Weka – Evaluation: Assessing the performance

Lab3 (in-class): 21 NOV 2016, 13:00-15:00, CHOMSKY

ACKNOWLEDGEMENTS: INFORMATION, EXAMPLES AND TASKS IN THIS LAB COME FROM SEVERAL WEB SOURCES.

Learning objectives

In this assignment you are going to:

- interpret the results generated by weka

Preliminaries: *peer-peer interaction*

In this lab class, we continue exploring the pros and cons of the *peer-peer practice* (aka *novice–novice pairing*). This approach in practical lab sessions can produce results significantly better than two students working independently.

For practical labs, we get some inspiration from *pair programming* (but *pair work* is applied in many other fields, not only in software development). As you can read on wikipedia, "Pair programming is an agile[1] software development technique in which two programmers work together at one workstation. One, the **driver**, writes code while the other, the observer or **navigator**, reviews each line of code as it is typed in. *The two programmers switch roles frequently*." Since we are not writing code, we apply this general behavior to our specific activity of the day.

One important element common to all agile methodologies and to time managing in genre is the concept of **timeboxing**. Timeboxing allocates a fixed time period, called a time box, to each planned activity.

Here, I list a few reminders:
1) Switch role frequently, at least once or twice per task (remind your companion if s/he forgets).
2) If you are the "driver" (the one actively using the computer), verbalize your actions so the "navigator" knows about your next move.
3) If you are the "navigator", suggest additional paths or solutionsto explore.
4) Discuss and agree on what needs to be done. If you do not agree, implement both solutions.
5) IMPORTANT: Apply timeboxing. See the time that you should allocate for each activity and make time-boxed decisions to complete your time-boxed tasks!

[1] Is you wish to know more about Agile methodologies, read here:
<https://en.wikipedia.org/wiki/Agile_software_development >

Task 1: Warming-up – The Course Ratings dataset [max time 10min]

You are already familiar with the Course Rating dataset. You created your arff version in Lab1. The original format of this dataset was a table similar to this one: http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/course_ratings.txt .

Open weka → Explorer interface, and upload the following dataset: http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/course_ratings_class_beginning.arff

- Observe and (re)analyse the dataset in the Preprocess tab, ie open the dataset in the Viewer (Edit button), analyse the attributes, the graphs and the statistics, etc. In short, observe and take notes of everything that you think is important to know before you start the classification task.

- Run J48 with default parameters.

Q1: what is the name of the class attribute? How many class values in this dataset? what's the classification accuracy? what's the name of the top node? what's the size of the tree? What's the value of k statistic, what does this value mean? Does this classification make sense?

Upload the following dataset:
http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/course_ratings_class_end.arff

- Run J48 with default parameters

Q2: reply to all the questions asked in Q1.

- Set the unpruned option to true (you must se a –U on the box next to Choose). Now run J48 again on this dataset.

Q3: what is the classification accuracy, what is k value and what does it mean? what's the size of the tree? what are the main attributes here? What do you think?

- Set the –M parameter to 1.

Q4: what happens?

Set the option Percentage split to two values of your choice and motivate these choices.

Run J48 with these two different values.

Q5: do you observe any variations in the performance?

Q6: draw general conclusions from this case study.

Tasks 2 – Tuning parameters via Development Set (aka validation set) [max time 20min]

Dowload the iris datset: http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/iris.arff
Open the iris dataset and create: a training set, a development set and a test set. Decide reliable proportions. Make sure that training set, development set and test set do no share any instances (ie. do not have instances in common).

Upload the training set to weka explorer. Run J48 and test it on the development set with default parameters. Write down the results. Then run J48 with the unpruned option. Write down the results.

Select the best model. Once you have identified the optimal parameters (ie the parameters of your selected model), merge training set and development set in one single training set. Reload the merged training set. Run J48 with your optimal parameters, and test the model on the test set.

Q7: what is the size of the trees, check accuracy, k statistic, precision, recall, errors ecc. Explore the Result panel and interpret all the values. Help yourselves out with the information in the Appendix. What do you think of your classifier, is it good or bad?

Q8: Now compare the performance of you selected models with the performance of a J48 run with default parameters on the whole iris dataset and tested with 10 fold-cross validation. Which classifier performs better?

Task 3: Comparing two models : Roc Curves and Auc [max time: 30min]

Download the spam base dataset:
http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/spambase.arff

```
the nominal class indicates: spam (1) or not (0),
Change @attribute class {0,1}
to @attribute class {notspam,spam}
```

Modify the instances accordingly (some quick text processing: use find&replace or whatever can be helpful. Remember: an arff is a comma-separated file, can this information help you somehow?)

