Unsupervised Learning

CS534
Supervised vs Unsupervised Learning

• So far we have assumed that the training examples were labeled with their class label or target variable --- supervised learning
  \[ D = \{(x_i, y_i)\}_{i=1}^{N} \]

• We assume now that all one has is a collection of examples without being told what the target variable is --- unsupervised learning
  \[ D = \{x_i\}_{i=1}^{N} \]
What can we learn from unlabeled data?

- Group of clusters in the data
- Low dimensional structure
Clustering:

• Clustering: the process of grouping a set of objects into classes of similar objects
  • high intra-class similarity
  • low inter-class similarity
• It is the most common form of unsupervised learning
Example Applications

• Find genes that are similar in their functions
• Group documents based on topics
• Categorize customers based on their buying habit
• Group images based on their contents
• ......
Issues for clustering

• What is a natural grouping among these objects?
  – Definition of "groupness"
• What makes objects “related”?
  – Definition of "similarity/distance"
• Representation for objects
  – Fixed length vector? Feature normalization?
• How many clusters?
  – Fixed a priori?
  – Completely data driven?
  – Avoid “trivial” clusters - too large or small
• Clustering Algorithms
  – Partitional algorithms
  – Hierarchical algorithms
What is a natural grouping among these objects?

By color? By pattern? By weight?

The definition of natural grouping is subjective.

This is why we call clustering exploratory data analysis.
What is similarity

• The real meaning of similarity is a philosophical question. We will take a more pragmatic approach
  – Depends on representation and algorithm. For many rep./alg., easier to think in terms of a distance (rather
    than similarity) between vectors

Hard to define but
We know it when we see it
What properties should a distance measure have?

• \( D \) must be Symmetric
  – \( D(A, B) = D(B, A) \)
  – Otherwise, we can say \( A \) looks like \( B \) but \( B \) does not look like \( A \)

• Positivity, and self-similarity
  – \( D(A, B) \geq 0 \), and \( D(A, B) = 0 \) iff \( A = B \)
  – Otherwise there will different objects that we cannot tell apart

• Must satisfy triangle inequality
  – \( D(A, B) + D(B, C) \geq D(A, C) \)
  – Otherwise one can say “\( A \) is like \( B \), \( B \) is like \( C \), but \( A \) is not like \( C \) at all”
Distance Measures: Minkowski Metric

• Suppose two object $x$ and $y$ both have $d$ features
  $- x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$
• The Minkowski metric of order $r$ is defined by
  $$d(x, y) = \sqrt[r]{\sum_{i} |x_i - y_i|^r}$$
• Common Minkowski metrics:
  - Euclidean ($r=2$): $d(x, y) = \sqrt{\sum (x_i - y_i)^2}$, also called $L_2$ distance
  - Manhattan distance ($r=1$): $d(x, y) = \sum |x_i - y_i|$, also called $L_1$ distance
  - “Sup” distance ($r = +\infty$): $d(x, y) = \max_i |x_i - y_i|$, also called $L_\infty$ distance
An Example

1: Euclidean distance: \[ \sqrt{4^2 + 3^2} = 5. \]
2: Manhattan distance: \[ 4 + 3 = 7. \]
3: "sup" distance: \[ \max\{4,3\} = 4. \]
Other Distances

- Hamming distance (Manhattan distance on binary features)
  - # of features that differ
  - e.g.: distance of two sites based on their species composition

<table>
<thead>
<tr>
<th></th>
<th>sp1</th>
<th>sp2</th>
<th>sp3</th>
<th>sp4</th>
<th>sp5</th>
<th>sp6</th>
<th>sp7</th>
<th>sp8</th>
<th>sp9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A:</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Site B:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ D(A, B) = 4 \]

- Mahalanobis distance (assuming \( x, y \) follows a Gaussian distribution with covariance matrix \( \Sigma \))

\[
D(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}
\]
Similarities

• Cosine similarities – commonly used to measure document similarity

\[ \cos(x, x') = \frac{\langle x \cdot x' \rangle}{|x| |x'|} \]

• Kernels – e.g., RBF (Gaussian) Kernel

\[ S(X, X') = \exp \left( -\frac{|X - X'|^2}{2\sigma^2} \right) \]
Clustering algorithms

• Hierarchical algorithms
  – Bottom up – agglomerative
  – Top down – divisive

• Partition algorithms (Flat)
  – K-means
  – Mixture of Gaussian
  – Spectral Clustering
Hierarchical Clustering

• Given a set of objects, build a tree-based taxonomy

• Hierarchies are convenient way for organizing information
Hierarchical Agglomerative Clustering (HAC)

- Starts with each object in a separate cluster
- Repeatedly joins the closest pair of clusters
- until there is only one cluster

The history of merging forms a tree of hierarchy

**Question**: how to measure the “closeness” of two clusters?
Closest Pair of Clusters

The distance between two clusters is defined as the distance between:

