Machine Learning Tutorial: Weka

1.) **Load in and work with Features/Attributes in WEKA**
2.) **Test ML Classification Algorithms**
3.) **Load in Validation Test Set**
4.) **Select Features**
5.) **Optimize Hyperparameters**

Why Use Machine Learning in Neuroimaging Analysis?

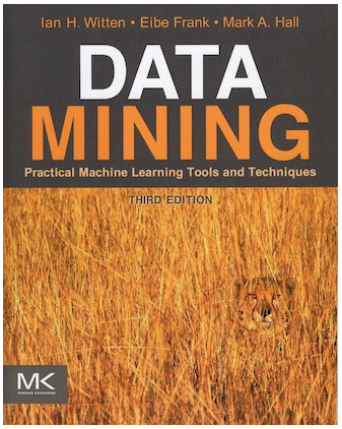
- *Decoding/Brainreading (Haxby et al. 2004)*
- *Identify Noise Components in Data (Poldrack & Tohka 2009)*
- *Computer Aided Diagnosis, or classifying populations (Anderson et al. 2010)*

Part I: Loading Attribute file with Feature Vector

What is WEKA? Weka is data mining software written in Java. It is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java/Perl code. Weka contains tools for data pre-processing, classification, regression, clustering, and association rules. Weka is open source and freely available at: http://www.cs.waikato.ac.nz/ml/weka/.

Weka only deals with "flat" files. The input file format that Weka operates on is called an attribute relation file format (.arff) file. The external representation of an instances class is an .arff file, which consists of a header assigning attribute variable types and data exemplars with labels.

The Header section contains the relation and attribute declarations. Attribute declarations take the form of an ordered sequence of @attribute statements. Each attribute in the data set has its own @attribute statement, which uniquely defines the name of that attribute and its data type.



The <datatype> can be any of the four types supported by Weka:

numeric

integer is treated as *numeric*

real is treated as *numeric*

<nominal-specification>

string

date [<date-format>]

Note, for most neuroimaging purposes, we are interested in either real or numeric data file formats. However, some behavioral data may take on the form of a string.

The next part of the .arff file contains the data as a comma separated list. Each Instance consists of a number of attributes, any of which can be nominal (= one of a predefined list of values), numeric (= a real or integer number) or a string (= an arbitrary long list of characters, enclosed in "double quotes"). Each

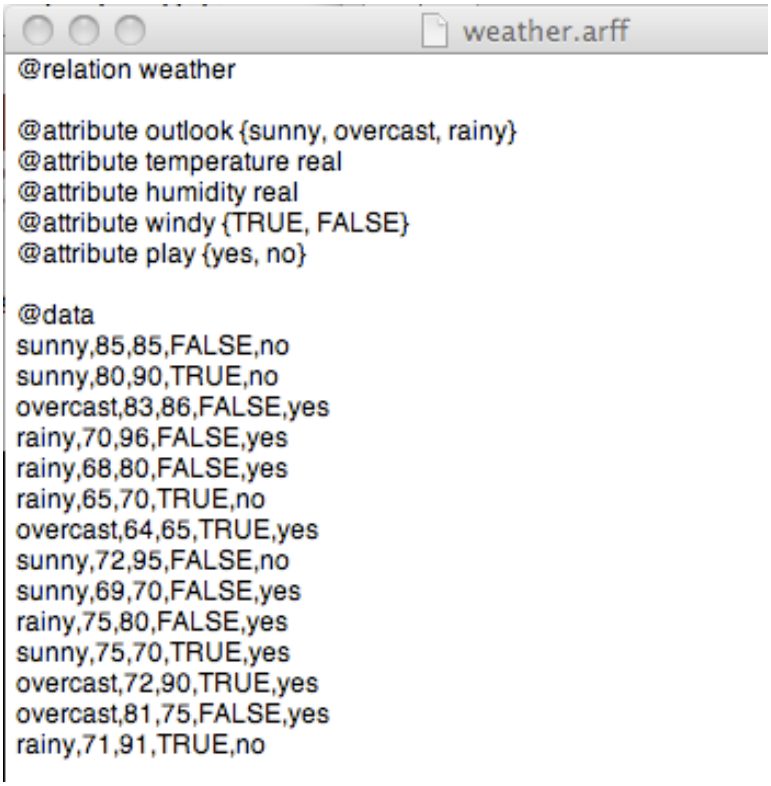
instance is surrounded by curly braces, and the format for each entry is: <index> <space> <value> where index is the attribute index (starting from 0).