Once you have modified the classes, save your file. Upload the newly saved dataset into Weka Explorer.

First, run J48 default parameters, 10-fold cross-validation. Then run J48 with the unpruned option.

Q9: Which classifier performs better? Compare accuracy, P, R, F-score, k statistic, Roc Curve/AUC and precision/recall curves for the spam class. What are your conclusions?

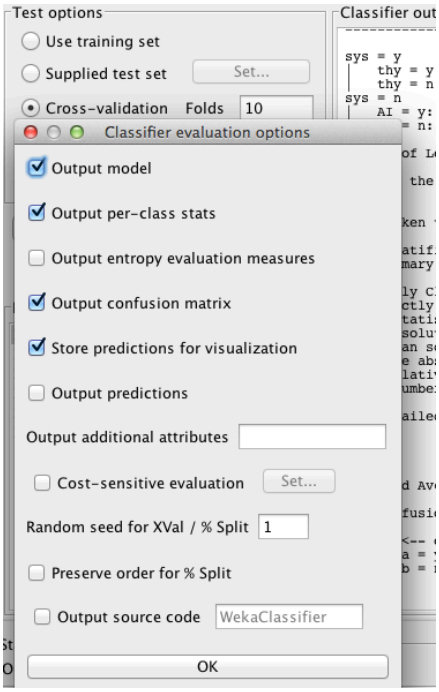
Task 4: Randomize seeds in 10-cross-validation 10 times [max 10min]

It is common practice in many language technology experiments to evaluate the results using 10-fold cross-validation 10 times with randomized seeds.

The **Random seed for XVal / % Split** option is the seed input for randomising the data. Changing the seed value produces a different set of cross-validation folds. The default value is 1. See picture below.

Run J48 on the spam datasets with 10 random different numbers for the **Random seed for XVal / % Split** option. You will get 10 slightly different sets of results. Average the accuracy results of the 10 runs and get a single value.

Q10: Compare this averaged accuracy value and the accuracy you get with defaults. Are they different?



Appendix. Interpretation of Weka results in a nutshell

Rule of thumb: you should have **at least** 5 instances per class to get some kind of sensible statistical results. The more example the better. For example, 5+ examples per the spam class and 5+ examples for the notspam class. The larger the sample/dataset, the more trustworthy results will be.

The stratified cross-validation paints the most reliable picture of a classifier's performance.

Weka results output:
TP = true positives: number of examples predicted positive that are actually positive
FP = false positives: number of examples predicted positive that are actually negative
TN = true negatives: number of examples predicted negative that are actually negative
FN = false negatives: number of examples predicted negative that are actually positive

Weka Confusion Matrix: (wiki primer)
The confusion matrix is more commonly named contingency table. The number of correctly classified instances is the sum of diagonals in the matrix; all others are incorrectly classified.

	a	b <-- classified as
actual a=0	TP	FN
actual b=1	FN	TP

Example

```
a b   <-- classified as
7 2 | a = yes
3 2 | b = no
```

The **True Positive (TP) rate** (it is equivalent to **Recall**) is the proportion of examples which were classified as class x, among all examples which truly have class x, i.e., how much of the class was captured correctly. In the confusion matrix, this is the diagonal element divided by the sum over the relevant row, i.e., 7/(7+2)=0.778 for class yes and 2/(3+2)=0.4 for class no in our example.

The **False Positive (FP) rate** is the proportion of examples which were classified as class x, but belong to a different class, among all examples which are not of class x. In the matrix, this is the column sum of class x minus the diagonal element, divided by the row sums of all other classes; i.e. 3/5=0.6 for class yes and 2/9=0.222 for class no.

The Precision is the proportion of the examples which truly have class x among all those which were classified as class x. In the matrix, this is the diagonal element divided by the sum over the relevant column, i.e. 7/(7+3)=0.7 for class yes and 2/(2+2)=0.5 for class no.

In other words:
Recall is the TP rate (also referred to as sensitivity), ie: what fraction of those that are actually positive were predicted positive? → TP / actual positives

Precision is TP / predicted Positive, ie what fraction of those predicted positive are actually positive?

