Performance comparisons of Support Vector Machine and Neural Network classifiers on the MUSK data-set

Saurabh Sivakumar
Department of Chemical Engineering
University of California, Davis
sausiva@ucdavis.edu

Sudheesh Kumar Ethirajan
Department of Chemical Engineering
University of California, Davis
sethirajan@ucdavis.edu

Abstract

The MUSK data-set taken from the UCI Machine Learning repository is used here for the study. Principal Component Analysis method is used to extract the essential features of the data-set. Support Vector Machine (SVM) and Neural Network (NN) classifiers are used to learn the MUSK data-set. Further, different kernel types for SVM and different optimisers for NN are studied in order to build an effective classifier. Moreover, learning curves for both SVM and NN classifiers are plotted to identify the dependence of the training data-set size on the model accuracy.

1 Introduction

Classification is probably the most widely used form of machine learning (supervised) and is used to solve many interesting real-world problems [1]. There are many applications in classification in many domains such as credit approval, medical diagnosis, target marketing, etc. It is the process of predicting the class of given data points. Classes are sometimes called targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (F) from input variables (X) to discrete output variables (Y). In this project, we have considered the MUSK data-set for our study as it was developed for accelerating drug research.

1.1 Accelerating drug discovery

Most drugs are small molecules that work by binding to much larger protein molecules such as enzymes and cell-surface receptors [2]. The potency of a drug is determined by the degree to which it binds to the larger, target molecule. The "right" molecular shape conforms closely to the shape of the binding site and presents electronically active surface atoms near complementary binding site atoms.

The goal of drug activity prediction is to predict the activity of new (not yet synthesized) molecules by analyzing a collection of training examples consisting of previously synthesized molecules and their observed activities when binding to a binding site of medical interest [3]. By focusing the expensive and time-consuming efforts of chemists on synthesizing the most promising candidate molecules, accurate drug activity predictions could yield large savings in time and money for pharmaceutical companies.

An even greater benefit of applying machine learning to drug activity prediction would be to guide the process of drug design. If chemists could obtain a three-dimensional model of the requirements for drug activity, they would be able to design better drugs. In many practical cases [2, 4], however, the shape of the binding site is unknown, and machine learning methods might be able to provide a three-dimensional shape hypothesis to support drug design. Such "rational" drug design could cut years off the time required to discover new drugs. It could also result in drugs with higher potency and fewer side-effects. The MUSK data-set was developed specifically to accelerate and aid the process of drug discovery.

The rest of this paper is organized as follows. In the next section, we review the literature, describe the data-set, and also define our project goals. In section 3, we describe different Machine learning methods considered in this project. Then in section 4, we present our results and discussions. Finally, we conclude in section 5.

2 Project overview

Building effective classifiers for the MUSK data-set could be tricky at times as it is a "multi-instance learning" problem, and hence a proper fundamental understanding of how different classifiers work with the data-set is necessary. When learning a classifier for this data, the classifier should classify a molecule as "musk" if ANY of its conformations is classified as a musk. A molecule should be classified as "non-musk" if NONE of its conformations is classified as a musk. The goal is to learn to predict whether new molecules will be musks or non-musks. In this work, we consider the version 2 of the MUSK data-set. It is available on the UCI Machine Learning repository [5]. It is a "multiple instance problem" data-set.

2.1 Literature review

Dietterich et al. (1997) [2] introduced the multiple-instance learning (MIL) model motivated by the problem of predicting whether a molecule would bind at a particular site. Since the shape of a molecule largely determines binding affinity, they represented each molecule by a high-dimensional vector that describes its shape, and labeled molecules that bind at a site as positive examples and those that do not bind as negative. The motivation for the MIL model [8] is the fact that a single molecule can have multiple conformations (shapes), and only one conformation needs to bind at the site for the molecule to be considered positive. Thus when an example is negative, all conformations in it are negative, but if an example is positive, then it may be the case that only one conformation of the set is positive, and the learner does not know which one.

In general, MIL refers to the many-to-one relationship between feature vectors and classes. Since its introduction, the MIL model has been applied to content-based image retrieval (Maron and Ratan, 1998, Zhang et al., 2002) [6,7], where each instance in a multi-instance example (bag) represents a feature of an image, and it is not known which feature corresponds to the content the user wants to retrieve. As with binding prediction, the MIL model used for content-based image retrieval assumes that the label of an example is a dis-junction of the labels of the instances in the example.

