



Data Mining and Machine Learning

Course Work 3

Issa Haddad

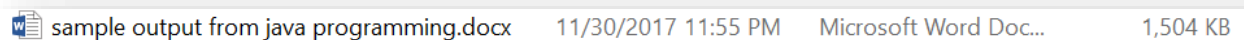
H00278537

Nov 30, 2017

Code and screenshots are all available in the link mentioned below. The file is structured in a way that is easy to follow. Code folder source classes are also attached, I used it to prepare the files for Weka loading and to build all classifiers in this coursework. You can open the code project file in eclipse to view code for entire coursework once. Results folder includes screenshots, and output results. Results folder is also divided in such a way to incorporate a file for each item in this coursework.

**Note:** I used programming (java in eclipse) to prepare the code for Weka. I also used programming (java in eclipse) to build all classifiers.

**Note:** I did the programming for the classifiers after I finished performing all the tasks in this coursework using Weka GUI. So screenshots are taken from Weka GUI, but I performed Sample run on each of the classifiers using Java programming, and I recorded their screenshots in a word document called: They match exactly with the outputs of Weka.



sample output from java programming.docx 11/30/2017 11:55 PM Microsoft Word Doc... 1,504 KB

- I programmed J48 & LMT tree using 10-fold cross validation + test (exactly like I did with Weka). I also took care of applying the correct classifier parameters for J48
- I programmed NN-multilayer perceptron using the training set + test (exactly like I did with Weka). I also took care of applying the correct classifier parameters for multilayer perceptron

Link: <https://www.dropbox.com/sh/iac31epwe1nhr5i/AAAouYekuhJbCcmLcWWFlvrFa?dl=0>

## Variation in performance with size of the training and testing sets

Before starting, I prepared the ARFF files using Weka embedded java in eclipse. I did so by using the 5 steps mentioned below:

1. Convert CSV to ARFF
2. Keeping the 10 top attributes from each emotion
3. Removing some of the instances using the filter: “unsupervised RemoveMisClassified instances”.
4. Randomize the instances
5. Creating 3 datasets by moving 30% of the testing instances to the training set each turn.

I ended up with 3 datasets that have the following number of instances:

1. Dataset 1: Training = 6160, Testing = 1611
  2. Dataset 2: Training = 6643, Testing = 1128
  3. Dataset 3: Training = 7126, Testing = 645
- I ran 10-fold cross validation on J48 and LMT decision trees + testing set
  - I ran training set on neural networks + testing set

		Dataset 1	Dataset 2	Dataset 3
J48	Acc. on training	76.7532%	74.6237%	72.8076%
	Acc. on testing	55.4313%	58.4738%	57.764%
LMT	Acc. on training	82.2078%	80.1023%	78.6446%
	Acc. on testing	57.3557%	61.5794%	66.9255%
NN	Acc. on training	70.0325%	68.6033%	69.5384%
	Acc. on testing	62.1974%	62.9104%	62.5776%

In all of the classifiers used above, we can notice that as we increase the size of the training set and reduce the size of the testing set, accuracy behaves vice versa. Accuracy decreases for training and increases for testing. It is also quite clear that overfitting decreases as we increase the size of the training set. This is because the model is now tested on a bigger chunk, so the probability of it performing better on the test set increases.

I have used the LMT decision tree which applies both the logistic prediction and the decision tree. It also uses linear regression model on its leaves [1]. Although this is why it yields better accuracy than J48, it also takes longer periods of time to execute the model. On average J48 took 1.15 seconds to load a model while LMT took around 60 seconds on average per model. LMT was better than J48 but was 52 times slower in executing its model.

### Variation in performance with change in the learning paradigm (Decision trees versus Neural Nets)

In general, both the neural networks and the decision tree apply supervised techniques to classify our instances. Decision trees allow for an easier extraction of the knowledge when compared to NN. That is, just by looking at the tree, you can clearly understand how the data is split and how the different instances are classified [2]. This is not as easy using neural networks.

Referring to the table from the previous section, we can conclude that although the J48 was better at classifying the training set, it was definitely worse at properly classifying the test set. J48 is clearly more prone to overfitting than NN. Thus we can conclude that although neural networks are less readable than J48, it was better overall in correctly classifying the instances.

