Machine Learning Practical
NITP Summer Course 2013

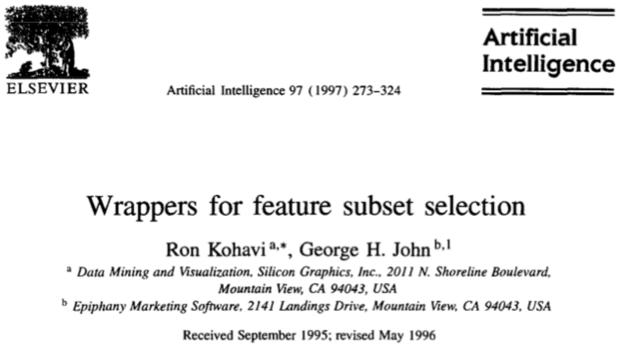
Pamela K. Douglas
UCLA Semel Institute
Email: pamelita [at] ucla [dot] edu

Topics Covered

Part I: WEKA Basics ☺

Part II: MONK Data Set & Feature Selection (from the Kohavi & John 1997)

- Test how Classification Algorithms choice effects results
- How does removal of redundant features change accuracy?



ELSEVIER

Artificial Intelligence 97 (1997) 273–324

Artificial Intelligence

Wrappers for feature subset selection

Ron Kohavi [a,*], George H. John [b,1]

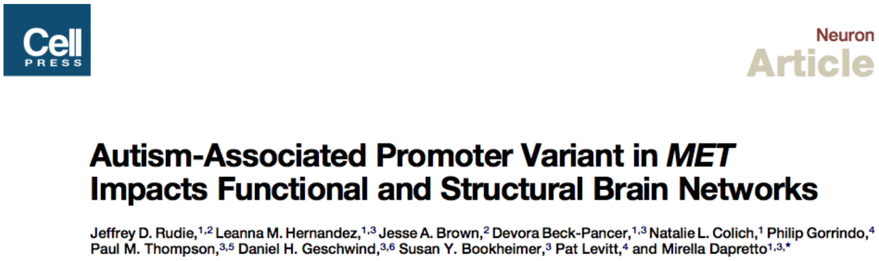
[a] *Data Mining and Visualization, Silicon Graphics, Inc., 2011 N. Shoreline Boulevard, Mountain View, CA 94043, USA*

[b] *Epiphany Marketing Software, 2141 Landings Drive, Mountain View, CA 94043, USA*

Received September 1995; revised May 1996

Part II: Functional Connectivity Matrix Decoding

- Data from Autism Spectrum Disorder & TD Subjects (Data from Rudie et al. 2012; Decoding from Douglas et al. 2012 OHBM)
- Use Resting State Connectivity Matricies
- Nested Cross Validation for Parameter Tuning



Cell PRESS

Neuron

Article

Autism-Associated Promoter Variant in *MET* Impacts Functional and Structural Brain Networks

Jeffrey D. Rudie,[1,2] Leanna M. Hernandez,[1,3] Jesse A. Brown,[2] Devora Beck-Pancer,[1,3] Natalie L. Colich,[1] Philip Gorrindo,[4] Paul M. Thompson,[3,5] Daniel H. Geschwind,[3,6] Susan Y. Bookheimer,[3] Pat Levitt,[4] and Mirella Dapretto[1,3,*]

Part 1. Weka Basics

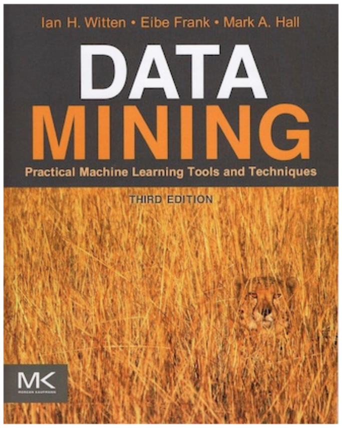
1.1. Background

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.

1.2. Loading Attribute file with Feature Vector

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.


Ian H. Witten • Eibe Frank • Mark A. Hall
DATA
MINING
Practical Machine Learning Tools and Techniques
THIRD EDITION
MK

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).

