Application of Shared Weight Topology and Genetic Algorithms in Feed-forward Neural Networks for Multi-class Vehicle Type Classification

Renfei Yang

Research School of Computer Science, Australian National University
u6209973@anu.edu.au

Abstract. Neural networks are widely used in the multi-class classification problems. However, the potential noise in the dataset and the instability of the data are two main problems that will influence the performance of neural networks. Genetic algorithms apply mechanisms inspired by biological genetic to determine the quality of the solutions, which can be used to select useful features. Besides, shared weight topology can contribute to enhancing the robustness of the data. In this paper, genetic algorithms and shared weight topology are applied in feed-forward neural networks for multi-class vehicle classification. The comparisons of the evaluation results demonstrate the practicability of genetic algorithms while shared weight topology will influence the performance of neural networks. Finally, the results are compared with another academic paper on the same dataset.
Keywords: Neural Networks, Genetic Algorithms, Shared Weight Topology and Standard Scores

1 Introduction

Multi-class vehicle classification, which refers to recognitions of vehicle types, is widely applied to transportation management to detect potential traffic congestions and efficiently manage the transportation. Recently, this technique has also been used into the research of autonomous cars. The accurate recognition of vehicles is the significant part of designing autonomous cars, which can greatly avoid car accidents (Sun, Bebis & Miller, 2006). Some researchers have already applied some machine learning methods, like support vector machines (Zhong & Fukushima, 2007), in order to get better recognition rate. However, the recognition rates those methods achieve are not high enough for applying on autonomous cars. In this paper, a feed-forward neural network is designed to achieve better recognition rate. However, large existences of the noise and instable encrypting the underlying structure of the raw data will influence the process of learning. Shared weight topology derives from the reduction in free parameters and the faster training of the input to hidden weights (Gedeon, Catalan & Jin, 1997). Besides, genetic algorithms apply mechanisms inspired by biological genetic to determine the quality of the solutions, which can be used to select useful features (Wang, Wu, Liu, Cao & Xie, 2017). Both of them may contribute to learning. Therefore, genetic algorithms and shared weight topology are applied in this paper to reduce the noise and enhance the data's robustness.

Statlog Vehicle Silhouettes dataset (Mowforth & Shepherd, 1993) is used in this paper for vehicle type recognition. I use this dataset because it is a famous multi-class vehicle classification dataset and features in this dataset are interesting. These features are from measurement and there are some physical significances in them. In detail, there are 18 attributes, 946 instances but 100 are retained in case of dispute, thus, there are actually 846 instances available. These attributes include compactness, circularity, etc. Besides, a double decker bus, Cheverolet van, Saab 9000 and an Opel Manta 400 are 4 target classes for classification. This particular combination of vehicles is chosen with the expectation that these four vehicles are easy to distinguish while the differences between the cars are difficult to distinguish. Considering the data directly from measurement, some errors can hardly be avoided. More seriously, some features might mislead the recognition. In order to deal with those features, some pre-processing methods applied in this paper are expected to largely decrease the effects of those aspects.

In order to solve the problem of recognition, I design a feed-forward neural network and use standard scores, genetic algorithms and shared weight topology to extract useful and robust features to do classification. Then, I utilize precision, recall, F1 score and test accuracy to evaluate the original and improved neural networks. The results demonstrate the practicability of genetic algorithms while shared weight topology will influence the performance of neural networks. Finally, I compare my results with another academic paper's results on the same dataset.

2 Methods

2.1 Data Pre-processing

There are 9 original data files. I read them into a .csv file. Some mathematical analysis about this dataset is shown in **Table 1**. The different attributes range widely, which may lead to irrational distribution of the weights. Thus, they influence the training. In order to solve this issue, the basic principle is to avoid encrypting the underlying structure of the data (Bustos & Gedeon, 1995). Therefore, I use standard scores to normalize the raw data, which can contribute to decryption of the raw data. The standard scores can be calculated as:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where z is standard score, x is the original data, μ is the mean of the population and σ is the standard deviation of the population. Besides, I replace 4 target class names with 0-3 in order to do classification.

