Machine Learning Summary

Connectionist and Statistical Language Processing

Frank Keller

keller@coli.uni-sb.de

Computerlinguistik

Universität des Saarlandes

Part I: Core Concepts

Types of Learning

These are the main machine learning problems:

- *Classification:* learn to put instances into pre-defined classes.
- *Association:* learn relationships between attributes.
- *Numeric prediction:* learn to predict a numeric quantity instead of a class.
- *Clustering:* discover classes of instances that belong together.

Clustering vs. Classification

Classification: the task is to learn to assign instances to predefined classes.

Clustering: no predefined classification is required. The task is to learn a classification from the data.

Clustering algorithms divide a data set into *natural groups* (clusters). Instances in the same cluster are *similar* to each other, they share certain properties.

Supervised vs. Unsupervised Learning

Supervised learning: classification requires supervised learning, i.e., the training data has to specify what we are trying to learn (the classes).

Unsupervised learning: clustering is an unsupervised task, i.e., the training data doesn't specify what we are trying to learn (the clusters).

Learning Bias

To generalize successfully, a machine learning system uses a *learning bias* to guide it through the space of possible concepts. Also called *inductive bias* (Mitchell 1997).

Language bias: the language in which the result is expressed determines which concepts can be learned.

Search bias: the way the space of possible concepts is searched determines the outcome of learning.

Overfitting-avoidance bias: avoid learning a concept that overfits, i.e., just enumerates the training data: this will give very bad results on test data, as it lacks the ability to generalized to unseen instances.

Part II: Evaluation

Training and Test Set

For classification problems, we measure the performance of a model in terms of its *error rate:* percentage of incorrectly classified instances in the data set.

We build a model because we want to use it to classify new data. Hence we are chiefly interested in model performance on new (unseen) data.

The *resubstitution error* (error rate on the training set) is a bad predictor of performance on new data.

The model was build to account for the training data, so might *overfit* it, i.e., not generalize to unseen data.

Crossvalidation

Testing is done either on a hold-out part of the data or using *k-fold crossvalidation*:

- Divide data randomly into k folds (subsets) of equal size.
- Train the model on $k-1$ folds, use one fold for testing.
- Repeat this process k times so that all folds are used for testing.
- Compute the average performance on the k test sets.

This effectively uses all the data for both training and testing. Typically $k = 10$ is used.

Comparing against a Baseline

An error rate in itself is not very meaningful. We have to take into account how hard the problem is.

This means comparing against a *baseline model* and showing that our model performs significantly better than the baseline.

The simplest model is the *chance baseline*, which assigns a classification randomly.

Problem: a chance baseline is not useful if the distribution of the data is skewed.

We need to compare against a *frequency baseline* instead. A frequency baseline always assigns the most frequent class.

Precision and Recall

Measures commonly used in information retrieval, based on true positives, false positives, and false negatives:

Precision: number of class members classified correctly over total number of instances classied as class members.

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|} \tag{1}$$

Recall: number of class members classified correctly over total number of class members.

$$\text{Recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|} \tag{2}$$

Evaluating Clustering Models

Problem: How do we evaluate the performance on the test set? How do we know if the clusters are correct? Possible solutions:

- Test the resulting clusters *intuitively*, i.e., inspect them and see if they make sense. Not advisable.
- Have an *expert* generate clusters manually, and test the automatically generated ones against them.
- Test the clusters against a predefined *classification* if there is one.
- Perform *task-based evaluation*, i.e., test if the performance of some other algorithm can be improved by using the output of the clusterer.

Part III: Core Algorithms

ID3 Algorithm

Informal formulation of the ID3 alorithm for decision tree induction:

- Determine the attribute that has the highest information gain on the training set.
- Use this attribute as the root of the tree, create a branch for each of the values that the attribute can take.
- for each of the branches, repeat this process with the subset of the training set that is classified by this branch.