Attribute Relation File for Weather Example:

Try opening up a sample .arff file for viewing to get an idea of how the input file is formatted.

Open: weather.arff

Then select File>>Open With and navigate to Wordpad/Textedit. This should display the .arff file format described above. This file represents one of the simplest examples that contains a mixture of data types.



Freely Available Data Sets

In the ML community, there are a number of datasets that are used for benchmarking purposes. One of the most popular repositories for these data sets is the UCI Repository available here: http://www.ics.uci.edu/~mlearn/MLRepository.html

Part II: Testing Machine Learning Classifiers

Why Test Multiple Classifiers?

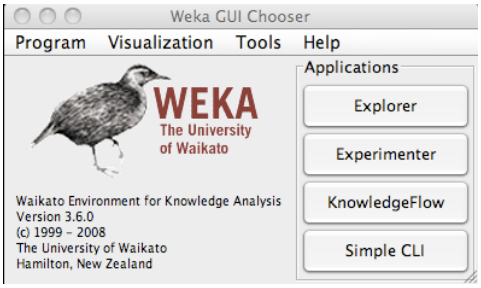
According to the Wolpert & MacGreedy "no free lunch" theorem, there is no single learning algorithm that universally performs best across all domains. Most Supervised ML algorithms differ in the model g(xlθ)

complexity that they use to describe inputs, x, using parameters θ, (the inductive bias), the loss function used, and/or the optimization procedure used to best fit the model parameters to the data. As such, a number of classifiers should be tested. Let's try a few using the WEKA gui on a sample data set.

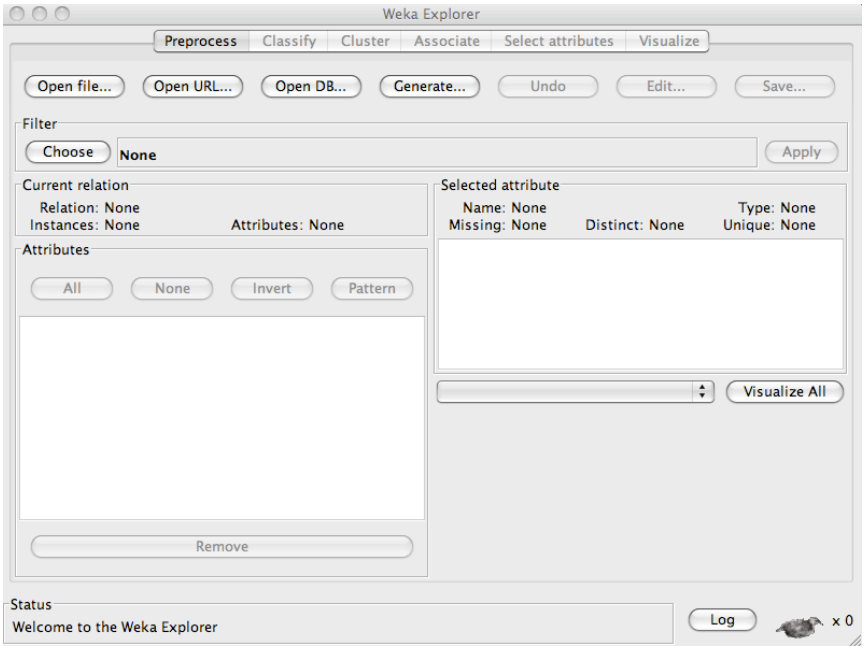
Launch the graphical user interface for weka by navigating to WEKA-3-6.