Table 1. Summary of Raw Data

Attributes	Min	Max	Mean
COMPACTNESS	73	119	93.7
CIRCULARITY	33	59	44.9
DISTANCE CIRCULARITY	40	112	82.1
RADIUS RATIO	104	333	168.9
PR. AXIS ASPECT RATIO	47	138	61.7
MAX.LENGTH ASPECT RATIO	2	55	8.6
SCATTER RATIO	112	265	168.8
ELONGATEDNESS	26	61	40.9
PR. AXIS RECTANGULARITY	17	29	20.6
MAX.LENGTH RECTANGULARITY	118	188	148
SCALED VARIANCE (MAJOR)	130	320	188.6
SCALED VARIANCE (MINOR)	184	1018	439.9
SCALED RADIUS OF GYRATION	109	268	174.7
SKEWNESS ABOUT MAJOR AXIS	59	135	72.5
SKEWNESS ABOUT MINOR AXIS	0	22	6.4
KURTOSIS ABOUT MINOR AXIS	0	41	12.6
KURTOSIS ABOUT MAJOR AXIS	176	206	188.9
HOLLOWS RATIO	181	211	197.4

2.2 Neural Network

I design a feed-forward neural network with shared weight topology in order to achieve better test accuracy. Besides, I will introduce my decisions on the activation function, loss function and optimizer.

2.2.1 Neural Network Architecture

I utilize shared weight topology implementing a neural network to do some pre-processing on training data in order to improve the data's robustness. The neural network contains 18 input neurons, 15 hidden neurons and 18 output neurons. I follow the simplest method (Gedeon, Catalan & Jin, 1997) to back-propagate errors and update weights as in standard back-propagation without shared weight, and then after the values of the input-hidden weight and hidden-output weight have diverged to average their values. I train the neural network until it converges and apply the output of it as the input for the second neural network.

As for the second neural network, I implement a 2-layer neural network and a 3-layer neural network. There are 18 input neurons, 36 hidden neurons and 4 output neurons for the 2-layer neural network. As for the 3-layer neural network, it contains 18 input neurons, 36 hidden neurons for each hidden layer and 4 output neurons. Then, I compare their test accuracies and find there are not significant differences between them, while 2-layer neural network can learn faster because fewer hidden neurons largely reduce calculation amount. Therefore, I use the 2-layer neural network for the training.

Finally, I compare the results between using shared weight topology with the neural network and only using the neural network. **Fig. 1** shows the whole process.

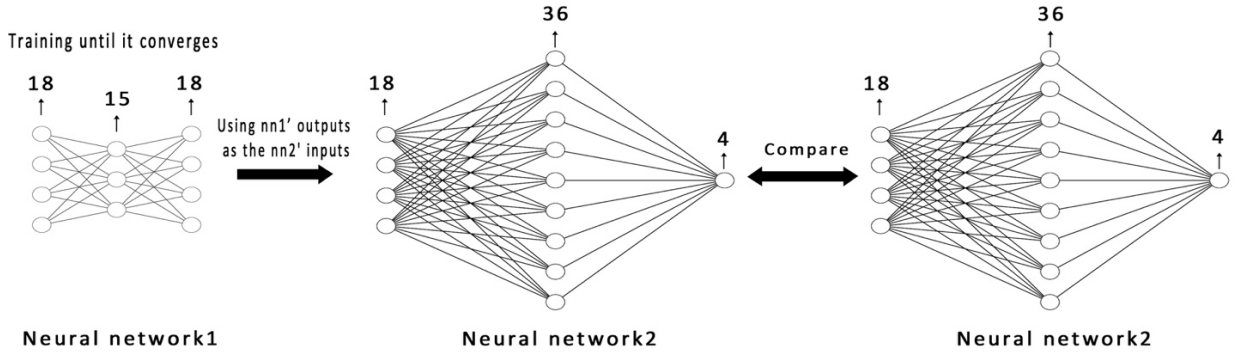

Fig. 1. Neural Network Architecture with Shared Weight Topology or Not

2.2.2 Activation Function

As for the hidden neurons, I use Rectified Linear Unit (ReLU) as the activation function. The ReLU can be calculated as:

$$\mathrm{f(x)} = \max(x, 0) \tag{2}$$

A ReLU has output 0 if the input is less than 0, and raw output otherwise. The ReLU loses less information during training and it is efficient for training large neural networks (Ramachandran, Zoph & Le, 2018). I compare Sigmoid activation function and ReLU activation function. Actually, there is not a big difference for the test accuracy.

