Supervised learning

Ziyun Zhu
Outline

• Introduction
• Classification
  – SVM
  – Evaluation
• Implementation
Introduction

• Inferring a function from \textit{labeled} training data:
  – Training set \((X, y)\)
  – Model \(y' = f(X ; p)\)
  – Criterion \(c(y, y')\)
Introduction

• Training set \((X, y)\)
  – If \(y\) is unknown \(\rightarrow\) unsupervised learning
  – If \(y\) is binary \(\rightarrow\) detection
  – If \(y\) is categorical \(\rightarrow\) classification
  – If \(y\) is continuous \(\rightarrow\) regression
Tools

• Language
  – Python:
  – R
Example

• Point fitting
  – Labeled data: \( (x, y) \)
  – System training:
    • Model: Linear
    • Criterion: Least square
Example

• Point fitting
Example

• Truth
  - \( y = \exp\{x\} + n \)
Classification

• Techniques
  – Logistic regression
  – Naïve Bayes
  – SVM
  – Decision tree
SVM

• Support Vector Machine
• Intuition
  – Training data
    • A point in feature space
  – Classifier
    • hyperplane
Example

- Two groups (randomly generated)
  - X1
    - Y=0: N(0, 0.5)
    - Y=1: N(1, 0.5)
  - X2
    - U(0, 1)
Example

- Hyperplane
  - Truth: $X_1 = 0.5$
  - Estimator:
    - Slope = 10
Evaluation

• Metric
  – True positive rate \( TP/(TP+FN) \) (Recall)
  – False positive rate \( FP/(FP+TN) \)
  – Precision \( TP/(TP+FP) \)

<table>
<thead>
<tr>
<th></th>
<th>TRUE (Truth)</th>
<th>FALSE (Truth)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE (Detector)</td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>FALSE (Detector)</td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>
Evaluation

• K-fold Cross validation
  – Partition data randomly to k subsamples
  – Training data: k-1 subsamples
  – Testing data: 1 subsample
Implementation

• Steps
  1. Select model
  2. Select features
  3. Train the model
  4. Evaluate the system
Model selection

- SVM
  ```python
  >>> from sklearn import svm
  >>> X = [[0, 0], [1, 1]]
  >>> y = [0, 1]
  >>> clf = svm.SVC()
  >>> clf.fit(X, y)
  ```

- Naïve bayes
  ```python
  >>> from sklearn.naive_bayes import GaussianNB
  >>> clf = GaussianNB()
  >>> clf.fit(X, Y)
  ```

- Decision tree
  ```python
  >>> from sklearn import tree
  >>> X = [[0, 0], [1, 1]]
  >>> Y = [0, 1]
  >>> clf = tree.DecisionTreeClassifier()
  >>> clf = clf.fit(X, Y)
  ```
Feature Selection

- Remove features with low variance

```python
from sklearn.feature_selection import VarianceThreshold
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
sel.fit_transform(X)
```

### Evaluation

```python
>>> from sklearn import metrics
>>> scores = cross_validation.cross_val_score(clf, iris.data, iris.target,
                                           cv=5, scoring='f1_weighted')
```

<table>
<thead>
<tr>
<th>Scoring</th>
<th>Function</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'accuracy'</td>
<td><code>metrics.accuracy_score</code></td>
<td></td>
</tr>
<tr>
<td>'average_precision'</td>
<td><code>metrics.average_precision_score</code></td>
<td></td>
</tr>
<tr>
<td>'f1'</td>
<td><code>metrics.f1_score</code></td>
<td>for binary targets</td>
</tr>
<tr>
<td>'f1_micro'</td>
<td><code>metrics.f1_score</code></td>
<td>micro-averaged</td>
</tr>
<tr>
<td>'f1_macro'</td>
<td><code>metrics.f1_score</code></td>
<td>macro-averaged</td>
</tr>
<tr>
<td>'f1_weighted'</td>
<td><code>metrics.f1_score</code></td>
<td>weighted average</td>
</tr>
<tr>
<td>'f1_samples'</td>
<td><code>metrics.f1_score</code></td>
<td>by multilabel sample</td>
</tr>
<tr>
<td>'log_loss'</td>
<td><code>metrics.log_loss</code></td>
<td>requires predict_proba support</td>
</tr>
<tr>
<td>'precision' etc.</td>
<td><code>metrics.precision_score</code></td>
<td>suffixes apply as with 'f1'</td>
</tr>
<tr>
<td>'recall' etc.</td>
<td><code>metrics.recall_score</code></td>
<td>suffixes apply as with 'f1'</td>
</tr>
<tr>
<td>'roc_auc'</td>
<td><code>metrics.roc_auc_score</code></td>
<td></td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'adjusted_rand_score'</td>
<td><code>metrics.adjusted_rand_score</code></td>
<td></td>
</tr>
<tr>
<td><strong>Regression</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'mean_absolute_error'</td>
<td><code>metrics.mean_absolute_error</code></td>
<td></td>
</tr>
<tr>
<td>'mean_squared_error'</td>
<td><code>metrics.mean_squared_error</code></td>
<td></td>
</tr>
<tr>
<td>'median_absolute_error'</td>
<td><code>metrics.median_absolute_error</code></td>
<td></td>
</tr>
<tr>
<td>'r2'</td>
<td><code>metrics.r2_score</code></td>
<td></td>
</tr>
</tbody>
</table>
Clustering Algorithms