### Variation in performance with varying learning parameters in Decision Trees

Binary Split	parameter	False				True			
	Number of leaves	656				656			
	Size of tree	1311				1311			
	Accuracy on training	76.7532%				76.7532%			
	Accuracy on testing	55.4314%				55.4314%			
unpruning	parameter	False				True			
	Number of leaves	656				656			
	Size of tree	1311				1311			
	Accuracy on training	76.7532%				76.9156%			
	Accuracy on testing	55.4314%				55.4314%			
Confidence threshold	parameter	0.005	0.05	0.01	0.25	0.5	1.0		
	Number of leaves	533	630	635	656	656	656		
	Size of tree	1065	1259	1269	1311	1311	1311		
	Accuracy on training	74.789%	76.3312%	76.54%	76.75%	76.76%	76.86%		
	Accuracy on testing	56.4246%	55.8659%	55.61%	55.43%	55.43%	55.43%		

Minimum objects	parameter	1	2	3	5	10	20	50	100
	Number of leaves	657	656	573	407	233	105	34	18
	Size of tree	1313	1311	1145	813	465	209	67	35
	Accuracy on training	81.4773%	76.7532%	74.17%	71.83%	67.43%	64.85%	62.59%	60.6%
	Accuracy on testing	55.1831%	55.4314%	54.81%	56.36%	56.797%	57.25%	60.83%	59.34%

Binary split: This setting forces the splitting of the nodes to only two branches [3]. It did not affect the size nor the performance of the tree. That is because this parameter is only used for nominal attributes.

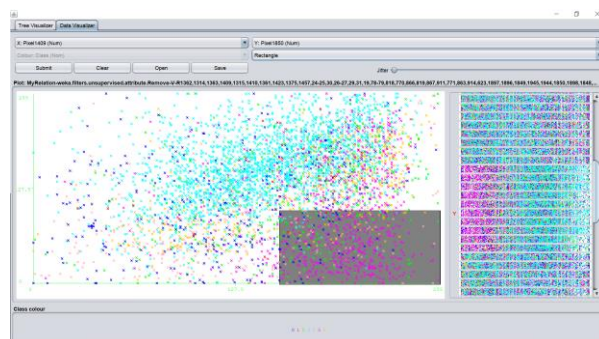
Pruning: does not affect the performance, if it is enabled, it tries to remove nodes and branches without affecting the performance of the algorithm.

Confidence threshold: confidence factor affects the amount of post pruning proportionally. We can notice from the graph that as the confidence factor increases, the size of the tree increases up to a certain factor. We can also notice that accuracy on test set decrease up to the same level. Overfitting increases as accuracy on test set decreases. Overfitting and tree size seem to converge at confidence factor = 0.25

Minimum objects: minimum number of instances per leaf. Number of leaves rapidly as the minimum number of permissible instances increase. Also, the size of the tree follows accordingly. Similarly, overfitting is reduced as the accuracy on the testing set increases and the accuracy of the training set decreases as the parameter increases.

I have used the LMT decision tree which applies both the logistic prediction and the decision tree. It also uses linear regression model on its leaves. Although this is why it yields better accuracy than J48, it also takes longer periods of time to execute the model. On average J48 took 1.15 seconds to load a model while LMT took around 60 seconds on average per model. LMT was better than J48 but was 52 times slower in executing its model.

User classifier enabled me to create my own tree by manually splitting the data. Screenshots for my implementation of it can be viewed in the link mentioned above. Using 2-fold cross validation, I only got 46.1% on the training set and 53.4% on the test set. See example screenshot below.



Correctly Classified Instances	2844	46.1688 %
Incorrectly Classified Instances	3316	53.8312 %
Kappa statistic	0.173	
Mean absolute error	0.1992	
Root mean squared error	0.3167	
Relative absolute error	91.854 %	
Root relative squared error	96.172 %	
Total Number of Instances	6160	

--- Detailed Accuracy By Class ---									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.647	0.126	0
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.488	0.016	1
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.524	0.108	2
0.855	0.633	0.476	0.855	0.611	0.242	0.710	0.578	0.578	3
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.653	0.135	4
0.557	0.202	0.425	0.557	0.482	0.324	0.729	0.381	0.381	5
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.561	0.099	6
Weighted Avg.	0.462	0.297	0.281	0.462	0.348	0.166	0.667	0.356	

