Chapter 9

Classification and Clustering
Classification and Clustering

- Classification/clustering are classical pattern recognition/machine learning problems

- Classification, also referred to as categorization
  - Asks “what class does this item belong to?”
  - Supervised learning task

- Clustering
  - Asks “how can I group this set of items?”
  - Unsupervised learning task

- Items can be documents, emails, queries, entities, images

- Useful for a wide variety of search engine tasks
Classification is the task of automatically applying labels to items.

Useful for many search-related tasks:
- Spam detection
- Sentiment classification
- Online advertising

Two common approaches:
- Probabilistic
- Geometric
How to Classify?

- How do humans classify items?
- For example, suppose you had to classify the healthiness of a food

  - Identify set of features indicative of health: fat, cholesterol, sugar, sodium, etc.
  - Extract features from foods
    - Read nutritional facts, chemical analysis, etc.
  - Combine evidence from the features into a hypothesis
    - Add health features together to get “healthiness factor”
  - Finally, classify the item based on the evidence
    - If “healthiness factor” is above a certain value, then deem it healthy
Ontologies

- Ontology is a labeling or *categorization scheme*

- **Examples**
  - Binary (spam, not spam)
  - Multi-valued (red, green, blue)
  - Hierarchical (news/local/sports)

- Different classification tasks require different ontologies
Naïve Bayes Classifier

- Probabilistic classifier based on Bayes’ rule:

\[
P(C|D) = \frac{P(D|C)P(C)}{P(D)} = \frac{P(D|C)P(C)}{\sum_{c \in C} P(D|C = c)P(C = c)}
\]

- \( C(D) \) is a random variable corresponding to the class (input)

- Based on the term independence assumption, the Naïve Bayes’ rule yields:

\[
P(c \mid d) = \frac{P(d \mid c)P(c)}{\sum_{c \in C} P(d \mid c)P(c)} = \frac{\prod_{i=1}^{n} P(w_i \mid c)P(c)}{\sum_{c \in C} \prod_{i=1}^{n} P(w_i \mid c)P(c)} \quad \text{(Chain rule)}
\]
Random variables are *non-deterministic*

- Can be *discrete* (finite number of outcomes) or *continuous*
- Model *uncertainty* in a variable

\[ P(X = x) \text{ means “the probability that random variable } X \text{ (i.e., its value is not fixed) takes on value } x” \]

**Examples.**

- Let \( X \) be the outcome of a coin toss, \( P(X = \text{heads}) = P(X = \text{tails}) = 0.5 \)
- \( Y = 5 - 2X \). If \( X \) is *random/deterministic*, then \( Y \) is *random/deterministic*
  - *Note:* “Deterministic” just means \( P(X = x) = 1.0! \)
Naïve Bayes Classifier

- Documents are classified according to

\[
\text{Class}(d) = \arg \max_{c \in \mathcal{C}} P(c|d)
\]

\[
= \arg \max_{c \in \mathcal{C}} \frac{P(d|c)P(c)}{\sum_{c \in \mathcal{C}} P(d|c)P(c)}
\]

- Must estimate \( P(d \mid c) \) and \( P(c) \)
  
  - \( P(c) \) is the probability of observing class \( c \)
  
  - \( P(d \mid c) \) is the probability that document \( d \) is observed given the class is known to be \( c \)
Estimating $P(c)$

- $P(c)$ is the probability of observing class $c$
- Estimated as the *proportion* of training documents in class $c$:
  \[
  P(c) = \frac{N_c}{N}
  \]
- $N_c$ is the number of *training documents* in class $c$
- $N$ is the total number of *training documents*
Estimating $P(d \mid c)$

- $P(d \mid c)$ is the probability that document $d$ is observed given the class is known to be $c$

- Estimate depends on the event space used to represent the documents

- What is an event space?
  - The set of all possible outcomes for a given random variable
  - e.g., for a coin toss random variable the event space is $S = \{ \text{heads, tails} \}$

- The probability of an event space $S$
  - A probability is assigned to each event/outcome in $S$
  - The sum of the probabilities over all the events in $S$ must equal to one
Multiple Bernoulli Event Space

- Documents are represented as *binary vectors*
  - One entry for every *word* in the vocabulary
  - Entry $i = 1$, if word $i$ occurs in the document; 0, otherwise

- Multiple Bernoulli distribution is a natural way to model distributions over binary vectors

- Same event space as used in the classical *probabilistic retrieval model*
Example.