Yantao Zhang
Topics

• K-means clustering
• Hierarchical Clustering
• Evaluation Techniques
• Assignment
Motivation

• Grouping data into clusters
  – High intra-cluster similarity
  – Low inter-cluster similarity
Distance Measure

- Euclidean Distance
- Manhattan Distance
- Cosine Distance
- Jaccard Distance
K-Means

• Split point set S into k groups S1, S2,..Sk such that:
  – Each bucket Si has a center ci
  – $Cost(S_1, \ldots, S_k) = \sum_{i=1}^{k} \sum_{x \in S_i} (x - c_i)^2$

• The cost should be minimized. NP-hard Problem.
Heuristic K-Means

• Split S into k groups S1, ..., Sk
• Compute mean mi of each cluster
• For each element s in S
  – Find group Si such that distance between mi and s is minimized.
  – Put s in the group.
• Repeat the steps until Converge
Kmeans in Sklearn

• Build kmeans model
  – sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1)

• Predict Labels
  – fit_predict(X[, y])
  – get the cluster index for each sample
Hierarchical Clustering

• Algorithm
  – Place each data point into its singleton group
  – Repeat: iteratively merge the two closest groups
  – Until: all the data are merged in a single cluster

• Linkage
  – Average: Use the average distance for cluster distance
  – Complete: Use the maximum distance for cluster distance
Hierarchical Clustering in Sklearn

• Build the model
  – Sklearn. cluster. AgglomerativeClustering
    (n_clusters=2, affinity='euclidean', memory=Memory(cachedir=None), connectivity=None, n_components=None, compute_full_tree='auto', linkage = 'ward')

• Predict Labels
Hierarchical Clustering Parameters

• Affinity : Distance Metrics
  – L1, L2, Euclidean, cosine, etc
  – Reference: 
    sklearn.metrics.pairwise_distances.html%23sklearn.metrics.pairwise_distances

• Linkage
  – Complete
  – Average
Evaluation
Optimal Number of Clusters

• Silhouette Score

\[ s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \]

• For each data i

  – a(i) is the average dissimilarity of i with all other data in the same cluster  =>  Compactness

  – b(i) is the lowest average dissimilarity of i to any other cluster  =>  Separation
Evaluation Example

Silhouette analysis for KMeans clustering on sample data with n_clusters = 2

The silhouette plot for the various clusters.

The visualization of the clustered data.
Evaluation Example

Silhouette analysis for KMeans clustering on sample data with $n_{clusters} = 3$

The silhouette plot for the various clusters.

The visualization of the clustered data.

Cluster label

The silhouette coefficient values

Feature space for the 2nd feature

Feature space for the 1st feature
Evaluation Example

Silhouette analysis for KMeans clustering on sample data with n_clusters = 4

The silhouette plot for the various clusters.

The visualization of the clustered data.

Cluster label

-0.1 0.0 0.2 0.4 0.6 0.8 1.0

The silhouette coefficient values

-14 -12 -10 -8 -6 -4 -2 0 2

Feature space for the 1st feature

Feature space for the 2nd feature
Code Samples

• Sklearn Library: [http://scikit-learn.org/](http://scikit-learn.org/)
  – KMeans located under sklearn.cluster
  – Agglomerative Clustering under sklearn.cluster
  – Distance Metrics sklearn.metrics.pairwise distance
  – TFIDF
    sklearn.feature_extraction.text.TfidfTransformer
  – Silhouette Score
Assignment

• Data Description
  – Sample of 3000 machines
  – One week sample period
  – Usage information of top 10 used applications (numerical data)
  – Machines all have at least one crash during sample period
Assignment

• Intuition
  – Try to find if certain applications are used more on specific set of machines.
  – Help to perform other machine learning algorithms in different groups.
  – Can the usage lead to crashes?
Assignment

• Tasks
  – Compute TFIDF
  – Kmeans clustering
    • TF-IDF
  – Agglomerative clustering
    • TF-IDF
  – Evaluation the clustering
    • Silhouette Score
Term Frequency Inverse Document Frequency (TF-IDF)

• A list of documents and a list of words in those documents.
• TF-IDF = TF x IDF
• TF: the frequency of a term $t$ in a document $d$. (Ex. Raw frequency, boolean frequency)
• IDF: the frequency of a term $t$ across all documents

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad N = |D|$$
Assignment

• Files Available
  – Error_Machines.csv: The first column is the number of errors in the sampling period and the other columns are the number of uses of application in a sample period.
  – clustering_exercise.py: The program for kMeans and Hierarchical clustering.