As for the output neurons, I use Softmax as the activation function. Softmax can be calculated as:

$$\sigma(\mathrm{z})_f = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3}$$

where z is a vector of the inputs to the output layer, K is the number of dimensions of the vector and j is the index of each dimension of z. Softmax performs well on multi-class classifications.

2.2.3 Loss Function

In the shared weight topology, I use MSE Loss, which is for regression problems, as the loss function. MSE Loss measures the mean squared error between n elements in the input x and target y. It can be calculated as:

$$\ell(\mathrm{x}, \mathrm{y}) = \{l_1, \ldots, l_N\}^T, \qquad l_n = (x_n - y_n)^2 \tag{4}$$

In the second neural network, I use Cross Entropy Loss, which is for classification problems, as the loss function. The losses are averaged across observations for each minibatch. Besides, it is particularly useful when you have an unbalanced training set. Cross Entropy Loss for an instance can be calculated as:

$$\mathrm{loss(x, class)} = -x[class] + \log\left(\sum_j \exp\left(x[j]\right)\right) \tag{5}$$

2.2.4 Optimizer

In order to achieve better test accuracy and train within less time, I compare 3 different optimizers. They are Adam, SGD with momentum = 0 and SGD with momentum = 0.9 respectively. Adam (Kingma & Ba, 2014) combines the advantages of two recently popular optimization methods: the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objectives. The results in **Fig. 2** do indicate Adam not only can achieve better test accuracy but also can learn faster.

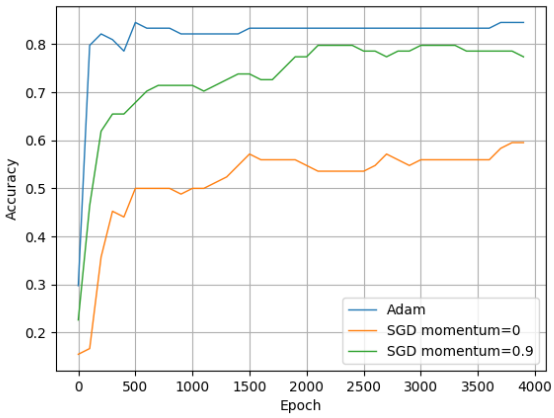

Fig. 2. Comparison of Optimizers

2.2.5 10-fold Cross-validation

The test accuracy ranges from 74% to 86% if the data is directly divided into training data and testing data because there are some noises in the dataset. The results fluctuate so significantly that it is difficult to apply any heuristic algorithm in order to achieve better test accuracy, because any result can be explained to be lucky or unlucky.

In 10-fold cross-validation, the original sample is randomly partitioned into 10 equal sized subsamples. A single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds can then be averaged to produce a single estimation (McLachlan, Do & Ambroise, 2005).

In order to make full use of the data and decrease fluctuations, I use 10-fold cross-validation. It only wastes 10% of the data. Besides, the test accuracy only ranges from 80% to 82%, which is more stable.

2.3 Genetic Algorithms

Genetic algorithms often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape. It applies mechanisms inspired by biological genetic to determine the quality of the solutions, which can be used to select useful features. Especially, some attributes in this particular dataset might not contribute to learning at all as all the attributes are directly calculated from the statistics. More than that, they might even destroy the process of learning. Therefore, it may be a good idea to select some useful features by genetic algorithms rather than using all of them.

