JADS Assignments 4

March 28, 2017

1 Data Mining (JADS) - Assignment 4

This assignment is about SVMs, Neural nets and preprocessing. We will again use several new datasets from https://www.openml.org. This time you will also use OpenML to upload your own experiments.

The same rules as the previous assignments regarding report length and formatting apply.

1.1 Exercise 1: Support Vector Machines and scaling (1 point)

Evaluate the performance of a Support Vector Machine on the Seismic Bumps dataset, using 10-fold cross-validation. More information on the dataset here: https://www.openml.org/d/1500. Download with the download icon. If the downloaded file does not have an .arff extension, simply add that extension.

- In WEKA Explorer, load the dataset and look at the distribution of the attribute values. Select the same attribute (e.g. attribute 1) both in the list (left) as the visualization target (right (in the lower right bar chart), using 'class' as the class attribute. Are the values normally distributed?
- Now, under filters, find the 'Standardize' filter and apply it to your data. Describe you the effect on the distrubution of your data. Reload the data, and now try the 'Normalize' filter instead. What is the difference?
- Reload the data, and now apply an SVM classfier (called functions.SMO in WEKA). First, build and evaluate an SVM model (with default hyperparameters) and report the performance (accuracy, i.e. percent correct).
- Now, in the hyperparameter settings, switch off the internal normalization by setting 'filterType' to 'No normalization/standardization'. What is the effect on performance? Does it matter whether you choose the normalize or standardize filter?

1.2 Exercise 2: Feature Selection (2 points)

Study the effect of feature selection on the Arrythmia dataset https://www.openml.org/d/5. It measures heart arrythmia based on sensor measurements, but not all sensors are equally useful.

- Load the dataset and select the most relevant features using correlation-based feature selection (CfsSubsetEval). Do this in the 'Select Attributes' tab. How many of the 280 features are selected?
- Build a normal k-Nearest Neighbor (kNN) classifier (use k=3) and report the performance (accuracy).

- Now perform feature selection and then run the same nearest neighbor classifier. This can be done with the AttributeSelectedClassfier. Use the standard evaluator (CfsSubsetEval), but be sure to set kNN as the classifier. Again, report the performance. Give a clear explanation why the performance is different.
- Repeat the above, but now with a decision tree (J48) instead of kNN. Do you still see a performance difference? Why (not)?
- Finally, replace the evaluator with the 'InfoGainAttributeEval' (also select 'Ranker' under Search), and compare its performance to J48 without feature selection. Does this yield different performances? Explain why (not).

1.3 Exercise 3: PCA (1 point)

Study the effect of PCA on the Glass dataset https://www.openml.org/d/41.

- Load the dataset, and visualize the first two attributes in the Visualize tab. Do the first two features neatly separate the classes (colors)? Include a screenshot and discuss.
- Run PCA (the Principal Component filter) on your data, and Visualize the first two principle components (those are the first two attributes after applying PCA). Is the data better separated? Include a screenshot and discuss.

1.4 Exercise 4: PCA and kNN (1 point)

Study the effect of PCA on the performance of kNN on the Isolet dataset https://www.openml.org/d/300.

- Load the dataset, and train a normal kNN classifier (use k=3). Report the performance (accuracy).
- Run PCA (the Principal Component filter) on your data, and set the number of retured attributes to 40. Now, build the same kNN classifier on the 40 principal component. Report the difference in performance and explain. Does the PCA manage to retain most of the information in 40 of the 618 features?

Note: building these models may take a few minutes.

1.5 Exercise 5: Kernel Selection (2 points)

Study the effect of different kernels in SVMs on the EEG Eye State dataset https://www.openml.org/d/1471.

- Build 3 models, using the default Linear (Polynomial with degree 1), Polynomial (degree 2), and RBF kernel. Report the performances.
- For the Polynomial kernel, change the degree to values [2,3,4,5,10,50]. Report the performances (e.g. use a table or a line plot) and discuss. When do you think the SVM is under/overfitting?
- For the RBF kernel, change the gamma parameter to values [0.001,0.01,0.1,1,10,100,1000]. Again, report the performances (e.g. use a table or a line plot) and discuss. When do you think the SVM is under/overfitting?

1.6 Exercise 6: Neural Networks (2 points)

Evaluate MultilayerPerceptrons on the Covertype dataset https://www.openml.org/d/150.

- Because this is a large dataset (and MLPs are slow), it is best to take a stratified sumsample of the data. Use the supervised.instance.Resample filter, and take a 0.1% (that's 0.001) subsample.
- Build a MultiLayerPerceptron (default settings) using the default settings and report its performance (accuracy).
- Observe the outputted weights. For Node 0, which incoming node has the strongest connection with this node (highest absolute weight)?

- Vary the learning rate using values [0.1,0.2,0.3,0.5,1]. Which gives you the best performance? Where do you think the optimal value lies?
- Vary the number of hidden layers using values [1,2,3,4,5]. You can keep the default size of the layers (denoted by 'a'). Hence, to build a 5-layer network, you need to set the `hiddenLayers` field to 'a,a,a,a,a'. Which gives you the best performance? Explain why you think this is.

1.7 Exercise 7: OpenML experiments (1 point)

Upload your first experiment to OpenML, using task https://www.openml.org/t/3951 (Classification on the Desharnais dataset).

- From the WEKA Package Manager, install the OpenmlWeka package.
- Read the guidelines for using the graphical interface here: https://www.openml.org/guide#!plugin_weka
- Load task 3951, select one (or a few) classifiers, and run them.
- On OpenML, find your runs under your profile, and report the run ID so that we can check that everything worked correctly. One run per team is ok,but mention the name of the student that uploaded that run.
- Note: by default, OpenML doesn't upload experiments that are identical to earlier experiments. If your experiment does not upload (and no errors are reported), try to run different algorithms (or use different hyperparameters).
- Note: for fun, you can compare your performance to other students: https://www.openml.org/t/3951#people