Then double click on the Weka Icon.

From here, select the icon for Explorer on the main Weka menu (see below).

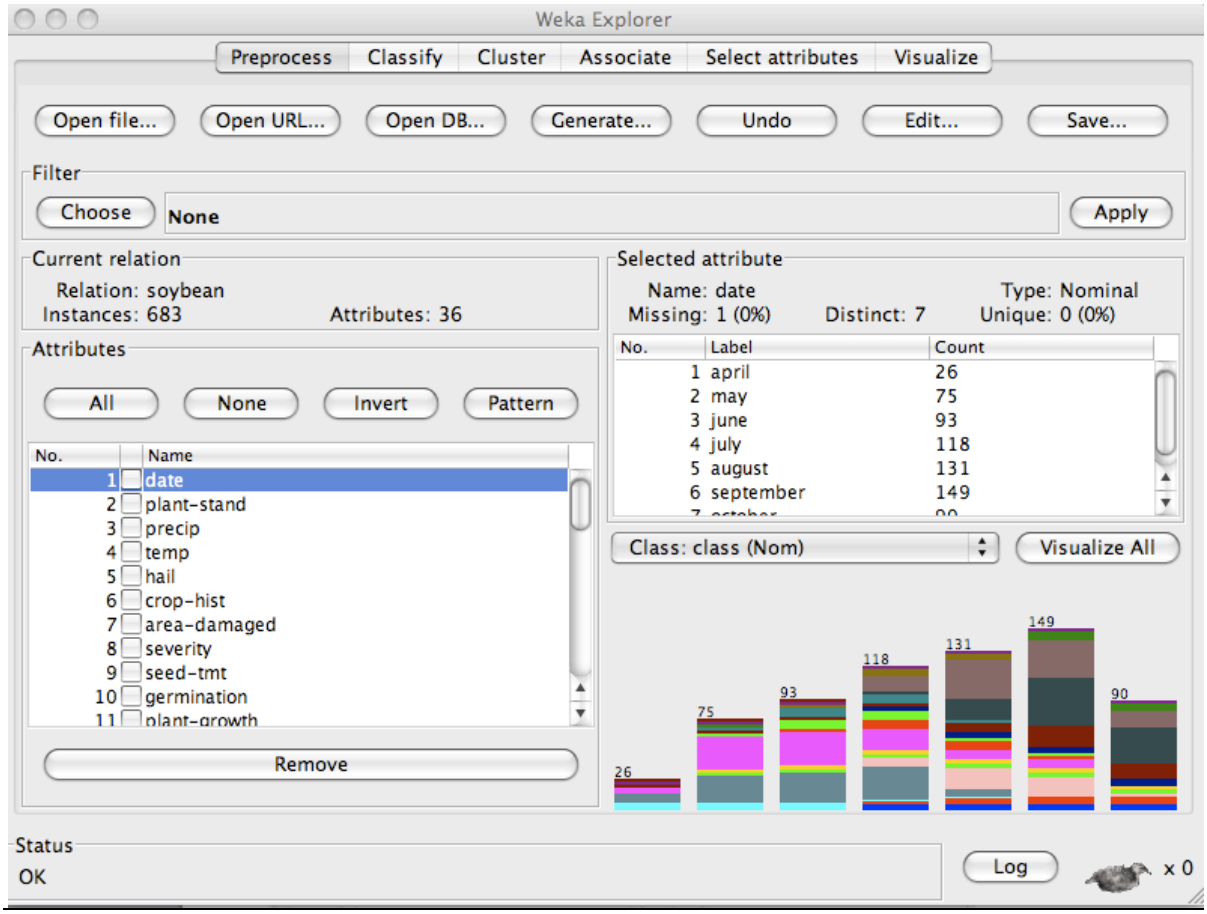


You should now see a screen like that shown below.