In this paper, I randomly generate 100 chromosomes as the population, choose 0.8 as cross rate and 0.002 as mutation rate respectively. Besides, there are 18 binary genes, which stand for 18 input features, in each chromosome. As for selection, I use propositional selection to choose chromosome. There is a positive correlation between the possibility of a chromosome to be selected and the fitness which is test accuracy in this paper. The possibility can be calculated as:

$$\varphi_s(x_i(t)) = \frac{f_Y(x_i(t))}{\sum_{I=1}^{n_s} f_Y(x_I(t))} \tag{6}$$

where $\varphi_s(x_i(t))$ is the possibility of a chromosome to be selected, $f_Y(x_i(t))$ is a fitness value and $\sum_{I=1}^{n_s} f_Y(x_I(t))$ is total fitness values. Then I randomly replace some genes from one parent with the genes from another parent to generate a child chromosome and remove one parent. After that, I randomly mutate genes with very low possibility. The whole process is shown in **Fig. 3**.

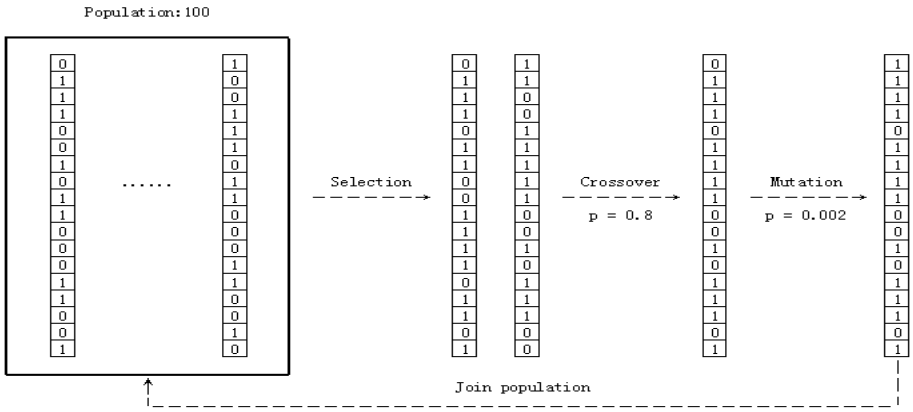

Fig. 3. Genetic Algorithms

3 Results and Discussion

3.1 Evaluation Methods

I use precision, recall, F1 score and test accuracy to evaluate the neural network. They all can be directly calculated from the confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. There are four basic terms:

1. **true positives (TP)**: It correctly predicts the target vehicle type.
2. **true negatives (TN):** It correctly predicts the other vehicle types.
3. **false positives (FP):** It incorrectly predicts the target vehicle type.
4. **false negatives (FN):** It incorrectly predicts the other vehicle types.

The accuracy, precision, recall, f1 score can be calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{9}$$

$$\text{F1 score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{10}$$

The accuracy is only good for symmetric data sets because it can achieve high accuracy by only predicting the most frequent label if the data set is far from symmetric. Precision looks at the ratio of correct positive observations while recall is the ratio of correctly predicted positive events. They both focus on the performance of positives rather than negatives. As for F1 score, it is the harmonic mean of precision and recall, which means it takes both false positives and false negatives into account. Therefore, I use all of them to evaluate my results in order to eliminate biases from only applying one method.

3.2 Evaluation on Multi-Layer Neural Network

The test results during training of the 2-layer neural network (I design in **2.2.1**) and 3-layer neural network (I design in **2.2.1**) are shown in **Table 2**. The 2-layer neural network converges more quickly than the 3-layer neural network and the final test accuracies are same. Therefore, I choose the 2-layer neural network.

Table 2. Evaluation on 2-layer NN and 3-layer NN

	2-layer neural network				3-layer neural network			
epoch	accuracy	precision	recall	F1 score	accuracy	precision	recall	F1 score
0	0.25	0.24	0.25	0.25	0.21	0.22	0.18	0.19
40	0.68	0.65	0.60	0.62	0.48	0.43	0.47	0.45
80	0.78	0.73	0.79	0.75	0.62	0.64	0.63	0.64
120	0.81	0.82	0.84	0.83	0.71	0.71	0.73	0.72
160	0.82	0.82	0.83	0.83	0.81	0.81	0.83	0.82
200	0.81	0.79	0.81	0.80	0.81	0.80	0.81	0.80

3.3 Evaluation on Shared Weight Topology

I train 2 neural networks with different architectures and evaluate the test results for both of them. The two different neural networks are:

1. The original neural network.
2. The original neural network with the new input which is from the output of shared weight topology.

