training fibers with corresponding anatomical bundles
  - error-prone
  - effortful
- Can we free the human?



What is clustering?

- Organizing data into groups, or clusters, such that there is:

What is clustering?

- Organizing data into groups, or clusters, such that there is:
  - High similarity within groups

What is clustering?

- Organizing data into groups, or clusters, such that there is:
  - High similarity within groups
  - Low similarity between groups

What is clustering?

- Organizing data into groups, or clusters, such that there is:
  - High similarity within groups
  - Low similarity between groups
- Unsupervised, so no labels to rely on for clustering

How to measure “similarity” between/within clusters?

- Any function that takes two data points as input and produces a real number as output

How to measure “similarity” between/within clusters?

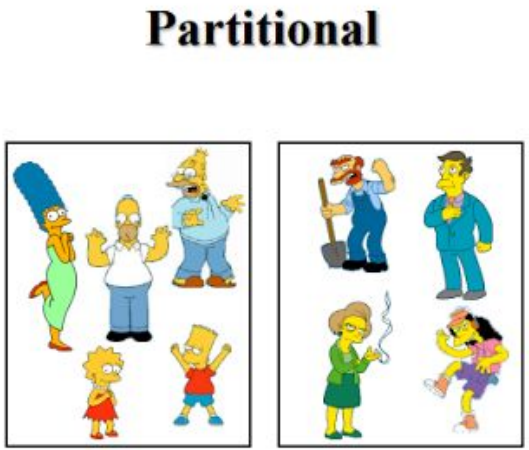
- Any function that takes two data points as input and produces a real number as output
- Examples:
  - Euclidean distance (as a measure of dissimilarity): $d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$

How to measure “similarity” between/within clusters?

- Any function that takes two data points as input and produces a real number as output
- Examples:
  - Euclidean distance (as a measure of dissimilarity): $d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$
  - Correlation coefficient: $s(x,y) = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y}$

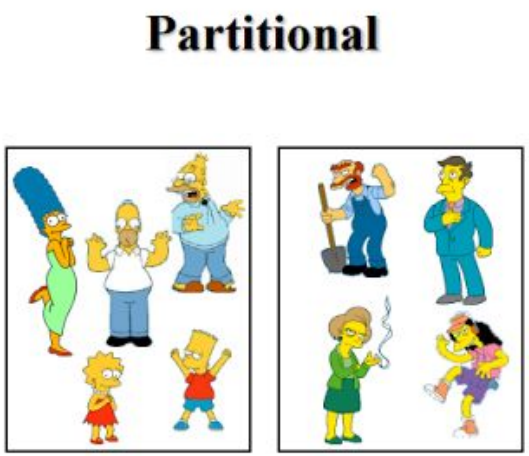
Clustering algorithms divide into 2 main types

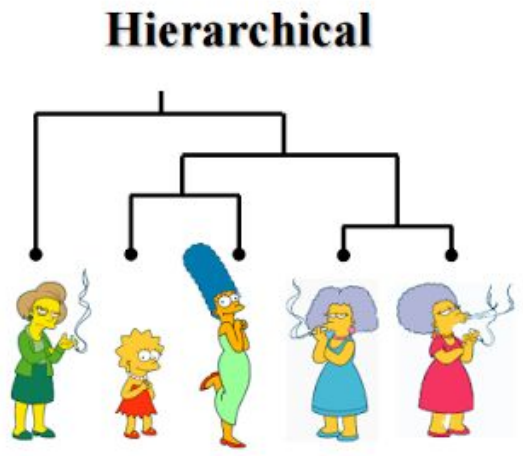
- Partitional algorithms
  - Construct various partitions and then evaluate the partitions by some criterion



Clustering algorithms divide into 2 main types

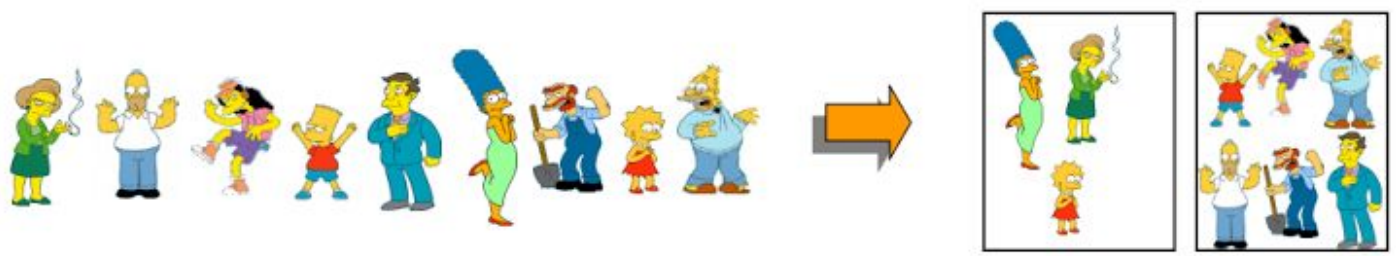
- Partitional algorithms
  - Construct various partitions and then evaluate the partitions by some criterion
- Hierarchical algorithms
  - Create a hierarchical decomposition of the set of objects using some criterion





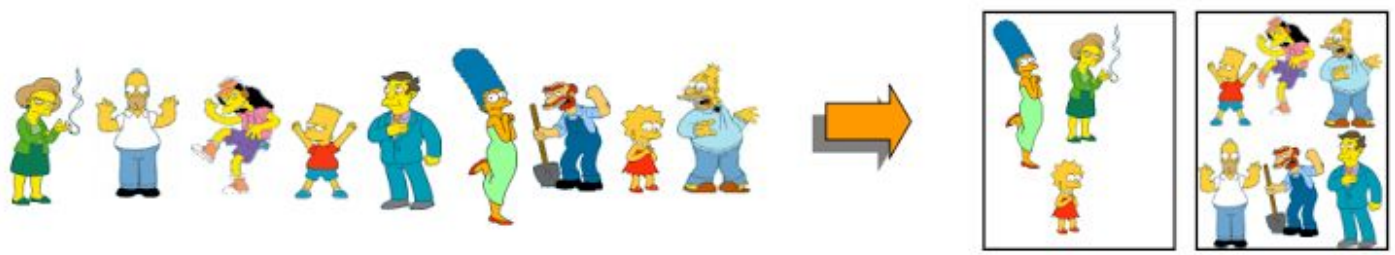
Partitional clustering

- Each data instance is placed in exactly one of K non-overlapping clusters



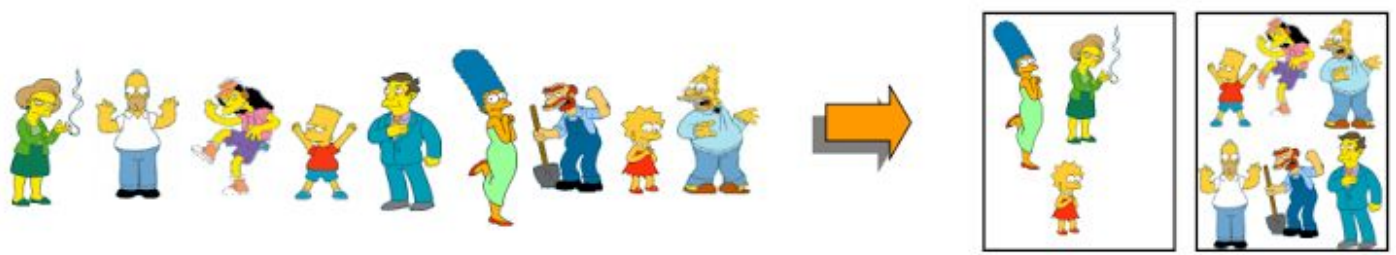
Partitional clustering

- Each data instance is placed in exactly one of K non-overlapping clusters
- The user must specify the number of clusters K



Partitional clustering

- Each data instance is placed in exactly one of K non-overlapping clusters
- The user must specify the number of clusters K
- We'll discuss 2 such algorithms:
  - k-means
  - spectral clustering



K-means algorithm: 4 easy steps!

- Input desired number of clusters k

K-means algorithm: 4 easy steps!

- ❏ Input desired number of clusters k
- ❏ Initialize the k cluster centers (randomly if necessary)

K-means algorithm: 4 easy steps!

- Input desired number of clusters k
- Initialize the k cluster centers (randomly if necessary)
- Iterate:
  - Assign all data instances to the nearest cluster center

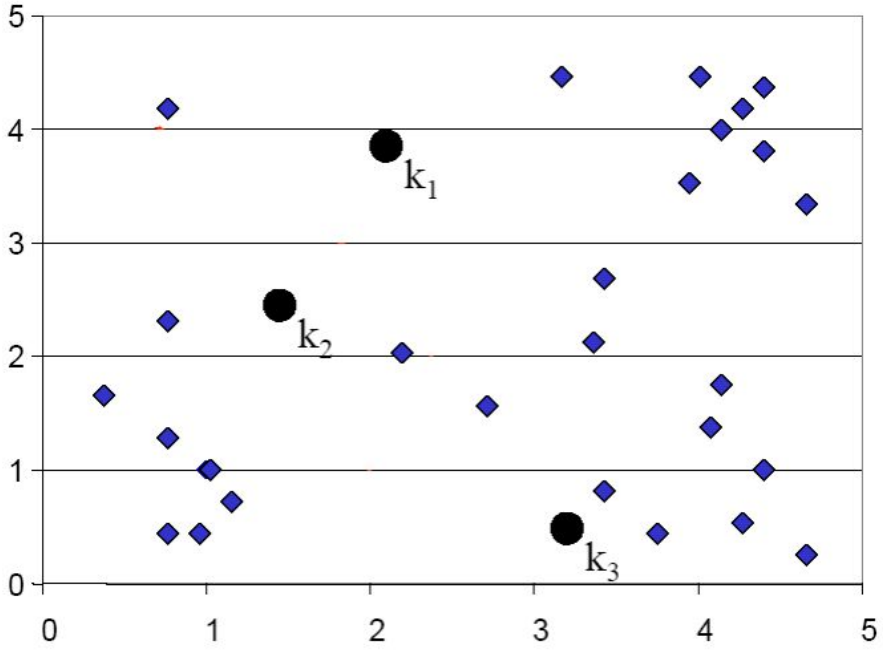
K-means algorithm: 4 easy steps!

- Input desired number of clusters k
- Initialize the k cluster centers (randomly if necessary)
- Iterate:
  - Assign all data instances to the nearest cluster center
  - Re-estimate the k cluster centers (aka cluster mean) based on current assignments

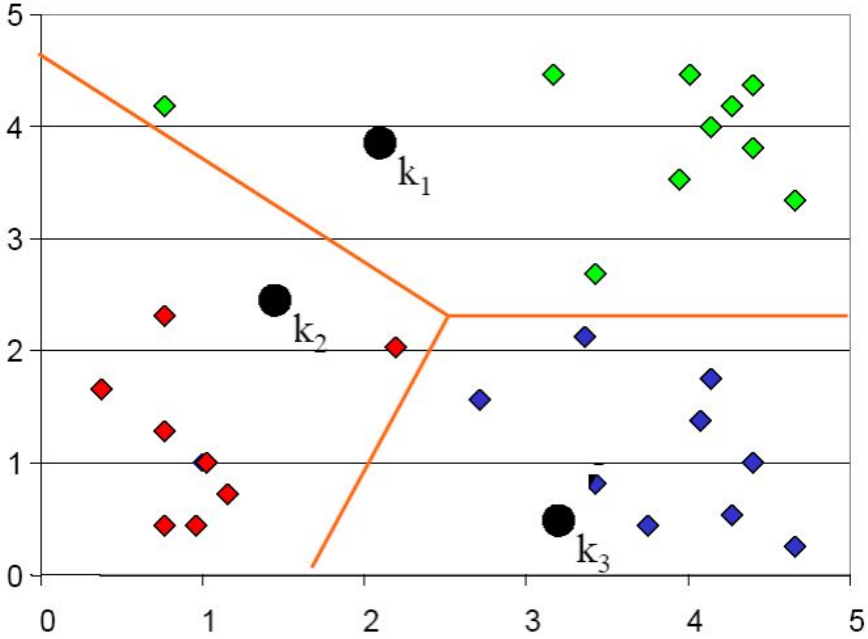
K-means algorithm: 4 easy steps!