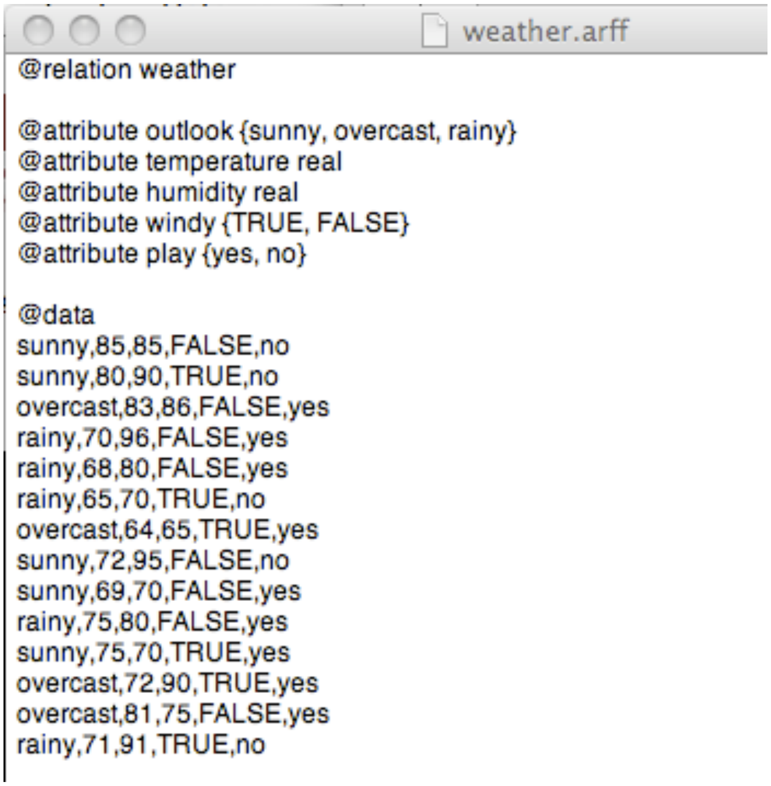
1.3. Attribute Relation File for Weather Example.