<table>
<thead>
<tr>
<th>document id</th>
<th>cheap</th>
<th>buy</th>
<th>banking</th>
<th>dinner</th>
<th>the</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>not spam</td>
</tr>
</tbody>
</table>
Multiple-Bernoulli: Estimating $P(d | c)$

- $P(d | c)$ is computed (in the Multiple-Bernoulli model) as:

\[
P(d|c) = \prod_{w \in \mathcal{V}} P(w|c)^{\delta(w,d)} (1 - P(w|c))^{1-\delta(w,d)}
\]

where $\delta(w, d) = 1$ iff term $w$ occurs in $d$; $P(d | c) = 0$ if $\exists w \in d$ never occurred in $c$ in the training set, the “data sparseness” problem, which can be solved by the “smoothing” methods.

- Laplacian smoothed estimate:

\[
P(w|c) = \frac{df_{w,c} + 1}{N_c + 1}
\]

where $df_{w,c}$ denotes the number of documents in $c$ including term $w$.

$N_c$ is the number of documents belonged to class $c$.

- Collection smoothed estimate:

\[
P(w|c) = \frac{df_{w,c} + \mu \frac{N_w}{N}}{N_c + \mu}
\]

where $\mu$ is a tunable parameter and $N_w$ is the no. of doc. including $w$.13
Multinomial Event Space

- Documents are represented as vectors of term frequencies
  - One entry for every word in the vocabulary
  - Entry $i$ = number of times that term $i$ occurs in the document

- Multinomial distribution is a natural way to model distributions over frequency vectors

- Same event space as used in the language modeling retrieval model
Example.

<table>
<thead>
<tr>
<th>document id</th>
<th>cheap</th>
<th>buy</th>
<th>banking</th>
<th>dinner</th>
<th>the</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>not spam</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>spam</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>not spam</td>
</tr>
</tbody>
</table>
Multinomial: Estimating $P(d | c)$

- $P(d | c)$ is computed as:

$$P(d|c) = P(|d|) \left( t_{fw_1,d} t_{fw_2,d} \ldots t_{fw_{|V|},d} \right)! \prod_{w \in V} P(w|c)^{tf_{w,d}}$$

- Laplacian smoothed estimate:

$$P(w|c) = \frac{tf_{w,c} + 1}{|c| + |V|}$$

  where $|c|$ is the number of terms in the training documents of class $c$

  $|V|$ is the number of distinct terms in the training documents

- Collection smoothed estimate:

$$P(w|c) = \frac{tf_{w,c} + \mu c_{fw}}{|c| + \mu}$$

  Number of term $w$ in a training set $c$

  Number of terms in all training documents

  Tunable parameter
The Multinomial model is consistently outperform the Multiple-Bernoulli model.

Implementing both models is relatively straightforward.