- Input desired number of clusters k
- Initialize the k cluster centers (randomly if necessary)
- Iterate:
  - Assign all data instances to the nearest cluster center
  - Re-estimate the k cluster centers (aka cluster mean) based on current assignments
- Terminate if none of the assignments changed in the last iteration

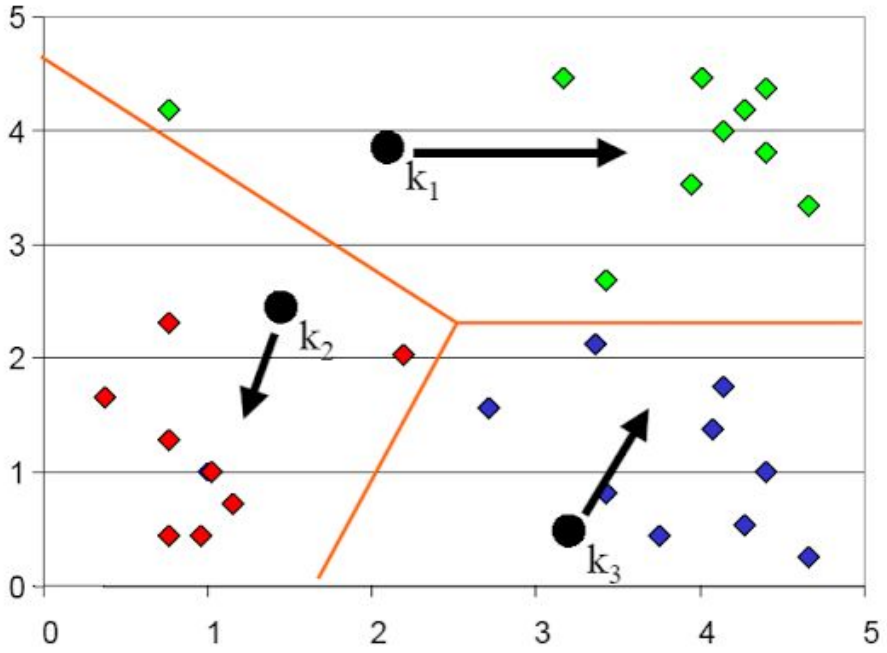
Step 1 + 2: input number of clusters k, initialize their positions



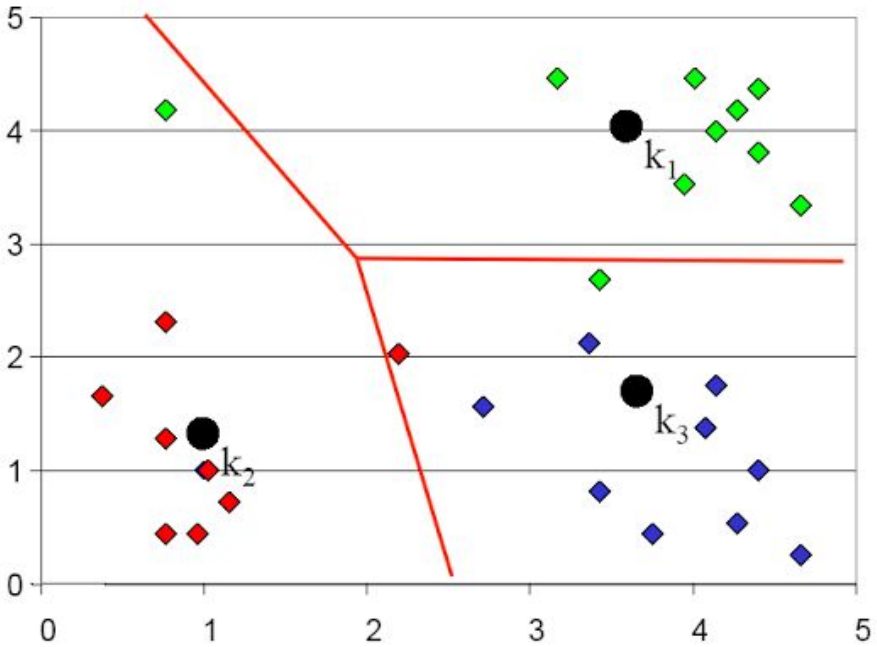
Step 3.1: assign all data to nearest cluster



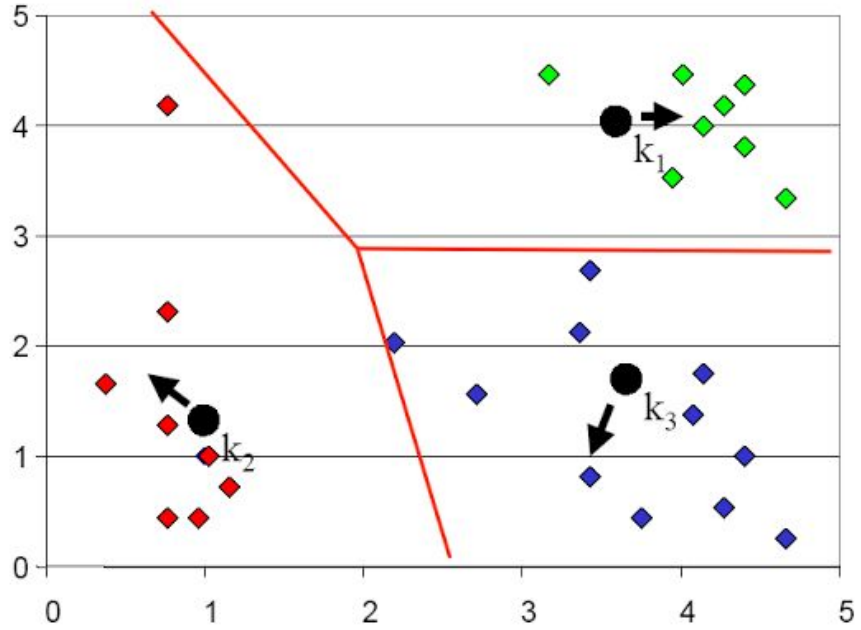
Step 3.2: re-estimate k cluster means



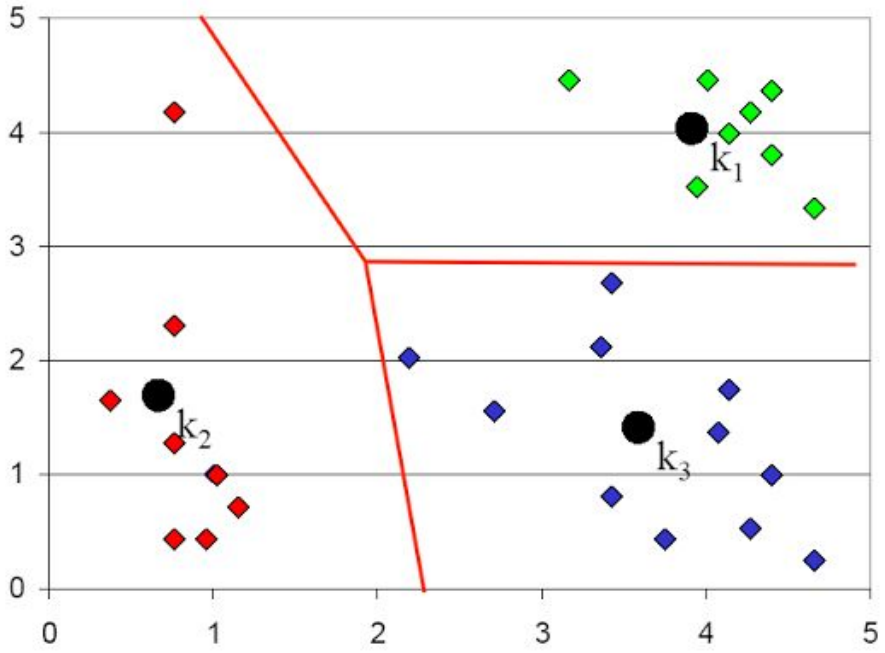
Step 3.2: re-estimate k cluster means



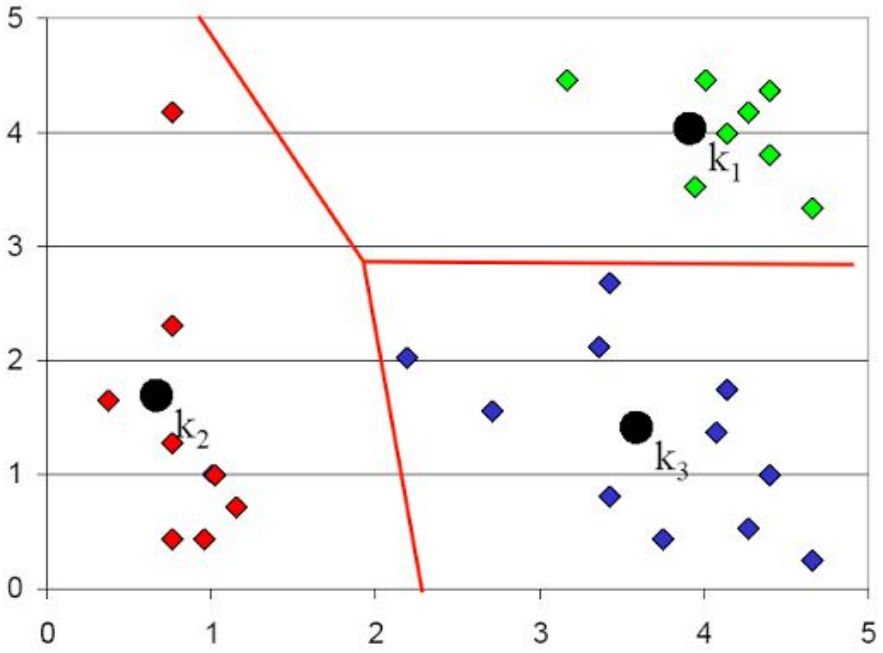
Step 3.1 again: assign all data to nearest cluster



