A Comparison of Classifiers for Classification of Steel Plate Faults and Motion Capture Data

Michael C. Hagenow
Department of Mechanical Engineering
`mhagenow@wisc.edu`

Abstract

In this paper, classifier performance of least squares, support vector machines (SVM), and simple neural networks are compared for two multiclass datasets: faults in steel plates and hand gestures using motion capture. Results show that all three methods perform similarly on the steel plate fault dataset. The neural network significantly outperforms the other methods for the motion capture dataset. These results highlight that in certain cases, simple methods can perform just as well as complicated methods and that in other cases, limitations of methods (e.g., linear decision boundaries) can greatly affect performance.

1 Introduction

With many recent advances related to artificial neural networks and in particular, deep learning, it is common to see these techniques quickly applied to many problems without consideration of more basic regression techniques that can sometimes yield similar results.

In this work, a variety of classifiers are applied to multi-class classificiation problems to explore the relative performance of methods. In particular, two datasets from the UC Irvine Machine Learning Repository [1] are used as examples. The two sets are for identifying faults in steel plates based on various plate measurements[1] and identifying hand gestures from motion capture hand poses[2]. These two datasets were selected as examples that could be useful in collaborative robotics for manufacturing. All of the code discussed in this article is provided as an open-source implementation[3].

2 Preprocessing and Dataset Details

For each of the datasets, the original delimitied data was converted to csv format and imported into NumPy [2] for convenient manipulation. The following sections describe additional preprocessing that was done before applying the learning algorithms.

2.1 Steel Plate Faults

The steel plate faults dataset consists of 1941 samples, each containing 27 features. The labels consisted of seven different types of plate faults. Each plate contained only a single fault. The data consists of measurements such as plate thickness and luminosity as well as discrete properties such as type of steel. In order to negate scaling associated with varying units, each column of the data was normalized.

[1]https://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults
[2]https://archive.ics.uci.edu/ml/datasets/Motion+Capture+Hand+Postures
[3]https://github.com/mhagenow01/ECE532ClassifierComparison

2.2 Motion Capture Data

The motion capture dataset consists of 78,095 samples, which each contain a variable number of unlabeled Cartesian positions (i.e., x, y, and z coordinates) for markers attached to the participant's hand. The number of available markers in each frame varies from 3 to 12. The labels consist of the gesture the tracked hand was making. The five recorded gestures include fist, stop, one finger point, two finger point, and grabbing. The data comes from demonstrations from 12 subjects.

To provide some consistency with the number of inputs, the dataset was pruned to samples where six motion capture measurements could be used as features (no samples with less, samples with more than six only use the first six). This reduced the input dataset size from 78,095 to 65,072. Requiring a minimum set of datapoints and computing features agnostic of the number of labels is also possible, however, using the reduced data points allowed for faster feature computation.

With unordered data, it was necessary to compute features agnostic to the specific order of motion capture markers. For each sample, six features were computed: (1-3) minimum/maximum/average distance between markers and (4-6) minimum/maximum/average angle formed from triples of markers. Finally, similar to the steel plate faults dataset, the final columns of motion capture data were normalized to create consistency between distances and angles.

3 Approach

The comparison methods were selected to span a breadth of basic regression technologies including linear methods, such as least squares and support vector machines (SVM) as well as a nonlinear method, a simple neural network with a nonlinear activation function. In the future, kernel methods will also be explored. Since the data from both sets consists of multiple classifications, each classifier was developed using the 'one' vs 'all' methodology. This leads to a classifier for each class label. To select the overall most likely class label and to break any potential ties, the highest function output is used as the label (corresponding to the furthest from the decision boundary in the positive direction).

3.1 Least Squares

The least squares approach is implemented in closed form, weighted closed form, and gradient descent. For the closed form with sample weighting which is used for the experiments presented later, the weights are calculated according to the literature [3]:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{y} \tag{1}$$

where $\mathbf{w}$ are the weights to be learned, $\mathbf{X}$ is the feature matrix, $\mathbf{W}$ is a diagonal matrix of weights which are inversely proportional to the number of instances in each class, and $\mathbf{y}$ is a vector of labels. The gradient methods similarly scale the gradient step update based on the same weighting.