ID3 Algorithm: Information Gain

$$\text{Gain}(S,A) = E(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} E(S_v) \quad (3)$$

$\text{Values}(A)$: set of all possible values of attribute A
S_v: subset of S for which A has value v
$|S|$: size of S; $|S_v|$: size of S_v

The information gain $\text{Gain}(I,A)$ is the *expected reduction in entropy* caused by knowing the value of the attribute A.

Naive Bayes Classifier

Assumption: training set consists of instances described as **conjunctions of attributes values**, target classification based on finite set of classes V.

The task of the learner is to predict the correct class for a new instance $\langle a_1, a_2, \ldots, a_n \rangle$.

Key idea: assign most probable class v_{MAP} using Bayes Rule.

$$
\begin{aligned}
(4) \qquad v_{\text{MAP}} &= \arg\max_{v_j \in V} P(v_j | a_1, a_2, \ldots, a_n) \\
&= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\
&= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n | v_j) P(v_j)
\end{aligned}
$$

Naive Bayes Classifier

Estimating $P(v_j)$ is simple: compute the relative frequency of each target class in the training set.

Estimating $P(a_1, a_2, \ldots, a_n | v_j)$ is difficult: typically not enough instances for each attribute combination in the training set: *sparse data problem.*

Independence assumption: attribute values are conditionally independent given the target value: **naive** Bayes.

$$P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j) \tag{5}$$

Hence we get the following classifier:

$$v_{\mathrm{NB}} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) \tag{6}$$

Linear Regression

Linear regression is a technique for numeric predictions that's widely used in psychology, medical research, etc.

Key idea: find a linear equation that predicts the target value x from the attribute values $a_1, \ldots, a_k$:

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k \tag{7}$$

Here, $w_1, \ldots w_k$ are the *regression coefficients*, w_0 is called the *intercept*. These are the model parameters that need to be induced from the data set.

Linear Regression

The regression equation computes the following *predicted value* x'_i for the i-th instance in the data set.

$$(8) \qquad x'_i = w_0 + w_1 a_{1,i}, w_2 a_{2,i}, \ldots, w_k a_{k,i} = w_0 + \sum_{j=1}^{k} w_j a_{j,i}$$

Key idea: to determine the coefficients $w_0, \ldots w_k$, minimize e, the squared difference between the predicted and the actual value, summed over all n instances in the data set:

$$(9) \qquad e = \sum_{i=1}^{n} (x_i - x'_i)^2 = \sum_{i=1}^{n} \left(x_i - w_0 - \sum_{j=1}^{k} w_j a_{j,i} \right)^2$$

The method for this is called *Least Square Estimation* (LSE).

The k-means Algorithm

Iterative, hard, flat clustering algorithm based on Euclidian distance. Intuitive formulation:

- Specify k, the number of clusters to be generated.
- Chose k points at random as cluster centers.
- Assign each instance to its closest cluster center using Euclidian distance.
- Calculate the centroid (mean) for each cluster, use it as new cluster center.
- Reassign all instances to the closest cluster center.
- Iterate until the cluster centers don’t change any more.

The k-means Algorithm

Each instance $\vec{x}$ in the training set can be represented as a vector of n values, one for each attribute:

$$\vec{x} = (x_1, x_2, \ldots, x_n) \tag{10}$$

The Euclidian distance of two vectors $\vec{x}$ and $\vec{y}$ is defined as:

$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{11}$$

The mean $\vec{\mu}$ of a set of vectors c_j is defined as:

$$\vec{\mu} = \frac{1}{|c_j|} \sum_{\vec{x} \in c_j} \vec{x} \tag{12}$$

References

Howell, David C. 2002. *Statistical Methods for Psychology*. Pacific Grove, CA: Duxbury, 5th edn.

Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Mitchell, Tom. M. 1997. *Machine Learning*. New York: McGraw-Hill.

Witten, Ian H., and Eibe Frank. 2000. *Data Mining: Practical Machine Learing Tools and Techniques with Java Implementations*. San Diego, CA: Morgan Kaufmann.