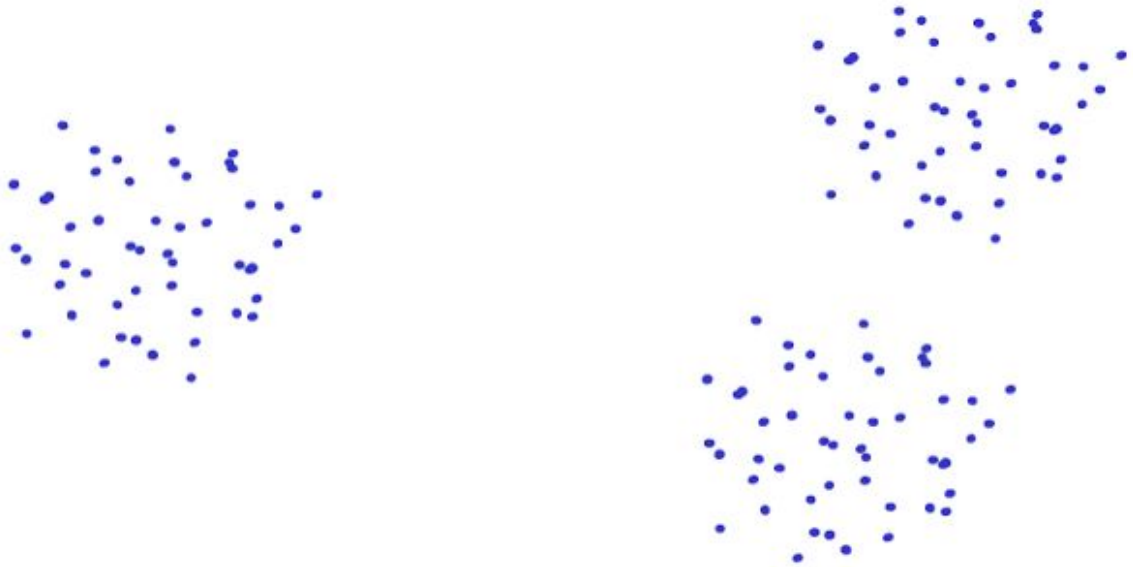
Step 3.2: re-estimate k cluster means



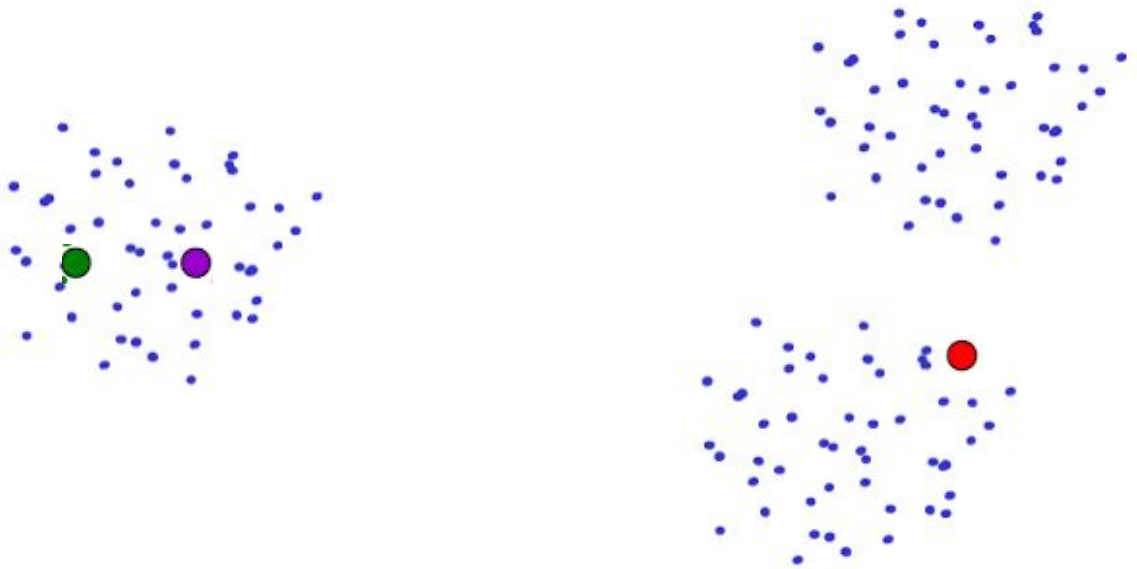
Step 4: terminate when cluster assignments don't change



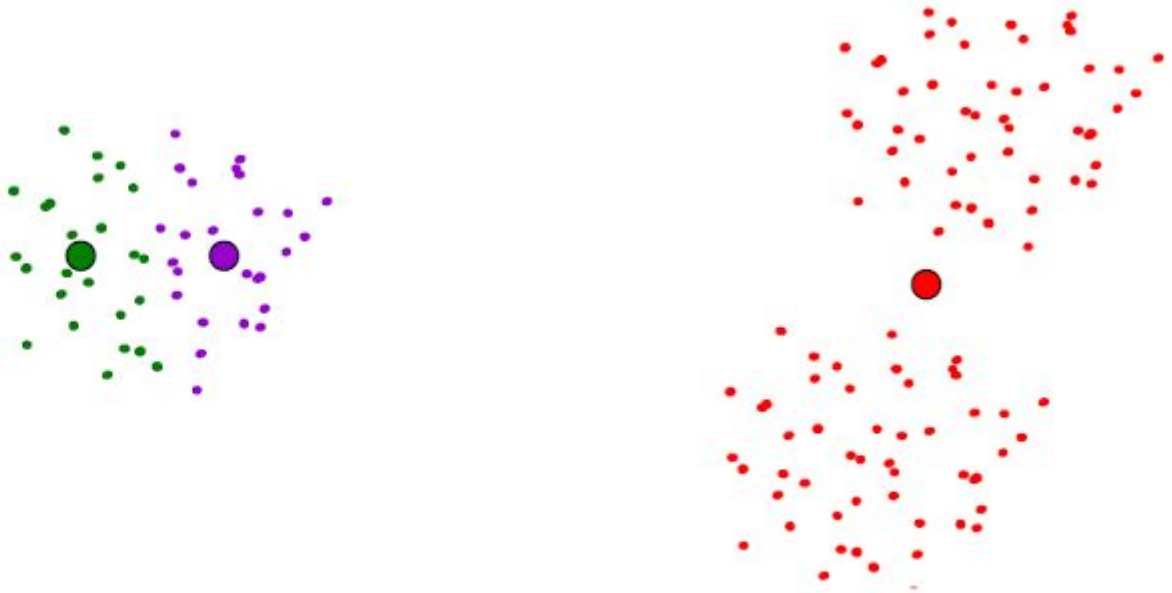
K-means problems: sensitive to cluster initialization



K-means problems: sensitive to cluster initialization



K-means problems: sensitive to cluster initialization



K-means problems: sensitive to cluster initialization

- Very important to try multiple starting points for the initialization of cluster means

K-means problems: sensitive to cluster initialization

- Very important to try multiple starting points for the initialization of cluster means
- Consider using k-means++ initialization

K-means problems: sensitive to cluster initialization

- Very important to try multiple starting points for the initialization of cluster means
- Consider using k-means++ initialization
  - initializes cluster means to be generally distant from each other

K-means problems: sensitive to cluster initialization

- Very important to try multiple starting points for the initialization of cluster means
- Consider using k-means++ initialization
  - initializes cluster means to be generally distant from each other
  - provably better results than random initialization

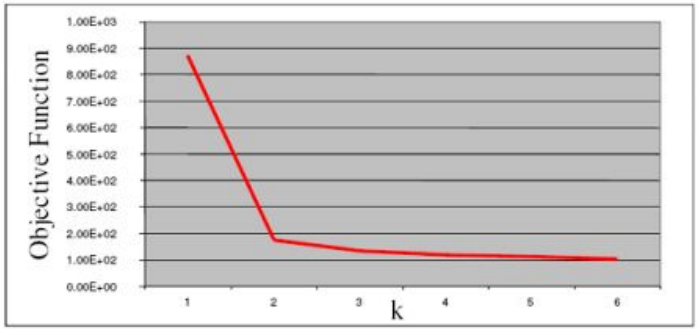
K-means problems: choosing k

- Objective function: $\sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$

K-means problems: choosing k

- Objective function: $\sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$

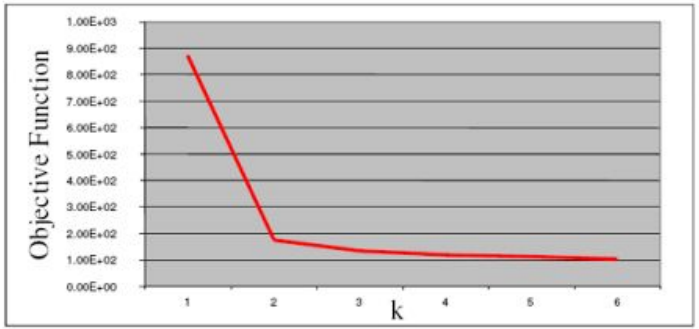
- In practice, look for “knee”/”elbow” in objective function:



K-means problems: choosing k

- Objective function: $\sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$

- In practice, look for “knee”/”elbow” in objective function:

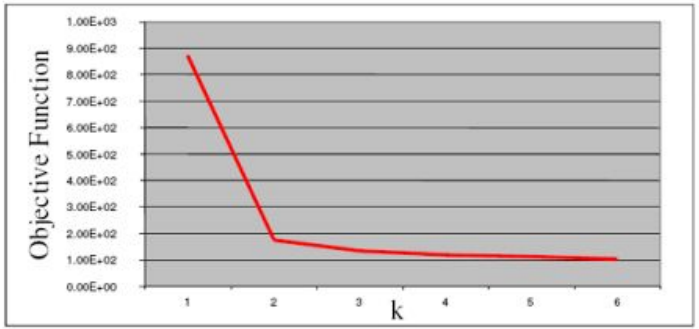


- Can we choose k by minimizing the objective over k?

K-means problems: choosing k

- Objective function: $\sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$

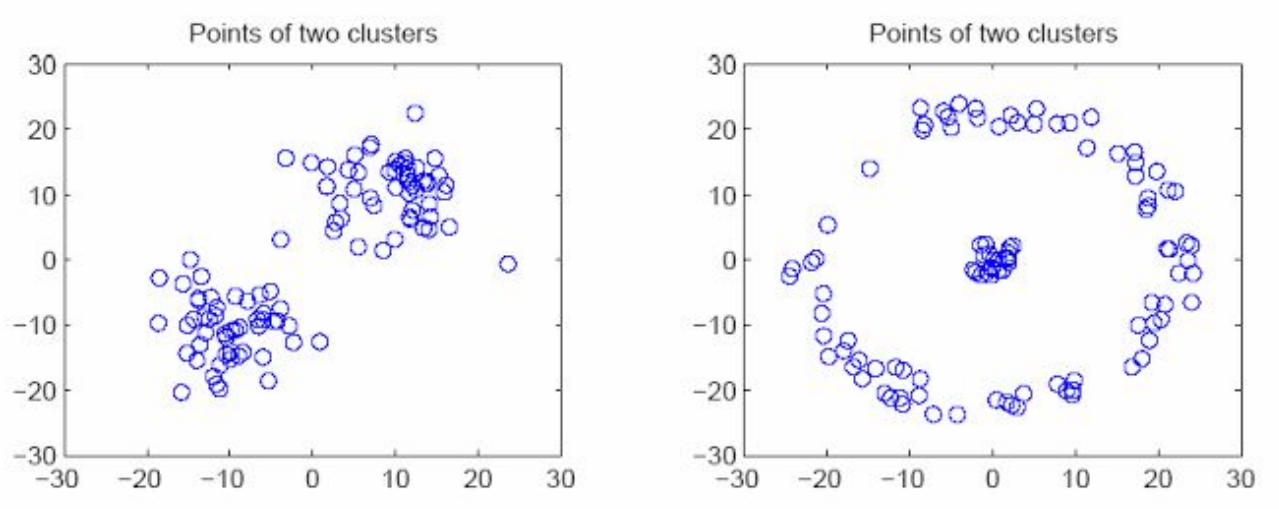
- In practice, look for “knee”/”elbow” in objective function:



- Can we choose k by minimizing the objective over k? No! The objective will go to 0 as the number of clusters approaches the number of centers!