3.2 Support Vector Machines

Support vector machines are implemented using gradient descent for a hinge loss objective function with $\ell 2$ regularization. The weights at each step were updated according to:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \tau\Big(\sum_{i=1}^{N} -\gamma_i d_i \mathbf{x}_i \mathbb{1}_{d_i \mathbf{x}_i^T \mathbf{w}^{(t)} < 1} + 2\lambda \mathbf{w}^{(t)}\Big) \tag{2}$$

where $\mathbf{w}$ are the weights, τ is the learning rate, γ_i is the weighting factor, d_i is the label, $\mathbf{x}_i$ is the feature vector, and λ is the regularization coefficient . The same weighting factors (inversely proportional to number of samples in the class) are used for the SVM method. The value of τ was initially set to the inverse of the operator norm of the feature matrix (max bound for least squares and reasonable starting point for other methods) and further tuned to give the best balance of convergence and performance based on trial runs. While the SVM method tended to perform better when given additional iterations, to allow for tractable runs, the number of iterations was capped at 500. There was also a weight norm update tolerance exit condition of 0.001 though it was rarely met.

3.3 Neural Network

All of the neural networks are implemented using Pytorch. The networks consist of a single hidden layer which contain 1000 nodes. The hidden nodes use a rectified linear (RELu) activation function and the output nodes use a sigmoid activation function. Multiple values of hidden layers were tested, but empirically yielded similar results, thus a single layer was chosen for simplicity.

The networks are trained using two topologies. For the 'one' vs 'all' testing (i.e., for a more direct comparison with least squares and SVM), a single output node is used with Boolean labels (0 or 1). The multiclass network is trained with a set of output nodes equal to the length of the number of classes which is again trained using a Boolean label (i.e., 1 for the true class, 0 for all other classes). Unless otherwise specified, the network uses an Adam solver, 5000 epochs, and a learning rate of 0.01. These parameters were selected based on the best empirical results balancing convergence and performance. While it is possible to either explicitly regularize the network weights or otherwise prevent overfitting using techniques such as dropout, the networks used in the paper use a basic mean squared error loss to minimize additional parameters and tuning.

4 Results

4.1 General Multiclass Classification

To assess the high-level performance of each algorithm, first the multiclass accuracy was determined using five-fold cross validation. For the least squares and support vector methods, no regularization term was used for the first analysis (consequently, SVM is referred to as Hinge-Loss for this experiment). The effect of regularization is explored in the next section. The data was also assessed on the training set in order to gain an intuition for whether the data could be separated. Table 1 displays the results. The neural network was able to fit the training data well, however, the performance significantly diminished in the cross validation. This result might imply that the neural network overfit the training data and demonstrates that high representational power does not always imply an algorithm will generalize well. However, it does demonstrate the power of a neural network as an approximator to complicated functional relationships. Interestingly, the least squares method in

	LSQ	Hinge-Loss	NN
Training Data	0.684	0.592	0.933
Cross-Validation (n=5)	0.585	0.544	0.583

Table 1: Results for the classification using 'One' vs 'All' without regularization. There are seven labels for the faults so the random-guess baseline would be 0.143.

both cases achieves a greater classification accuracy than the hinge loss of the SVM method, which is unexpected in theory given that outliers can skew the least squares results. One key difference is that the LSQ method is closed form whereas the SVM method is implemented using gradient descent. To explore this effect, the SVM method was run using five-fold cross validation for a variety of number of iterations. The results confirmed that the SVM method converged to a similar classification accuracy as the least squares as the number of iterations was increased (nearly identical with 50,000 iterations).