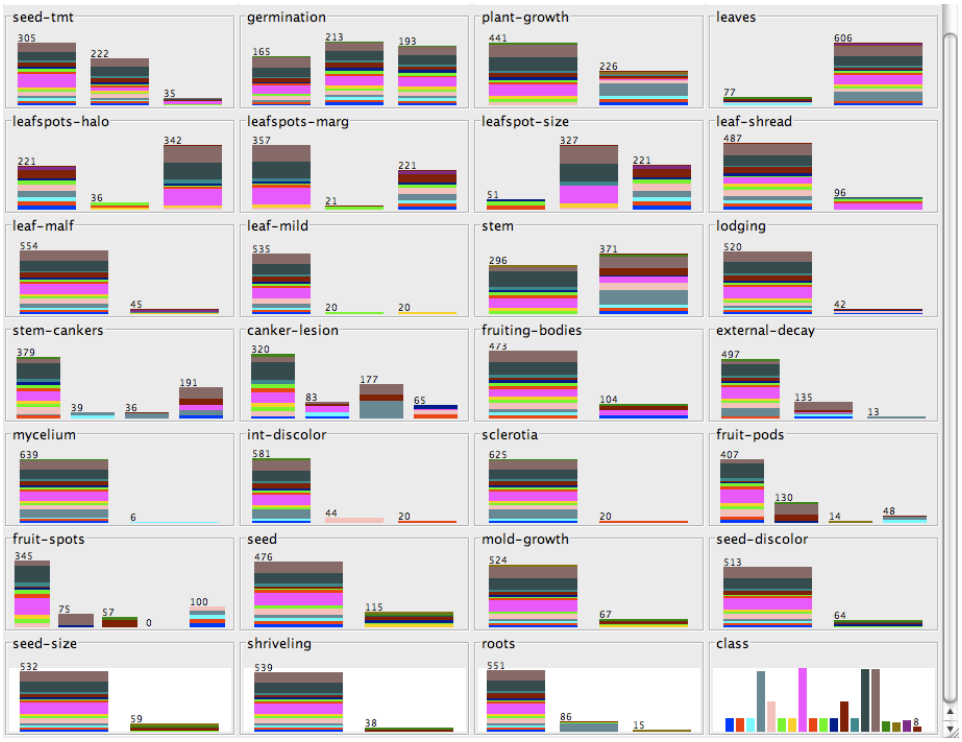


From the Preprocess menu, select 'Open File…', and navigate to the soybean.arff file.

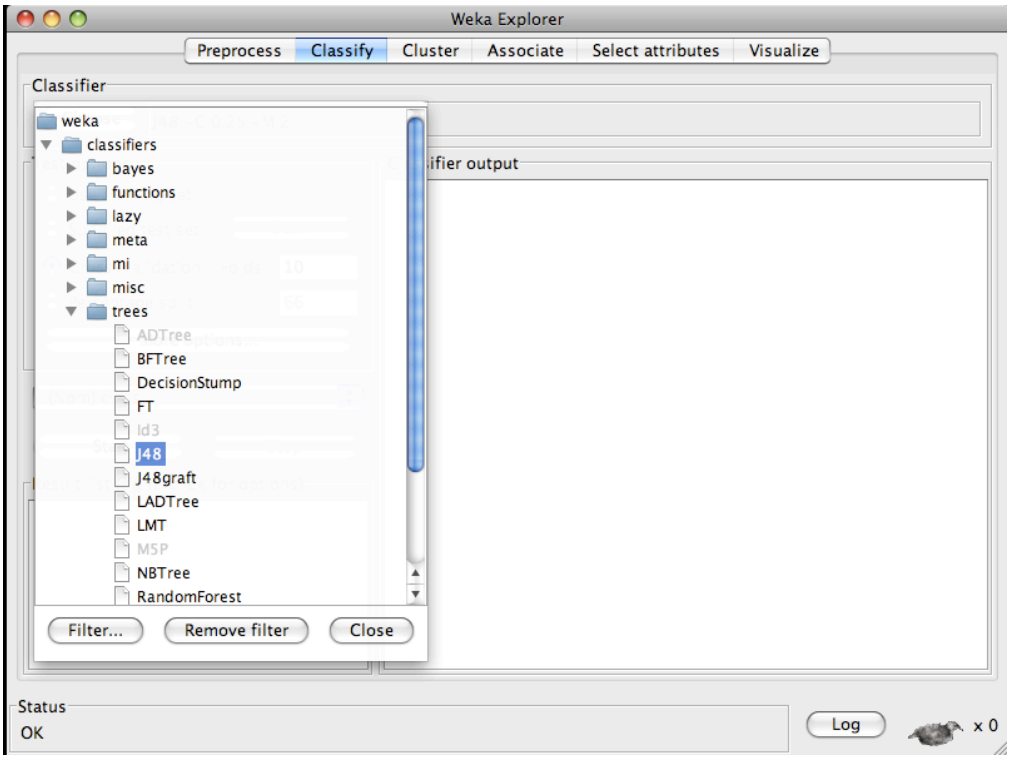
Once you have selected this file, the data should be loaded in, and should look like what you see below.