- **Single-link**
  - The nearest pair of points

- **Complete-link**
  - The furthest pair of points

- **Centroid**
  - The center of gravity

- **Average-link**
  - Average of all cross-cluster pairs
Single Link Method
Complete Link Method

(1) 

(2) 

(3)
Visualization: Dendrogram

- Height of the joint = the distance between the two merge clusters
- The merge distance monotonically increases as we merge more and more for:
  - Single, complete and average linkage methods
  - Not for the centroid method

This example is shown upside down.
Interpreting Dendrogram

- Dendrogram can be used to identify
  - the number of clusters in data
    - A horizontal cut creates a partition
    - Moving the cut from root down creates more clusters
    - Large gaps indicate good cutting points
  - well-formed clusters
    - Some clusters are better formed than the other
    - This can be easy to see on a dendrogram
Example
Single Link vs. Complete Link

- Single-link creates straggly clusters due to chaining effect
Partitional Clustering

• Given a data set of n points, we know that there are k clusters in the data, how to find these clusters?
• Roughly speaking there are $O(k^n)$ ways to partition the data, Which one is better?
• One intuition says that we want tight clusters, i.e., points should be in a tight ball
• This leads to the following objective function
  $$\sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$  --- squared distance between data point x and its cluster center
• Optimizing this objective is a combinatorial optimization problem
  — Exhaustive search for an optimal solution is not feasible
Combinatorial optimization: An iterative solution

• **Initialization**: Start with a random partition of the data

• **Iterative step**: the cluster assignments and cluster centers are updated to improve the objective

• **Stopping criterion**: if no improvement can be achieved.
  
  Iterative greedy descent
  – convergence is guaranteed, but to local optimal
K-Means

Algorithm

Input – Desired number of clusters, \( k \)

Initialize – the \( k \) cluster centers (randomly if necessary)

Iterate –

1. Assigning each of the \( N \) data points to its nearest cluster centers
2. Re-estimate the cluster center by assuming that the current assignment is correct

\[
\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x
\]

Termination –

If none of the data points changed membership in the last iteration, exit. Otherwise, go to 1
K-Means Example (K=2)

Pick seeds
Reassign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!
Computational Complexity

• At each iteration:
  – Reassigning clusters: $O(kn)$ distance computations
  – Computing centroids: Each instance vector gets added once to some centroid: $O(n)$

• Assume these two steps are each done once for $I$ iterations: $O(lkn)$.

• Linear in all relevant factors, assuming a fixed number of iterations, more efficient than $O(n^2)$ HAC

• Does it always converge?
Kmeans Convergence

Objective

\[ \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 \]

1. Fix \( \mu \), optimize \( C \):

\[ \min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{C} \sum_{i=1}^{n} |x_i - \mu_{x_i}|^2 \]

2. Fix \( C \), optimize \( \mu \):

\[ \min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 \]

- Take partial derivative of \( \mu_i \) and set to zero, we have

\[ \mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \]

**Step 1 of kmeans**

**Step 2 of kmeans**

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge.
Impact of Initial Seeds

• Highly sensitive to the initial seeds

• Multiple random trials: choose the one with best sum of squared loss (important!)

• Heuristics for choosing better centers
  – choose initial centers to be far apart – furthest first traversal
  – Initialize with results of other clustering method
More Comments

• K-Means is exhaustive:
  – Cluster every data point, no notion of outlier
  – Outliers cause problems
    • Become singular clusters
    • Bias the centroid estimation

• K-medoids methods is more robust to outliers
  – Cluster medoid: the point that has minimum sum squared distance to all data points in the cluster
  – More expensive to compute
    • For each pt: sum squared dist with all other pts in cluster $O(|C|^2)$

• Need to specify $k$: difficult in practice
  – Automatically deciding $k$?  more on this later...
Soft Clustering

• Hard clustering:
  – Data point is deterministically assigned to one and only one cluster
  – But in reality clusters may overlap

• Soft-clustering:
  – Data points are assigned to clusters with certain probabilities

• Model-based clustering
Side track: Gaussian Bayes Classifier

- We have $k$ classes in our data
- Each class contains data generated from a particular Gaussian distribution
- Data is generated by
  - first randomly select one of the classes according to class prior $p(y)$
  - draw random samples from the Gaussian distribution of that particular class

\[
P(x, y) = P(x | y)P(y)
\]
\[
P(x | y = i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}
\]
Back to Unsupervised Learning

• Now assume we know our data is generated in the same way
• If we know the labels, we can estimate the mean and covariance of the each class using ML estimation
  – Bayes Gaussian Classifier
  – If assuming shared covariance, this leads to LDA
• But for unsupervised learning, we don’t have the labels
• How can we learn the correct model from the incomplete data?
Gaussian Mixture Model

\[
P(x) = \sum_{i=1}^{k} P(x, y = i) = \sum_{i=1}^{k} P(x \mid y = i)P(y = i) = \sum_{i=1}^{k} \alpha_i P(x \mid \theta_i)
\]