The results for the cross-validation are otherwise mediocre, however, not unexpected given that there are no guarantees that the data is linearly separable nor contains sufficient patterns to be decoded efficiently with a neural network. In fact, before any analysis was run, there was some question to how successful the results would be given that the features didn't appear on the surface to be good direct indicators of what type of fault was present (e.g. perimeter of plate). The results are further broken down by class in Figure 1. Generally, all methods perform best on Z-scratches, K-scratches and stains.

Finally, one state-of-the-art method was run for comparison which focuses on countering noise and errors in the labels, Cleanlab [4]. This method was run for the same cross-validation using the logistic regression functionality with a small amount of $\ell 2$ regularization from scikit-learn. On this dataset, the method had a lower classification accuracy of 0.41. Looking at the results, this processing appeared to overfit the 'other faults' category (0.97 accuracy on this class. The next highest was 0.35).

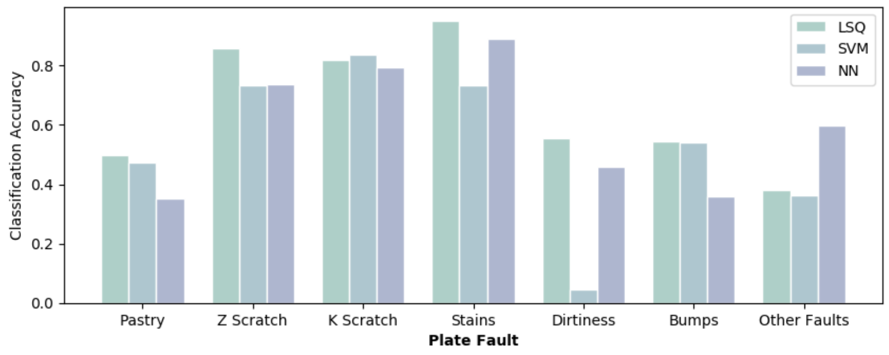

Figure 1: Classification accuracy during five-fold cross-validation broken down by each class. SVM generally performs comparably to the other methods, except in the 'dirtiness' fault class where it performs significantly worse. Note that the classes have different number of instances.

This result again highlights that depending on the nature of the dataset, certain advanced classifiers may not perform favorably.

4.2 Regularization and Feature Selection

To further assess overfitting and to evaluate the impact of individual features on the classification, an analysis of regularization was conducted. First, the least squares and SVM methods were evaluated using cross-validation for a logarithmically-spaced range of regularization values. The results are shown in Figure 2. For both classifiers, increased regularization tended to almost monotonically decrease performance. As expected, with sufficiently large regularization, both classifiers begin to perform poorly as the bias-variance too heavily favors small weights. With a maximum performance with no regularization, this might suggest that the solution for the dataset might have similar examples in the test and training sets (i.e., minimizing bias was the most important factor).

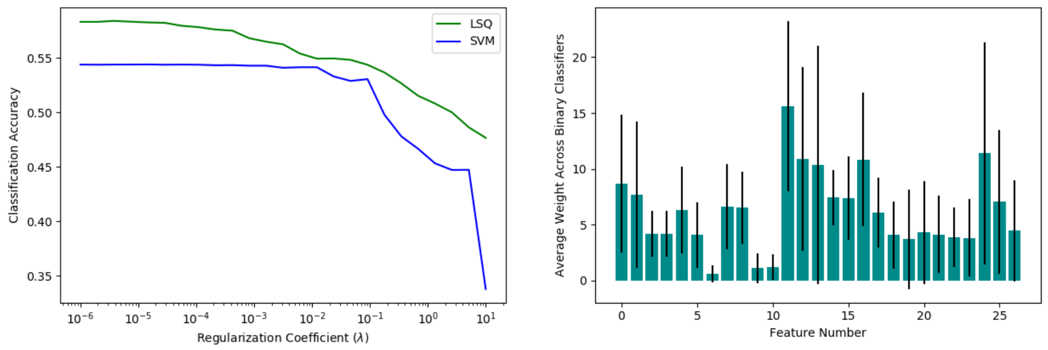

Figure 2: (Left) Impact of l2 regularization on classification accuracy of least squares and SVM. (Right) Impact of l1 regularization on least squares feature weights.

