SEMANTIC COMPUTING

Lecture 5: Supervised and Semi-Supervised Machine Learning

Dagmar Gromann
International Center For Computational Logic

TU Dresden, 28 May 2018
Overview

- Supervised Machine Learning algorithms
- Support Vector Machines (SVM)
- Decision Trees
- Paradigm of Semi-Supervised Machine Learning
Supervised Machine Learning algorithms
Supervised Machine Learning algorithms

- Naive Bayes for classification problems
- Linear regression for regression problems
- Support vector machines for classification and regression problems
- Decision trees for classification and regression problems
- Random forest for classification and regression problems
Definition

SVM is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. The optimal separating hyperplane maximizes the distance to the nearest training points (called the margin). SVMs allow you to use multiple different kernels to find non-linear decision boundaries.

Why maximize margin to training examples?
Because it increases the robustness of the classification algorithm. This means it is more resistant to noise in the data.
Kernel

**Definition**

A kernel is a mathematical function of distance that is used to determine the weight of each training example.

**Linear**  |  **Polynomial**  |  **Radial Basis Function (RBF)**

Kernel

Since SVM basically linearly separates data how can it create decision boundaries such as the two non-linear above?

Answer
Using a mathematical transformation, SVMs move the original data set into a new (usually high dimensional) mathematical space in which the decision boundary is easy to describe/linear.
Transformation

Data projected to $\mathbb{R}^2$ (nonseparable)

Data in $\mathbb{R}^3$ (separable)
The Kernel Trick

We do not have to manually add features but instead select the kernel function for the SVM that does the “kernel trick”.

**Kernel trick**

Instead of actually mapping the points into the higher-dimensional space (which could be expensive if there are a lot of points and the mapping is complicated), we can use a kernel function to compute dot products in the higher-dimensional space and use those to find a hyperplane.
Types of Kernels

Each type of kernel is based on a specific kernel function $k(x, x')$ that computes the inner product of two projected vectors

- **Linear**: provides a linear separator with the 'normal' dot product; $k(x, x')$

- **Poly**: kernel induces a space of a polynomial combination of our features up to a certain degree (a parameter in the SVM implementation of scikit learn); $k(x, x') = (\gamma(x, x') + r)^d$

- **Radial Basis Function**: induces space of Gaussian distributions; it calculates the squared Euclidean distance with a kernel coefficient gamma to generate radial areas around training points; $k(x, x') = (\exp(-\gamma||x - x'||^2)$
Parameters of SVMs

- **C**: penalty parameter C of the error term
  - tradeoff between how smooth the decision boundary is and how well it classifies examples
  - default = 1.0
  - large C = smoother boundary or more points classified correctly?

- **Gamma**: kernel coefficient for rbf, poly, and sigmoid; defines the influence of a single training example

```python
import numpy as np
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
from sklearn.svm import SVC
clf = SVC(C=1.0, gamma='auto', kernel='rbf')
clf.fit(X, y)
print(clf.predict([[-0.8, -1]]))  # Output: [1]
```


Dagmar Gromann, 28 May 2018  Semantic Computing
(Dis)Advantages of SMV

Advantages:

• effective in complicated domains with a clear margin of separation between classes
• memory efficient: uses a subset of training points in the decision function (called support vectors)
• flexible: different kernel functions can be specified

Disadvantages:

• do not work well with lots of noise
• slow and prone to overfitting on very large datasets with many features
• do not directly provide probability estimates (expensive to calculate, e.g. using 5-fold cross validation)
Decision Trees

**Definition**

Decision Tree learning is one of the most widely used methods for inductive inference; it is a method to approximate discrete-valued target functions that is robust to noise and capable of learning disjunctive expressions (represent a disjunction of conjunctions on the constraints of the attribute values of instances).
Decision Tree example

Decision Tree for “Playing Tennis on Saturday”

(Outlook = Sunny \land Humidity = Normal)
\lor (Outlook = Overcast)
\lor (Outlook = Rain \land Wind = Weak)


