Recap: Classification

- Classification systems:
  - Supervised learning
  - Make a rational prediction given evidence
  - We’ve seen several methods for this
  - Useful when you have labeled data (or can get it)

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns

- What could “similar” mean?
  - One option: small (squared) Euclidean distance
    \[ \text{dist}(x, y) = (x - y)^T (x - y) = \sum_i (x_i - y_i)^2 \]
K-Means Example

K-Means as Optimization
- Consider the total distance to the means:
  \[ \phi(x_i, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_k) \]
- Each iteration reduces \( \phi \)
- Two stages each iteration:
  - Update assignments: fix means \( c \), change assignments \( a \)
  - Update means: fix assignments \( a \), change means \( c \)

Phase I: Update Assignments
- For each point, re-assign to closest mean:
  \[ a_i = \arg \min_k \text{dist}(x_i, c_k) \]
- Can only decrease total distance \( \phi \):
  \[ \phi(x_i, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, a_i, c_i) \]

Phase II: Update Means
- Move each mean to the average of its assigned points:
  \[ c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i : a_i = k} x_i \]
- Also can only decrease total distance!
- Why?
- Fun fact: the point \( y \) with minimum squared Euclidean distance to a set of points \( \{x\} \) is their mean

Initialization
- K-means is non-deterministic
- Requires initial means
- It does matter what you pick!
- What can go wrong?
- Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics

K-Means Getting Stuck
- A local optimum:
K-Means Questions

- Will K-means converge?
  - To a global optimum?

- Will it always find the true patterns in the data?
  - If the patterns are very very clear?

- Will it find something interesting?

- Do people ever use it?

- How many clusters to pick?

Clustering for Segmentation

- Quick taste of a simple vision algorithm

- Idea: break images into manageable regions for visual processing (object recognition, activity detection, etc.)

Representing Pixels

- Basic representation of pixels:
  - 3 dimensional color vector \( <r, g, b> \)
  - Ranges: \( r, g, b \in [0, 1] \)
  - What will happen if we cluster the pixels in an image using this representation?

- Improved representation for segmentation:
  - 5 dimensional vector \( <r, g, b, x, y> \)
  - Ranges: \( x \in [0, M] \), \( y \in [0, N] \)
  - \( M, N \) makes position more important
  - How does this change the similarities?

- Note: real vision systems use more sophisticated encodings which can capture intensity, texture, shape, and so on.

K-Means Segmentation

- Results depend on initialization!
  - Why?

- Note: best systems use graph segmentation algorithms

Other Uses of K-Means

- Speech recognition: can use to quantize wave slices into a small number of types (SOTA: work with multivariate continuous features)

- Document clustering: detect similar documents on the basis of shared words (SOTA: use probabilistic models which operate on topics rather than words)

Agglomerative Clustering

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters

- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there’s only one cluster left

- Produces not one clustering, but a family of clusterings represented by a dendrogram
Agglomerative Clustering

- How should we define "closest" for clusters with multiple elements?
- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Distance between centroids (broken)
  - Ward’s method (my pick, like k-means)
- Different choices create different clustering behaviors

Back to Similarity

- K-means naturally operates in Euclidean space (why?)
- Agglomerative clustering didn’t require any mention of averaging
  - Can use any function which takes two instances and returns a similarity
  - (if your similarity function has the right properties, can adapt k-means too)
- Kinds of similarity functions:
  - Euclidian (dot product)
  - Weighted Euclidian
  - Edit distance between strings
  - Anything else?

Similarity Functions

- Similarity functions are very important in machine learning
- Topic for next class: kernels
  - Similarity functions with special properties
  - The basis for a lot of advance machine learning (e.g. SVMs)

Case-Based Reasoning

- Similarity for classification
  - Case-based reasoning
  - Predict an instance’s label using similar instances
- Nearest-neighbor classification
  - 1-NN: copy the label of the most similar data point
  - K-NN: let the k nearest neighbors vote (have to devise a weighting scheme)
- Trade-off:
  - Small k gives relevant neighbors
  - Large k gives smoother functions
  - Sound familiar?
- [DEMO]

Parametric / Non-parametric

- Parametric models:
  - Fixed set of parameters
  - More data means better settings
- Non-parametric models:
  - Complexity of the classifier increases with data
  - Better in the limit, often worse in the non-limit
- (K)NN is non-parametric

http://www.cs.cmu.edu/~zhuxj/courseproject/knn/demos/KNN.html
Collaborative Filtering

- Ever wonder how online merchants decide what products to recommend to you?
- Simplest idea: recommend the most popular items to everyone
  - Not entirely crazy! (Why)
  - Can do better if you know something about the customer (e.g. what they've bought)
- Better idea: recommend items that similar customers bought
- A popular technique: collaborative filtering
  - Define a similarity function over customers (how?)
  - Look at purchases made by people with high similarity
  - Trade-off: relevance of comparison set vs confidence in predictions
- How can this go wrong?

Next Class

- Kernel methods / SVMs
- Basis for a lot of SOTA classification tech