True Negative Rate is also called Specificity. (TN / actual negatives)
1-specificity is x-axis of ROC curve: this is the same as the FP rate (FP / actual negatives)
what fraction of those that are actually negative were found to be positive?
(hopefully very low)

The F-Measure is simply 2*Precision*Recall/(Precision+Recall), a combined measure for precision and recall. These measures are useful for comparing classifiers.

The **weighted average** is computed in this way:
https://www.youtube.com/watch?v=4iJYJXv5yYg

Att!: all the error statistics compare true values to their estimates, but do it in a slightly different way. They all tell you "how far away" are your estimated values from the true value of. Sometimes square roots are used and sometimes absolute values - this is because when using square roots the extreme values have more influence on the result.

The MAE and the RMSE can be used together to diagnose the variation in the errors in a set of forecasts. The RMSE will always be larger or equal to the MAE; the greater difference between them, the greater the variance in the individual errors in the sample. If the RMSE=MAE, then all the errors are of the same magnitude.

Mean absolute error (MAE): The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. Expressed in words, the MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average.

Root mean squared error (RMSE): The RMSE is a quadratic scoring rule which measures the average magnitude of the error. Expressing the formula in words, the difference between forecast and corresponding observed values are each squared and then averaged over the sample. The root of the mean squared error, indicate the accuracy of the probability estimates that are generated by the classification model.
Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable.
Remember!
Absolute values have the same units as the quantities measured. For example, 0.2314 grams, or plus or minus 0.02 mL.
Relative values are ratios, and have no units. The ratios are commonly expressed as fractions (e.g. 0.562), as percent (fraction x 100, e.g. 56.2%), as parts per thousand

(fraction x 1000, e.g. 562 ppt), or as parts per million (fraction x 106 , e.g. 562,000 ppm).

K values= Agreement of the prediction with the true class
K removes the agreement **by chance** (watch this video, if this concept is still unclear: https://www.youtube.com/watch?v=fOR_8gkU3UE)
Generally speaking, the range of possible values of kappa is from -1 to 1, though it usually falls between 0 and 1. Unity represents perfect agreement, indicating that the raters agree in their classification of every case. Zero indicates agreement no better than that expected by chance, as if the raters had simply "guessed" every rating. *A negative kappa would indicate agreement worse than that expected by chance*.

In weka, Kappa is a chance-corrected measure of agreement between the classifications and the true classes. It's calculated by taking the agreement expected by chance away from the observed agreement and dividing by the maximum possible agreement. A value greater than 0 means that your classifier is doing better than chance (it really should be!).

ROC Area and AUC
The ROC curves and AUC areas are discussed in Section 5.7 (page 172ff) of the weka book. A ROC curve can be generated by right-clicking on an entry in the result list and selecting Visualize threshold curve. This gives a plot with FP Rate on the x-axis and TP Rate on the y-axis. Depending on the classifier used, this plot can be quite smooth or it can be fairly irregular.

Each instance in the ROC curve has a threshold value of class. If the instance highly belongs to this class, its threshold will be close to 1 so it will have red colour, the higher threshold of instance the dark colour it will have in the ROC curve[2].

The Classifier Output also gives the ROC area (also known as AUC), which, as explained in Section 5.7 (page 177ff), is the probability that a randomly chosen positive instance in the test data is ranked above a randomly chosen negative instance, based on the ranking produced by the classifier. The best outcome is that all positive examples are ranked above all negative examples, in which case the AUC is 1. In the worst case it is 0. In the case where the ranking is essentially random, the AUC is 0.5, and if it is significantly less than this the classifier has performed anti-learning!

Sources/References

<http://stats.stackexchange.com/questions/118/why-square-the-difference-instead-of-taking-the-absolute-value-in-standard-devia >
< http://weka.wikispaces.com/Primer >
< http://www.eumetcal.org/resources/ukmeteocal/verification/www/english/msg/ver_cont_var/uos3/uos3_ko1.htm >
< http://stats.stackexchange.com/questions/131267/how-to-interpret-error-measures-in-weka-output >
< http://www.cs.usfca.edu/~pfrancislyon/courses/640fall2014/WekaDataAnalysis.pdf >
< http://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/ >

[2] More specifically, color depends on what have you chosen. In your case, you chosen Colour: Threshold, so the color represents threshold value set to get this pair of true FPR/TPR point. In other words - ROC is a parametric curve of the threshold, so point (tpr, fpr) belongs to ROC if and only if there exists threshold value t for which experiment results in True positive rate=tpr and False positive rate=fpr. However, it is just an existance, so ROC as such does not show what is the exact value of t used. Coloring the curve gives you this one missing information. (<>)