Data for this example can be found here: Applications/Weka/weka-4.1.3/data

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.



```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

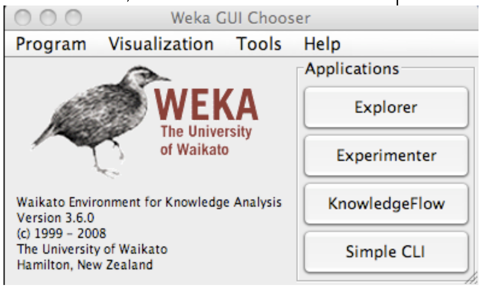
1.4. Testing Machine Learning Classifiers

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 $g(x|\theta)$ 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.

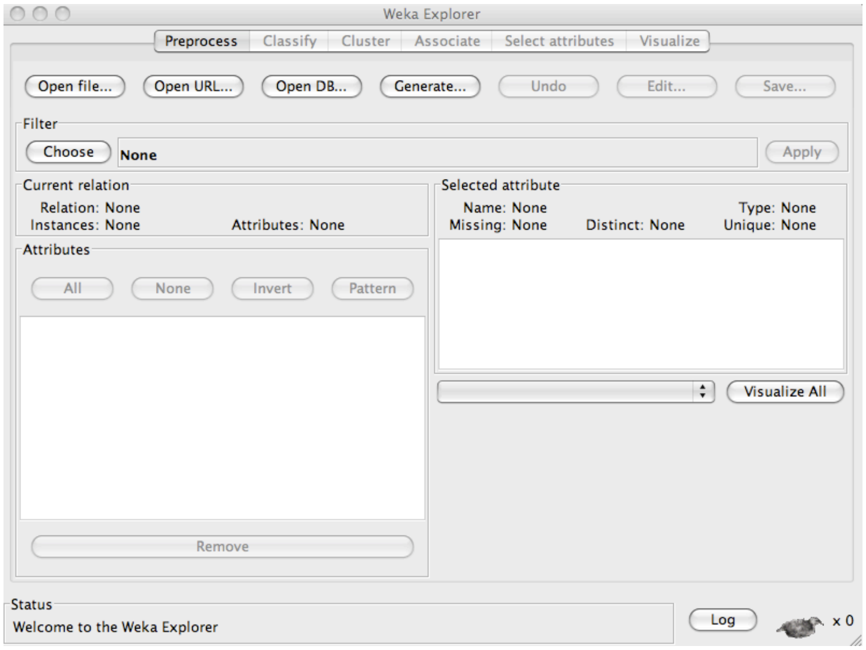
Launch the graphical user interface for weka by navigating to WEKA-3-6.
Then double click on the Weka Icon.

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



Waikato Environment for Knowledge Analysis
Version 3.6.0