Both classifiers are:
- efficient, since their statistical data can be stored in memory
- accurate in document classification
- popular and attractive choice as a general-purpose classifier
Evaluating Classifiers

- Common classification metrics
  - Accuracy (precision at rank 1)
  - Precision
  - Recall
  - F-measure
  - ROC curve analysis

- Differences from IR metrics
  - “Relevant” replaced with “classified correctly”
  - Micro-averaging more commonly used in classification
    - Computes a metric for every test instance (document) and then averages over all the instances, whereas
    - Macro-averaging computes the average of the per-class metrics
Classes of Classifiers

- Types of classifiers
  - Generative (e.g., Naïve-Bayes): assumes some underlying probability distribution on generating (writing) both documents and classes using $P(c)$ and $P(d \mid c)$
  - Non-parametric (e.g., Nearest Neighbor)

- Types of learning
  - Supervised (e.g., Naïve-Bayes, SVMs)
  - Semi-supervised (e.g., Rocchio, Relevance models)
  - Unsupervised (e.g., Clustering)
Generative Classification Models

- Generative models
  - Assumes documents and classes are drawn from joint distribution $P(d, c)$
  - Typically $P(d | c)$ decomposed to $P(d | c) P(c)$
  - Effectiveness depends on how $P(d | c)$ is modeled
  - Typically more effective when little training data exists; otherwise, discriminative (e.g., SVM) or non-parameter models (e.g., nearest neighbor) should be considered
Naïve Bayes Generative Process

Class 1

Generate class according to $P(c)$

Class 2

Generate document according to $P(d \mid c)$

Class 3
Nearest Neighbor Classification

- Non-parametric classifier: given an unseen example, find the training example that is nearest to it
Document classifiers can have a very large number of features

- Not all features are useful
- Excessive features can increase computational cost of training and testing

Feature selection methods reduce the number of features by choosing the most useful features

- which can significantly improve efficiency (in terms of storage and processing time) while not hurting the effectiveness much (in addition to eliminating noisy)
Feature Selection

- Feature selection is based on entropy/information gain
  - The law of large numbers indicates that the symbol $a_j$ will, on the average, be selected
    \[ n \times p(a_j) \]
    times in a total of $n$ selections
  - The average amount of information obtained from $n$ source outputs for each $a_j$ ($j \geq 1$) with $- \log_2 p(a_j)$ bits is
    \[ n \times p(a_1) \log_2 p(a_1)^{-1} + \ldots + n \times p(a_j) \log_2 p(a_j)^{-1} \]
    bits. Divided by $n$ obtain the average amount of information per source output symbol, which is known as uncertainty, or the entropy, $E(p)$:
    \[ E = - \sum_{i=1}^{\sigma} p_i \log_2 p_i \]
Information Gain

- Information gain is a commonly used feature selection measure based on information theory.
  - It tells how much “information” is gained (about the class labels) if we observe some feature.
  - Entropy is the expected information contained in $P(c)$.

- Rank features by information gain and then train model using the top $K$ (where $K$ is typically small).

- The information gain for a Multiple-Bernoulli Naïve Bayes classifier is computed as:

\[
IG(w) = H(C) - H(C|w) = - \sum_{c \in C} P(c) \log P(c) + \sum_{w \in \{0,1\}} P(w) \sum_{c \in C} P(c|w) \log P(c|w)
\]

- Entropy of $P(c)$
- Conditional Entropy
Example. The information gain for the term “cheap”

\[
IG(\text{cheap}) = -P(\text{spam}) \log P(\text{spam}) - P(\overline{\text{spam}}) \log P(\overline{\text{spam}}) + \\
P(\text{cheap})P(\text{spam}|\text{cheap}) \log P(\text{spam}|\text{cheap}) + \\
P(\text{cheap})P(\overline{\text{spam}}|\text{cheap}) \log P(\overline{\text{spam}}|\text{cheap}) + \\
P(\overline{\text{cheap}})P(\text{spam}|\overline{\text{cheap}}) \log P(\text{spam}|\overline{\text{cheap}}) + \\
P(\overline{\text{cheap}})P(\overline{\text{spam}}|\overline{\text{cheap}}) \log P(\overline{\text{spam}}|\overline{\text{cheap}})
\]

\[
= - \frac{3}{10} \log \frac{3}{10} - \frac{7}{10} \log \frac{7}{10} + \frac{4}{10} \cdot \frac{3}{4} \log \frac{3}{4} \\
+ \frac{4}{10} \cdot \frac{1}{4} \log \frac{1}{4} + \frac{6}{10} \cdot \frac{1}{6} \log \frac{1}{6} + \frac{6}{10} \cdot \frac{6}{6} \log \frac{6}{6}
\]

\[
= 0.2749
\]

where \(P(\overline{\text{cheap}})\) denotes \(P(\text{cheap} = 0)\)