Select the button, "Visualize All," to view each attribute's distribution. Are there some attributes that are more informative than others?



Now select the "Classify" from the top menu. We will now select the classifier to use on the soybean data set. We will start by using the J48 Decision Tree algorithm, which implements the C4.5 Decision Tree, as described originally by Quinlan (1993). You can find this algorithm by selecting classifiers>>trees>>J48.



There are various approaches to determine the performance of classifiers, however cross-validation seems to be the most popular.

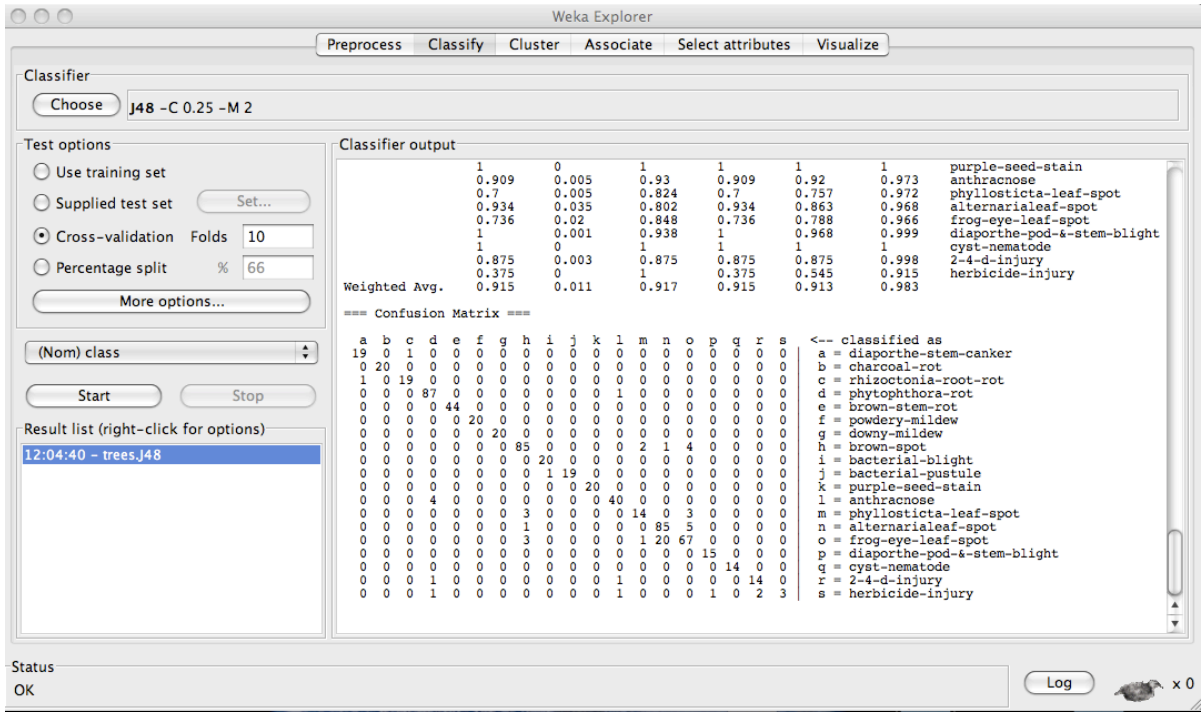
In cross-validation, a number of folds n is specified. The dataset is randomly reordered and then split into *n* folds of equal size. In each iteration, one fold is used for testing and the other *n-1* folds are used for training the classifier. The test results are collected and averaged over all folds. This gives the cross-validation estimate of the accuracy. The folds can be purely random or slightly modified to create the same class distributions in each fold as in the complete dataset. In the latter case the cross-validation is called *stratified*. Leave-one-out (loo) cross-validation signifies that *n* is equal to the number of examples. Out of necessity, loo cv has to be non-stratified, i.e. the class distributions in the test set are not related to those in the training data. Therefore loo cv tends to give less reliable results. However it is still quite useful in dealing with small datasets since it utilizes the greatest amount of training data.