(c) 1999 - 2008
The University of Waikato
Hamilton, New Zealand

You should now see a screen like that shown below.



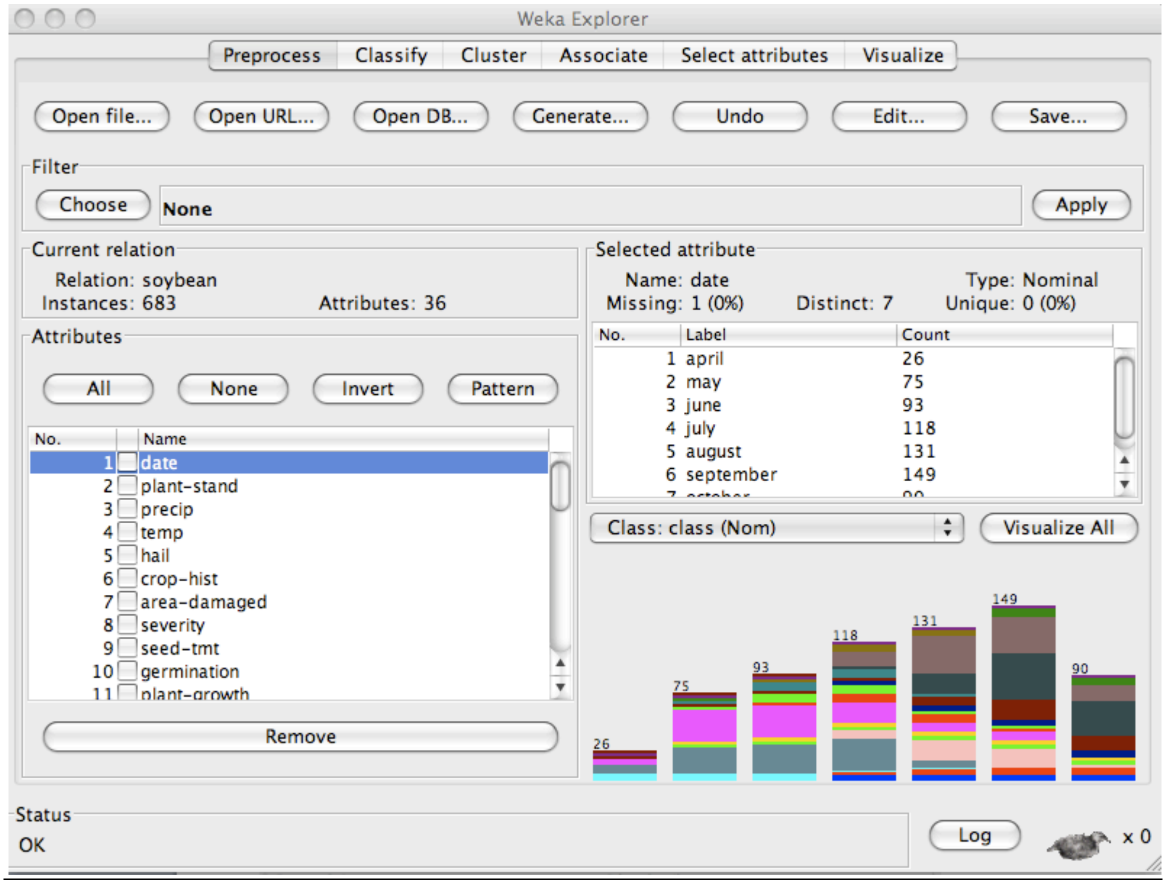
Filter: None

Relation: None
Instances: None
Attributes: None

Name: None
Missing: None
Distinct: None
Type: None
Unique: None

Welcome to the Weka Explorer

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.



Filter: None

Relation: soybean
Instances: 683
Attributes: 36

Name: date
Missing: 1 (0%)
Distinct: 7
Type: Nominal
Unique: 0 (0%)

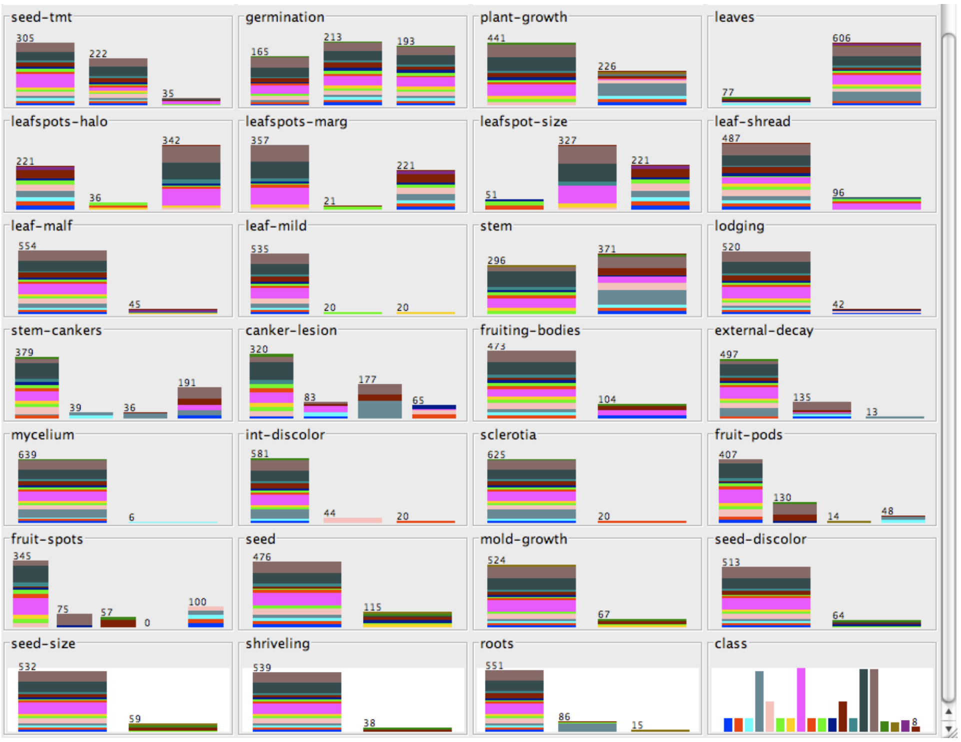
No.	Label	Count
1	april	26
2	may	75
3	june	93
4	july	118
5	august	131
6	september	149
7	october	90

No.		Name