K-means problems: shape of clusters

- Assumes convex clusters



K-means takeaways

- A simple, iterative way to do clustering

K-means takeaways

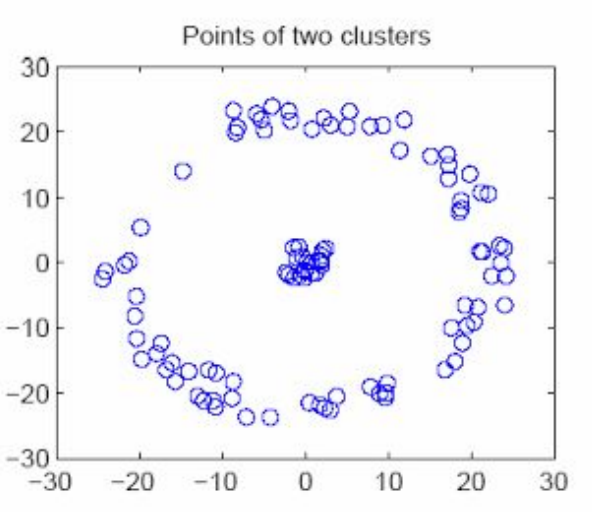
- A simple, iterative way to do clustering
- Sensitive to initialization of cluster means

K-means takeaways

- A simple, iterative way to do clustering
- Sensitive to initialization of cluster means
- Assumes convexity of clusters

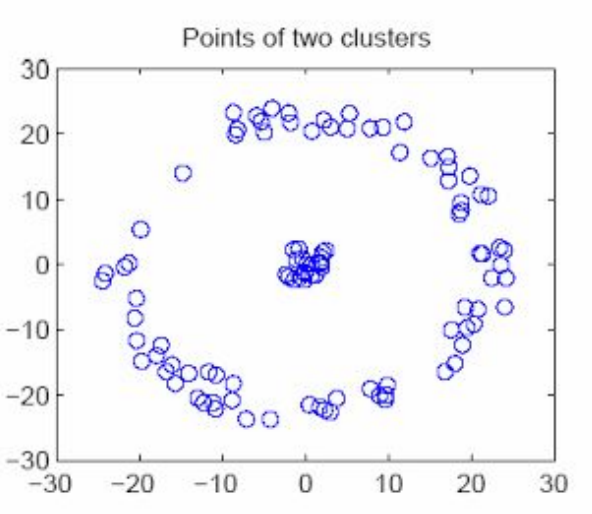
What if clusters aren’t convex?

- We can first do laplacian eigenmaps to reduce dimensionality



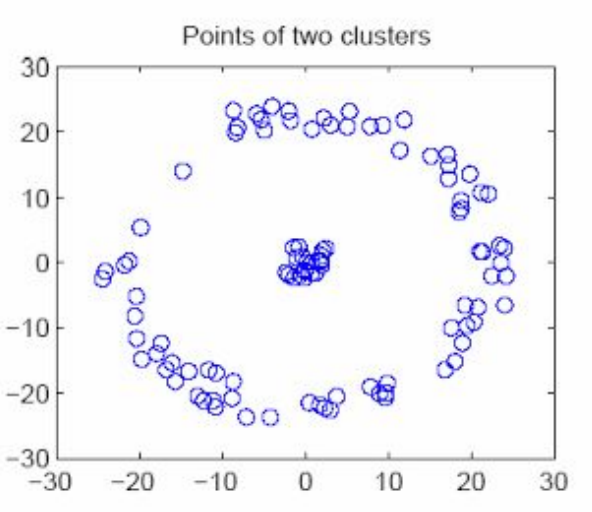
What if clusters aren’t convex?

- We can first do laplacian eigenmaps to reduce dimensionality
- Then we can do k-means on the embedded points in lower-dimension



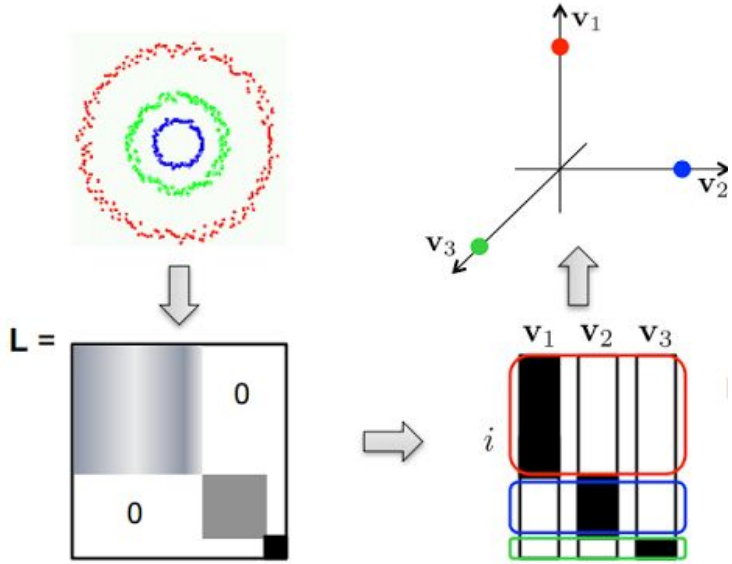
What if clusters aren’t convex?

- ❏ We can first do laplacian eigenmaps to reduce dimensionality
- ❏ Then we can do k-means on the embedded points in lower-dimension
- ❏ This is called spectral clustering



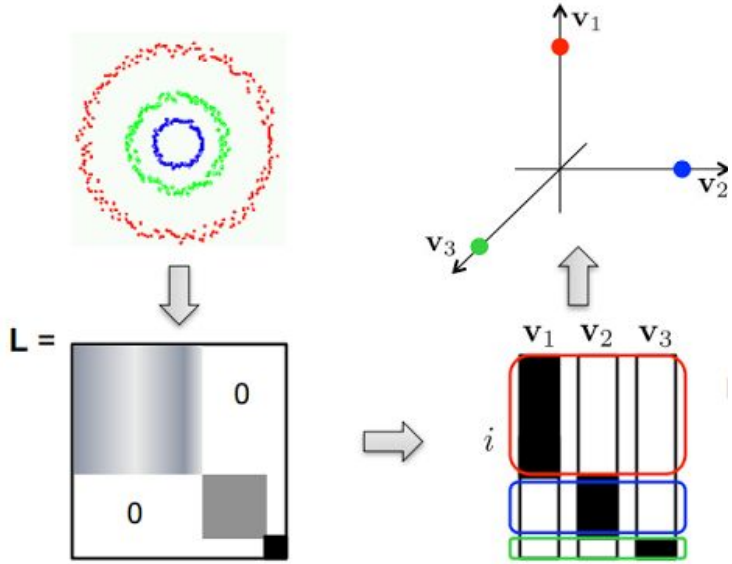
Spectral clustering: intuition

- Laplacian eigenmaps constructs a graph. If there are separable clusters, the corresponding graph should have disconnected subgraphs



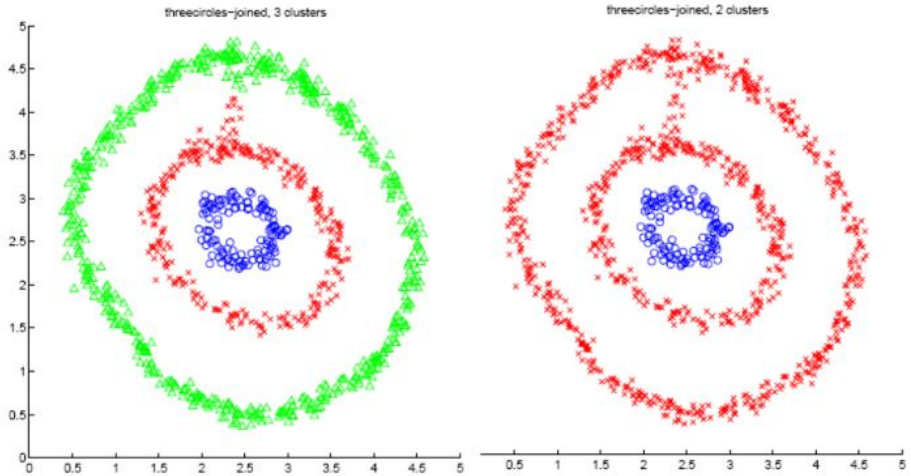
Spectral clustering: intuition

- Laplacian eigenmaps constructs a graph. If there are separable clusters, the corresponding graph should have disconnected subgraphs
- Points are easy to cluster in the embedded space using k-means



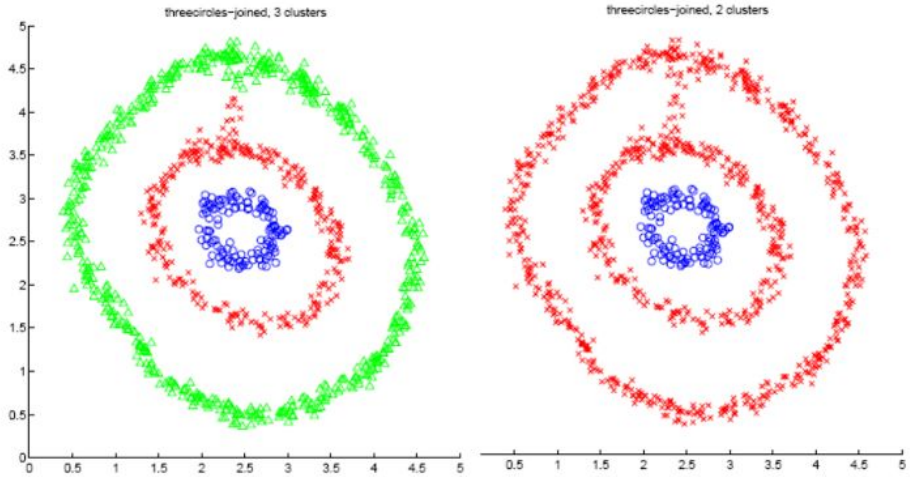
Spectral clustering problems

- Same problems as laplacian eigenmaps (choice of k for k-NN, or ε)



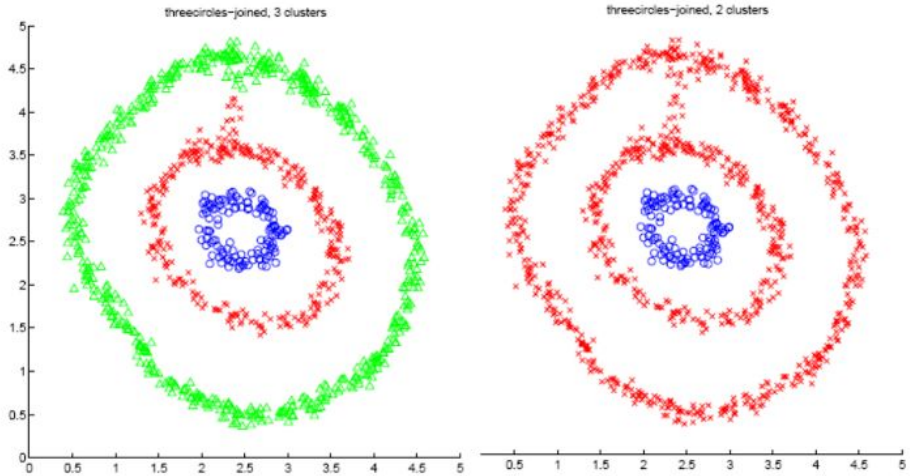
Spectral clustering problems

- ❏ Same problems as laplacian eigenmaps (choice of k for k-NN, or ε)
- ❏ Also need a way to choose number of clusters k

