Machine Learning

Chao Lan
Clustering and Dimensionality Reduction

Clustering
- Kmeans
- DBSCAN
- Gaussian Mixture Model

Dimensionality Reduction
- principal component analysis
- manifold learning

Other Feature Processing

Learning Theory
Clustering is the task of grouping instances based on their similarities.

- many ways to measure similarity; a common way is Euclidean distance

\[ \| x - x' \|_2 = \| \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} - \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_p \end{bmatrix} \|_2 = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_p - x'_p)^2} \]
Clustering is Subjective and not Unique

What is a natural grouping among these objects?

Clustering is subjective

- Simpson's Family
- School Employees
- Females
- Males
1. K-means

Assign an example to a cluster to minimize the total within-cluster distance.
Lloyd Algorithm

Alternately update (1) cluster center and (2) cluster assignment, until converge.
Impact of Initial Cluster Centers

Clustering depends on initial cluster centers (choose the best over random trials).
Impact of Cluster Number

Choosing $K$ in practice is challenging, and larger $K$ often implies smaller total distance.
2. DBSCAN

If two points’ distance is no greater than $\varepsilon$, they are assigned to the same group. If a group has more than $m$, it becomes a cluster; otherwise it is a group of outliers.
The Clustering Algorithm
Advantages of DBSCAN

Compared to Kmeans…

No need to specify cluster number k.

No dependence on initial points.

Can find nonlinearly separable clusters.
3. Gaussian Mixture Model

Assume examples are generated from a mixture of Gaussian distributions. Each example is assigned to the distribution that contributes most to its generation.

- can also assign to different distributions with different probabilities (soft-assignment)
Generative Component Identification

\[ p(x) = \sum_{k=1}^{K} p(x, z_k) = \sum_{k=1}^{K} p(x \in z_k) \cdot p(x | z_k) = \sum_{k=1}^{K} \pi_k \cdot N(x | \mu_k, \Sigma_k) \]

\[ p(z_k | x_i) = \frac{\pi_k \cdot N(x_i | z_k)}{\sum_{k} \pi_k \cdot N(x_i | z_k)} = \frac{p(z_k) \cdot p(x_i | z_k)}{\sum_{k} p(z_k) \cdot p(x_i | z_k)} \]

Unknown parameters can be estimated by EM.
```python
import numpy as np
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn import metrics

# Data Preparation
iris = datasets.load_iris()
X = iris.data[:, np.newaxis, 2]
X_train = X[:-20]
X_test = X[-20:]
y_train = iris.target[:-20]
y_test = iris.target[-20:]

# model construction
model = KMeans(n_clusters=3)

# model training
model.fit(X_train)

# model evaluation
y_pred = model.predict(X_train)
ARI = metrics.adjusted_rand_score(y_train, y_pred) # ignore permutation
MI = metrics.adjusted_mutual_info_score(y_train, y_pred)
print([ARI, MI])
```
```python
import numpy as np
from sklearn import datasets
from sklearn.cluster import DBSCAN
from sklearn import metrics

# Data Preparation
iris = datasets.load_iris()
X = iris.data[:, np.newaxis, 2]
X_train = X[:-20]
X_test = X[-20:]
y_train = iris.target[:-20]
y_test = iris.target[-20:]

# model construction
model = DBSCAN(eps=0.3, min_samples=10)

# model training
model.fit(X_train)

# model evaluation
y_pred = model.labels_
ARI = metrics.adjusted_rand_score(y_train, y_pred)  # ignore permutation
MI = metrics.adjusted_mutual_info_score(y_train, y_pred)
print([ARI, MI])
```
GMM in Python

```python
import numpy as np
from sklearn import datasets
from sklearn import mixture
from sklearn import metrics

# Data Preparation
iris = datasets.load_iris()
X = iris.data[:, np.newaxis, 2]
X_train = X[:-20]
X_test = X[-20:]
y_train = iris.target[:-20]
y_test = iris.target[-20:]

# model construction
model = mixture.GaussianMixture(n_components=3)

# model training
model.fit(X_train)

# model evaluation
y_pred = model.predict(X_train)
ARI = metrics.adjusted_rand_score(y_train, y_pred) # ignore permutation
MI = metrics.adjusted_mutual_info_score(y_train, y_pred)
print([ARI, MI])
```
Clustering and Dimensionality Reduction

Clustering
- Kmeans
- DBSCAN
- Gaussian Mixture Model

Dimensionality Reduction
- principal component analysis
- manifold learning

Other Feature Processing

Learning Theory
Concept: Projection

We can project data from one space $\mathbb{R}^p$ to another space $\mathbb{R}^r$ by a mapping function $W$.

