PROBLEM STATEMENT

Compare the performance (quality and computational effort) of LVQ, Backprop, and SVM on a dataset with at least ten input attributes, e.g., chosen from the UCI database. How sensitive to algorithm parameter are the results of each algorithm?

ANALYSIS

1. DATASET

Using Cardiotocography dataset from the UCI Machine Learning Dataset Repository for this experiment and below is the link and brief description of the Dataset. The three classes are NSP - fetal state class code (N=normal; S=suspect; P=pathologic) represented by 1,2,3 for N, S, P respectively.
http://archive.ics.uci.edu/ml/datasets/Cardiotocography

	Number of instances	Number of attributes	Number of Classes
Cardiotocography	2126	23	3

2. CODE

Modified code of backpropagation [1] and learning vector quantization [2] and combined the code [3] to call backpropagation and learning vector quantization algorithms with same training and testing data subsets.

LVQ: Euclidean distance between every pair of data points is calculated and the pair of data points are said to be similar if the distance is smaller. LVQ learns patterns from the training data and such pattern is called a codebook. A library of such patterns is called a codebook vector. Initially these codebooks are randomly initialized and as data points arrive distance between these codebook vectors and the data point is calculated. Most similar codebook vector (least distance) is considered as the best matching unit(BMU). The randomly initialize codebooks iteratively adapt to model the training data through the number of trials or epochs, which for each epoch adapts the model to each of the data points and within that through each of the features in each data point resulting in a training pattern. The difference between such a training pattern and the BMU represents the error. For a class match this error is added to BMU (to make it closer to pattern) or else this error is subtracted from BMU (to drive it away from the pattern). As per [2] our lvq code uses linear decay learning rate.

Backpropagation: This algorithm initializes a network (each neuron with weight vector and a bias), for each data point there is forward propagation (constituting neuron activation, neuron transfer using sigmoid function and forward propagation), evaluation of error based on expected outputs and forward propagated outputs and then back propagate this error to each of the hidden layer neurons to update weights resulting in training of the network through backpropagation.

Line	Code
97	predicted_lvq = learning_vector_quantization(train_set, test_set, *args)
100	accuracy_lvq = accuracy_metric(actual, predicted_lvq)
103	predicted_bkp = back_propagation(train_set, test_set, *args)
106	accuracy_bkp = accuracy_metric(actual, predicted_bkp)

args* refers to the parameters passed to each algorithm. The parameters we change to compare the algorithms are n_epochs (number of iterations or trials each algorithm is run), n_hidden or n_codebooks (number of hidden neurons in case of backpropagation algorithm and number of codebook vectors in case of LVQ algorithm), learning rate and n_folds (cross validation split). We vary these parameters for both LVQ and backpropagation algorithms to compare the performance.

SVM: Used Linear SVM classifier to train a model from 80% of the data points and then tested the model with 20% of the data points. The simplified version of code from [4] has been used for data preprocessing, training and testing is available at [5] to achieve the desired results.

3. EXPERIMENT AND RESULTS

We ran SVM code [5] and found the accuracy to be 91.76% with the computational time of 0.03s. A linear SVM model achieved almost same accuracy with any number of trials. So we are comparing the performance of backpropagation and LVQ by varying their parameters with the fixed results of SVM as mentioned earlier.

Parameters:
Trails: 100 (epochs from 1 to 100), 10 folds, 0.1 learning rate, significance level $p<0.05$

After running both backpropagation and LVQ with same parameters as mentioned above and only changing the count of hidden neurons (backpropagation) and the count of codebook vectors(lvq) we computed the mean of the accuracies and did a t-test on the results of 100 such means from 100 epochs and have listed the results in below table. An increase in the number of hidden neurons reduces the overall accuracy for backpropagation (figure 1) where as an increase in the number of codebook vectors for LVQ is increasing the accuracy, though not by a significant factor (figure 2).