Here, we will start by using the default, 10-fold cross validation, for our accuracy assesement.

Now click Start.

Notice how the Weka bird paces back and forth while you classify (yay!).

The output should look like what you see below:

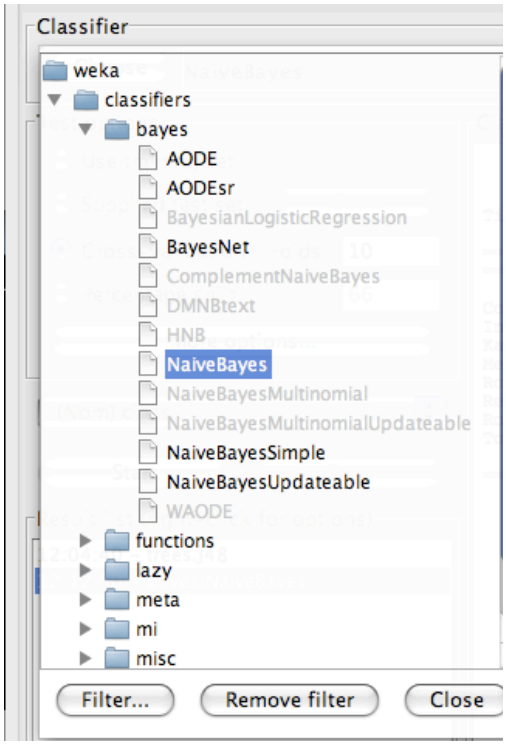


You should see the confusion matrix. Now scroll upwards to view the % of correctly classified instances.

```
Time taken to build model: 0.08 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         625               91.5081 %
Incorrectly Classified Instances        58                8.4919 %
Kappa statistic                          0.9068
Mean absolute error                      0.0135
Root mean squared error                  0.0842
Relative absolute error                 14.0484 %
Root relative squared error             38.4134 %
Total Number of Instances              683
```

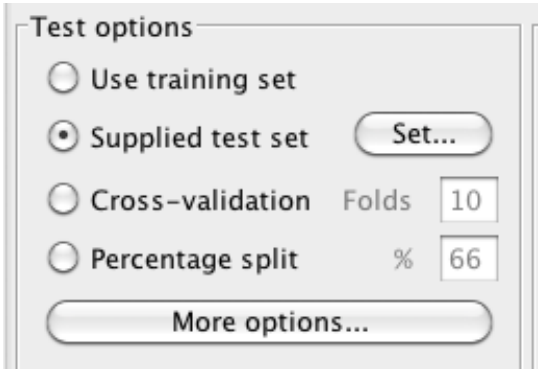


Now try out a different classifier, Naïve Bayes.

How does Naïve Bayes compare to the J48 Tree?