1	☐	date
2	☐	plant-stand
3	☐	precip
4	☐	temp
5	☐	hail
6	☐	crop-hist
7	☐	area-damaged
8	☐	severity
9	☐	seed-tmt
10	☐	germination
11	☐	plant-growth

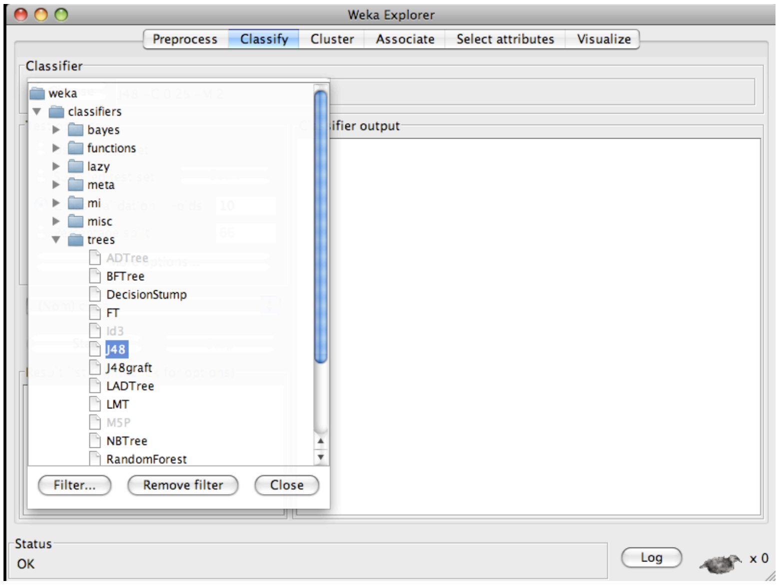
Class: class (Nom)

OK

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.



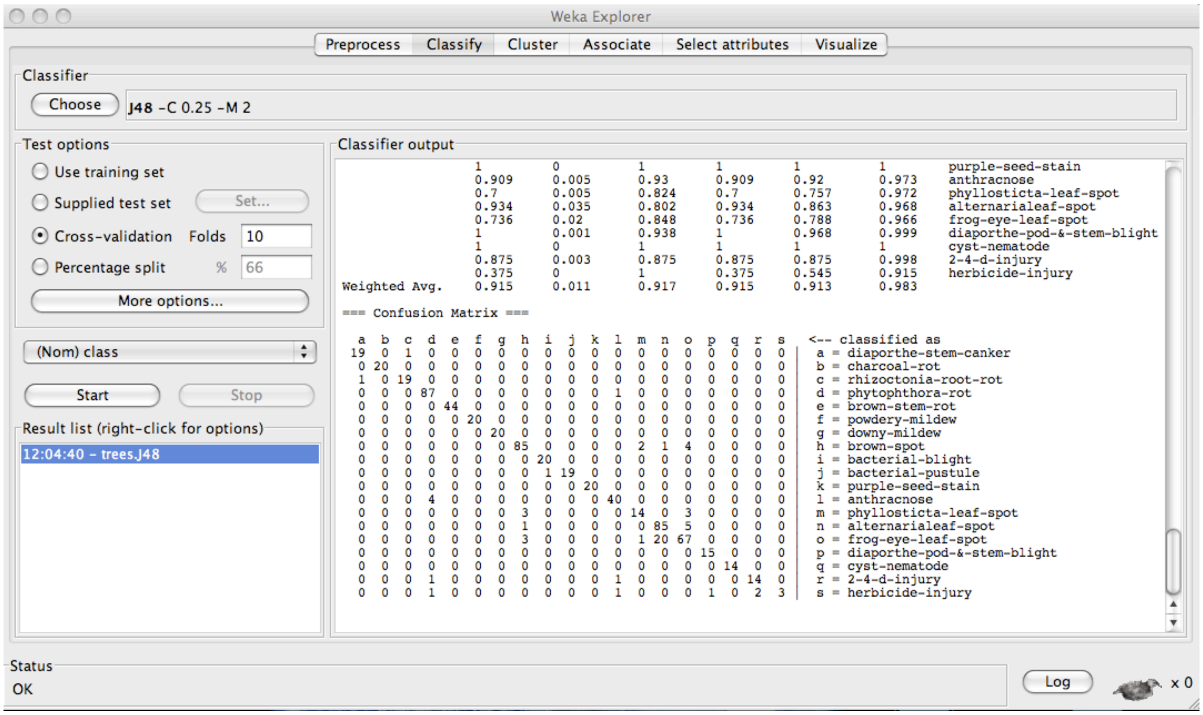
There are various approaches to determine the performance of classifiers, however cross-validation seems to be the most popular.

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:



Classifier: J48 -C 0.25 -M 2

○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) class

12:04:40 - trees.J48

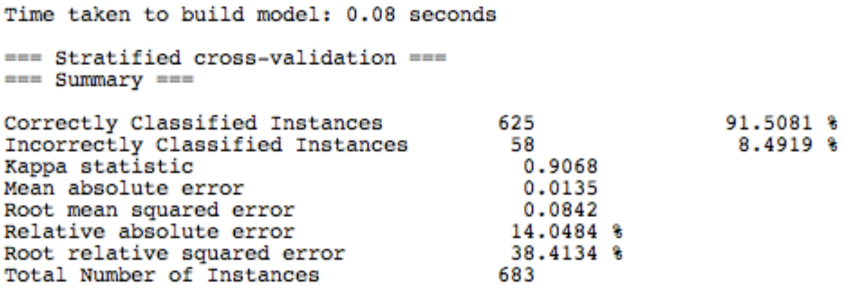
Status: OK