	Mean	t-test
Backpropagation (10 hidden layers)	86.65	t = 7.54
LVQ (10 codebook vectors)	80.94	p<0.00001
Backpropagation (15 hidden layers)	85.79	t = 3.83
LVQ (15 codebook vectors)	82.32	p<0.00017
Backpropagation (20 hidden layers)	73.48	t = -5.43
LVQ (20 codebook vectors)	83.34	p<0.00001

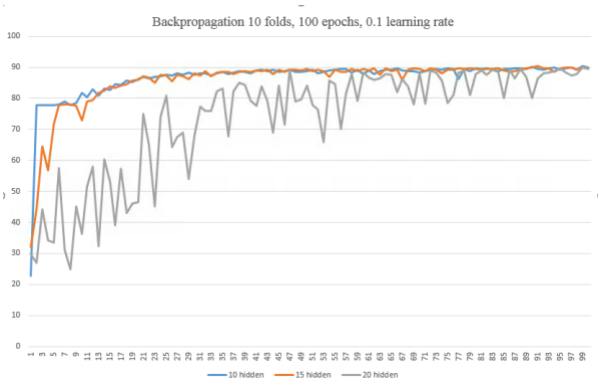

Figure 1

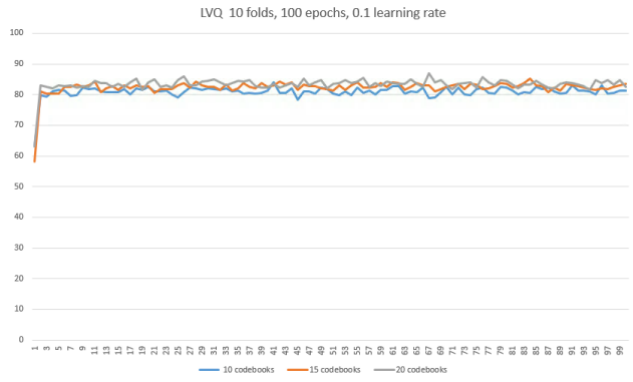

Figure 2

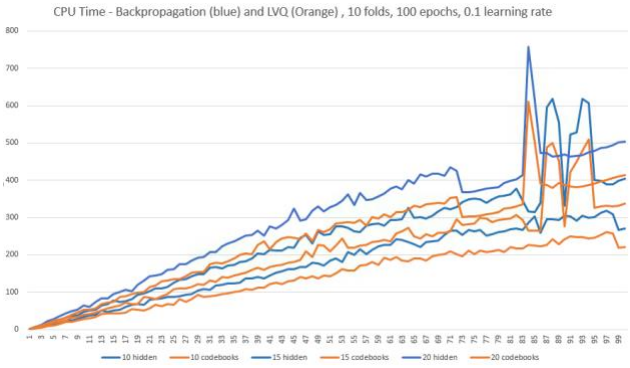

Figure 3

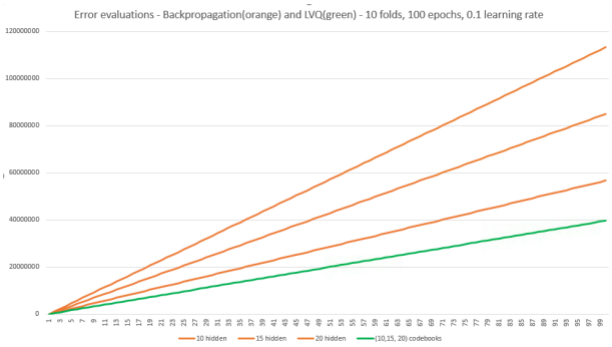

Figure 4

From figure 3 it is clear that LVQ is significantly faster than backpropagation even with the increasing number of hidden neurons or codebook vectors. Since the number of times the codebook vectors get updated is same irrespective of their number as per the green line in figure 4 and the corresponding error updating in case of backpropagation is found to be higher than learning vector quantization and also increase when the number of hidden neurons increases.

Parameters:
Trails: 100 (epochs from 1 to 100), 10 hidden (backpropagation) and 10 codebooks (learning vector quantization), 0.01 learning rate, significance level $p<0.05$

After running both backpropagation and learning vector quantization with same parameters as mentioned above and only by changing the number of folds of cross-validation splits for both backpropagation and learning vector quantization we computed the mean of the accuracies and did a t-test using the online tool [6] on the results of 100 such means from 100 epochs and have listed the results in below table. The details of all t-test calculations are available at [7]. The results of t-test in each category of 5, 10 and 15-fold cross validation are listed and are found to be very closer. An increase in the folds of cross-validation split has found not much change in the accuracy of backpropagation (though it is found to increase a little bit) or the accuracy of learning vector quantization (which is also increasing a little bit).

