Machine Learning
Toolbox for Text Analysis: Unsupervised Learning

Jessy Li, UT Austin
With many slides from Byron Wallace
So far: Supervised learning

- Labeled data \( \mathcal{L} \)
- Unlabeled data \( \mathcal{U} \)
- Learner
- Predictive model
This time: Unsupervised learning
Unsupervised learning

• Given a bunch of data, make sense of it by making some assumptions and imposing them

• Example: clustering -- put data into (usually discrete) groups or clusters in a way that is useful/makes sense/whatever
Example
(from reading)

Figure 1a: Initial points.

Figure 1b: Two clusters.

Figure 1c: Six clusters

Figure 1d: Four clusters.
To cluster, we must specify...

- A distance function between points
- What makes a good clustering (i.e., an objective)
- A procedure to maximize this objective
What’s a “cluster”?
Cluster (def 1)

Well-separated set of points s.t. any point in cluster \( c \) is closer to other points \( c \) than to points in other clusters \( c' \)
Cluster (def 2)

Center-based a cluster is a set of objects that any point in c is closer to the center c than to the center of other clusters c'
Cluster (def 3)

**Contiguous** a set of points such that any point in c is closer to one or more points in c than to any point not in c

![Figure 5: Eight contiguous clusters of 2 dimensional points.](image)
OK… but how do we calculate ‘nearness’?

We need a *dissimilarity* measure $d(i,j)$!

- What should be the properties of this measure?
We need a dissimilarity measure $d(i,j)$!

Properties:

1) $d(i,i) = 0$ for all $i$ (points are not dissimilar from themselves)
2) $d(i,j) = d(j,i)$ (symmetry)
3) $d(i,j) \geq 0$ for all $i,j$ (positivity)

OK... but how do we calculate ‘nearness’?
We need a *dissimilarity* measure \( d(i,j) \)!

**Properties:**

1) \( d(i,i) = 0 \) for all \( i \) (points are not dissimilar from themselves)
2) \( d(i,j) = d(j,i) \) (symmetry)
3) \( d(i,j) \geq 0 \) for all \( i,j \) (positivity)
4) \( d(i,j) = 0 \) iff \( i = j \)
5) Triangle inequality

OK… but how do we calculate ‘nearness’?

then it’s a distance metric!
Hierarchical / bottom-up clustering

(1) Assign every point to its own cluster
(2) Find two most similar clusters and join them
(3) Repeat until there is only one cluster
K-means

• K-means is a **partitional clustering** algorithm

• Let the set of data points (or instances) $D$ be
  $$\{x_1, x_2, \ldots, x_n\},$$
  where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a **vector** in a real-valued space $X \subseteq \mathbb{R}^r$, and $r$ is the number of attributes (dimensions) in the data.
K-means

• K-means is a **partitional clustering** algorithm

• Let the set of data points (or instances) $D$ be
  \[
  \{x_1, x_2, \ldots, x_n\},
  \]
  where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a **vector** in a real-valued space $X \subseteq R^r$, and $r$ is the number of attributes (dimensions) in the data.

• The $k$-means algorithm partitions the given data into $k$ clusters.
  – Each cluster has a cluster **center**, called **centroid**.
  – $k$ is specified by the user

Credit: Bing Liu, UIC
K-means algorithm

Given $k$, the *k*-means algorithm works as follows:

1) Randomly choose $k$ data points (**seeds**) to be the initial **centroids**, cluster centers

2) Assign each data point to the closest **centroid**

3) Re-compute the **centroids** using the current cluster memberships. (Just an average!)

4) Go back to 2.

Is this correct?
K-means algorithm

Given $k$, the *k-means* algorithm works as follows:

1) Randomly choose $k$ data points (seeds) to be the initial **centroids**, cluster centers

2) Assign each data point to the closest centroid

3) Re-compute the **centroids** using the current cluster memberships. (Just an average!)

4) If a convergence criterion is not met, go to 2).

*Credit:* Bing Liu, UIC
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the sum of squared error (SSE).

- $C_j$ is the $j$th cluster,
- $m_j$ is the centroid of cluster $C_j$ (the mean vector of all the data points in $C_j$),
- $\text{dist}(x, m_j)$ is the distance between data point $x$ and centroid $m_j$.

$$
\sum_{j=1}^{\infty} \sum_{x \in C_j} \text{dist}^2(x, m_j),
$$
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the sum of squared error (SSE),

\[
SSE = \sum_{j=1}^{k} \sum_{x \in C_j} \text{dist}(x, m_j)^2
\]  \hspace{1cm} (1)

- \( C_j \) is the \( j \)th cluster, \( m_j \) is the centroid of cluster \( C_j \) (the mean vector of all the data points in \( C_j \)), and \( \text{dist}(x, m_j) \) is the distance between data point \( x \) and centroid \( m_j \).
Calculating centroids

• Say cluster 1 = {(1,4), (3,5), (8,4)}
• We just average these to get the centroid.
  \[-(1,4) + (3,5) + (8,4) = (12, 13)/3 = (4, 4.333)\]
K-means demo