- $W(x)$ is the mapping of $x$ in the new space

If $W$ is linear, it is a matrix $W \in \mathbb{R}^{p \times r}$.

- $W(x) = W^T x$

Dimensionality reduction finds optimal $W$.

- many criterions and techniques
Example of Linear Projection

\[ x = \begin{pmatrix} \text{google} \\ \text{lottery} \\ \text{cat} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \]

\[ w_1 = \begin{pmatrix} 0.3 \\ 0.1 \\ 0.7 \end{pmatrix} \]

\[ w_2 = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.1 \end{pmatrix} \]

\[ w_1^T x = \begin{pmatrix} 0.3, 0.1, 0.7 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 0.3 + 0.1 + 0 = 0.4 \]

\[ w_2^T x = \begin{pmatrix} 0.5, 0.2, 0.1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 0.5 + 0.2 + 0 = 0.7 \]

\[ W = [w_1, w_2] \]

\[ W^T x = \begin{pmatrix} 0.4 \\ 0.7 \end{pmatrix} \]
Linear Projection for Data Visualization

1st new feature $w_1^{T^*x}$

2nd new feature $w_2^{T^*x}$
Reconstruction from the Projected Space

\[
x = \begin{pmatrix}
\text{google} \\
\text{lottery} \\
\text{cat}
\end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}
\]

\[
W^T x = \begin{pmatrix} 0.4 \\ 0.7 \end{pmatrix}
\]

\[
x' = \begin{pmatrix} 0.8 \\ 1.1 \\ 0.1 \end{pmatrix}
\]

\[
\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0.8 \\ 1.1 \\ 0.1 \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.1 \\ -0.1 \end{pmatrix}
\]

reconstruction error = \(0.2^2 + (0.1)^2 + (-0.1)^2 = 0.06\)
1. Principal Component Analysis (PCA)

Learn a linear projection where data variance is maximally preserved.

Can be applied to

- compress data
- data visualization (exploratory data analysis)
- anomaly detection
Example of Eigenface
Kernel Principal Component Analysis (KPCA)
```python
import numpy as np
from sklearn import datasets
from sklearn.decomposition import PCA

# Data Preparation
iris = datasets.load_iris()
X = iris.data
X_train = X[:-20]
X_test = X[-20:]
y_train = iris.target[:-20]
y_test = iris.target[-20:]

# model construction
model = PCA(n_components=2)

# model training
model.fit(X_train)

# apply model to reduce dimension
X_pca = model.transform(X_train)

# reconstruct original data from the reduced dimension
X_train_reconstructed = model.inverse_transform(X_pca)

# evaluate reconstruction error (total/average squared loss)
loss = np.sum((X_train_reconstructed - X_train)**2, axis=1)
print(np.mean(loss))
```
```python
import numpy as np
from sklearn import datasets
from sklearn.decomposition import KernelPCA

# Data Preparation
iris = datasets.load_iris()
X = iris.data
X_train = X[:-20]
X_test = X[-20:]
y_train = iris.target[:-20]
y_test = iris.target[-20:]

# model construction
model = KernelPCA(kernel="rbf", gamma=1, fit_inverse_transform=True)

# model training
model.fit(X_train)

# apply model to reduce dimension
X_kpca = model.transform(X_train)

# reconstruct original data from the reduced dimension
X_train_reconstructed = model.inverse_transform(X_kpca)

# evaluate reconstruction error (total/average squared loss)
loss = np.sum((X_train_reconstructed - X_train)**2,axis=1)
print(np.mean(loss))
```
2. Manifold Learning

Learn a projection where data manifold structure is maximally preserved.

Local Linearity
import numpy as np
from sklearn import datasets
from sklearn.manifold import locally_linear_embedding

# Data Preparation
iris = datasets.load_iris()
X = iris.data
X_train = X[:-20]
X_test = X[-20:]
y_train = iris.target[:-20]
y_test = iris.target[-20:]

# reduced sample and reconstruction error
X_lle, err = locally_linear_embedding(X_train, n_neighbors=12, n_components=2)
Clustering and Dimensionality Reduction

Clustering
- Kmeans
- DBSCAN
- Gaussian Mixture Model

Dimensionality Reduction
- principal component analysis
- manifold learning

Other Feature Processing

Learning Theory
1. Feature Selection

Feature Selection

- combinatorial method (NP-hard)
- greedy method
  - backward stepwise
  - forward stepwise
- Lasso

