Tutorial: Memory based learning with TiMBL
TiMBL

- TiMBL: Tilburg Memory Based Learner
- We follow TiMBL manual and examples
- Can be downloaded from http://ilk.uvt.nl/timbl
TiMBL

- Memory-Based Learning (MBL) is based on the idea that intelligent behavior can be obtained by analogical reasoning, rather than by the application of abstract mental rules as in rule induction and rule-based processing.

- In particular, MBL is founded in the hypothesis that the extrapolation of behavior from stored representations of earlier experience to new situations, based on the similarity of the old and the new situation, is of key importance.
TiMBL

- MBL algorithms take a set of examples (fixed-length patterns of feature-values and their associated class) as input, and produce a classifier which can classify new, previously unseen, input patterns.

- Can in principle be applied to any kind of classification task with symbolic or numeric features and discrete (non-continuous) classes for which training data is available.
TiMBL

Example task: prediction of Dutch diminutive suffixes (data set included in the TiMBL distribution).
Example task

- Diminutives are formed by a productive morphological rule which attaches a form of the Germanic suffix -tje to the singular base form of a noun.

- The task we consider here is to predict which suffix form is chosen for previously unseen nouns on the basis of their form.
### Table 4.1: Allomorphic variation in Dutch diminutives.

<table>
<thead>
<tr>
<th>Noun</th>
<th>Form</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>huis (house)</td>
<td>huisje</td>
<td>-je</td>
</tr>
<tr>
<td>man (man)</td>
<td>mannetje</td>
<td>-etje</td>
</tr>
<tr>
<td>raam (window)</td>
<td>raampje</td>
<td>-pje</td>
</tr>
<tr>
<td>woning (house)</td>
<td>woninkje</td>
<td>-kje</td>
</tr>
<tr>
<td>baan (job)</td>
<td>baantje</td>
<td>-tje</td>
</tr>
</tbody>
</table>

Our goal is to use TiMBL in order to train a classifier that can predict the class of new, previously unseen words as correctly as possible, given a set of training examples that describe the features given above. Because the basis of classification in TiMBL is the storage of all training examples in memory, a test of the classifier's accuracy must be done on a separate test set. We will call these datasets `dimin.train` and `dimin.test`, respectively. The training set `dimin.train` contains 2999 words and the test set contains 950 words, none of which are present in the training set. Although a single train/test partition suffices for the purpose of explanation, it does not factor out the bias of choosing this particular split. Unless the test set is sufficiently large, a more reliable generalization accuracy measurement is used in real experiments, e.g., 10-fold cross-validation (Weiss and Kulikowski, 1991). This means that 10 separate experiments are performed, and in each “fold” 90% of the data is used for training and 10% for testing, in such a way that each instance is used as a test item exactly once. Another reliable way of testing the real error of a classifier is leave-one-out (Weiss and Kulikowski, 1991). In this approach, every data item in turn is selected once as a test item, and the classifier is trained on all remaining items. Accuracy of the classifier is then the number of data items correctly predicted. With the option `-t leave-one-out`, this test methodology is used by TiMBL. We will use this option in the tutorial on the file `dimin.data`, the union of `dimin.train` and `dimin.test`.

### 4.2 Using TiMBL

Different formats are allowed for training and test data files. TiMBL is able to guess the type of format in most cases. We will use comma-separated values here, with the class as the last value. This format is called C4.5 format in TiMBL because it is the same as that used in Quinlan's well-known C4.5 program for induction of decision trees (Quinlan, 1993). See Section 6.2 for more information about this and other file formats.

An experiment is started by executing TiMBL with the two files (`dimin.train` and `dimin.test`)
Representation of nouns in terms of their syllable structure as training material.

For each of the last three syllables of the noun, four different features are collected:

- whether the syllable is stressed or not (values - or +),
- the string of consonants before the vocalic part of the syllable (i.e. its onset),
- its vocalic part (nucleus),
- and its post-vocalic part (coda).