\(P(\text{spam})\) denotes \(P(\text{not spam})\)

\(0 \log 0 = 0\)

and \(IG(\text{buy}) = 0.0008, IG(\text{banking}) = 0.043, IG(\text{dinner}) = 0.361, IG(\text{the}) = 0\)
Classification Applications

Classification is widely used to enhance search engines

Example applications

- Spam detection
- Sentiment classification
- Semantic classification of advertisements (based on the semantically-related, not topically-related scope)
- Others …
Spam, Spam, Spam

- Classification is widely used to detect various types of spam

- There are many types of spam
  - Link spam
    - Adding links to message (blog) boards
    - (Joining) Link exchange networks
    - Link farming (spammers buys/creates a large no. of domains/sites)
  - Term spam
    - URL (archive) term spam
    - Dumping (fill documents with unrelated words)
    - Phrase stitching (combine words/sentences from various sources)
    - Weaving (add spam terms into a valid source and post it again)
Website:

BETTING NFL FOOTBALL PRO FOOTBALL SPORTSBOOKS NFL FOOTBALL LINE ONLINE NFL SPORTSBOOKS NFL

Players Super Book

When It Comes To Secure NFL Betting And Finding The Best Football Lines Players Super Book Is The Best Option! Sign Up And Ask For 30% In Bonuses.

MVP Sportsbook

Football Betting Has Never been so easy and secure! MVP Sportsbook has all the NFL odds you are looking for. Sign Up Now and ask for up to 30% in Cash bonuses.

Term spam:

pro football sportsbooks nfl football line online nfi sportsbooks nfl football gambling odds online pro nfi betting pro nfl gambling online nfl football spreads offshore football gambling online nfi gambling spreads online football gambling line online nfi betting nfl sportsbook online online nfl betting spreads betting nfl football online online football wagering online gambling online gambling football online nfl football betting odds offshore football sportsbook online nfl football gambling ...

Link spam:

Spam Detection

- Useful features
  - Unigrams
  - Formatting (invisible text, flashing, etc.)
  - Misspellings
  - IP address (Black list)

- Different features are useful for different spam detection tasks

- Email and Web page spam are by far the most widely studied, well understood and easily detected types of spam
Example: Spam Assassin Output

To: ...
From: ...
Subject: non profit debt
X-Spam-Checked: This message probably not SPAM
X-Spam-Score: 3.853, Required: 5
X-Spam-Level: *** (3.853)
X-Spam-Tests: BAYES_50,DATE_IN_FUTURE_06_12,URIABL_BLACK
X-Spam-Report-rig: ---- Start SpamAssassin (v2.6xx-cscf) results
    2.0 URIABL_BLACK Contains an URL listed in the URIABL blacklist
        [URLs: bad-debtyh.net.cn]
    1.9 DATE_IN_FUTURE_06_12 Date: is 6 to 12 hours after Received: date
    0.0 BAYES_50 BODY: Bayesian spam probability is 40 to 60%
        [score: 0.4857]

Say good bye to debt
Acceptable Unsecured Debt includes All Major Credit Cards, No-collateral
Bank Loans, Personal Loans,
Medical Bills etc.
http://www.bad-debtyh.net.cn
Blogs, online reviews/forum posts are often opinionated

Sentiment classification attempts to automatically identify the polarity of the opinion

- Negative opinion
- Neutral opinion
- Positive opinion

Sometimes the strength of the opinion is also important

- “Two stars” vs. “four stars”
- Weakly negative vs. strongly negative
Classifying Sentiment