The **Table 3** shows the evaluation on both neural networks. As the results show, it is not surprised that accuracy, precision, recall and F1 score have decreased to some extent. The reason is the data is compressed when applying shared weight topology. Therefore, these evaluation values all decrease. As for the different proportions of decrease, it is because there are different effects of compression on different types of vehicles.

Table 3. Evaluation on Shared Weight Topology or Not

	original neural network				original neural network with shared weight topology			
vehicle type	accuracy	precision	recall	F1 score	accuracy	precision	recall	F1 score
bus	0.82	0.84	0.86	0.85	0.72	0.72	0.73	0.72
saab	0.75	0.72	0.76	0.74	0.63	0.65	0.64	0.64
opel	0.85	0.83	0.85	0.84	0.83	0.82	0.81	0.81
van	0.81	0.81	0.84	0.82	0.72	0.73	0.73	0.73
average	0.82	0.82	0.83	0.83	0.75	0.73	0.74	0.73

In order to identify whether the data become robustness or not, I apply the distinctiveness (Gedeon & Harris, 1991) to remove insignificant hidden neurons and retrain the neural network to see if the data is robust. Specifically, I simply prune one neuron in the pair of neurons whose distinctiveness is less than 15 and both neurons in the pair of neurons whose distinctiveness is bigger than 165. The **Table 4** shows the evaluation on both neural networks. The results seem strange. In fact, they do not conflict with experimental results from Gedeon. As Gedeon, Catalan & Jin (1997) mentioned that the former data using standard back-propagation were better than that using shared weights. However, the test accuracy of using shared weight topology after pruning is very low, so it is meaningless to prune one more hidden neuron with distinctiveness near 60. Therefore, I decide not using shared weight topology.

Table 4. Evaluation on Shared Weight Topology or Not After Pruning

	original neural network				original neural network with shared weight topology			
vehicle type	accuracy	precision	recall	F1 score	accuracy	precision	recall	F1 score
bus	0.75	0.74	0.75	0.75	0.58	0.54	0.57	0.56
saab	0.65	0.63	0.64	0.64	0.32	0.33	0.35	0.34
opel	0.73	0.73	0.75	0.74	0.57	0.56	0.54	0.55
van	0.71	0.71	0.73	0.72	0.52	0.53	0.51	0.52
average	0.71	0.72	0.73	0.72	0.52	0.54	0.54	0.54

3.4 Evaluation on Genetic Algorithms

I apply genetic algorithms to select input features because some features may not contribute to learning. I stop processing until the most fitted DNA does not change. I do this method several times to see whether the same features are extracted every time. Actually, the results are mixed. On the one hand, I cannot get the same features every time. The reasons for it are:

1. The whole search space is large. The genetic algorithm on this dataset cannot search the whole search space and has random start. Therefore, it may miss the global optimum chromosome and choose the local optimum chromosome instead.
2. The randomness affects every result. Every time there are random initial weights for training the neural network and test accuracy has a small fluctuation as the result. Therefore, it may not provide the best test accuracy just because of unluckiness.

On the other hand, the good news is we can always get better and more stable results if there are a big population size and a large enough generation size. I randomly generate 100 chromosomes and process until it converges. I do above 10 times and find there are 4 times that I get the same chromosomes with the best fitness in the end. The average test accuracy is 83.6%, it is 1.5% higher than the original features. Therefore, genetic algorithms can contribute to features selection. The best features I find are to remove "skewness about major axis", "elongatedness" and "Pr. axis rectangularity" from the original features and there are 15 input neurons as the results. **Table 5** shows the evaluation on the improved neural network.

Table 5. Evaluation on New Neural Network Using Genetic Algorithms

vehicle type	accuracy	precision	recall	F1 score
bus	0.86	0.87	0.86	0.87
saab	0.77	0.75	0.76	0.75
opel	0.84	0.83	0.83	0.83
van	0.82	0.82	0.84	0.83
average	0.84	0.83	0.85	0.84

3.5 Comparison with Another Academic Paper

Pappa, Freitas & Kaestner (2002) provide a multi-objective genetic algorithm for attribute selection. It is based on the wrapper approach to discover the best subset of attributes for a given classification algorithm, namely C4.5, which tries to minimize the error rate and the size of the tree. Their method achieves 26.03 ± 1.78% for the error rate on the Vehicle dataset. Actually, the test accuracy they achieve can be directly calculated by $(1 - \text{error rate})$ which is 73.97 ± 1.78%.