How about Support Vector Machine? (hint: its located under functions, and is called SMO)

Part III: Loading Test/Validation Set – Example Monk1 problem



The dangers of 'circular logic' have been discussed in detail in the neuroimaging literature (see Kriegeskorte. Circular analysis in systems neuroscience: the dangers of double dipping. Nat Neurosci. 2009). In order to avoid this pitfall, one should set aside a test set for model validation.

Lets try this using the classic Monk-1 dataset (Thrun et al. 1991), available on the UCI database repository (*http://archive.ics.uci.edu/ml/machine-learning-databases/monks-problems/monks.names*). The Monk dataset was the first one used in an international competition applying ML algoriths to the same dataset.

Loading the monk1_train.arff using the preprocess tab at the top. Next click on classify, and select 'supplied test set.' Navigate to the file called monk1_test.arff. Click close. Try using AdaBoost to classify this data set (located under the 'meta' menu tab).

Part IV: Feature Selection Step

Although feature selection may be considered an optional step depending on the data set one is working with and the model parameters used, it can often be useful when there are features that might be highly correlated. Misclassification rates generally decline at first as more features are considered. However a large number of 'noise' features, or a number of redundant features may cause the classifier to decline in performance, depending on the algorithm and hyperparameters used.

With the Monk-1 data set, Features 1 and 2 are highly correlated. Try going back to the 'preprocess' screen, and select the second feature for removal.

Using AdaBoost, try classifying the data set again. (Note: you will need to use the validation set called monk1_test_minus2.arff).

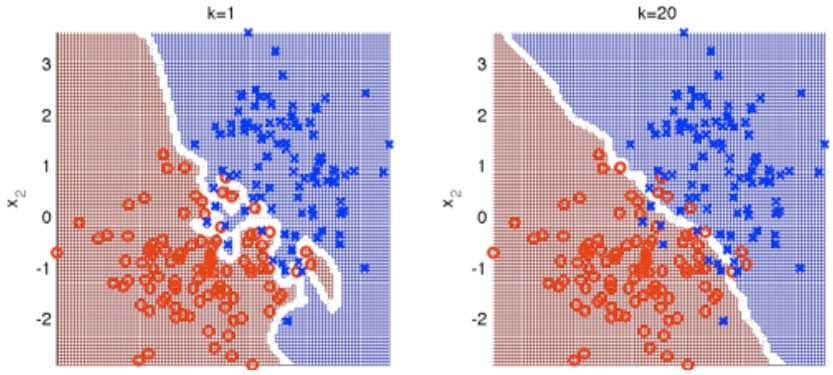
Did the classifier perform better?

There are a number of approaches to feature subset selection. Forward selection begins with an empty set of features; whereas *backward elimination* refers to a search that begins at the full set of features. Each of these methods are generally performed iteratively.

Part V: Optimize Hyperparameters

Why do this step?

One example that illustrates why

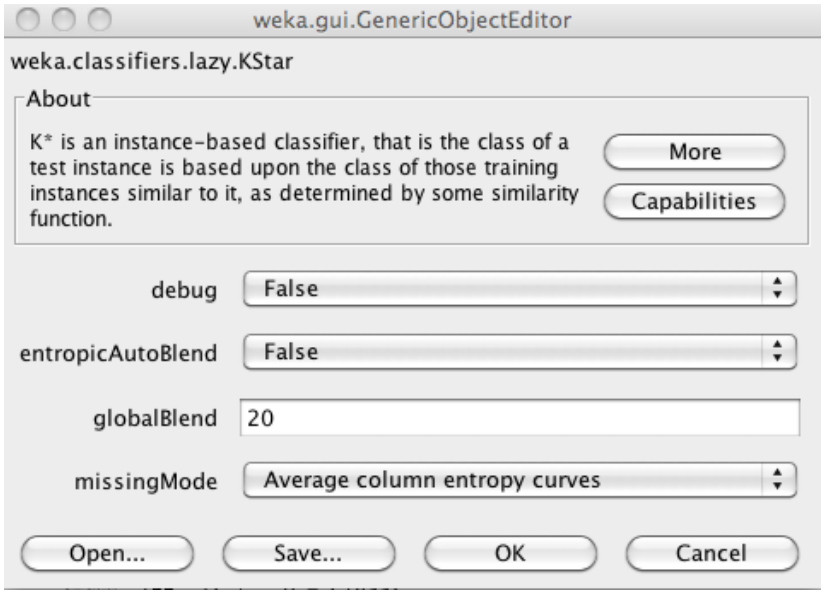


this is important is shown on the right from the Alypaydin (2004) textbook. When too many neighbors are used in K-NN, the algorithm begins to overfit, and won't generalize well to incoming data sets.