- **Useful features**
  - Unigrams
  - Bigrams
  - Part of speech tags
  - Adjectives

- SVMs with *unigram* features have been shown to be outperform hand built rules
Sentiment Classification Example

All user reviews

- General Comments (148 comments) 82% positive
- Ease of Use (108 comments) 78% positive
- Screen (92 comments) 97% positive
- Software (78 comments) 35% positive
- Sound Quality (59 comments) 89% positive
- Size (59 comments) 76% positive
Unlike traditional search, online advertising goes beyond “topical relevance”

A user searching for ‘tropical fish’ may also be interested in pet stores, local aquariums, or even scuba diving lessons.

These are semantically related, but not topically relevant!

We can bridge the semantic gap by classifying ads and queries according to a semantic hierarchy.
Semantic Classification

- Semantic hierarchy ontology
  - Example: Pets / Aquariums / Supplies

- Training data
  - Large number of queries and ads are manually classified into the hierarchy

- Nearest neighbor classification has been shown to be effective for this task

- Hierarchical structure of classes can be used to improve classification accuracy
A set of unsupervised algorithms that attempt to find latent structure in a set of items.

Goal is to identify groups (clusters) of similar items, given a set of unlabeled instances.

Suppose I gave you the shape, color, vitamin C content and price of various fruits and asked you to cluster them.

- What criteria would you use?
- How would you define similarity?

Clustering is very sensitive to how items are represented and how similarity is defined!
Clustering

- General outline of clustering algorithms
  1. Decide how items will be represented (e.g., feature vectors)
  2. Define similarity measure between pairs or groups of items (e.g., cosine similarity)
  3. Determine what makes a “good” clustering
  4. Iteratively construct clusters that are increasingly “good”
  5. Stop after a local/global optimum clustering is found

- Steps 3 and 4 differ the most across algorithms
Hierarchical Clustering

- Constructs a *hierarchy of clusters*
  - Starting with some initial clustering of data and iteratively trying to improve the “quality” of clusters
  - The *top* level of the hierarchy consists of a *single* cluster with *all* items in it
  - The *bottom* level of the hierarchy consists of *N* (# of items) *singleton* clusters

- Different objectives lead to different types of clusters

- Two types of hierarchical clustering
  - Divisive (“*top down*”)
  - Agglomerative (“*bottom up*”)

- Hierarchy can be visualized as a *dendogram*
Example Dendrogram

Height indicates the similarity of the clusters involved.
Divisive & Agglomerative Hierarchical Clustering

- Divisive
  - Start with a *single* cluster consisting of all of the items
  - Until only *singleton clusters* exist…
    - *Divide* an existing cluster into two (or more) new clusters

- Agglomerative
  - Start with *N* (number of items) singleton clusters
  - Until a single cluster exists…
    - *Combine* two (or more) existing cluster into a new cluster

- How do we know how to divide or combined clusters?
  - Define a *division or combination cost*
  - Perform the division or combination with the *lowest cost*
Divisive Hierarchical Clustering
Agglomerative Hierarchical Clustering
Algorithm 1 Agglomerative Clustering

1: procedure AGGLOMERATIVECLUSTER($X_1, \ldots, X_N, K$)
2:     $A[1], \ldots, A[N] \leftarrow 1, \ldots, N$
3:     $ids \leftarrow \{1, \ldots, N\}$
4:     for $c = N$ to $K$ do
5:         bestcost $\leftarrow \infty$
6:         bestclusterA $\leftarrow$ undefined
7:         bestclusterB $\leftarrow$ undefined
8:     for $i \in ids$ do
9:         for $j \in ids - \{i\}$ do
10:             $c_{i,j} \leftarrow COST(C_i, C_j)$
11:             if $c_{i,j} < bestcost$ then
12:                 bestcost $\leftarrow c_{i,j}$
13:                 bestclusterA $\leftarrow i$
14:                 bestclusterB $\leftarrow j$
15:             end if
16:         end for
17:     end for
18:     $ids \leftarrow ids - \{bestClusterA\}$
19:     for $i = 1$ to $N$ do
20:         if $A[i]$ is equal to bestClusterA then
21:             $A[i] \leftarrow bestClusterB$
22:         end if
23:     end for
24: end procedure
Clustering Costs