Whenever a feature value is not present (e.g. a syllable does not have an onset, or the noun has less than three syllables), the value ‘=’ is used.

The class to be predicted is either E (-etje), T (-tje), J (-je), K (-kje), or P (-pje).
CHAPTER 4. QUICK-START TUTORIAL

Noun Form Suffix

huis (house) huisje -je
man (man) mannetje -etje
raam (window) raampje -pje
woning (house) woningkje -kje
baan (job) baantje -tje

Table 4.1: Allomorphic variation in Dutch diminutives.

collected: whether the syllable is stressed or not (values - or +), the string of consonants before the vocalic part of the syllable (i.e. its onset), its vocalic part (nucleus) and its post-vocalic part (coda). Whenever a feature value is not present (e.g. a syllable does not have an onset, or the noun has less than three syllables), the value '=' is used. These last ob e s p e d i c e s i t h e r E ( -etje ), T ( -tje ), J ( -je ), K ( -kje ), or P ( -pje ).

Some examples are given below (the word in the rightmost column is only provided for convenience and is not used). The values of the syllabic content features are given in phonetic notation.

| +bi | - | zm | Ant | J | biezenmand |
| = = | = | = | = | = | big |
| = = | = | +b | K | -b | an | T | bijbaan |
| = = | = | = | +b | K | -b | @ | l | T | bijbel |

Our goal is to use TiMBL in order to train a classifier that can predict the class of new, previously unseen words as correctly as possible, given a set of training examples that describe the features given above. Because the basis of classification in TiMBL is the storage of all training examples in memory, a test of the classifier’s accuracy must be done on a separate test set. We will call these datasets dimin.train and dimin.test, respectively. The training set dimin.train contains 2999 words and the test set contains 950 words, none of which are present in the training set. Although a single train/test partition suffices for our purpose so explanation, it does not factor out the bias of choosing this particular split. Unless the test set is sufficiently large, a more reliable generalization accuracy measurement is used in real experiments, e.g. 10-fold cross-validation (Weiss and Kulikowski, 1991). This means that 10 separate experiments are performed, and in each “fold” 90% of the data is used for training and 10% for testing, in such a way that each instance is used as a test item exactly once. Another reliable way of testing the real error of a classifier is leave-one-out (Weiss and Kulikowski, 1991). In this approach, every data item in turn is selected once as a test item, and the classifier is trained on all remaining items. Accuracy of the classifier is then the number of data items correctly predicted. With the option -t leave one out, this testing methodology is used by TiMBL. We will use this option in the tutorial on the file dimin.data, the union of dimin.train and dimin.test.

4.2 Using TiMBL

Different formats are allowed for training and test data files. TiMBL is able to guess the type of format in most cases. We will use comma-separated values here, with the class as the last value. This format is called C4.5 format in TiMBL because it is the same as that used in Quinlan’s well-known C4.5 program for induction of decision trees (Quinlan, 1993). See Section 6.2 for more information about this and other file formats.

An experiment is started by executing TiMBL with the two files (dimin.train and dimin.test)
Goal: train a classifier that can predict the class of new, previously unseen words as correctly as possible, given a set of training examples that are described by the features given above.

Because the basis of classification in TiMBL is the storage of all training examples in memory, a test of the classifier’s accuracy must be done on a separate test set.