\(\alpha_i = p(y=i): \text{the class prior}\)

\(\theta_i = \{\mu_i, \Sigma_i\}\)

Goal of unsupervised learning:

• Given a set of x’s, estimate \(\{\alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k\}\)

• Once the model is identified, we can compute \(p(y = i \mid x)\) for \(i = 1, \ldots, k\),
Maximum Marginal Likelihood

\[
\arg \max_{\theta} \prod_j P(x^j) = \arg \max_{\theta} \prod_j \sum_{i=1}^k P(x^j, y^j = i)
\]
\[
= \arg \max_{\theta} \sum_{j=1}^n \log \sum_{i=1}^k P(x^j, y^j = i)
\]

log sum is difficult to optimize!

Gradient ascent is doable but very inefficient
Expectation Maximization (EM)

• A highly used approach for dealing with hidden (missing) data
  – Here the cluster labels are hidden
• Much simpler than gradient methods
• It is an iterative algorithm that starts with some initial guess of the model parameters
• Iteratively performs two linked steps:
  – **Expectation (E-step):** given current model parameters $\lambda_t$, compute the expectation for the hidden (missing) data
  – **Maximization (M-step):** re-estimate the parameters $\lambda_{t+1}$ assuming that the expected values computed in the E-step are the true values
• We will first show how it works for mixture of Gaussian
EM – simple case

• A simple case:
  – We have unlabeled data $x^1, \ldots, x^m$
  – We know there are $k$ classes
  – We know $\alpha_1 = P(y = 1), \ldots, \alpha_k = P(y = k)$
  – We don’t know $\mu_1 \ldots \mu_k$, but know the common variance $\sigma^2$

Start with an initial guess for $\mu_1, \ldots, \mu_k$,

1. If we know $\mu_1, \ldots, \mu_k$, we can easily compute probability that a point $x^j$ belongs to class $i$:

$$p(y = i|x^j) \propto \exp \left( -\frac{1}{2\sigma^2} |x^j - \mu_i|^2 \right)p(y = i)$$

Simply evaluate this, then normalize

2. If we know the probability that each point belongs to each class, we can estimate the $\mu_1, \ldots, \mu_k$

$$\mu_i = \frac{\sum_{j=1}^{m} p(y=i|x^j)x^j}{\sum_{j=1}^{m} p(y=i|x^j)}$$
EM – Axis-aligned Gaussian

- We have unlabeled data \(x^1, \ldots, x^m\)
- We know there are \(k\) classes
- We know that the Gaussians are axis aligned

Start with an initial guess for \(\mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k, \alpha_1, \ldots, \alpha_k\),

1. If we know the parameters, we can easily compute probability that a point \(x^j\) belongs to class \(i\):

\[
p(y = i | x^j) \propto p(x^j | \mu_i, \Sigma_i) p(y = i)
\]

Simply evaluate this, then normalize

2. If we know the probability that each point belongs to each class, we can estimate the \(\mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k, \alpha_1, \ldots, \alpha_k\),

\[
\mu_i = \frac{\sum_{j=1}^m p(y = i | x^j) x^j}{\sum_{j=1}^m p(y = i | x^j)} \quad \alpha_i = \frac{\sum_{j=1}^m p(y = i | x^j)}{m} \quad \sigma_{il}^2 = \frac{\sum_{j=1}^m p(y = i | x^j) (x^j_i - \mu_{il})^2}{\sum_{j=1}^m p(y = i | x^j)}
\]
EM – General Gaussian

Start with an initial guess for $\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k, \alpha_1, ..., \alpha_k$,

1. If we know the parameters, we can easily compute probability that a point $x^j$ belongs to class $i$:

$$ p(y = i|x^j) \propto p(x^j|\mu_i, \Sigma_i) p(y = i) $$

Simply evaluate this, then normalize

2. If we know the probability that each point belongs to each class, we can estimate the $\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k, \alpha_1, ..., \alpha_k$,

$$ \mu_i = \frac{\sum_{j=1}^{m} p(y=i|x^j)x^j}{\sum_{j=1}^{m} p(y=i|x^j)} $$

$$ \alpha_i = \frac{\sum_{j=1}^{m} p(y=i|x^j)}{m} $$

$$ \Sigma_i = \frac{\sum_{j=1}^{m} p(y=1|x^j)(x^j-\mu_i)(x^j-\mu_i)^T}{\sum_{j=1}^{m} p(y=j|x^j)} $$
Gaussian Mixture Example: Start
After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration
Q: Why are these two points red?
Behavior of EM

• It is guaranteed to converge
  – Convergence proof is based on the fact that $P(x|\theta)$ must increase or remain same between iterations (not obvious)
  – In practice it may converge slowly, one can stop early if the change in log-likelihood is smaller than a threshold

• It converges to a local optimum
  – Multiple restart is recommended