2.2 Data-set description

The data-set describes a set of 102 molecules of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. However, the 166 features that describe these molecules depend upon the exact shape, or conformation, of the molecule. Because bonds can rotate, a single molecule can adopt many different shapes. To generate this data-set, all the low-energy conformations of the molecules were generated to produce 6,598 conformations. Then, a feature vector was extracted that describes each conformation.

2.3 Objectives

As the original data-set has 166 columns, Principal Component Analysis (PCA) can be used to reduce the dimensionality of the data-set and extract the essential features. Support Vector Machine (SVM) and Neural Network (NN) classifiers will be used to learn the MUSK data-set. Study with different kernel types for SVM and different optimisers for NN are needed to build an effective classifier. Moreover, learning curves for both SVM and NN classifiers can be plotted to better understand the dependence of the size of (training) data-set on the model accuracy.

3 Machine learning methods

This section briefly covers the theoretical aspects of PCA, SVM, and NN methods used in this project.

3.1 Intuition

Based on our understanding of the methods, we hypothesize PCA+SVM < SVM < NN in terms of model accuracy. As the PCA is a dimensionality reduction method and when used with SVM, the complexity of model will be greatly reduced. On the other hand, NN can be very dense and fully connected to account for maximum complexity which can be used to capture all the variations and trends in the data-set.

3.2 Principal component analysis

Principal component analysis (PCA) is a technique that is useful for the compression and clustering of data. The purpose is to reduce the dimensionality of a data set (sample) by finding a new set of variables, smaller than the original set of variables, that nonetheless retains most of the sample's information. By information we mean the variation present in the sample, given by the correlations between the original variables. The new variables, called principal components (PCs), are uncorrelated, and are ordered by the fraction of the total information each retains.

Consider an $\boldsymbol{n \times p}$ data matrix, $\boldsymbol{X}$, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the $\boldsymbol{n}$ rows represents a different repetition of the experiment, and each of the $\boldsymbol{p}$ columns gives a particular kind of feature. Mathematically, the transformation is defined by a set of size $\boldsymbol{l}$ of $\boldsymbol{p}$-dimensional vectors of weights or coefficients $\boldsymbol{w_{(k)} = (w_1, \ldots, w_p)_{(k)}}$ that map each row vector $\boldsymbol{x_{(i)}}$ of $\boldsymbol{X}$ to a new vector of principal component scores $\boldsymbol{t_{(i)} = (t_1, \ldots, t_l)_{(i)}}$, given by

$$\boldsymbol{t_{k(i)} = x_{(i)} \cdot w_{(k)}} \qquad \text{for} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, l$$

in such a way that the individual variables $\boldsymbol{t_1, \ldots, t_l}$ of $\boldsymbol{t}$ considered over the data set successively inherit the maximum possible variance from $\boldsymbol{X}$, with each coefficient vector $\boldsymbol{w}$ constrained to be a unit vector (where $\boldsymbol{l}$ is usually selected to be strictly less than $\boldsymbol{p}$ to reduce dimensionality).

The $\boldsymbol{k}$-th component can be found by subtracting the first $\boldsymbol{k-1}$ principal components from $\boldsymbol{X}$:

$$\boldsymbol{\hat{X}_k = X - \sum_{s=1}^{k-1} X w_{(s)} w_{(s)}^T}$$

and then finding the weight vector which extracts the maximum variance from this new data matrix

$$\boldsymbol{w_{(k)} = \underset{w=1}{\operatorname{argmax}} \left\{ \left\| \hat{X}_k w \right\|^2 \right\} = \arg\max \left\{ \frac{w^T \hat{X}_k^T \hat{X}_k w}{w^T w} \right\}}$$

It turns out that this gives the remaining eigenvectors of $\boldsymbol{X^T X}$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues. Thus the weight vectors are eigenvectors of $\boldsymbol{X^T X}$.

The $\boldsymbol{k}$-th principal component of a data vector $\boldsymbol{x_{(i)}}$ can therefore be given as a score $\boldsymbol{t_{k(i)} = x_{(i)} . w_{(k)}}$ in the transformed coordinates, or as the corresponding vector in the space of the original variables, $\boldsymbol{\{x_{(i)} . w_{(k)}\} w_{(k)}}$, where $\boldsymbol{w_{(k)}}$ is the $\boldsymbol{k}$-th eigenvector of $\boldsymbol{X^T X}$.

