Tutorial 10: Connectionist and Statistical Language Processing

Linear Models

Date: 23.01.2002
Student's name:

1 The Task: Modeling Word Order Preferences

We will again use the Weka machine learning package in this tutorial. The files for the tutorial reside in:

`/courses/connectionism.winter.01/tutorial10/`

Your task is to build and evaluate a model of word order preferences in German. We will use two data sets that contain average grammaticality judgments of native speakers for different word order patterns (the data are based on Keller 2000).

In these data set, the following attributes are represented, each of which corresponds to a linguistic constraint on word order. An attribute takes as its value the number of times the constraint is violated in the corresponding structure.

(1) `VerbFinal`: the verb has to be in sentence final position.
(2) `NomAlign`: nominative NPs have to precede accusative NPs.
(3) `DatAlign`: dative NPs have to precede accusative NPs.
(4) `ProAlign`: pronouns have to precede full NPs.
(5) `GroundAlign`: ground NPs have to be in sentence final or sentence initial position.[1]

For each instance in the data set, the target attribute `Grammaticality` is specified. It represents the average grammaticality score assigned by the experimental subjects to the structure represented by this instance.

The first data set, `wordorder1.arff`, deals with ditransitive verbs such as *vorschlagen*, that take two NPs as arguments. Here are example sentences together with their attribute values:

(6) a. `VerbFinal = 0, NomAlign = 0, DatAlign = 0, ProAlign = 0, GroundAlign = 0`
Ich glaube, dass der Produzent dem Regisseur den Schauspieler vorschlägt.
b. `VerbFinal = 0, NomAlign = 1, DatAlign = 1, ProAlign = 0, GroundAlign = 0`
Ich glaube, dass den Schauspieler der Produzent dem Regisseur vorschlägt.
c. `VerbFinal = 0, NomAlign = 1, DatAlign = 1, ProAlign = 2, GroundAlign = 0`
Ich glaube, dass den Schauspieler der Produzent ihm vorschlägt.

The second data set `wordorder2.arff`, deals with transitive verbs like *kaufen* that take two arguments:

(7) a. `VerbFinal = 0, NomAlign = 0, DatAlign = 0, ProAlign = 0, GroundAlign = 0`
Maria glaubt, dass der Vater den Wagen kauft.
b. `VerbFinal = 1, NomAlign = 1, DatAlign = 0, ProAlign = 0, GroundAlign = 0`
Maria glaubt, dass kauft den Wagen der Vater.
c. `VerbFinal = 1, NomAlign = 1, DatAlign = 0, ProAlign = 1, GroundAlign = 0`
Maria glaubt, dass kauft den Wagen er.

Both data sets are quite small: the first one contains 24 instances, the second one 40 instances.

[1] Ground is a term in the literature on information structure. A constituent is ground if it's contextually given, and focus if it isn't.

2 Building a Linear Regression Model

We will first build a linear regression model for the dataset `wordorder1.arff`. Start Weka, load the dataset and use the complete base relation to build the model (no filtering required). Then go to the `Classify` menu and select `LinearRegression` as the classifier to work with. In the classifier selection window also set `attributeSelectionMethod` to `No attribute selection`. This tells Weka include all attributes in the regression equation.

Under `Test options` specify `Use training set` to test the performance of the model on the training set. As target attribute specify `Grammaticality`. Now train the model and report the correlation coefficient and the root mean squared error (RMS error).

Which regression equation does the algorithm come up with? Try to interpret the regression coefficients. What does it mean if some regression coefficients are bigger than others? Why are some coefficients zero? Inspect the data set (using `emacs` or `more`) to find out.

Now run the linear regression algorithm on the second dataset `wordorder2.arff`. Report again the correlation coefficient and the RMS error. Compare the regression equations you get for `wordorder1.arff` and `wordorder2.arff`. Do the coefficients differ? Is this expected?

3 Evaluating the Model

We will now perform *attribute selection* on the two models we trained in the last question. To select an attribute selection method, click on `LinearRegression` under `Classify`. In the classifier window choose `Greedy method` for `attributeSelectionMethod`. This implements the forward selection method that was discussed in the lecture.

With this setting, train regression models on `wordorder1.arff` and `wordorder2.arff`. Does the Weka come up with different regression equations than in Question 2? Does the performance of the model decrease by eliminating attributes?

So far, we have only tested our regression models on the training set. As discussed in the lecture, this is not a good idea, as it fails to test the ability of the model to generalize to unseen data.

For the models `wordorder1.arff` and `wordorder2.arff` perform a series of different evaluations by manipulating the options under `Test options`:

- Test on a held out part of the data by setting `Percentage split` to 66%. This tells Weka to use 66% of the data for training and the rest for testing. Also try 50%, 75%, and 90%.
- Perform 10-fold crossvalidation by selecting the option `Cross-validation` and setting the number of folds to 10.
- Perform leave-one-out crossvalidation by selecting the option `Cross-validation` and setting the number of folds to the number of instances in the data set.
- Finally, use a totally unseen test set by selecting `Supplied test set` and specifying the other data set as test set (i.e., test the `wordorder1.arff` model on `wordorder2.arff` and vice versa).

Report correlation coefficient and RMS error for both data sets for all four evaluation procedures. Compare to the evaluation results on the training set obtained in Question 2. Which evaluation do you think is the most stringent one? Does it give the worst performance?

4 Using Regression Trees and Model Trees

In the lecture, we discussed two ways of combining linear regression with decision trees: regression trees and model trees. We will now try these approaches on the word order data sets.

First build a regression tree, i.e., a decision tree that has numeric attributes at its branches and that outputs a numeric target value. Under `Classifier` choose `m5.M5Prime`. This selects the M5′ algorithm for the induction of model trees (of which regression trees are a spe-

cial case). Set the following classifier options: `useUnsmoothed: true`, `pruningFactor: 0`, `modelType: Regression tree`, and `verbosity: 0`.

Then train the model on both data sets and test using 10-fold crossvalidation. Draw the regression trees. Do they use the same attributes as the regression equations Which performance do you get? Why is the performance worse than using linear regression?

Finally, we will train a model tree on our word order data. A model tree is a regression tree, but instead of having target values at the leaves, it has linear equations (a separate one for each leaf). Under `Classifier` choose again `m5.M5Prime`. Set the following classifier options: `useUnsmoothed: false`, `pruningFactor: 2`, `modelType: Model tree`, and `verbosity: 0`.

Now train model trees for both data sets and test again using 10-fold crossvalidation. Draw the model trees and give the regression equations. Does performance improve with respect to the regression trees?

Compare the model trees that you get on `wordorder1.arff` and `wordorder2.arff`. What does this tell us about the two data sets? Is it appropriate to fit a single linear equation to `wordorder2.arff` (as we did in Questions 2 and 3)? Or is there are non-linear relationship? How does the model tree account for this? Look in particular at the attribute `VerbAlign`.

References

Keller, Frank. 2000. Evaluating competition-based models of word order. In Lila R. Gleitman and Aravid K. Joshi, eds., *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, 747–752. Mahwah, NJ: Lawrence Erlbaum Associates.