## Variation in performance with varying learning parameters in Neural Networks

epochs	parameter	50	100	200	500	1000	2000		
	Accuracy on training	66.67%	67.46%	68.97%	70.0325%	70.74%	72.02%		
	Accuracy on testing	64.86%	63.62%	63.19%	62.1974%	61.26%	61.20%		
Hidden layer	parameter	1	2	3	5	10	30	50	100
	Accuracy on training	55.66%	59.26%	62.06%	65.19%	70.03%	80.66%	84.88%	88.96%
	Accuracy on testing	56.92%	59.09%	61.01%	61.63%	62.19%	62.25%	61.82%	62.25%
Learning rate	parameter	0.01	0.05	0.1	0.2	0.3	0.5		
	Accuracy on training	68.08%	70.12%	70.82%	70.53%	70.03%	68.94%		
	Accuracy on testing	64.30%	62.44%	61.63%	61.26%	62.19%	62.13%		
momentum	parameter	0.01	0.05	0.1	0.2	0.5			
	Accuracy on training	70.35%	70.61%	69.87%	70.03%	69.82%			
	Accuracy on testing	61.94%	61.32%	62.25%	62.19%	61.39%			
Validation threshold	parameter	1	2	5	10	20	50		
	Accuracy on training	70.03%	70.03%	70.03%	70.03%	70.03%	70.03%		
	Accuracy on testing	62.19%	62.19%	62.19%	62.19%	62.19%	62.19%		

Epoch: is in the number of iterations that in which the data is used to train the network. As the parameter increase the accuracy on the training set increases as well. The accuracy on the test set decreases, that means that overfitting is increasing as the parameter increases.

Hidden layer: I used 1 hidden as it is enough to solve a nonlinear problem of that sort. The accuracy on both the training set as well as the test set increases up to a certain value and then overfitting starts to take over after that. The best number of neurons in that hidden layer I found was roughly equal to half the number of attributes in my data.

Learning rate: affects how quickly I will find a solution or how efficiently. If I increase learning rate, there is a chance that I might overshoot the optimum solution. If I decrease it. It will take more time to find a solution, but it will find a better one. From the graph we can see that overfitting is least when the parameter is lowest.

Momentum: is a multiplier of the learning rate, thus affects speed of finding solution too [4]. We can see from the graph that the overfitting is least when the value is between 0.1 and 0.2.

Validation threshold: a threshold that specifies when the error rate will cause the whole training validation epoch to be repeated. In our experiment, this threshold did not manipulate the performance of the neural network.

### Variation in performance according to different metrics (TP Rate, FP Rate, Precision, Recall, F Measure, ROC Area)

TP rate is equivalent to Recall and is also referred to as sensitivity. It measures how much of the real true cases have been detected. Meaning how many of the actual class is predicted as that class. TP rate and Recall change proportionally overall with the change of the accuracy.

FP rate also known as the false alarm is the probability of indicating a given condition exists when it does not. FP Rate changes opposite to accuracy.

Precision or positive predictive value is the ratio of how many of the predicted conditions are actually true. Precision changes proportionally overall with the change of the accuracy.

The F-measure is the weighted average or the harmonic mean between precision and recall, where the value reaches 1 if there is perfect precision and recall, and reaches zero vice versa. F-measure changes proportionally overall with the change of the accuracy.

ROC area is the area under the curve, its measures the probability that randomly chosen positive instance is ranked above randomly chosen negative one. Since the curve is usually a linear shape, we would want a value greater than or equal to 0.7. Roc Area changes proportionally overall with the change of the accuracy.

Accuracy refers to the prediction on all of the classes, the metrics above refer to the prediction of 1 class at a time. Therefore, accuracy alone is not a good enough predictor of the scenario. Different errors can yield different costs. Maybe for some scenarios you are more interested in 1 class than another class. It is always ideal to look at the metrics provided by the confusion matrix on top of the overall accuracy.

ified Instances	4558	68.6033 %						
ssified Instances	2086	31.3967 %						
	0.5757							
error	0.1178							
ed error	0.2549							
te error	54.3589 %							
quared error	77.4268 %							
Instances	6644							
curacy By Class ==>								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.354	0.035	0.483	0.354	0.409	0.368	0.767	0.414	0
0.080	0.002	0.421	0.080	0.134	0.179	0.733	0.117	1
0.314	0.064	0.357	0.314	0.334	0.265	0.746	0.329	2
0.894	0.151	0.800	0.894	0.844	0.732	0.934	0.890	3
0.509	0.039	0.579	0.509	0.542	0.498	0.835	0.522	4
0.810	0.081	0.725	0.810	0.765	0.701	0.917	0.794	5
0.492	0.036	0.576	0.492	0.531	0.490	0.840	0.541	6
0.686	0.094	0.665	0.686	0.671	0.594	0.876	0.694	

An example from the confusion matrix obtained by running multilayer perceptron on the second dataset. Let's say we are interested in class no.1, from an initial look you might think that the accuracy is acceptable with 68.6%, however, we can notice that the TP rate for class 1 is 0.80. this means that although accuracy is relatively good, class no. 1 was misclassified for the majority of the time.