<table>
<thead>
<tr>
<th>case ID</th>
<th>predictors</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_ID</td>
<td>CUST_GENDER</td>
<td>EDUCATION</td>
</tr>
<tr>
<td>101501</td>
<td>F</td>
<td>Masters</td>
</tr>
<tr>
<td>101502</td>
<td>M</td>
<td>Bach.</td>
</tr>
<tr>
<td>101503</td>
<td>F</td>
<td>HS-grad</td>
</tr>
<tr>
<td>101504</td>
<td>M</td>
<td>Bach.</td>
</tr>
<tr>
<td>101505</td>
<td>M</td>
<td>Masters</td>
</tr>
<tr>
<td>101506</td>
<td>M</td>
<td>HS-grad</td>
</tr>
<tr>
<td>101507</td>
<td>M</td>
<td>&lt; Bach.</td>
</tr>
<tr>
<td>101508</td>
<td>M</td>
<td>HS-grad</td>
</tr>
<tr>
<td>101509</td>
<td>M</td>
<td>Bach.</td>
</tr>
<tr>
<td>101510</td>
<td>M</td>
<td>Bach.</td>
</tr>
</tbody>
</table>
2. Categorical Data Encoding

Encode categorical data using dummy variables.

<table>
<thead>
<tr>
<th>EDUCATION</th>
<th>Naive</th>
<th>Dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masters</td>
<td>0</td>
<td>0001</td>
</tr>
<tr>
<td>Bach.</td>
<td>1</td>
<td>0010</td>
</tr>
<tr>
<td>HS-grad</td>
<td>2</td>
<td>0100</td>
</tr>
<tr>
<td>&lt; Bach.</td>
<td>3</td>
<td>1000</td>
</tr>
</tbody>
</table>
3. Missing Data Imputation

**Mean Imputation**: replace a missing value by the mean of observed ones.

- e.g., “?” = 34.5
- many variants

Other Techniques

- low-rank matrix factorization
- prediction
4. Data Normalization

Putting all features in the same scale may improve learning performance.

- mini-max normalization

\[ z = \frac{x - \text{min}(x)}{\text{max}(x) - \text{min}(x)} \]

- z-score normalization

\[ z = \frac{x_i - \mu}{\sigma} \]

<table>
<thead>
<tr>
<th>Fringe diameter (μm)</th>
<th>Fringes gap (μm)</th>
<th>Inlet flow velocity (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>50</td>
<td>0.01</td>
</tr>
<tr>
<td>1000</td>
<td>500</td>
<td>0.1</td>
</tr>
<tr>
<td>2000</td>
<td>1000</td>
<td>0.2</td>
</tr>
<tr>
<td>5000</td>
<td>2500</td>
<td>0.5</td>
</tr>
<tr>
<td>10000</td>
<td>5000</td>
<td>1</td>
</tr>
</tbody>
</table>
The Mini-Max Normalization Technique

\[ z = \frac{x - \text{min}(x)}{\text{max}(x) - \text{min}(x)} \]

Fringe Diameter

max(x) = 10000
min(x) = 100
max(x) - min(x) = 9900

<table>
<thead>
<tr>
<th>Fringe diameter (μm)</th>
<th>Fringe diameter (μm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>(100-100) / 9900 = 0</td>
</tr>
<tr>
<td>1000</td>
<td>(1000-100) / 9900 = 0.09</td>
</tr>
<tr>
<td>2000</td>
<td>(2000-100) / 9900 = 0.19</td>
</tr>
<tr>
<td>5000</td>
<td>(5000-100) / 9900 = 0.49</td>
</tr>
<tr>
<td>10000</td>
<td>(10000-100) / 9900 = 1</td>
</tr>
</tbody>
</table>
The Z-Score Normalization Technique

\[ z = \frac{x_i - \mu}{\sigma} \]

**Fringe Diameter**

\( \mu = 3620 \)

\( \sigma = 3591 \)

<table>
<thead>
<tr>
<th>Fringe diameter ((\mu m))</th>
<th>100</th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(100-3620) / 3591 = -0.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000-3620) / 3591 = -0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2000-3620) / 3591 = -0.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5000-3620) / 3591 = 0.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(10000-3620) / 3591 = 1.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Geometric Interpretation of Z-Score Normalization

$$z = \frac{x_i - \mu}{\sigma}$$

original data  zero-centered data  normalized data
Equalization can degrade clustering performance
Clustering and Dimensionality Reduction

Clustering
- Kmeans
- DBSCAN
- Gaussian Mixture Model

Dimensionality Reduction
- principal component analysis
- manifold learning

Other Feature Processing

Learning Theory
We are interested in the generalization performance of a prediction model $f$.