- http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html
Can use any dissimilarity measure with $k$-means

One possibility: the Euclidean distance!

$$p_{ij} = \left( \sum_{k=1}^{d} \left| x_{ik} - x_{jk} \right|^r \right)^{1/r}$$
<table>
<thead>
<tr>
<th>Names</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean distance</td>
<td>$|a - b|_2 = \sqrt{\sum_i (a_i - b_i)^2}$</td>
</tr>
<tr>
<td>Squared Euclidean</td>
<td>$|a - b|_2^2 = \sum_i (a_i - b_i)^2$</td>
</tr>
<tr>
<td>Manhattan distance</td>
<td>$|a - b|_1 = \sum_i</td>
</tr>
<tr>
<td>maximum distance</td>
<td>$|a - b|_\infty = \max_i</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>$\sqrt{(a - b)^T S^{-1} (a - b)}$ where $S$ is the Covariance matrix</td>
</tr>
<tr>
<td>Cosine similarity</td>
<td>$\frac{a \cdot b}{|a| |b|}$</td>
</tr>
</tbody>
</table>
K-means algorithm

Given $k$, the $k$-means algorithm works as follows:

1) Randomly choose $k$ data points (seeds) to be the initial centroids, cluster centers

2) Assign each data point to the closest centroid

3) Re-compute the centroids using the current cluster memberships. (Just an average!)

4) If a convergence criterion is not met, go to 2).

Credit: Bing Liu, UIC
How’d we do?
What if we kept going?
Strengths of k-means

- It is:
  - Simple
  - Efficient

- K-means is probably most popular clustering algorithm

- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity

Credit: Bing Liu, UIC
Weaknesses of k-means

• The algorithm is only applicable if the mean is defined.
  – For categorical data, k-mode - the centroid is represented by most frequent values.

• The user needs to specify $k$.

• The algorithm is sensitive to outliers
  – Outliers are data points that are very far away from other data points.
  – Outliers could be errors in the data recording or some special data points with very different values.

Credit:: Bing Liu, UIC
Weaknesses of k-means: Problems with outliers

(A): Undesirable clusters

(B): Ideal clusters

Credit: Bing Liu, UIC
Weaknesses of k-means (cont …)

$k$-means is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).

Credit: Bing Liu, UIC
What if we took a more contiguous approach?

**Contiguous** a set of points such that any point in c is closer to one or more points in c than to any point not in c.

*Figure 5: Eight contiguous clusters of 2 dimensional points.*
Density-based clustering

• Basic idea: clusters = regions that are dense; these are separated by sparse regions

• Advantages
  Discover clusters of arbitrary shape.
  Clusters – Dense regions of objects separated by regions of low density
Density-based clustering

• Can find things like this
DB-Scan

**Density Definition**

- $\varepsilon$-Neighborhood – Objects within a radius of $\varepsilon$ from an object.
  \[ N_\varepsilon(p) : \{ q \mid d(p, q) \leq \varepsilon \} \]

- “High density” - $\varepsilon$-Neighborhood of an object contains at least $\text{MinPts}$ of objects.

\[ \text{Density of } p \text{ is “high” (MinPts = 4)} \]
\[ \text{Density of } q \text{ is “low” (MinPts = 4)} \]

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
Given $\varepsilon$ and $\text{MinPts}$, categorize the objects into three exclusive groups.

A point is a **core point** if it has more than a specified number of points (MinPts) within $\varepsilon$—These are points that are at the interior of a cluster.

A **border point** has fewer than MinPts within $\varepsilon$, but is in the neighborhood of a core point.

A **noise point** is any point that is not a core point nor a border point.

$\varepsilon = 1 \text{ unit}, \text{MinPts} = 5$

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
Example

Original Points

Point types: core, border and outliers

\( \varepsilon = 10, \text{MinPts} = 4 \)

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
Density-reachability

- Directly density-reachable
  - An object \( q \) is directly density-reachable from object \( p \) if \( p \) is a core object and \( q \) is in \( p \)'s \( \varepsilon \)-neighborhood.

- \( q \) is directly density-reachable from \( p \)
- \( p \) is not directly density-reachable from \( q \)
- Density-reachability is asymmetric

MinPts = 4

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
Density-reachability

• Density-Reachable (directly and indirectly):
  – A point $p$ is directly density-reachable from $p_2$
  – $p_2$ is directly density-reachable from $p_1$
  – $p_1$ is directly density-reachable from $q$
  – $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain

• $p$ is (indirectly) density-reachable from $q$
• $q$ is not density-reachable from $p$

MinPts = 7
DB-Scan algorithm

**Algorithm:**

```plaintext
for each \( o \in D \) do
    if \( o \) is not yet classified then
        if \( o \) is a core-object then
            collect all objects density-reachable from \( o \)
            and assign them to a new cluster.
        else
            assign \( o \) to NOISE
```
DBSCAN Algorithm: Example