"To validate the generalization ability of a classifier with hyperparameters one has to perform a nested cross-validation. On each training set of the outer cross-validation, an inner cross-validation is performed for different values of the hyperparameters. The one with minimum (inner) cross-validation error is selected and evaluated on the test set of the outer cross-validation." – Muller et a. (2004)

Try adjusting some classifier parameters in WEKA. First right click on the Classifier bar. This should bring you to a window with all of the options for changing classifier hyperparameters. For a description of each one, click 'More'. This screen should also list the reference for the original paper that the classifier was based on.

Ok! Now you should be somewhat familiar with Weka. To run it recursively, you might want to try calling Weka via the command line using Perl or Java scripts.



Some Useful References

Pereira, F., Mitchell, T., & Botvinick, M. (2009). Machine learning classifiers and fMRI: a tutorial overview. *NeuroImage*, *45*(1 Suppl), S199-209. doi: 10.1016/j.neuroimage.2008.11.007.

Anderson, A., Dinov, I. D., Sherin, J. E., Quintana, J., Yuille, A. L., & Cohen, M. S. (2009). Classification of spatially unaligned fMRI scans. *NeuroImage*. doi: 10.1016/j.neuroimage.2009.08.036

Cox, D. D., & Savoy, R. L. (2003). Functional magnetic resonance imaging (fMRI) "brain reading": detecting and classifying distributed patterns of fMRI activity in human visual cortex. *NeuroImage*, *19*(2 Pt 1), 261-270

Poldrack, R. A. (2006). Can cognitive processes be inferred from neuroimaging data? *Trends in Cognitive Sciences*, *10*(2), 59-63. doi: 10.1016/j.tics.2005.12.004.

Poldrack, R. A., Halchenko, YO, Hanson, SJ. (2009) Decodeing the large-scale structure of brain fucntion by classifying mental States across individuals. Psychological Science. Nov 20(11)1364-72.

Tohka, J., Foerde, K., Aron, A. R., Tom, S. M., Toga, A. W., & Poldrack, R. A. (2008). Automatic independent component labeling for artifact removal in fMRI. *NeuroImage*, *39*(3), 1227-1245. doi: 10.1016/j.neuroimage.2007.10.013.

LaConte, S., Strother, S., Cherkassky, V., Anderson, J., & Hu, X. (2005). Support vector machines for temporal classification of block design fMRI data. *NeuroImage*, *26*(2), 317-329. doi: 10.1016/j.neuroimage.2005.01.048

Feature Subset Selection

Kohavi and John. "Wrappers for Feature Subset Selection," Artificial Intelligence 97(1997): 273-324.

D.W. Aha, "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms." Infernat. J. Man-Machine Studies 36 (1992) 267-287.

Liu, H. Li, J. and Wong, L. "A comparative study on feature selection and classification methods using gene expression profiles and proteomic pattern." Genomic Informatics, 13, 51-60, 2002

Dash, M. and Liu, H. "Feature selection for classification." International Journal of Intelligent Data Analysis, 1(3), 1997

Nested Cross Validation in fMRI and BCI

Douglas, PK, Harris, S. Yuille, A, Cohen, MS "Performance Comparison of Machine Learning Algorithms and Number of Independent Components Used in fMRI Decoding of Belief vs. Disbelief" Neuroimage 2010

Muller et al. "Machine Learning Techniques for Brain Computer Interfaces." Biomedical Technologies, 2004.