As an additional assessment of skepticism that the provided features could be useful for classification of plate faults, an $\ell 1$ regularization analysis was performed to see what features emerged as the solution was driven to be more sparse. The value of the regularization was experimentally set to be large enough to see some features driven towards zero (in this experiment, $\lambda = 0.1$). The results are also shown in Figure 2. Three features (Y perimeter, Maximum of Luminosity, and Length of Conveyer) had minimal contribution in terms of final weight which seem arbitrary. The largest weight was for a Boolean flag of whether the steel was type A300 which seems a reasonable indicator for potential faults. With some important Boolean features, it would make interesting future work to see how a decision tree might perform on the same data.

4.3 Neural Network Classification Topology

Neural networks offer flexibility in implementation due to choices in inputs, hidden layers, nodes, and outputs. Whereas the least squares and SVM methods use 'one' vs 'all' with binary classifiers to address the multiclass problem, the neural network can be used similarly or can directly produce a likelihood of each class by having an output node for each class.

Out of curiosity for whether this choice would have a strong impact on the results of the classification, a comparison was run between the two network topologies. Both networks were trained on 80 percent of the data and tested on the remaining 20 percent. Each network had a single hidden layer with 1000 nodes. The two conditions performed similarly (0.585 and 0.567 accuracy for 'one' vs 'all' and multiclass respectively). Performance was increased minimally with decreased hidden nodes, which might also indicate that the data is overfit.

4.4 Mocap Analysis

In addition to the steel plate faults used otherwise in the experiments, the use of these classifiers was also assessed for the motion capture data, where the features were manually chosen for the unlabeled data. Due to the large number of samples present in this dataset, the models are trained using 25 percent of the data (to avoid errors in the array sizes of the numerical methods) and tested on the remaining samples. The test and training sets were selected randomly from each class. The neural network is also trained using 'one' vs 'all' for consistency. The results from the classifiers are shown in Table 2.

	LSQ	SVM	NN
Test set	0.172	0.240	0.881

Table 2: Results for the motion capture classification averaged over three runs. There are five labels for the faults so the random-guess baseline would be 0.2.

Overall, both the least squares and SVM methods perform poorly in classifying the data. On the contrary, the single hidden layer network performed quite well with an overall classification accuracy of 0.88. This result might suggest that a clear nonlinear boundary exists in the selected features that could not be separated by the linear methods. In the neural network, the lowest classification occurred with the stop hand sign by a considerable margin (0.563 accuracy with next lowest 0.946). In the future, perhaps additional features such as the variance of distance could provide more differentiation to help with the classification.

5 Assessments, Strengths/Limitations, and Conclusions

One major benefit of the linear methods was the ability to easily set up and run the methods with a smaller amount of experimental evaluation and tuning. In particular, least squares estimation can be executed with a single line of code including the weighting of samples. The gradient descent associated with the SVM was also simple, but added more uncertainty and tuning to achieve practical results similar to the least squares. These methods were also able to be trained in significantly less time than the neural networks.

Naturally, the main drawback of these methods is that they cannot represent nonlinear decision boundaries. The impact of this was clear in the motion capture dataset. However, in the steel plate faults, all methods performed similarly which suggests that whatever relationship could be learned by the network was equally effectively learned with a basic linear decision boundary. The neural network has the advantage of representing arbitrary functions given sufficient representational power, however, some of the pragmatic details such as tuning and troubleshooting can be greatly inconvenient.

5.1 Conclusions

The results of this study are interesting to confirm and question some basic intuitions. First, the results demonstrate that depending on the dataset, advanced machine learning methods may or may not be needed over more basic regression techniques. Second, the results around regularization highlight the importance of thorough analysis in choosing regression parameters.

References

[1] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2.

[3] Strutz, Tilo. (2010). Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond).

[4] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: Estimating uncertainty in dataset labels," arXiv preprint arXiv:1911.00068, 2019.