- $f$ is trained from a sample $S$
- $f$ is expected to generalize well on the population $D$
- what’s the relation between $\text{err}(f)$ and $\text{err}_S(f)$?

\[
\text{err}(f) = \Pr_{x \sim D}[f(x) \neq f_*(x)] = E_{x \sim D}[1_{f(x) \neq f_*(x)}]
\]

\[
\hat{\text{err}}_S(f) = \frac{1}{|S|} \sum_{x \in S} 1_{f(x) \neq f_*(x)}
\]
Relation between $\text{er}(f)$ and $\text{er}_S(f)$

If sample is i.i.d., then the expectation of empirical error equals the generalization error.

$$E_{S \sim D^m}[\hat{er}(f)] = er(f)$$

$$er(f) = \Pr_{x \sim D}[f(x) \neq f_*(x)] = E_{x \sim D}[1_{f(x) \neq f_*(x)}]$$

$$\hat{er}_S(f) = \frac{1}{|S|} \sum_{x \in S} 1_{f(x) \neq f_*(x)}$$
Relation between $\text{er}(f)$ and $\text{er}_S(f)$

PAC guarantee on the sample complexity of $f$ in the consistent case.

**Theorem 1.** Let $H : X \to Y$ be a finite set of hypotheses mapping from population $X$ to a binary label set $Y$. Let $S \subseteq X$ be an i.i.d. sample of size $m$, Let $f \in H$ be any hypothesis consistent on $S$. Then, for any $\epsilon, \delta$, there is $\text{er}(f) \leq \epsilon$ if

$$m \geq \frac{1}{\epsilon} \left( \log |H| + \log \frac{1}{\delta} \right).$$

(4)

Above admits the following result: with probability at least $1 - \delta$, there is

$$\text{er}(f) \leq \frac{1}{m} \left( \log |H| + \log \frac{1}{\delta} \right).$$

(5)
Relation between $\text{er}(f)$ and $\text{er}_S(f)$

PAC guarantee on the **sample complexity** of $f$ in the **inconsistent case**.

**Theorem 4.** Let $H : X \rightarrow \{0, 1\}$ be a finite hypothesis space, and $S \subseteq X$ be an i.i.d. sample of size $m$. Then for any $\delta > 0$, with probability at least $1 - \delta$, all $f \in H$ satisfy

$$\text{er}(f) \leq \text{er}_S(f) + \sqrt{\frac{\log |H| + \log \frac{2}{\delta}}{2m}}.$$  \hspace{1cm} (18)

\[
\text{er}(f) = \Pr_{x \sim D}[f(x) \neq f_*(x)] = E_{x \sim D}[1_{f(x) \neq f_*(x)}]
\]

\[
\text{er}_S(f) = \frac{1}{|S|} \sum_{x \in S} 1_{f(x) \neq f_*(x)}
\]
VC Theory

We say a set of examples $S$ can be shattered by a set of models/hypotheses $H$ if every label assignment of $S$ can be implemented by some hypothesis in $H$. 
VC Theory

The VC dimension of H is the largest size of any data set S which can be shattered by H.
VC Theory

The VC dimension of H is the largest size of any data set S which can be shattered by H.

- larger VC dimension implies more complex model
VC Theory

The **VC dimension** of H is the largest size of *any* data set S which can be shattered by H.

- larger VC dimension implies more complex model
- VC-Dim of linear classifier is p+1
Relation between $\text{er}(f)$ and $\text{er}_S(f)$ with VC Theory

The **VC dimension** of $H$ is the largest size of *any* data set $S$ which can be shattered by $H$.

**Theorem 5.** Let $H : X \to \{0, 1\}$ be a hypothesis space with VC dimension $d$, and $S \subseteq X$ be an i.i.d. sample of size $m$. Then for any $\delta > 0$, with probability at least $1 - \delta$, all $f \in H$ satisfy

$$
\text{er}(f) \leq \hat{e}_S(f) + \sqrt{\frac{2d \log \frac{em}{d}}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.
$$

(24)

\[\text{er}(f) = \Pr_{x \sim D}[f(x) \neq f_*(x)] = E_{x \sim D}[\mathbb{1}_{f(x) \neq f_*(x)}]\]

\[\hat{e}_S(f) = \frac{1}{|S|} \sum_{x \in S} \mathbb{1}_{f(x) \neq f_*(x)}\]