- Training set: dimin.train (2999 words)

- Test set: dimin.test (950 words, not present in training set)

- In reality: 10-fold cross validation, leave-one-out, or wug testing

- Full set: dimin.data
Timbl -f dimin.train -t dimin.test

Timbl -f dimin.train -t dimin.test > dimin-exp1

-m parameter: choosing a distance metric

Timbl -mM -f dimin.train -t dimin.test

Timbl -mM -f dimin.train -t dimin.test

-k parameter: choosing a number of neighbors

Timbl -mM -k5 -f dimin.train -t dimin.test
Whenever more than one nearest neighbor is taken into account, it may be useful to weigh the influence of the neighbors on the final decision as a function of their distance from the test item. Several possible implementations of this distance function are provided. E.g., the following provides inverse distance:

Within the IB weighted overlap option, the default feature weighting method is gain ratio. Other feature relevance weighting methods are available as well. By setting the parameter `−w` to 0, an unweighted overlap definition of similarity is created where each feature is considered equally relevant. In that case, similarity reduces to the number of equal values in the same position in the two patterns being compared. As an alternative weighting, users can provide their own weights by using the `−w` parameter with a filename in which the feature weights are stored (see Section 6.2.2 for a description of the format of the weights file).

Table 4.2 shows a small matrix indicating the effect of distance metric (Overlap versus MVDM) and weighting method choice on generalization accuracy, using the same training and test set as before, and increasing $k$ from 1 to 3 and 5. While increasing $k$ leads to a deterioration of generalization accuracy with the Overlap function, it leads to improvements with MVDM. Another clear contrast is that the absence of feature weighting leads to the lowest scores with the Overlap function, and the highest score with MVDM and $k=5$. Given that TIMBL offers several more hyperparameters than only $k$, the distance metric, and the feature weighting metric, it should be obvious that even with a single training and test set experiment, a large experimental matrix can be explored. Unfortunately, the location of the cell with the highest number in this matrix cannot be predicted upfront. It is therefore useful to try out a large set of reasonable combinations of options by cross-validation on the training data to achieve best results with MBL (Van den Bosch, 2004b). The option `−t @f` where `f` is the name of a file, allows you to predefine various combinations of options to be tested and test them without having to retrain the stage repeatedly each time. See Chapter 6.1.

<table>
<thead>
<tr>
<th></th>
<th>no weight (overlap)</th>
<th>gain ratio</th>
<th>information gain</th>
<th>chi squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlap, $−k1$</td>
<td>86.4</td>
<td>96.8</td>
<td>96.7</td>
<td>96.7</td>
</tr>
<tr>
<td>Overlap, $−k3$</td>
<td>73.1</td>
<td>96.4</td>
<td>96.8</td>
<td>96.9</td>
</tr>
<tr>
<td>Overlap, $−k5$</td>
<td>62.6</td>
<td>95.4</td>
<td>96.1</td>
<td>96.1</td>
</tr>
<tr>
<td>MVDM, $−k1$</td>
<td>95.8</td>
<td>96.4</td>
<td>96.2</td>
<td>96.3</td>
</tr>
<tr>
<td>MVDM, $−k3$</td>
<td>97.3</td>
<td>97.6</td>
<td>97.6</td>
<td>97.6</td>
</tr>
<tr>
<td>MVDM, $−k5$</td>
<td><strong>97.8</strong></td>
<td>97.7</td>
<td>97.7</td>
<td>97.7</td>
</tr>
</tbody>
</table>
-d parameter: choosing a distance weighting function

Timbl -mM -k5 -dID -f dimin.train -t dimin.test
Ignoring features

Timbl -mM:l1-8,10 -k5 -w0 -f dimin.train -t dimin.test

(ignores everything but rime and stress of last syllable)
-v -v option: increasing and decreasing verbosity
  -v o: dumping option settings
  -v p: value-class conditional probabilities
  -v e: exact matches
  -v db: distributions
  -v cm: confusion matrix
  -v as: advanced statistics besides accuracy: micro-average and macro-average F-score and AUC
  -v cs: per-class advanced statistics
  -v n: the nearest neighbors on which decision are based
  -v k: just the class distributions per k-nearest distance per classified instance
  -v di: distances to the nearest neighbor
- Timbl +v db -f dimin.train -t dimin.test
- Timbl +v di -f dimin.train -t dimin.test
- Timbl +v cm -f dimin.train -t dimin.test
- Timbl +v as+cs -f dimin.train -t dimin.test