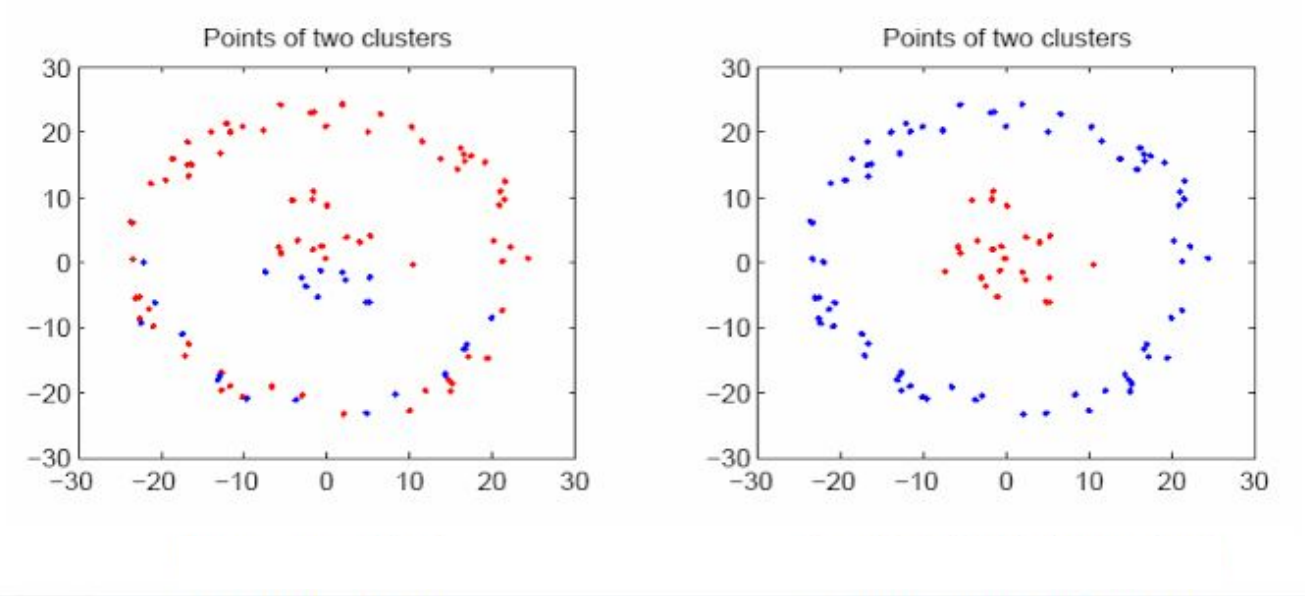


Spectral clustering problems

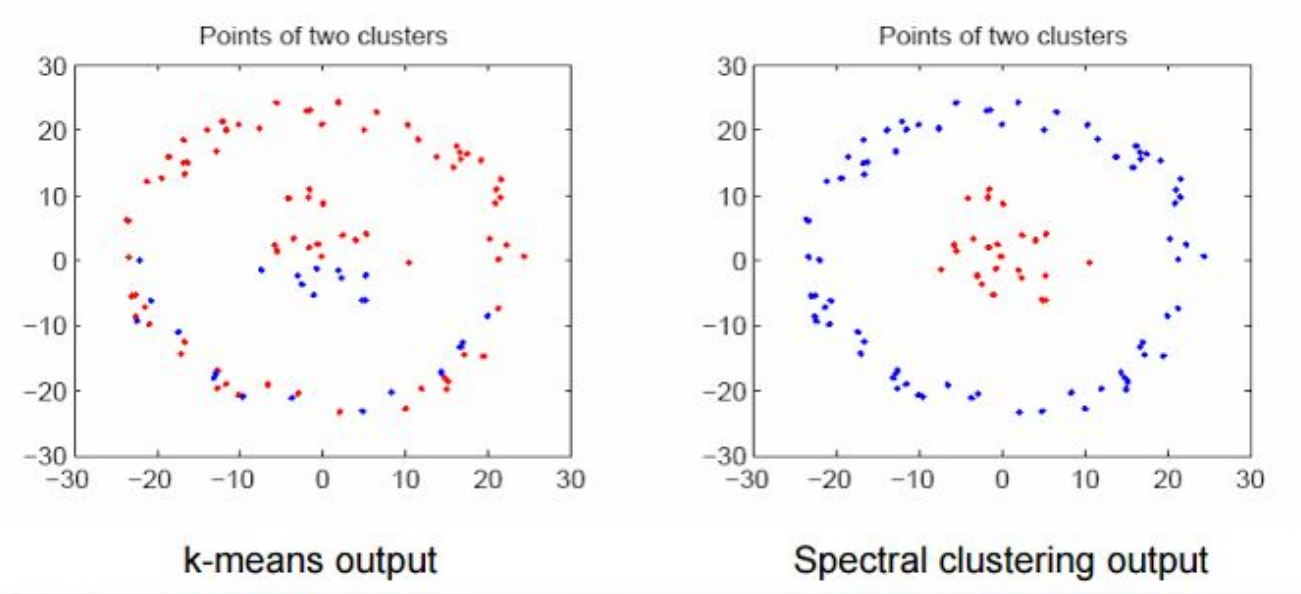
- Same problems as laplacian eigenmaps (choice of k for k-NN, or ε)
- Also need a way to choose number of clusters k
  - Most stable clustering is usually given by the value of k that maximizes the eigengap between consecutive eigenvalues



Spectral clustering vs k-means



Spectral clustering vs k-means



Spectral clustering takeaways

- Another type of partitional clustering

Spectral clustering takeaways

- ❏ Another type of partitional clustering
- ❏ Can deal with non-convex clusters

Spectral clustering takeaways

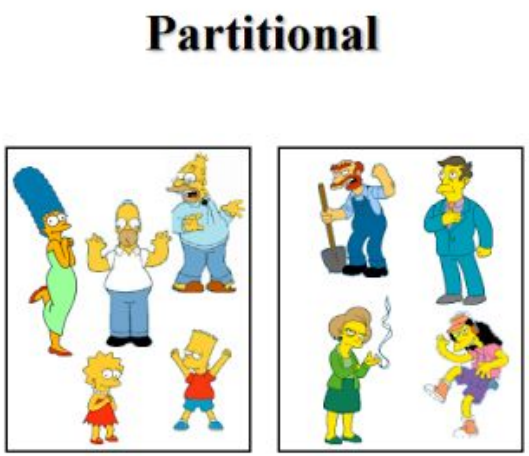
- Another type of partitional clustering
- Can deal with non-convex clusters
- First performs laplacian eigenmaps, and then clusters the embedded points using k-means

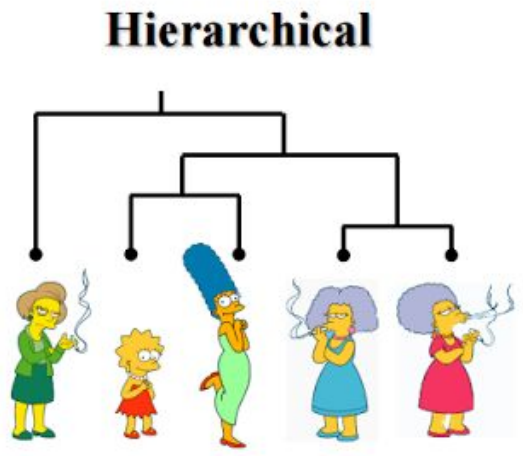
Spectral clustering takeaways

- Another type of partitional clustering
- Can deal with non-convex clusters
- First performs laplacian eigenmaps, and then clusters the embedded points using k-means
- Still need to choose the number of clusters k

Clustering algorithms divide into 2 main types

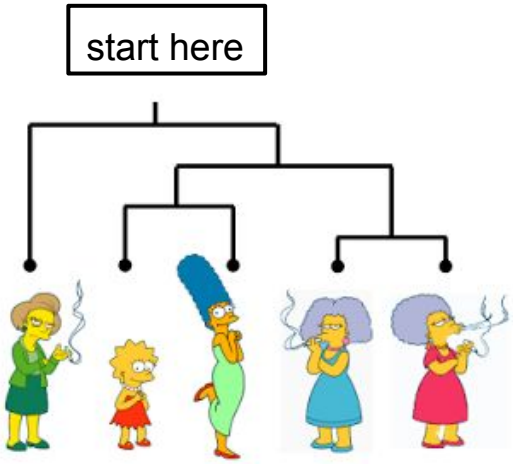
- Partitional algorithms
  - Construct various partitions and then evaluate the partitions by some criterion
- **Hierarchical algorithms**
  - **Create a hierarchical decomposition of the set of objects using some criterion**





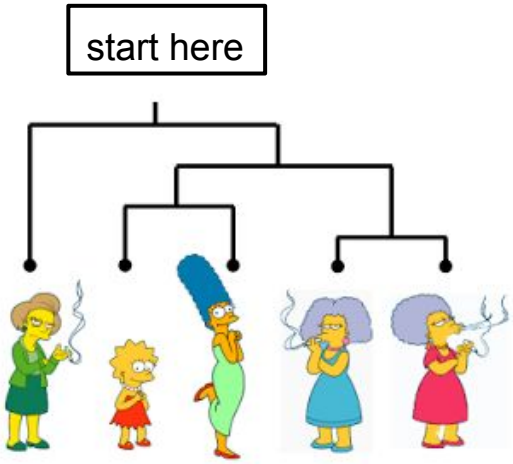
Hierarchical clustering: 2 types

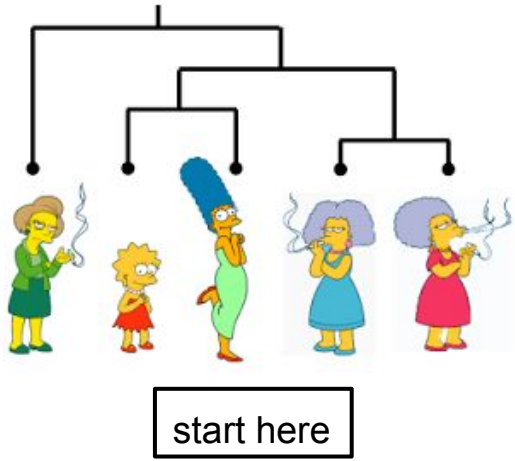
- Divisive (top-down)



Hierarchical clustering: 2 types

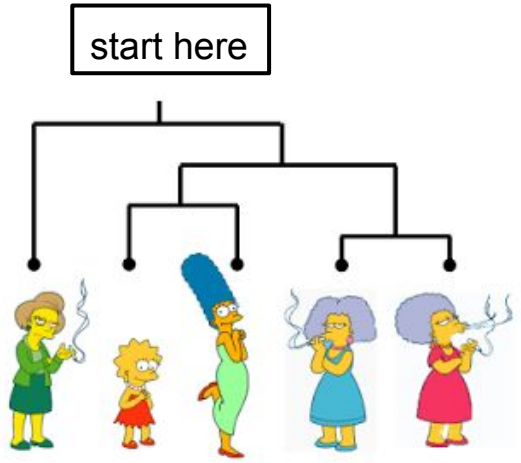
- Divisive (top-down)
- Agglomerative (bottom-up)





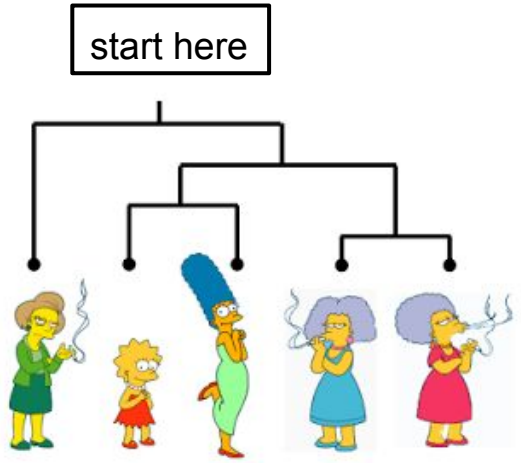
Divisive hierarchical clustering (top-down)

- Step 1: start with all data in one cluster



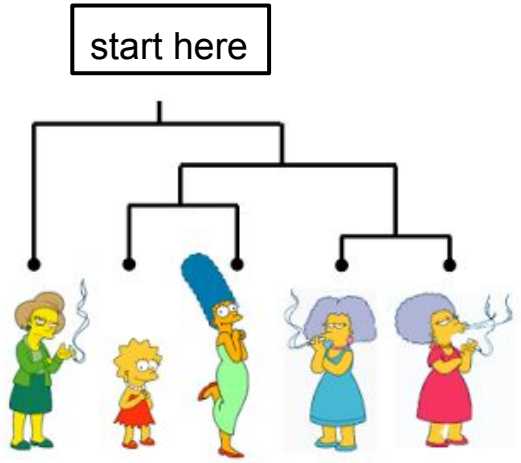
Divisive hierarchical clustering (top-down)

- Step 1: start with all data in one cluster
- Step 2: split cluster using a partitional clustering method