The full principal components decomposition of $\boldsymbol{X}$ can therefore be given as $\boldsymbol{T = XW}$ where $\boldsymbol{W}$ is a p-by-p matrix of weights whose columns are the eigenvectors of $\boldsymbol{X^T X}$.

3.3 Support vector machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. Polynomial and exponential kernels calculates separation line in higher dimension. This is called kernel trick.

Given training vectors $\boldsymbol{x_i \in R^p}$, i=1,..., n, in two classes, and a vector $\boldsymbol{y \in \{1, -1\}^n}$, our goal is to find $\boldsymbol{w \in R^p}$ and $\boldsymbol{b \in R}$ such that the prediction given by $\boldsymbol{sign(w^T \phi(x) + b)}$ is correct for most samples.

SVM solves the following primal problem:

$$\begin{gathered}\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to } y_i \left(w^T \phi \left(x_i \right) + b \right) \geq 1 - \zeta_i \\ \zeta_i \geq 0, i = 1, \ldots, n\end{gathered}$$

Ideally, the value $y_i \left(w^T \phi \left(x_i \right) + b \right)$ would be ≥ 1 for all samples, which indicates a perfect prediction. For non-separable samples, we allow it to be at a distance ζ_i from their correct margin boundary. The penalty term C controls the strength of this penalty.

The Lagrangian dual problem is obtained from the primal as shown below,

$$\begin{gathered}\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{subject to } y^T \alpha = 0 \\ 0 \leq \alpha_i \leq C, i = 1, \ldots, n\end{gathered}$$

where e is the vector of all ones, and Q is an n-by-n positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. The most commonly used kernel is radial basis function. This dual representation highlights the fact that training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ. Solution of the above optimization problem will yield the predicted class from the decision function.

3.4 Neural Networks

A neural network or a multi-layer perceptron is another method we employ to "learn" the data-set and make predictions. These are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. The neural network will transform the input features into a hidden layer through a weighted linear sum, followed by a non-linear activation function. The selection of activation function is a study on it's own end and not the focus of this paper. The weights for the linear sum are typically calculated iteratively using back-propagation algorithms until a loss function (usually Average cross entropy, MAE or MSE) associated with the network is minimized. Several optimization algorithms exist for this purpose such as Stochastic Gradient Descent, Adam, or L-BFGS which may have their own associated parameters to fine-tune. The output layer will receive the values from the last hidden layer and transforms them into output values. A major drawback to using these for a classification problem is the compute time required to tune several of the hyper parameters associated with the network for the task at hand. [1]

Consider a set of training examples for a supervised binary classification problem defined as $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ where $x_i \in R^n$ and $y_i \in (0, 1)$.

Mathematically the predicted value, $\hat{y}_n$ for a given y_n for a single hidden layer network is,

$$\hat{y}_i = W_2 * g(W_1^T x_i + b_1) + b_2$$

Here W_1 and W_2 represent the weight matrices associated with the neurons of the input layer and hidden layer, respectively; and b_1 and b_2 are scalars that represent the bias added to the hidden layer and the output layer, respectively. $g(.)$ denotes the activation function which could be anything, but commonly used functions are the sigmoid, hyperbolic tangent and RELU. The RELU function is defined as the $max(0, x)$. The loss function, the average cross entropy that is to be minimized for the classification task is then subsequently defined as follows.

$$L(\hat{y}, y, W) = \frac{-1}{n} \sum_{i=0}^{n} (y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)) + \frac{\alpha}{2n} ||W||_2^2$$

Here the last term $\alpha ||W||_2^2$ represents a L2-regularization term that penalizes complex models and α is a non-negative hyper-parameter that controls the magnitude of the penalty.

4 Results and discussions

In this section, we present some salient results from training the data-set on the ML algorithms discussed previously. Using PCA we identified the 2 major principal components as shown in Figure