## Research Question

### Comparative analysis of Neural Networks and Deep Neural Networks

Ever since Threshold logic was introduced in 1943 by Warren McCulloch and Walter Pitts to produce computational models for neural networks. NN is a supervised technique that has long been used to perform nonlinear statistical modelling. They have been used to discover complex relationships among both dependent and independent variables. Artificial neural networks were and still are a very successful method in solving many different classification and forecasting problems. However, ANN cannot determine relationships between input and output, and only relies on incremental improvements of accuracy, hence they are extremely prone to overfitting [5].

Neural Networks used to work with no more than two hidden layers, as more than that seemed impractical due to diminishing returns caused by the vanishing gradient problem. In ANN, the function of the derivative of each layer determines the updated weights on the next layer, therefore the update signal decreases as more layers are added [6]. Only with recent advancement in hardware processing powers, algorithmic techniques, and big data has this issue been resolved and the use of more hidden layers facilitated. This allowed for the emergence of a new application called the deep neural networks.

In 2006, these advancements in technology have paved the way for Geoff Hinton, Osindero, and Teh to exploit this many-layered deep neural network to the world's advantage. These researchers have managed to train this deep neural network in an unsupervised fashion before training it in a supervised fashion. This permitted the deep neural network to build on the concept of feature engineering.

Deep neural network name has found its way to a new sophisticated term called deep learning. The new architecture of deep learning allows it to build complex, hierarchical features of data. Because of its numerous hidden layers, deep learning allows itself to work on abstract raw data or high level representation of data. Each added hidden layer improves the process of pattern recognition through the algorithm itself [7].

Because we often work with very complex nonlinear structural data, deep learning has been adopted by many industrial sectors. For instance, deep learning has shown direct significant improvements on the advancements of various applications including; computer vision, speech recognition, medicine, natural language processing, and much more. It has a tremendous effect in revolutionizing the technology we use nowadays [8].

In a nutshell deep learning is a fairly new field in machine learning that has only evolved in the late twenty first century. Deep learning can be thought of as an improved branch of neural networks whose strengths and capabilities are yet to be stretched in future. Unlike traditional neural networks that are only used for supervised classification, deep learning can be used as either supervised, unsupervised, or both. It enables us to use feature extraction and make more sense of the input of the neural network. Most importantly, it enables us to work with high level raw or abstract data.



## Reference:

- [1] Purva Sewaiwar, Kamal Kant Verma :Comparative Study of Various Decision Tree Classification Algorithm Using WEKA. International Journal of Emerging Research in Management &Technology ,Vol 4, pp 2278-9359(2015).
  - [2] Chandra, R., Chaudhary, K. and Kumar, A. 2007. *The Combination and Comparison of Neural Networks with Decision Trees for Wine Classification.*, Lautoka, , Fiji: School of Science and Technology, The University of Fiji.
  - [3] Stiglic G, Kocbek S, Pernek I, Kokol P. Comprehensive decision tree models in bioinformatics. PloS one. 2012; 7(3), e33812 doi: 10.1371/journal.pone.0033812
  - [4] V. V. Phansalkar and P. S. Sastry, "Analysis of the back-propagation algorithm with momentum," in *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 505-506, May 1994.  
doi: 10.1109/72.286925
  - [5] Tu JV. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. J. Clin. Epidemiol. 1996;49:1225–1231. doi: 10.1016/S0895-4356(96)00002-9.
  - [6] Nielsen. M, (2017). Why are deep neural networks hard to train? Retrieved from:  
<http://neuralnetworksanddeeplearning.com/chap5.html>
  - [7] Wang. H, and Raj. B, (2017) On the origin of deep learning. arXiv:1702.07800v4.
  - [8] Parloff. R, (2016). Why Deep Learning is Suddenly Changing Your Life. Retrieved from:  
<http://fortune.com/ai-artificial-intelligence-deep-machine-learning/>
-