Divisive hierarchical clustering (top-down)

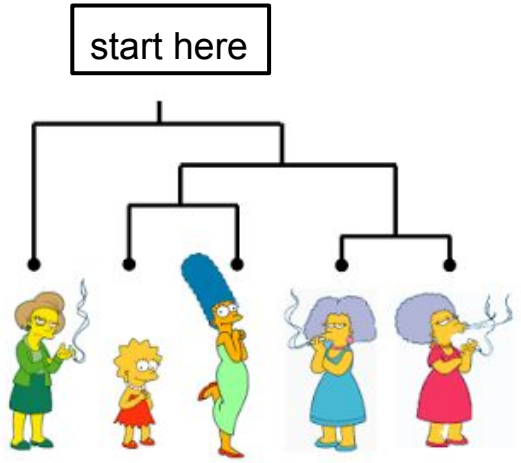
- Step 1: start with all data in one cluster
- Step 2: split cluster using a partitional clustering method
- Step 3: apply step 2 to every individual cluster until every data instance is in its own cluster



Divisive hierarchical clustering (top-down)

- Step 1: start with all data in one cluster
- Step 2: split cluster using a partitional clustering method
- Step 3: apply step 2 to every individual cluster until every data instance is in its own cluster

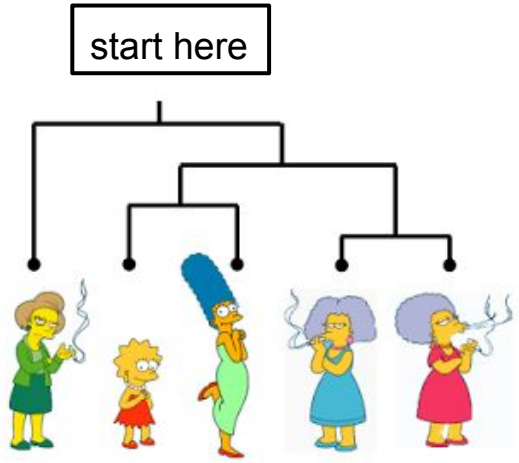
- Benefits from global information



Divisive hierarchical clustering (top-down)

- Step 1: start with all data in one cluster
- Step 2: split cluster using a partitional clustering method
- Step 3: apply step 2 to every individual cluster until every data instance is in its own cluster

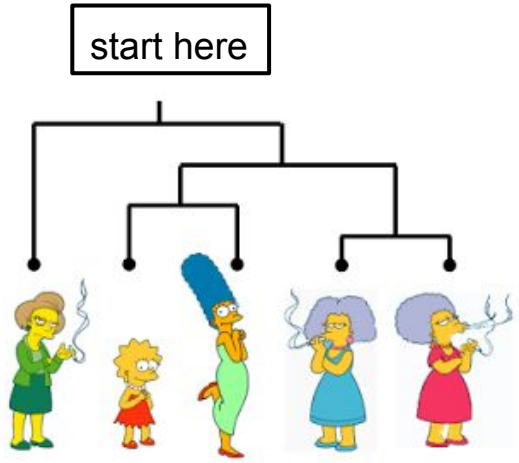
- Benefits from global information
- Can be efficient if:



Divisive hierarchical clustering (top-down)

- Step 1: start with all data in one cluster
- Step 2: split cluster using a partitional clustering method
- Step 3: apply step 2 to every individual cluster until every data instance is in its own cluster

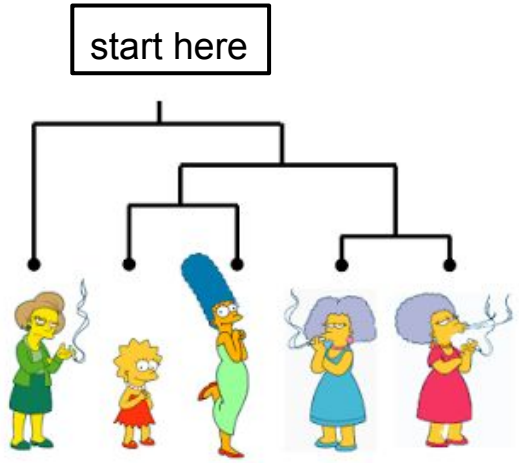
- Benefits from global information
- Can be efficient if:
  - Stopped early (not wait until all points are in own clusters)



Divisive hierarchical clustering (top-down)

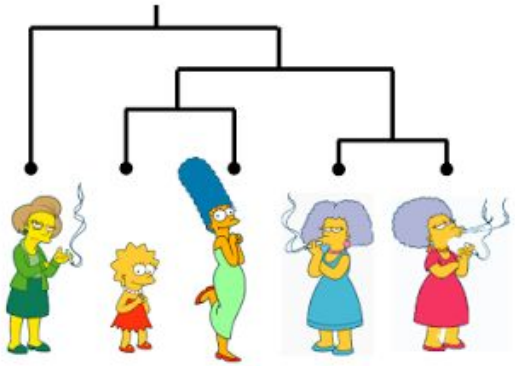
- Step 1: start with all data in one cluster
- Step 2: split cluster using a partitional clustering method
- Step 3: apply step 2 to every individual cluster until every data instance is in its own cluster

- Benefits from global information
- Can be efficient if:
  - Stopped early (not wait until all points are in own clusters)
  - Use an efficient partitional method (like k-means)



Agglomerative clustering (bottom-up)

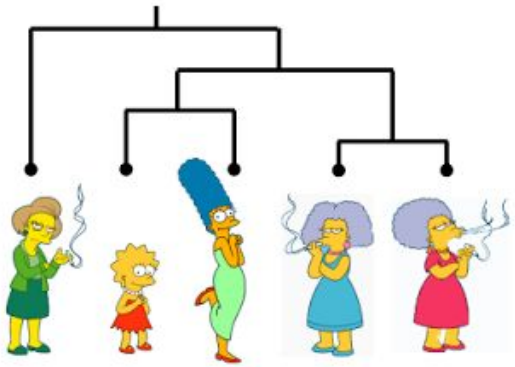
- Step 1: start with each item in own cluster



start here

Agglomerative clustering (bottom-up)

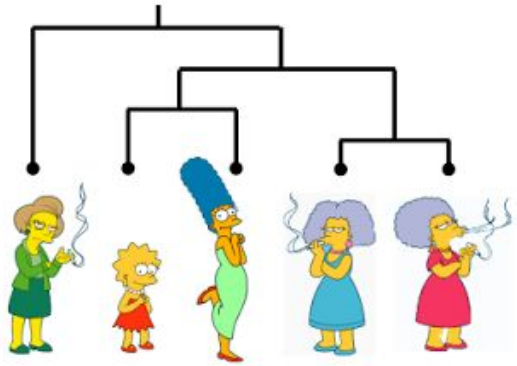
- Step 1: start with each item in own cluster
- Step 2: find best pair to merge into a new cluster



start here

Agglomerative clustering (bottom-up)

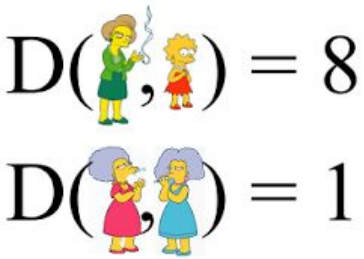
- Step 1: start with each item in own cluster
- Step 2: find best pair to merge into a new cluster
- Step 3: repeat step 2 until all clusters are fused together

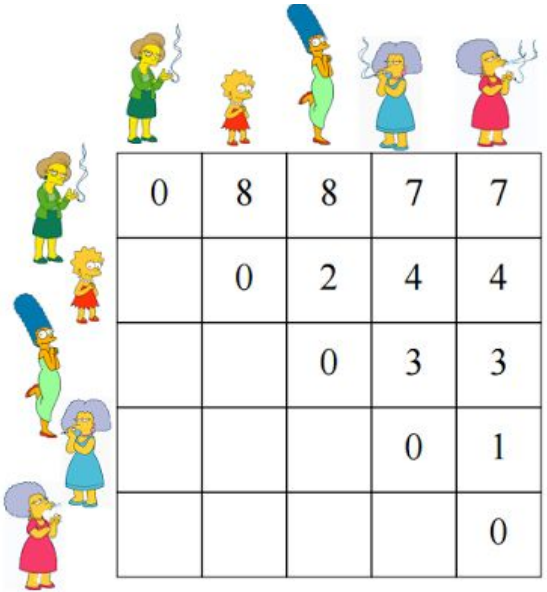


start here

But how do we find the best data points to merge?

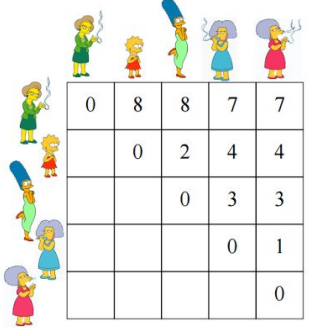
- Start with a distance matrix that contains the distances between every pair of data instances





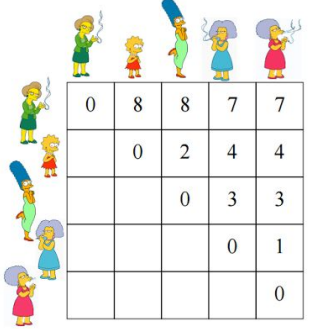
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

Step 2: find the best pair to merge in a cluster

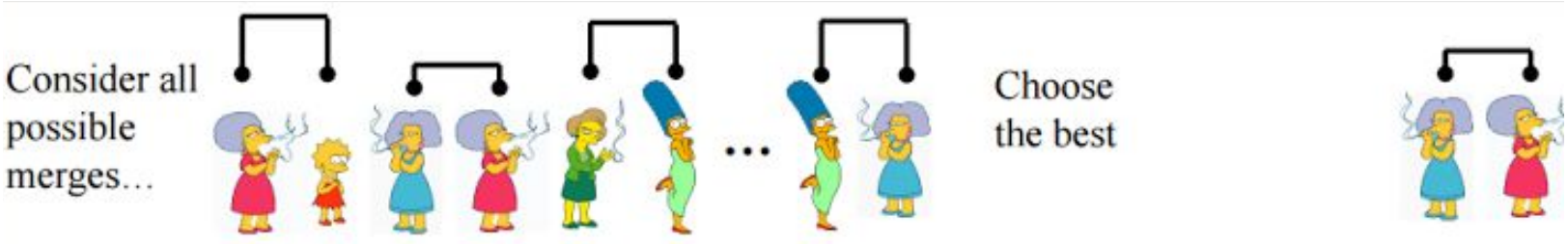


0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

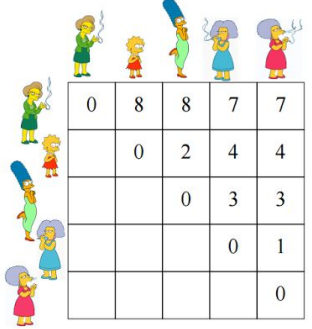
Step 2: find the best pair to merge in a cluster



0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

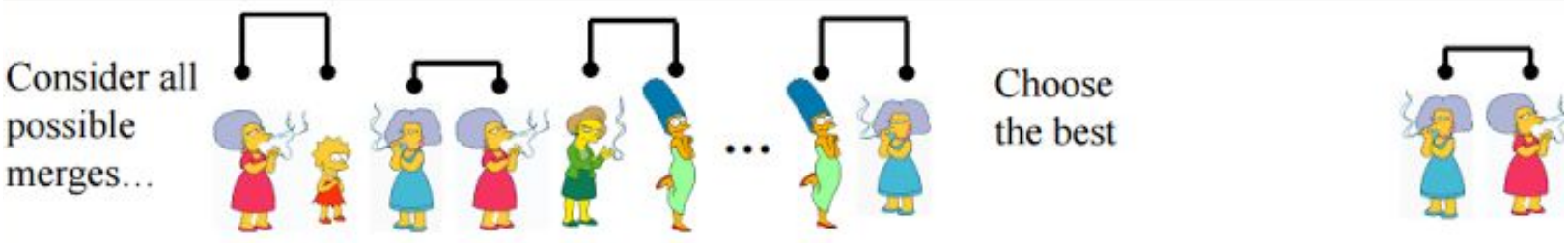


Step 2: find the best pair to merge in a cluster



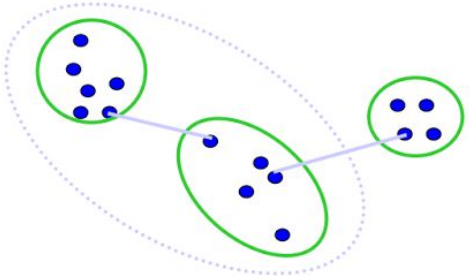
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

Now what? How do we compute distances between clusters with multiple data instances?