- Cost: a measure of how expensive to merge 2 clusters
  
  - Single linkage
    \[
    COST(C_i, C_j) = \min\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}
    \]
  
  - Complete linkage
    \[
    COST(C_i, C_j) = \max\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}
    \]
  
  - Average linkage
    \[
    COST(C_i, C_j) = \frac{\sum_{x_i \in C_i, x_j \in C_j} dist(X_i, X_j)}{|C_i||C_j|}
    \]
  
  - Average group linkage (\(U_C = \sum_{X \in C} X / |C|\) is the centroid of cluster C)
    \[
    COST(C_i, C_j) = dist(\mu_{C_i}, \mu_{C_j})
    \]
Clustering Strategies

- **Single Linkage**
- **Complete Linkage**
- **Average Linkage**
- **Average Group Linkage**
K-Means Clustering

- Hierarchical clustering constructs a hierarchy of clusters
- K-means always maintains exactly $K$ clusters
  - Clusters represented as centroids (“center of mass”)
- Basic algorithm:
  - Step 0: Choose $K$ cluster centroids
  - Step 1: Assign points to closest centroid
  - Step 2: Re-compute cluster centroids
  - Step 3: Goto Step 1
- Tends to converge quickly
- Can be sensitive to choice of initial centroids
- Must choose $K$!
K-Means Clustering

- Goal: find the cluster assignments (for the assignment vectors $A[1], \ldots, A[N]$) that minimize the cost function:

$$\text{COST}(A[1], \ldots, A[N]) = \sum_{k=1}^{K} \sum_{i: A[i] = k} \text{dist}(X_i, C_k)$$

where $\text{dist}(X_i, C_k) = \|X_i - \mu_{C_k}\|^2$, where $\mu_{C_k}$ is the centroid of $C_k$

$$= (X_i - \mu_{C_k}) \cdot (X_i - \mu_{C_k}), \text{ the Euclidean Distance}$$

- Strategy:
  1. Randomly select $K$ initial cluster centers (instances) as seeds
  2. Move the cluster centers around to minimize the cost function
      i. Re-assign instances to the cluster with the closest centroid
      ii. Re-compute the cost value of each centroid based on the current members of its cluster
K-Means Clustering

- Example.

(a) (b) (c)

selection of seeds

assignment of documents (iter. 1)

recomputation/movement of $\bar{\mu}$'s (iter. 1)

(d) (e)

$\bar{\mu}$'s after convergence (iter. 9)

movement of $\bar{\mu}$'s in 9 iterations
K-Means Clustering

- The $K$-means optimization problem:
  - A naïve approach is to try every possible combination of cluster assignments, which is infeasible for large data sets
  - The $K$-means algorithm should find an approximate, heuristic solution that iteratively tries to minimize the cost
  - Anticipated results:
    1. the solution is not guaranteed to be globally optimal
    2. the solution is not even guaranteed to be locally optimal
    3. despite the heuristic nature, the $K$-means algorithm tends to work very well in practice

- In practice, $K$-means clustering tends to converge quickly
  - Compared to hierarchical clustering $H$, $K$-means is more efficient and produces clusters of similar quality to $H$
  - Implementing $K$-means requires $O(KN)$, rather than $O(N^2)$
K-Means Clustering Algorithm