Dagmar Gromann, 28 May 2018  Semantic Computing 14
Parameters of Decision Trees

- **min_sample_split**: the minimum number of samples required to split an internal node (default = 2)
- **criterion**: The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

```python
class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

Entropy

Definition

Entropy “characterizes the (im)purity of an arbitrary collection of examples.” It measures the homogeneity of examples.


- **S**: collection of training examples
- **p⁺**: proportion of positive examples in S
- **p⁻**: proportion of negative examples in S

\[
Entropy(S) = H(S) = -p^+ \log_2 p^+ - p^- \log_2 p^-
\]

14 examples, 9+, 5- => H(S)?
Entropy Overview by Proportion of Positive Examples

Entropy Beyond Binary Classification

The above formula only applies to cases where the task at hand is a binary classification with two potential results/classes for the classifier. In all other cases the following formula applies:

\[ H(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \]

Where \( c \) is the number of target classes and \( p_i \) is the proportion of \( S \) belonging to class \( i \).
Information Gain

**Definition**

Information gain “measures how well a given attribute separates the training examples according to their target classification. ... It is the expected reduction in entropy caused by partitioning examples according to this attribute.”


\[
Gain(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)
\]

*Values*(A) is the set of all possible values for attribute A

*S\_v* is the subset of S for which Attribute A has value v
Bias-Variance Tradeoff

Necessity to simultaneously minimize two sources of errors that prevents the supervised algorithm from generalizing beyond its training sets.

- **Bias**: High bias (prior assumption) can cause an algorithm to miss the relevant relations between features and target outputs (underfitting)
- **Variance**: error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs. The smaller the test set, the greater the expected variance. (overfitting)

One way to minimize both: Random Forests = a collection of decision trees whose results are aggregated into one final result (ensemble methods)
(Dis)Advantages of Decision Trees

Advantages:

• simple to understand and interpret (easy to visualize)
• white box model: decision can be explained in boolean logic
• good at handling multi-output problems

Disadvantages:

• prone to overfitting (especially with a large number of features)
• unstable to small variations in the data
• easy to produce biased trees if some class dominates (necessary to balance the dataset prior to fitting)
Paradigm of Semi-Supervised Machine Learning
Motivation: Semi-Supervised Learning

- labeling data manually can be time-consuming
- manual labeling can introduce an undetected human bias
- unlabeled data are easy to obtain
- train on a combination of labeled and unlabeled data: semi-supervised or weakly supervised machine learning
- can be used on classification, regression, and clustering
Example of Semi-Labeled Data

Example fraud detection - for some cases the bank does not know for others data are available:

<table>
<thead>
<tr>
<th>Name</th>
<th>Loan Amount</th>
<th>Loan Repaid</th>
<th>Fraud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Donna</td>
<td>50.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anton</td>
<td>90.000</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Debby</td>
<td>1.000.000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Sara</td>
<td>10.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Martin</td>
<td>90.000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Anna</td>
<td>1.000.000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Semi-supervised machine learning is used to label the unlabeled data and then the algorithm is retrained on the fully labeled data, turning it into a supervised algorithm.

Dagmar Gromann, 28 May 2018
Algorithms

- **Self-training methods**: use the labeled data to train an initial model, the use that model to label the unlabeled data and retrain a new model.
- **Co-training**: two classifiers are used that have to correctly classify labeled examples and must agree on the classification for unlabeled examples.
- **Graph regularization methods**: build models based on a graph on instances, where edges in the graph indicate similarity. The regularization constraint is one of smoothness along this graph.
- **Structural learning**: uses unlabeled data to find a new, reduced-complexity hypothesis space by exploring regularities in feature space.
Example Algorithm: Semi-Supervised SVMs (S3VMs)

Maximizes unlabeled data margin (place the decision boundary in a way that there are few unlabeled data near it). Intuitively unlabeled data guide the decision boundary away from high-density areas.
(Dis)Advantages of Semi-Supervised Learning

Advantages:

• simple methods can be used, such as Naïve Bayes (using class probability values for each data instance)
• often used in real tasks for NLP

Disadvantages:

• no means to find out correctness of predictions on unlabeled data
• errors in the first training stages propagate to errors in final predictions on new test set/previously unseen examples

Review of Lecture 6

- What is supervised machine learning?
- Which supervised algorithms do you know?
- How do Support Vector Machines work?
- What is a kernel and what is the kernel trick?
- What are Decision Trees? What are their main advantages?
- What is entropy and how does it relate to information gain?
- What is the bias-variance tradeoff?
- How does supervised learning differ from semi-supervised learning?