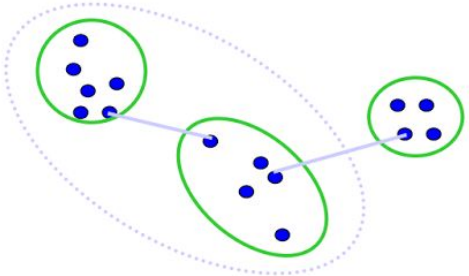
How do we compute distances between clusters?

- Distance between two closest members in each class
  - “single link”



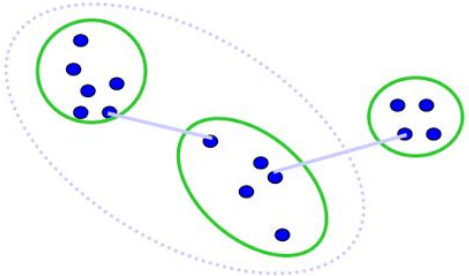
How do we compute distances between clusters?

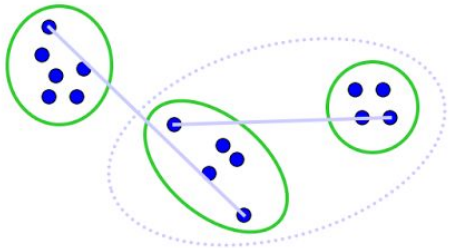
- Distance between two closest members in each class
  - “single link”
  - potentially long and skinny clusters



How do we compute distances between clusters?

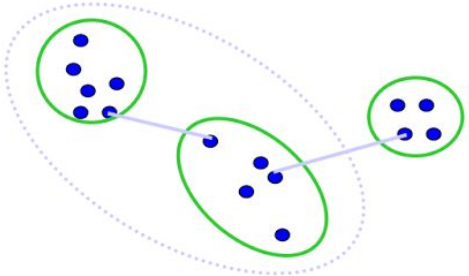
- Distance between two closest members in each class
  - “single link”
  - potentially long and skinny clusters
- Distance between two farthest members
  - “complete link”

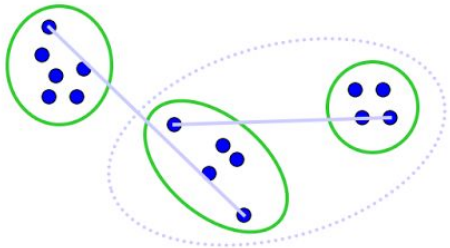




How do we compute distances between clusters?

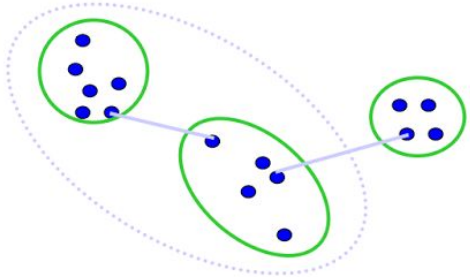
- Distance between two closest members in each class
  - “single link”
  - potentially long and skinny clusters
- Distance between two farthest members
  - “complete link”
  - tight clusters

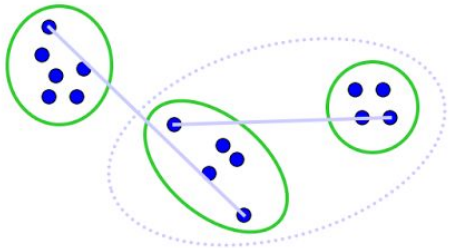


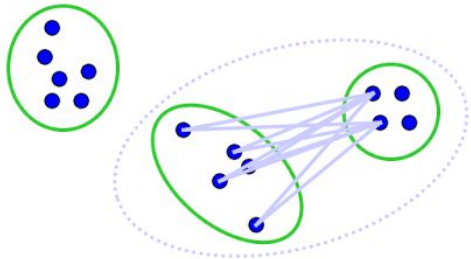


How do we compute distances between clusters?

- Distance between two closest members in each class
  - “single link”
  - potentially long and skinny clusters
- Distance between two farthest members
  - “complete link”
  - tight clusters
- Average distance of all pairs
  - “average link”

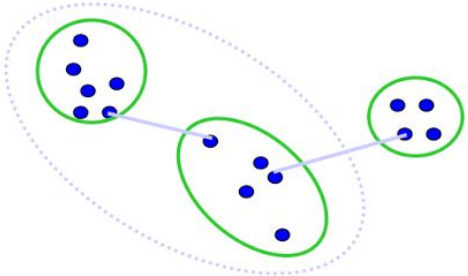


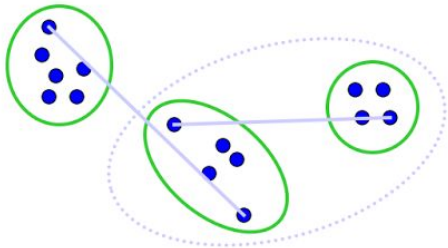


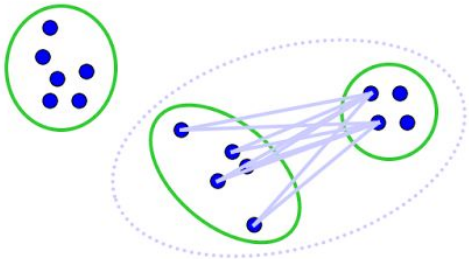


How do we compute distances between clusters?

- Distance between two closest members in each class
  - “single link”
  - potentially long and skinny clusters
- Distance between two farthest members
  - “complete link”
  - tight clusters
- Average distance of all pairs
  - “average link”
  - robust against noise

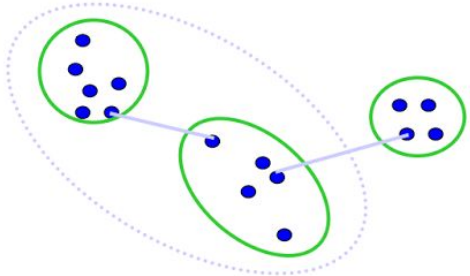


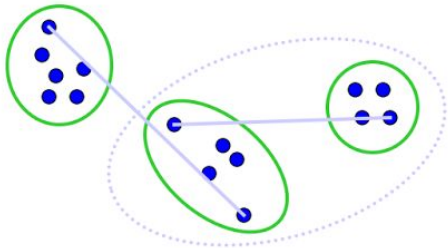


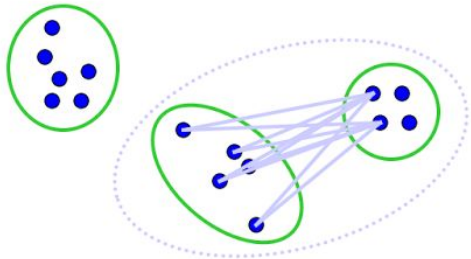


How do we compute distances between clusters?

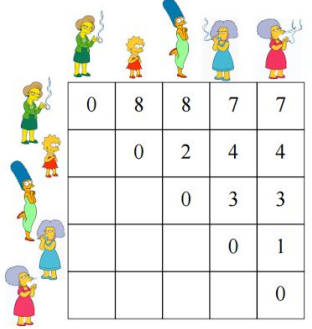
- Distance between two closest members in each class
  - “single link”
  - potentially long and skinny clusters
- Distance between two farthest members
  - “complete link”
  - tight clusters
- Average distance of all pairs
  - “average link”
  - robust against noise
  - most widely used





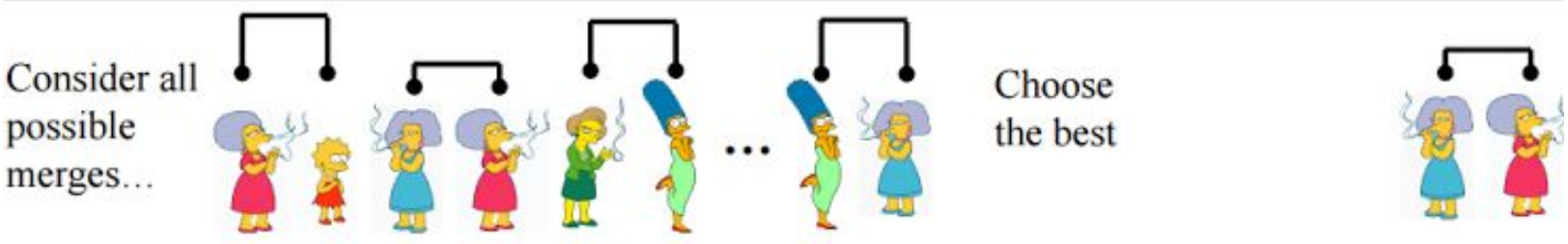


Step 2: find the best pair to merge in a cluster

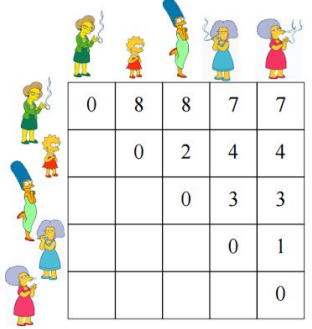


0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

Now what? How do we compute distances between clusters with multiple data instances?