	Mean	t-test
Backpropagation (5 fold)	76.9	t = -5.6
LVQ (5 fold)	80.78	p<0.00001
Backpropagation (10 fold)	76.77	t = -5.4
LVQ (10 fold)	80.82	p<0.00001
Backpropagation (15 fold)	77.13	t = -5.66
LVQ (15 fold)	80.89	p<0.00001

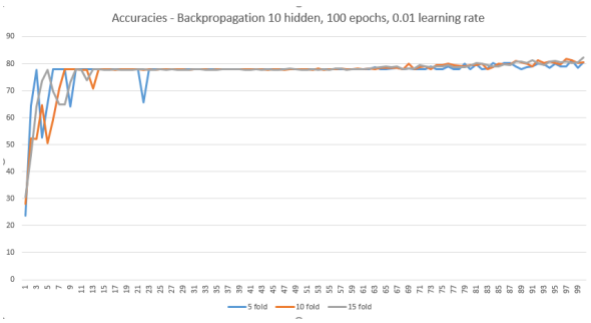

Figure 5

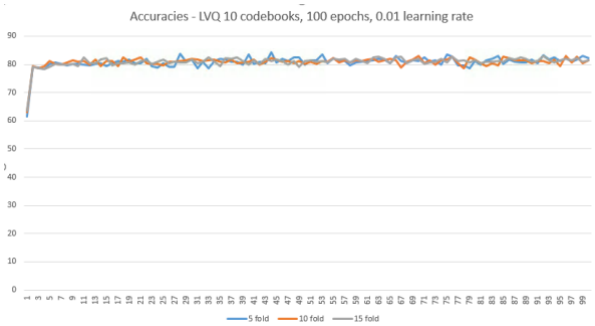

Figure 6

Figure 5 and 6 reiterate the insignificant change in the accuracies of backpropagation and learning vector quantization by increasing the number of folds of cross-validation split of the dataset used to train the models.

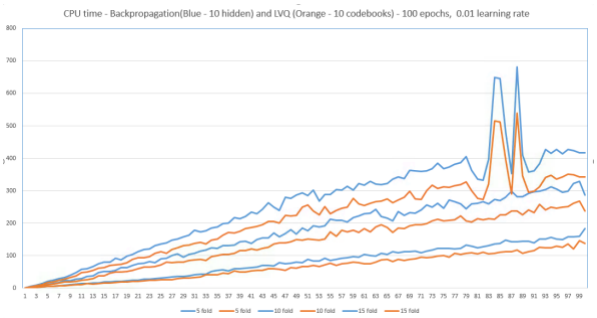

Figure 7

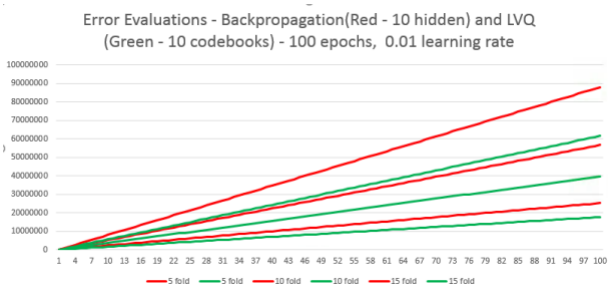

Figure 8

However, the computational effort in terms of CPU time is found to be higher in case of backpropagation compare to corresponding learning vector quantization as shown in figure 7. Error evaluations for backpropagation was found to be greater when compared to learning vector quantization as shown in figure 8. And the change in case of learning vector quantization despite the size of codebooks vector being same can be attributed to the number of folds of cross-validation split.

Parameters:

Trails: 100 (epochs from 1 to 100), 10 fold, 10 hidden (backpropagation) and 10 codebooks (learning vector quantization), significance level $p<0.05$.

With above parameters being same for both backpropagation and learning vector quantization but by changing the learning rate by 10 times we got the below average of accuracies. A t-test was performed on these accuracies obtained from 100 epochs.