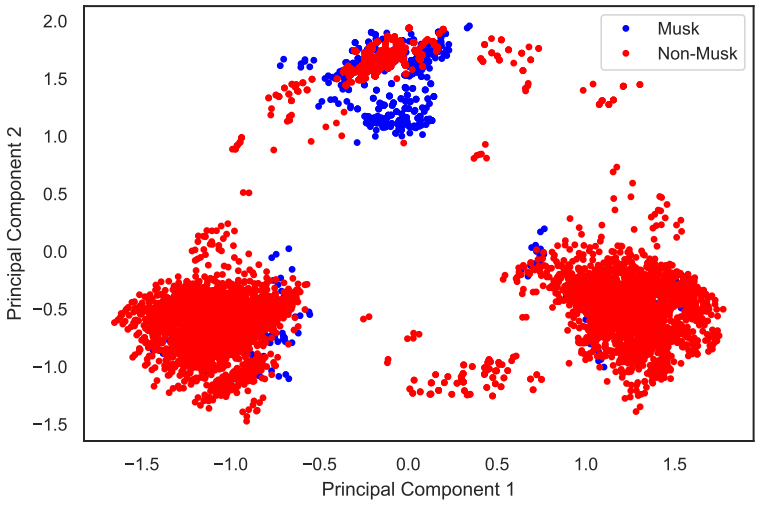

Figure 1: Pictorial representation of the data in 2 dimensions using the first two principal components

Table 1: Model training metrics

Model	**Accuracy (Test)**
NN (SGD)	96.5 %
NN (Adam)	99.5 %
SVM (rbf)	96.7 %
SVM (linear)	96.0 %

1. From this it is clear that in 2 dimensions that data is closely clustered and that more features are relevant to classify this problem. In fact we observed that 95 % of the variance can be represented by 21 % of the features. However, considering the size of the data-set, training with all the features is not time consuming, hence we choose to train both a multi-layer perceptron and a support vector machine with the full data-set including all the 166 features. The data-set is randomly shuffled then scaled by removing the mean and scaling to unit variance Finally the data is split into a train and test set with 80 % of the data as the training set (5278 data points) and the remaining as test (1320 data points) on which the model performance is measured. All the models are trained using the python package sklearn.[9]

A typical training procedure for a neural network will involve extensive hyper-parameter tuning using methods like grid search. Here considering time constraints, we use 5 fold cross validation to determine the number of neurons in a network with a single hidden layer, a RELU activation function and a fixed learning rate. Through this we identified that a network with 90 neurons is the most effective for the data. We subsequently trained 2 models using the SGD and Adam optimizer both with their default parameters defined in the package [9] (Generally accepted to be well suited for most tasks). The Adam optimizer displayed faster initial progress on the training loss and generalized well to the test data [10]. The accuracy, defined as the percentage of correct predictions on the test data-set are shown in Table 1.

Furthermore, we study the effectiveness of a support vector classifier on the data using 4 different kernels. The radial basis function kernel led to the best test data accuracy as shown in Table 1 and Figure 3. Another salient observation that can be noted from Figure 2 and Figure 3 was that increasing the number of data points in the training greatly improved the training and cross validation accuracy for both the models, a fact well established in previous literature. The testing accuracy obtained by the NN on the data is fairly high as shown in Table 1 and reinforces the notion that a NN can be an effective universal approximator of functions. [1]

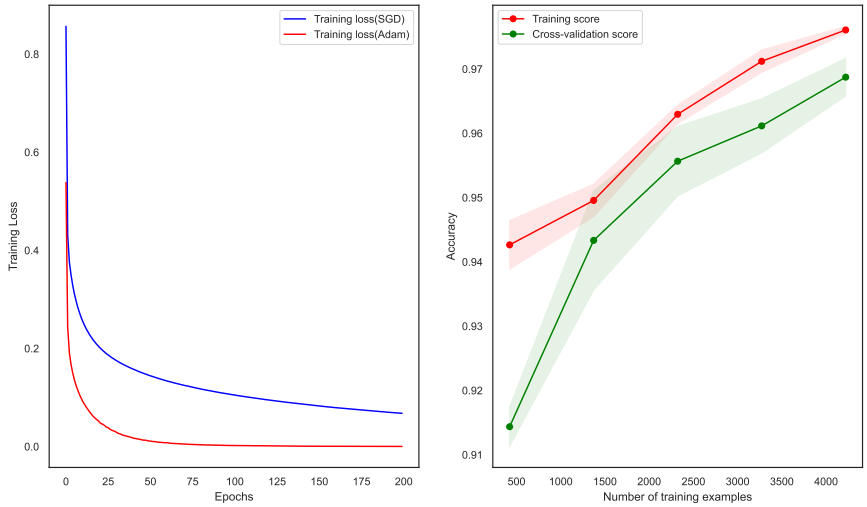

Figure 2: **Left:** The variation in the training loss function with a change in the optimizer. **Right:** Learning curve for the classifier using a SGD optimizer with variation in the size of the data-set. The improvement in the accuracy with increasing the number of training examples is evident from this graph