I achieve 82% ~ 84% on this dataset. Obviously, both of us apply genetic algorithms for feature selection. The different results may come from these points:

1. We implement different genetic algorithms. To be more specific, our selection, crossover, mutation and fitness function may be different.
2. The randomness affects every result. Every time there are random initial weights for training the neural network and test accuracy has a small fluctuation as the result.
3. I use Adam optimizer. Adam is considered to be more efficient and robust.

4 Conclusion and Future Work

In this paper, I implement a 2-layer neural network to recognise vehicle types on Vehicle Silhouettes dataset. I apply standard scores normalizing the initial data to do pre-processing. Besides, shared weight topology and genetic algorithms are used to improve the data's robustness and increase the test accuracy. However, I find the test accuracy is so low that I decide not using shared weight topology. As for genetic algorithms, the test accuracy averagely increases 1.5%, which indicates the practicability of genetic algorithms. Finally, I compare my results with another academic paper on the same dataset. My neural network performance is better because I use a more efficient genetic algorithm and Adam optimizer.

As for future work, there are several parts that I can improve my approach. As for the pre-processing, I can prune some instances that may affect training. Specifically, I plan to use bimodal distribution removal method (Slade & Gedeon, 1993). The idea of this method is to calculate errors and remove train data with big errors during training, which contributes to getting better test accuracy. It is quite similar with feature selection. Consequently, it must be very interesting to compare the results of these two methods. Besides, I can try other selection, crossover and mutation methods for genetic algorithms and compare the results. Last but not least, fussy logic can achieve good results for multi-class classification problems. I can generate fuzzy sets for every attributes and utilize these fuzzy sets as the input of neural network. I would like to try these methods in the future.

Reference

Bustos, R. A., & Gedeon, T. D. (1995). Decrypting neural network data: a GIS case study. *In Artificial Neural Nets and Genetic Algorithms*. Springer, Vienna.

Gedeon, T. D., Catalan, J. A., & Jin, J. (1997). Image compression using shared weights and bidirectional networks. *In Proceedings 2nd International ICSC Symposium on Soft Computing*.

Gedeon, T. D., & Harris, D. (1991). Network reduction techniques. *In Proceedings International Conference on Neural Networks Methodologies and Applications*, 1, 119-126.

Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

McLachlan, G., Do, K. A., & Ambroise, C. (2005). *Analyzing microarray gene expression data*. John Wiley & Sons.

Mowforth, P., & Shepherd, B. (1993). Statlog (Vehicle Silhouettes) data set. Retrieved from http://archive.ics.uci.edu/ml/datasets/Statlog+%28Vehicle+Silhouettes%29

Pappa, G. L., Freitas, A. A., & Kaestner, C. A. (2002). Attribute selection with a multi-objective genetic algorithm. *In Brazilian Symposium on Artificial Intelligence*. Springer, Berlin, Heidelberg.

Ramachandran, P., Zoph, B., & Le, Q. V. (2018). Searching for activation functions.

Slade, P., & Gedeon, T. D. (1993). Bimodal distribution removal. *In International Workshop on Artificial Neural Networks*. Springer, Berlin, Heidelberg.

Sun, Z., Bebis, G., & Miller, R. (2006). Monocular precrash vehicle detection: features and classifiers. *IEEE Transactions on Image Processing*, 15(7), 2019-2034.

Wang, Z., Wu, X., Liu, X., Cao, Y., & Xie, J. (2017). Research on feature extraction algorithm of rolling bearing fatigue evolution stage based on acoustic emission. *Mechanical Systems and Signal Processing.*

Zhong, P., & Fukushima, M. (2007). Regularized nonsmooth Newton method for multi-class support vector machines. *Optimisation Methods and Software*, 22(1), 225-236.