	Mean	t-test
Backpropagation (0.01 learning rate)	76.77	t = -5.4
LVQ (0.01 learning rate)	80.82	p<0.00001
Backpropagation (0.1 learning rate)	86.65	t = 7.54
LVQ (0.1 learning rate)	80.94	p<0.00001

Learning vector quantization was found to be consistent which can be attributed to the quick decay in learning rate after few iterations as shown in figure 9 (orange-0.01 and black-0.1). However, backpropagation with learning rate of 0.1 was found to perform better than itself and learning vector quantization (one green curve-0.1 above all other curves).

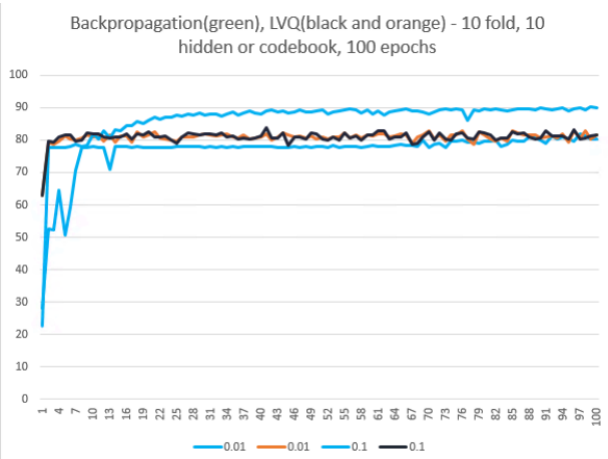

Figure 9

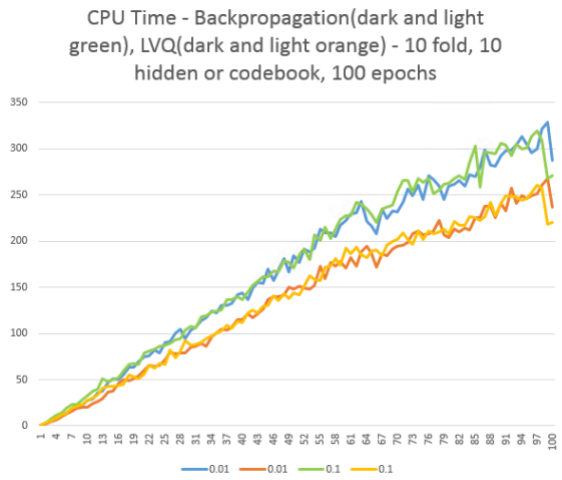

Figure 10

Yet again backpropagation took more CPU time irrespective of the learning time when compared to learning vector quantization as show in figure 10.

4. CONCLUSION

From our test we conclude the following outcomes.

- Learning vector quantization is significantly faster in training when compared to backpropagation and support vector machines. Hence computationally learning vector quantization did better.
- Quality wise SVM did better than other algorithms by achieving higher accuracy. This could be mainly due to lesser data points (only 2126) which might be less for training neural networks like backpropagation and learning vector quantization.
- Backpropagation showed significant improvement with increased learning rate as an algorithm parameter when compared to the other two algorithms.
- Learning vector quantization was found to achieve good results even with lesser number of codebook vectors indicating the fact that it was able to summarize the entire dataset and fit in such small number of codebook vectors.
- With lesser number of iterations learning vector quantization was found to achieve better accuracies much quicker than backpropagation.

5. REFERENCES

[1] Backpropagation code: https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

[2] LVQ code: https://machinelearningmastery.com/implement-learning-vector-quantization-scratch-python/

[3] My code Backpropagation and LVQ: https://github.com/sheshappanavar/ComparisonofLVQBackpropSVM/blob/master/LVQ_Backprop.py

[4] SVM Code: https://github.com/DataMiningPolyuProject/Cardiotocography/blob/master/SupportVectorMachine.R

[5] My SVM: https://github.com/sheshappanavar/ComparisonofLVQBackpropSVM/blob/master/SVM.R

[6] t-test tool: http://www.socscistatistics.com/tests/studentttest/Default2.aspx

[7] https://github.com/sheshappanavar/ComparisonofLVQBackpropSVM