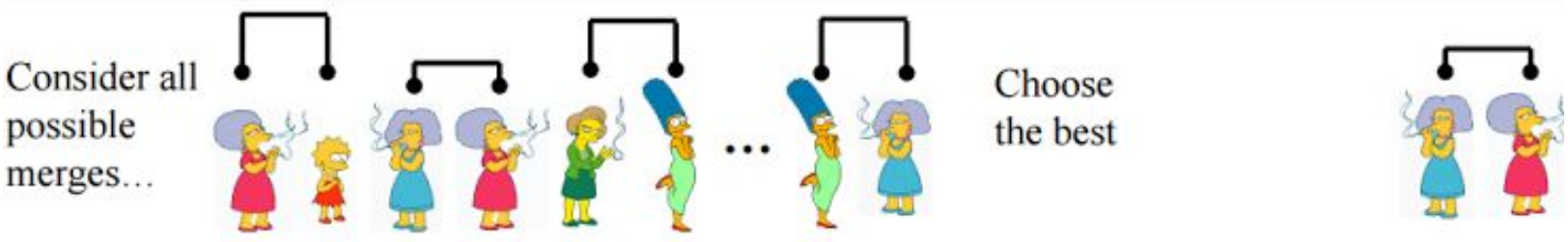


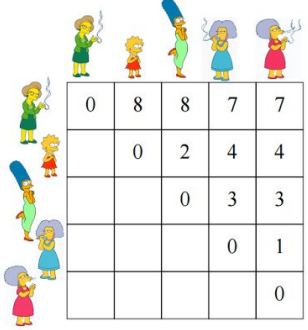
Step 2: find the best pair to merge in a cluster



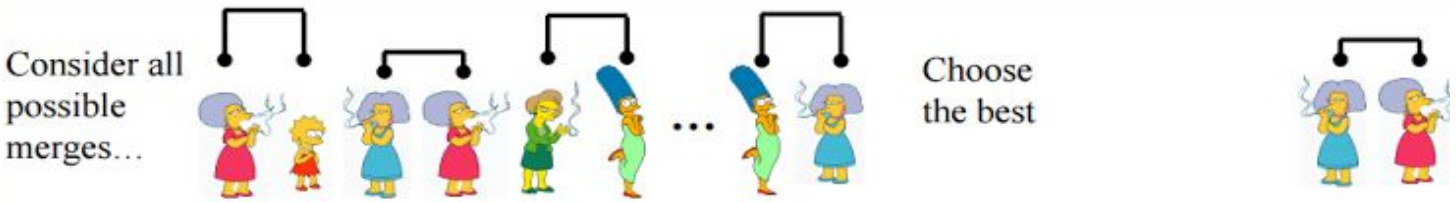
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

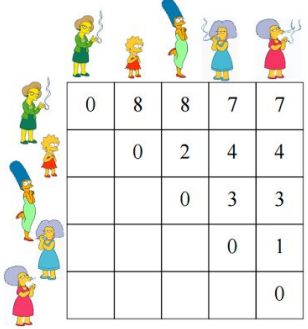
Now what? How do we compute distances between clusters with multiple data instances? => **use complete link!**



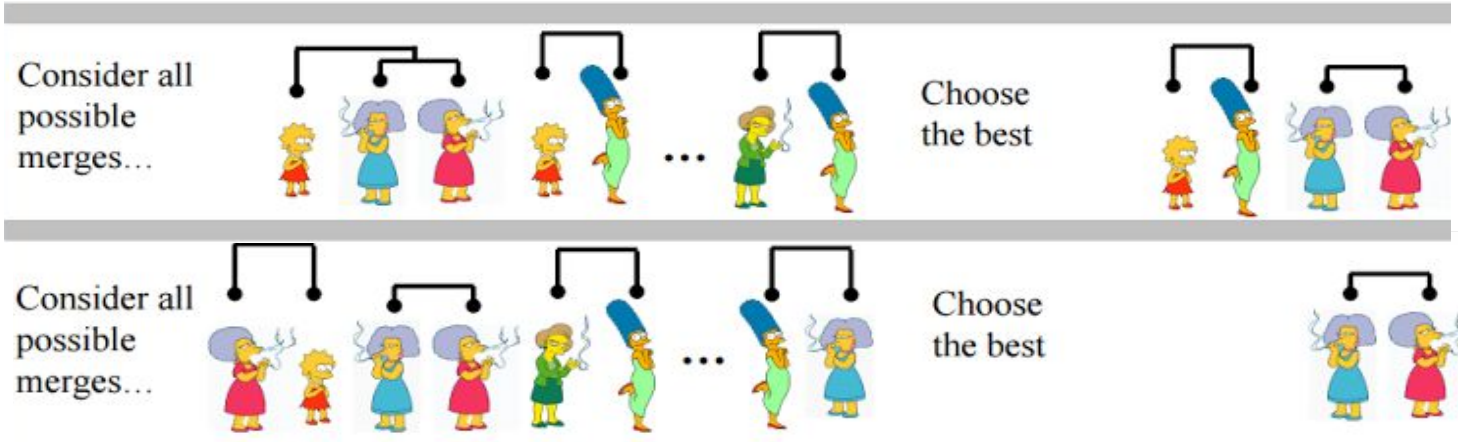


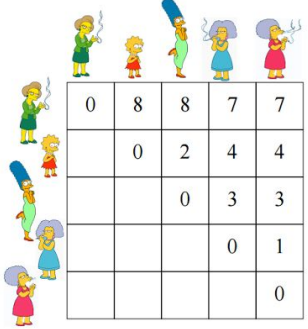
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0



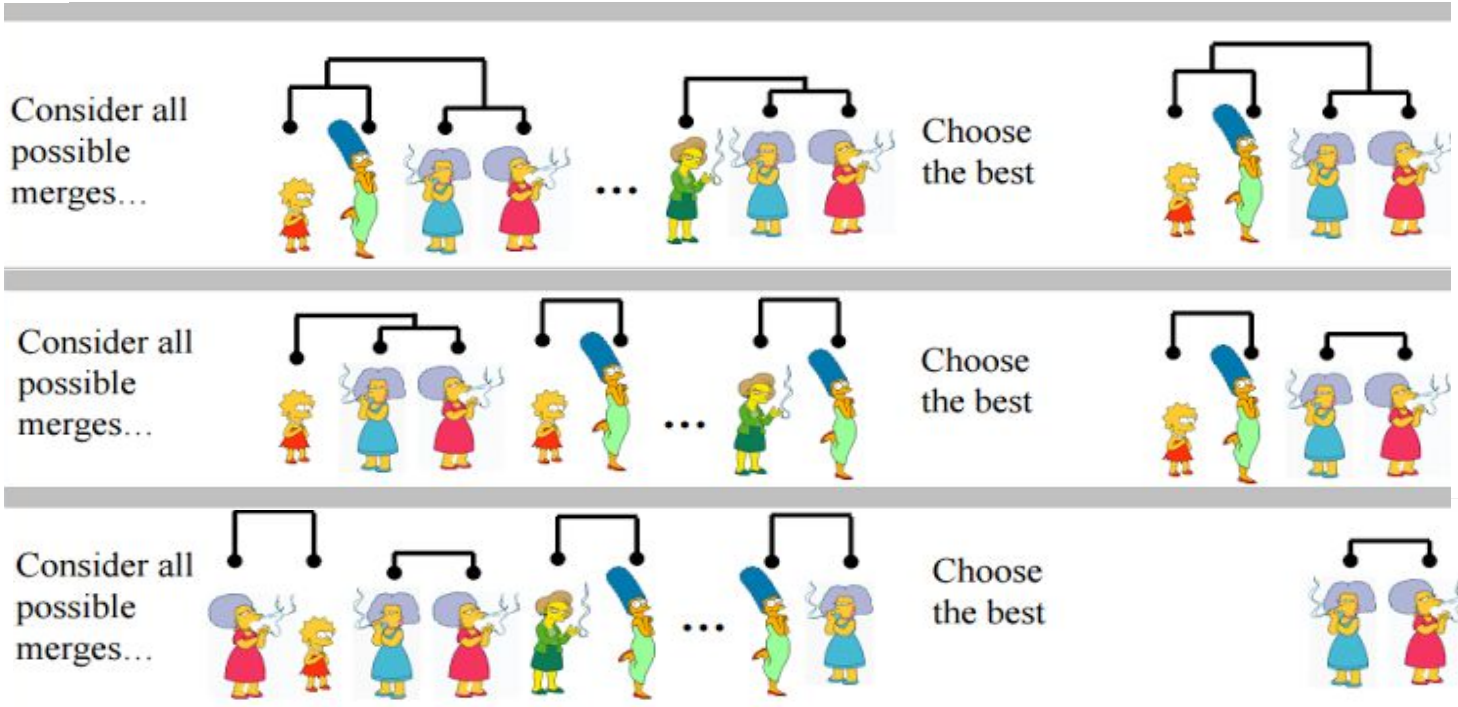


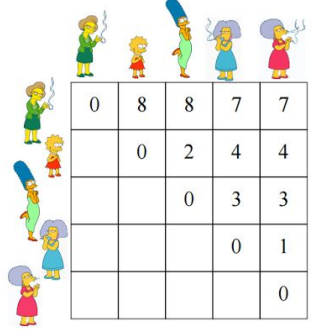
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0



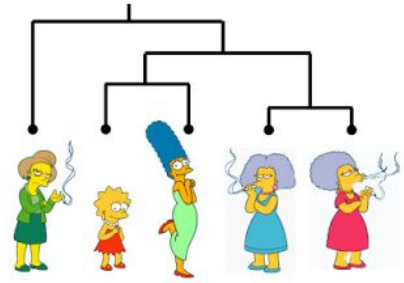


0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

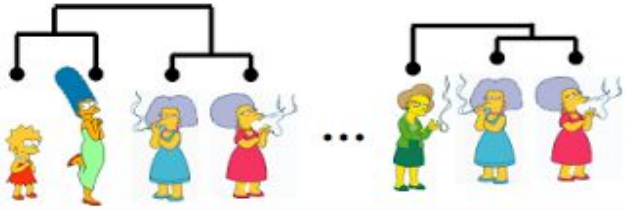




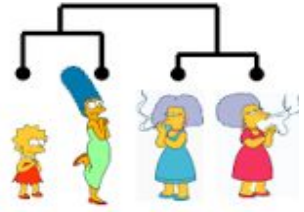
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0



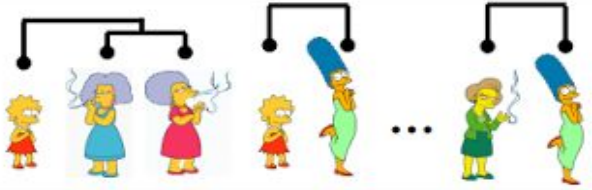
Consider all possible merges…



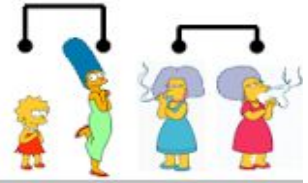
Choose the best



Consider all possible merges…

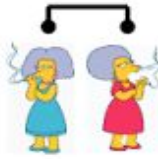


Choose the best



Consider all possible merges…

Choose the best



Hierarchical clustering takeaways

- Creates a hierarchical decomposition of groups of objects

Hierarchical clustering takeaways

- Creates a hierarchical decomposition of groups of objects
- There are two types: top-down and bottom-up

Hierarchical clustering takeaways

- Creates a hierarchical decomposition of groups of objects
- There are two types: top-down and bottom-up
- Neither is very efficient

Hierarchical clustering takeaways

- Creates a hierarchical decomposition of groups of objects
- There are two types: top-down and bottom-up
- Neither is very efficient
- But we don’t have to specify number of clusters apriori

Clustering takeaways

- Can be useful when there is a lot of unlabeled data

Clustering takeaways

- Can be useful when there is a lot of unlabeled data
- Evaluation of clustering algorithms is subjective because there is no ground truth (since there are no labels)

Clustering takeaways

- Can be useful when there is a lot of unlabeled data
- Evaluation of clustering algorithms is subjective because there is no ground truth (since there are no labels)
- It’s very important to understand the assumptions that each clustering algorithm makes when you use it

Main takeaways: dimensionality reduction and clustering

- Both are unsupervised methods that can be used when no labeled data is available

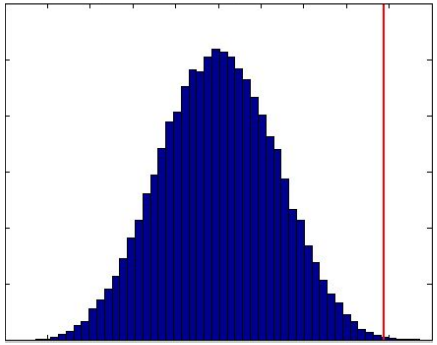
Main takeaways: dimensionality reduction and clustering

- Both are unsupervised methods that can be used when no labeled data is available
- Dimensionality reduction can uncover underlying or latent dimensions of the observed data that can better explain the variance of the data

Main takeaways: dimensionality reduction and clustering

- Both are unsupervised methods that can be used when no labeled data is available
- Dimensionality reduction can uncover underlying or latent dimensions of the observed data that can better explain the variance of the data
- Clustering can group different data instances without much domain knowledge

Next time: advanced topics!



- Evaluate results
  - cross validation (how generalizable are our results?)
  - nearly assumption-free significance testing (are the results different from chance?)
- **Complex data-driven hypotheses of brain processing**
  - **advanced topics: latent variable models, reinforcement learning, deep learning**