```
                 1         0         1         1         1         1         purple-seed-stain
                 0.909     0.005     0.93      0.909     0.92      0.973     anthracnose
                 0.7       0.005     0.824     0.7       0.757     0.972     phyllosticta-leaf-spot
                 0.934     0.035     0.802     0.934     0.863     0.968     alternarialeaf-spot
                 0.736     0.02      0.848     0.736     0.788     0.966     frog-eye-leaf-spot
                 1         0.001     0.938     1         0.968     0.999     diaporthe-pod-&-stem-blight
                 1         0         1         1         1         1         cyst-nematode
                 0.875     0.003     0.875     0.875     0.875     0.998     2-4-d-injury
                 0.375     0         1         0.375     0.545     0.915     herbicide-injury
Weighted Avg.    0.915     0.011     0.917     0.915     0.913     0.983

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s   <-- classified as
 19  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  a = diaporthe-stem-canker
  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = charcoal-rot
  1  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = rhizoctonia-root-rot
  0  0  0 87  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 |  d = phytophthora-rot
  0  0  0  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = brown-stem-rot
  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = powdery-mildew
  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0 |  g = downy-mildew
  0  0  0  0  0  0  0 85  0  0  0  0  2  1  4  0  0  0  0 |  h = brown-spot
  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0 |  i = bacterial-blight
  0  0  0  0  0  0  0  0  1 19  0  0  0  0  0  0  0  0  0 |  j = bacterial-pustule
  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0 |  k = purple-seed-stain
  0  0  0  4  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0 |  l = anthracnose
  0  0  0  0  0  0  0  3  0  0  0  0 14  0  3  0  0  0  0 |  m = phyllosticta-leaf-spot
  0  0  0  0  0  0  0  1  0  0  0  0  0 85  5  0  0  0  0 |  n = alternarialeaf-spot
  0  0  0  0  0  0  0  3  0  0  0  0  1 20 67  0  0  0  0 |  o = frog-eye-leaf-spot
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0 |  p = diaporthe-pod-&-stem-blight
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14  0  0 |  q = cyst-nematode
  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0 14  0 |  r = 2-4-d-injury
  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  1  0  2  3 |  s = herbicide-injury
```

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)

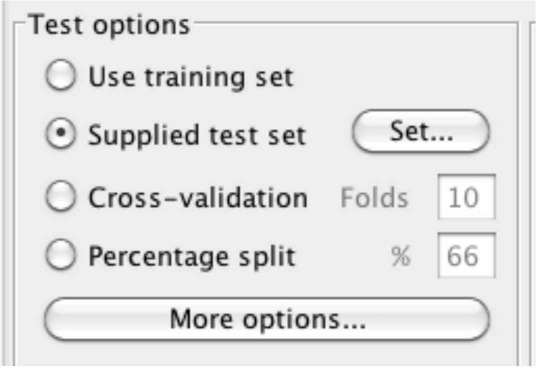
Part 2. Monk1 Data Set – Feature Selection

2.1. Background.

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*). was the first one used in an international competition applying ML algoriths to the same dataset. It is still used for ML benchmarking purposes. 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).