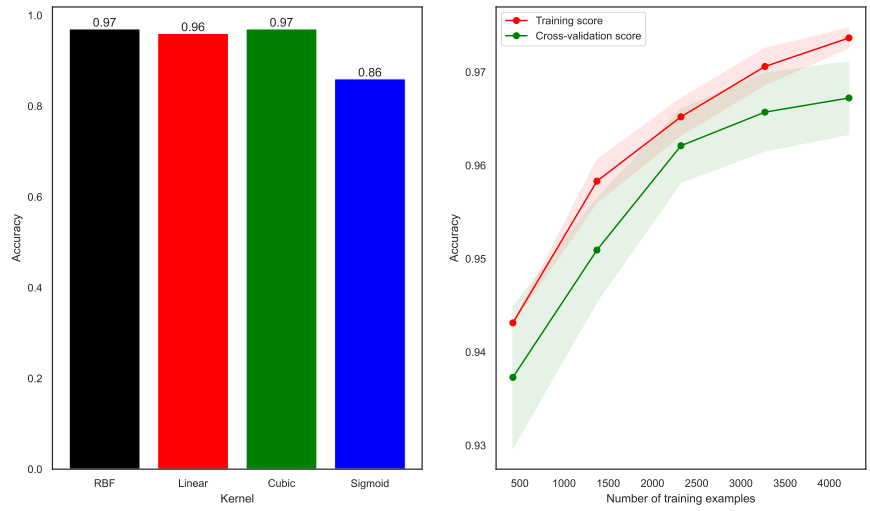

Figure 3: **Left:** The accuracy of the SV classifier with changing kernels **Right:** Learning curve for the SV classifier using a rbf kernel with variation in the size of the data-set.

5 Conclusion

In this study we train two machine learning models, a neural network and a support vector machine for the supervised classification problem represented by the musk data-set. Analysis of the data through it's principal components was also performed. Results indicate the neural network trained using the Adam optimizer to be the best fit for the data and also generalizes well to the test set. This is an active area of current research. Such models combined with high throughput computing and several larger data-sets have opened a lot of barriers in drug design and continue to do so. Moreover, these machine learning strategies to drug activity prediction will also be able to guide the process of drug design.

References

[1] Kevin P. Murphy (2022) *Probabilistic Machine Learning: An introduction*, Cambridge, MA: MIT Press.

[2] Dietterich, T. G. & Lathrop, R. H. & Lozano-Perez, T. (1997) *Solving the multiple-instance problem with axis-parallel rectangles*, *Artificial Intelligence 89*, pp. 31–71.

[3] Dietterich, T. G. & Jain, A. & Lathrop, R. & Lozano-Perez, T. (1994) *A comparison of dynamic reposing and tangent distance for drug activity prediction*, *Advances in Neural Information Processing Systems 6*, pp. 216–223.

[4] Jain, A. N. & Dietterich, T. G. & Lathrop, R. H. & Chapman, D. & Critchlow, R. E. & Bauer, B. E. & Webster, T. A & Lozano-Perez & T. Compass (1994) *A shape-based machine learning tool for drug design*, *Computer-Aided Molecular Design*, pp. 635–652.

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

[6] Maron, O. & Ratan, A. L. (1998) *Multiple-instance learning for natural scene classification*, *Proc. 15th International Conf. on Machine Learning*, pp. 341-–349.

[7] Zhang, Q. & Goldman, S. A. & Yu, W. & Fritts, J. E. (2002) *Content-based image retrieval using multiple-instance learning*, *Proc. 19th International Conf. on Machine Learning*, pp. 682-–689.

[8] Qingping Tao & Stephen Scott & N. V. Vinodchandran & Thomas T. Osugi. (2004) *SVM-based generalized multiple-instance learning via approximate box counting*, *International Conf. on Machine Learning*.

[9] Pedregosa, F et al, (2011) *Scikit-learn: Machine learning in Python* of machine learning research, pp. 2825—2830.

[10] Aman Gupta & Rohan Ramanath & Jun Shi & S. Sathiya Keerthi. (2021) *Adam vs. SGD: Closing the generalization gap on image classification OPT2021: 13th Annual Workshop on Optimization for Machine Learning*