Algorithm 1 K-Means Clustering

1: procedure KMEANSCluster ($X_1, \ldots, X_N, K$)
2: \hfill $A[1], \ldots, A[N] \leftarrow$ initial cluster assignment (* Either randomly or using some knowledge of the data *)
3: repeat
4: \quad change $\leftarrow$ false
5: \quad for $i = 1$ to $N$ do
6: \quad \quad $\hat{k} \leftarrow \arg \min_k \text{dist}(X_i, C_k)$ (* Each instance is assigned to the closest cluster *)
7: \quad \quad if $A[i]$ is not equal $\hat{k}$ then
8: \quad \quad \quad $A[i] \leftarrow \hat{k}$ (* The cluster of an instance changes; proceeds *)
9: \quad \quad end if
10: \quad end for
11: until change is equal to false return $A[1], \ldots, A[N]$
12: end procedure
K Nearest Neighbor Clustering

- Hierarchical and K-Means clustering partition items into clusters
  - Every item is in exactly one cluster
- K Nearest neighbor clustering forms one cluster per item
  - The cluster for item $j$ consists of $j$ and the $K$ nearest neighbors of $j$
  - Clusters now overlap
5 Nearest Neighbor Clustering
K Nearest Neighbor Clustering

- **Drawbacks** of the *K* Nearest Neighbor Clustering method
  - Often fails to find meaningful clusters
    - In *spare areas* of the input space, the instances assigned to a cluster are far away (e.g., *D* in the 5-NN example)
    - In *dense areas*, some related instances may be missed if *K* is not large enough (e.g., *B* in the 5-NN example)
  - Computational expensive (compared with *K*-means), since it computes distances between each pair of instances

- **Applications** of the *K* Nearest Neighbor Clustering method
  - Emphasize finding a small number (rather than all) of closely related instances, i.e., precision over recall
Evaluating Clustering

- Evaluating clustering is challenging, since it is an unsupervised learning task.
- If labels exist, can use standard IR metrics, e.g., precision/recall.
- If not, then can use measures such as “cluster precision”, which is defined as:

\[
ClusterPrecision = \frac{\sum_{i=1}^{K} |\text{MaxClass}(C_i)|}{N}
\]

where \( K (= |C|) \) is the total number of resultant clusters,

\(|\text{MaxClass}(C_i)|\) is the number of instances in cluster \( C_i \) with the most instances of (human-assigned) class label \( C_i \),

\( N \) is the total number of instances.

- Another option is to evaluate clustering as part of an end-to-end system, e.g., clusters to improve web ranking.
How to Choose K?

- *K*-means and *K* nearest neighbor clustering require us to choose *K*, the number of clusters
  - *No* theoretically appealing way of choosing *K*
  - Depends on the *application* and *data*; often chosen experimentally to evaluate the quality of the resulting clusters for various values of *K*

- Can use *hierarchical clustering* and choose the best level

- Can use *adaptive K* for *K*-nearest neighbor clustering
  - Larger (Smaller) *K* for *dense* (spare) areas
  - Challenge: choosing the *boundary* size

- Difficult problem with *no* clear solution
Adaptive Nearest Neighbor Clustering
Clustering and Search

- Cluster hypothesis
  - “Closely associated documents tend to be relevant to the same requests” – van Rijsbergen ‘79

- Tends to hold in practice, but *not always*

- Two retrieval modeling options
  - (Clustered-based) *Rank* clusters using
    \[ P(Q \mid C_j) = \prod_{i=1}^{n} P(q_i \mid C_j) \]
  - *Smooth* documents (language model) using *K-NN clusters*:
    \[
    P(w \mid D) = (1 - \lambda - \delta) \frac{f_{w,D}}{|D|} + \delta \sum_{C_j} \frac{f_{w,C_j}}{|C_j|} P(D \mid C_j) + \lambda \frac{f_{w,Coll}}{|Coll|}
    \]
    where
    - \( f_{w,D} = \text{frequency of word } w \text{ in } D \)
    - \( f_{w,C_j} = \text{frequency of word } w \text{ in } C_j \text{ that contains } D \)
    - \( f_{w,Coll} \text{ is the frequency of word } w \text{ in the collection Coll} \)