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.

** NOTE –Feature Selection should be run on your Training Data only!!! Using your Test Data in Feature Selection is another form of 'peeking'. !



Test options
○ Use training set
● Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

2.1 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?

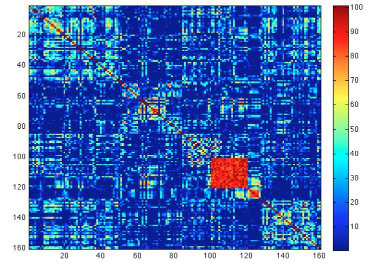
There are a number of approaches to feature subset selection. Forward selection begins with an empty set of features; whereas *backward elimination* refers to a search that begins at the full set of features. Each of these methods are generally performed iteratively.

Part 3. Connectivity Matricies – ADHD vs. Typically Developing

3.1. Background/ Data.

Data from this part of the experiment is simulated data that is meant to resemble resting state functional connectivity data from two populations. Data from genome wide associations and complexity measures could also be represented in this way, so the process for classification would be similar.

In our simulated data, each element in the matrix represents the correlation value between two ROI timecourses. Notice that the diagonal is unity, since each ROI timecourse should be perfectly correlated with itself. The matrix is also symmetric about the diagonal.



3.2. Generate ".ARFF" Files from Each Matrix

Open the MATLAB script, "NITP_weka.m." This script loads in data matricies for 30 subjects, 15 with ADHD, and 15 typically developing (TD).

The NITP_weka.m script should achieve the following:

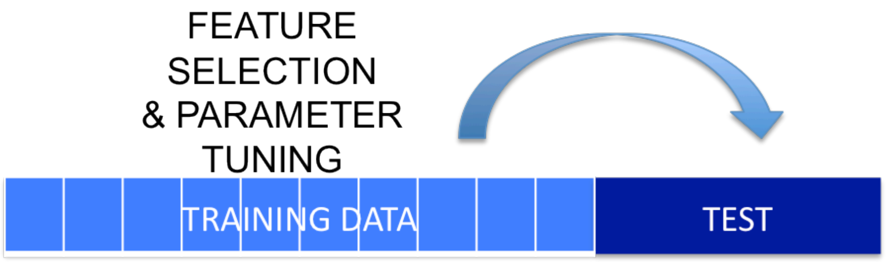
- Load in simulated data .mat files
- Extract Data & Dimensions
- Parse the data into a 3-fold cross validation
- Create Training & Testing Attribute Relation (.ARFF) Files for WEKA

ARFF files are created using the create_arff_nitp() function. This function reads in data, and parses it correctly into the Weka format. This function requires a path definition in order to know where to place the output Weka files. In order to run the script, you will need to make sure that you have set your path correctly.

If you like, you can check to see that your data was loaded into Matlab correctly. You can use the image() command for this, along with the colorbar() specification. For usage, type help at the Matlab command line.

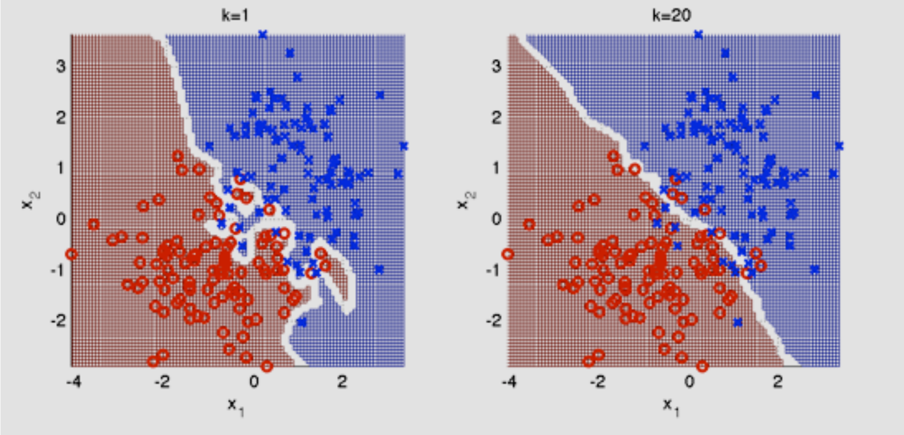
Now run the script.

The script should have created 2 Weka (.ARFF) files – one for training & one for testing. NOTE: we are only performing the first fold of an "outer" 3 fold cross validation here. Running additional folds can get tedious, and should therefore be run at the command line.



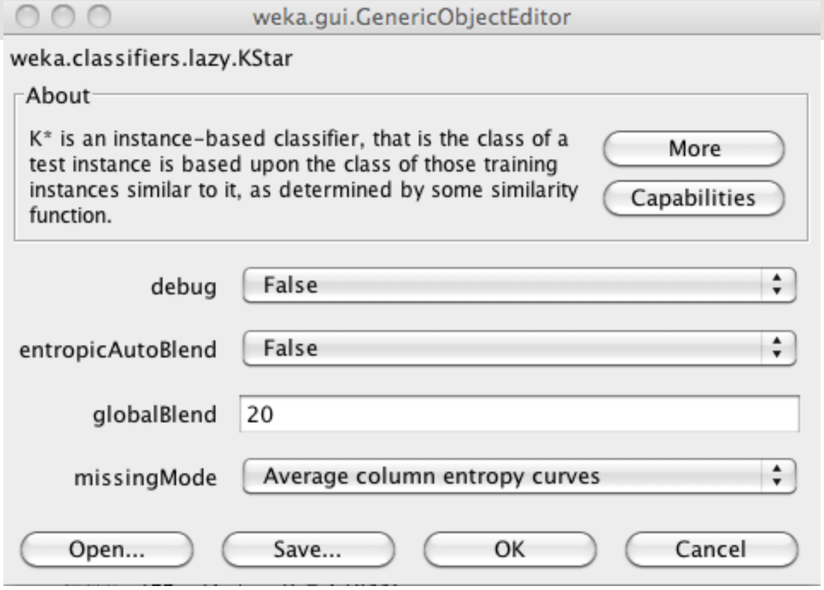
3.3. Tune Parameters Using a Nested 10-fold cross validation

Why perform parameter tuning? 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 out a 10 fold nested cross validation wrapped with an outer 3-fold cross validation in WEKA. Make sure to load the training set to tune your parameters on. 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.