- **Parameter**
  - $\varepsilon = 2$ cm
  - $MinPts = 3$

```plaintext
for each $o \in D$ do
  if $o$ is not yet classified then
    if $o$ is a core-object then
      collect all objects density-reachable from $o$
      and assign them to a new cluster.
    else
      assign $o$ to NOISE
  
Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
DBSCAN Algorithm: Example

- Parameter
  - $\varepsilon = 2$ cm
  - $MinPts = 3$

```plaintext
for each $o \in D$ do
  if $o$ is not yet classified then
    if $o$ is a core-object then
      collect all objects density-reachable from $o$
      and assign them to a new cluster.
    else
      assign $o$ to NOISE
  end if
end for
```

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
DBSCAN Algorithm: Example

- **Parameter**
  - $\varepsilon = 2$ cm
  - $MinPts = 3$

```plaintext
for each $o \in D$ do
  if $o$ is not yet classified then
    if $o$ is a core-object then
      collect all objects density-reachable from $o$
      and assign them to a new cluster.
    else
      assign $o$ to NOISE
  end if
end for
```

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
DBSCAN Algorithm: Pseudocode

DBSCAN(D, eps, MinPts)
    C = 0
    for each unvisited point P in dataset D
        mark P as visited
        NeighborPts = regionQuery(P, eps)
        if sizeof(NeighborPts) < MinPts
            mark P as NOISE
        else
            C = next cluster
            expandCluster(P, NeighborPts, C, eps, MinPts)

expandCluster(P, NeighborPts, C, eps, MinPts)
    add P to cluster C
    for each point P' in NeighborPts
        if P' is not visited
            mark P' as visited
            NeighborPts' = regionQuery(P', eps)
            if sizeof(NeighborPts') >= MinPts
                NeighborPts = NeighborPts joined with NeighborPts'
                if P' is not yet member of any cluster
                    add P' to cluster C

regionQuery(P, eps)
    return all points within P's eps-neighborhood (including P)
DBSCAN Algorithm: Pseudocode

DBSCAN(D, eps, MinPts)

C = 0

for each unvisited point P in dataset D
  mark P as visited
  NeighborPts = regionQuery(P, eps)
  if sizeof(NeighborPts) < MinPts
    mark P as NOISE
  else
    C = next cluster
    expandCluster(P, NeighborPts, C, eps, MinPts)

expandCluster(P, NeighborPts, C, eps, MinPts)

  add P to cluster C
  for each point P' in NeighborPts
    if P' is not visited
      mark P' as visited
      NeighborPts' = regionQuery(P', eps)
      if sizeof(NeighborPts') >= MinPts
        NeighborPts = NeighborPts joined with NeighborPts'
      if P' is not yet member of any cluster
        add P' to cluster C

regionQuery(P, eps)

  return all points within P's eps-neighborhood (including P)

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
When DBSCAN Works Well

- Resistant to Noise
- Can handle clusters of different shapes and sizes

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
When DBSCAN Does NOT Work Well

- Cannot handle varying densities
- Sensitive to parameters—hard to determine the correct set of parameters

Original Points

(MinPts=4, Eps=9.92).

(MinPts=4, Eps=9.75)

Credit: https://cse.buffalo.edu/~jing/cse601/fa13/materials/clustering_density.pdf
Take-away Message

• The basic idea of density-based clustering
• The two important parameters and the definitions of neighborhood and density in DBSCAN
• Core, border and outlier points
• DBSCAN algorithm
• DBSCAN’s pros and cons
Evaluating clusters

• This is hard!

• No gold standard.

• Data may be grouped in multiple ways, all of which defensible

• User (expert) input is critical

  • Somehow show the expert the induced clusterings

  • E.g., if we clustered documents, show a few samples from different clusters
Evaluating clusters

• Sometimes people “conflate” classification with clustering!

  • Take some labeled data, cluster it (ignoring labels) then evaluate with respect to labels!

• What’s wrong with this?

  • This assumes clusters = classes, which is not always obvious or warranted!!
Evaluating clusters

- **Purity**: proportion of cluster belonging to the majority label therein!
  
  - e.g., if we have docs with labels “world news” and “sports” and cluster c contains 100 points: 80 “world news” and 20 “sports”, then purity(c) = .8!
  
  - In general you would average the purity of all clusters!

- What is a trivial clustering case where purity is 1?
Evaluating clusters

- **Normalized mutual information** eliminates this trade-off; see https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html
Evaluating clusters

• Downstream evaluation: evaluate clusterings with respect to some other task!
  
  • E.g., if we’re making a recommendation engine, use clusterings to improve recommendations and assess with respect to this criteria!

• Cons?
  
  • Assumes that the downstream task has a reasonable evaluation!

  • This is great when applicable, but not always possible!
Data Reduction

Summarize data with many \((p)\) variables by a smaller set of \((k)\) derived (synthetic, composite) variables.
Dimensionality reduction

• “Residual” variation is information in $A$ that is not retained in $X$

• Balancing act between
  – clarity of representation, ease of understanding
  – oversimplification: loss of important or relevant information.