weka.classifiers.lazy.KStar

About

K* is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function.

debug: False

entropicAutoBlend: False

globalBlend: 20

missingMode: Average column entropy curves

→ Try changing the 'C' parameter in SVM. Does this make a difference? How about the global blend parameter in K*?

3.4. Remove Redundant Features

As noted earlier, our data are symmetric about the diagonal. Therefore by removing either the lower or upper triangle of the matrix, we reduce redundancy in the data.

→ Change the "create_arff_nitp" function to the one that's been commented out, "create_arff _nitp_NR." You will also need to change the commenting on the name of your .ARFF file, so that the original is not written over. Now run the script again. You should have produced 2 new .ARFF files.

The non-redundant ARFF Script should have the lower triangle as well as the diagonal of features removed. When you load your new .ARFF files, make sure that the number of features makes sense.

→ Repeat the exercise above by trying out your parameter tuning on your training set via the nested 10-fold cross validation. Does the accuracy change with less redundant features?

3.5. Further Explorations

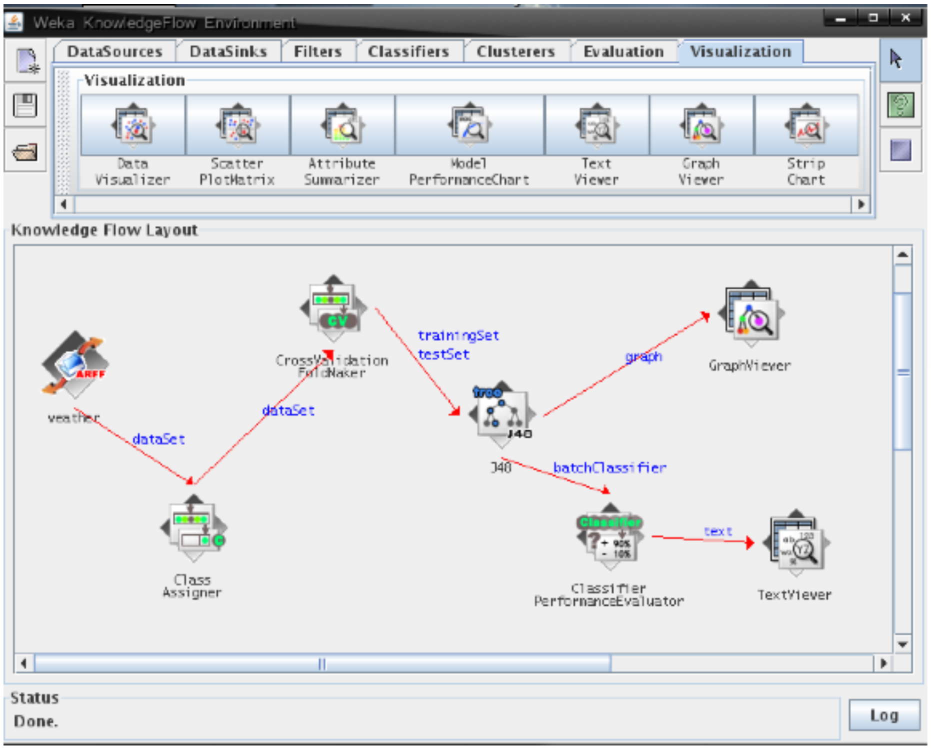
If time permits, you might also try WEKA's feature selection tools under the "SELECT ATTRIBUTES" menu. Try the linear forward search. Do the "informative features" found match the ones that you would expect by visual inspection of the matrix?

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.

To see what command you should use – try holding your curser over the main classifier window. WEKA should give you the command needed. This comes in handy when running a number of optimizations.

Interpretation of Classifier Weights and Outputs

You might also want to try Weka Knowledge Flow, as it may be useful in visualizing the structure and/or weights of your classifier model.



Useful Resources & References

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

- ADNI (Alzheimer's Disease Neuroimaging Initiative)
- ADHD 200

General Machine Learning in Neuroimaging

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

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

Neuroimaging & WEKA ☺

Pamela Douglas et al. "Single Trial Decoding of Belief Decision Making from EEG and fMRI Data Using ICA Features" Frontiers in Human Neuroscience (2013 in press)

Pamela Douglas, Sam Harris, Alan Yuille, Mark S. Cohen "Performance Comparison of Machine Learning Algorithms and Number of Independent Components Used in fMRI Decoding of Belief vs. Disbelief" *Neuroimage* 2011 May; 56(2): 544-53. **** Also Decoding using ICA*

Kerr, W., Anderson, A., Lau, E.P., et al. "Automated Diagnosis of Epilepsy Using EEG Power Spectrum." *Epilepsia (in press)*