Predicting Biodegradability of Chemicals from its Molecular Description using Support Vector Machines and Multilayer Perceptron Classifier

Utkarsh Patel 18EC30048
Akhil 18CS10070

Abstract

This report is submitted as part of third assignment to the course CS60050 – Machine Learning, IIT Kharagpur. We used the QSAR biodegradation dataset for classification task. The dataset contains molecular description of 1055 chemical samples described by 41 attributes. The class labels were 'readily biodegradable' (*RB*) and 'non-readily biodegradable' (*NRB*). Our task was to use and tune Support Vector Machines and Multilayer Perceptron Classifier to make predictions on unseen samples. This report contains a brief overview of our approach and the results we obtained in the process.

1 The Dataset

In this assignment, we used the QSAR biodegradation dataset, which was built in the Milano Chemometrics and QSAR Research Group (The University of Milano-Bicocca, Milano, Italy). The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under Grant Agreement n. 238701 of Marie Curie ITN Environmental Cheminformatics (ECO) project. The data have been used to develop QSAR (Quantitative Structure Activity Relationships) models for the study of the relationships between chemical structure and biodegradation of molecules. Biodegradation experimental values of 1055 chemicals were collected from the webpage of the National Institute of Technology and Evaluation of Japan (NITE). There were 699 chemical samples belonging to class *NRB*, while remaining 356 belonged to class *RB*. The objective in this assignment is to learn the relationship between the molecular description of samples in order to predict its class.

2 Pre-processing Data

While analyzing the dataset, we observed that there were no missing values and the attributes contained numeric values and class labels were *RB* and *NRB*. We split the dataset into training set and test set in 80:20 ratio and random shuffling. While dealing with numeric attributes, it is always better to standardize and normalize the data. In this case, we computed mean and variance of the training instances, used these parameters to normalize training as well as test instances. The class labels used were +1 for class *RB* and -1 for class *NRB*. Therefore, the problem can be specified as follows: We have training samples $\mathcal{X} = \{x^{(i)}, y^{(i)}\}, i = 1, 2, \dots, m$ where $x^{(i)} \in \mathbb{R}^{41}$, $x_j^{(i)} \sim N(0,1), j = 1, 2, \dots, 41$ and $y^{(i)} = \{-1, +1\}$ depending on whether $x^{(i)}$ belongs to class *NRB* or class *RB*. So, given any query instance $x^{(q)}$ which remains unseen to the model, we want to predict its class as precisely as possible.

3 Support Vector Machines

For purpose of classification, we used support vector machines with different kernel function viz. linear, quadratic and radial-basis, and tuned it using different regularization parameter *C*. The result we obtained is tabulated below.

Table 3.1 Accuracy observed on training set over different kernel function for different regularization constant

Regularization Constant C	**Linear Kernel**	**Quadratic Kernel**	**Radial Kernel**
1	0.895	0.840	0.906
2	0.898	0.872	0.914
5	0.893	0.899	0.931
10	0.895	0.926	0.943
20	0.895	0.933	0.960
50	0.893	0.945	0.970
100	0.898	0.960	0.977
200	0.897	0.961	0.985
500	0.897	0.974	0.993
1000	0.897	0.986	0.995

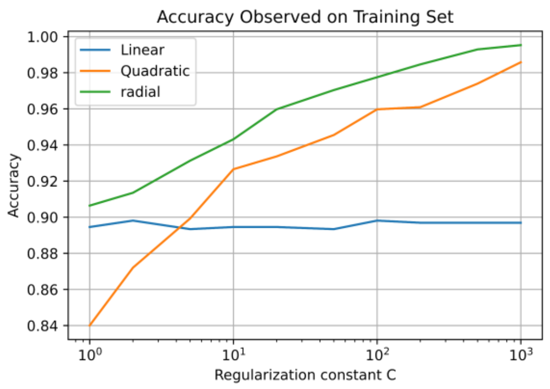

Fig 3.1 Plot of accuracy observed over training set

As it can be observed from Table 3.1 and Fig. 3.1, the linear kernel underfits the training samples, owing to less training accuracies for wide range of regularization constant. However, in contrast, the quadratic kernel and radial kernel tend to overfit the training samples as the regularization constant is increased.

Table 3.2 Accuracy observed on test set over different kernel function for different regularization constant

Regularization Constant C	**Linear Kernel**	**Quadratic Kernel**	**Radial Kernel**
1	0.853	0.773	0.848
2	0.858	0.801	0.863
5	0.858	0.844	0.877
10	0.858	0.863	0.896
20	0.863	0.853	0.878
50	0.867	0.834	0.863
100	0.858	0.867	0.863
200	0.858	0.863	0.853
500	0.858	0.848	0.839
1000	0.853	0.839	0.839

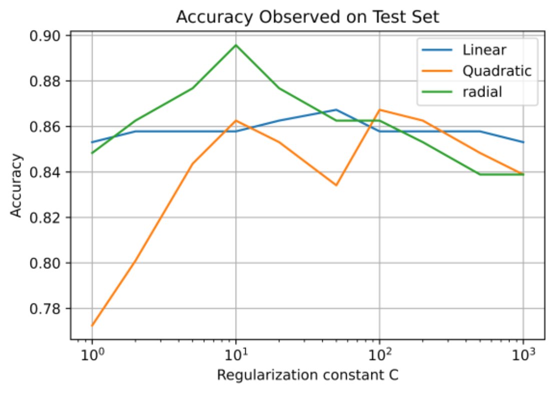

Fig 3.2 Plot of accuracy observed over test set

It can be observed from the Fig. 3.2 that radial basis kernel with $C = 10$ performs the best on the test set. It can also be observed that kernel with high C don't perform well on test set, unlike their performance on training set. Therefore, it can be implied that kernels with high C tend to overfit the training samples and perform badly on test set. Optimal value of C for various kernels is as follows:

Table 3.3 Accuracy observed for Optimal Regularization constant *C* for various kernels

Kernel	**Optimal *C***	**Train Accuracy**	**Test Accuracy**
Linear	50	0.893	0.867
Quadratic	100	0.960	0.867
Radial	10	0.943	0.896

4 Multilayer Perceptron Classifier

As the input feature vector dimension is 41 and output is a real number, hence, nodes in input and output layers are 41 and 1 respectively. For purpose of classification, we used different architectures of neural networks.

Table 4.1 Different architectures used in MLP classifier

Architecture	**Description**
M1	No hidden layers
M2	1 hidden layer with 2 nodes
M3	1 hidden layer with 6 nodes
M4	2 hidden layers with 2 and 3 nodes
M5	2 hidden layers with 3 and 2 nodes

We used *Stochastic Gradient Descent* as optimizer with momentum coefficient being 0.9. The loss function used was *cross-entropy loss function* which used logit units. We tuned each model for different values of learning rates. The results obtained are tabulated below.

Table 4.2 Accuracy observed on training set for different models for different learning rates

Learning Rate	**M1**	**M2**	**M3**	**M4**	**M5**
0.00001	0.667	0.553	0.646	0.357	0.437
0.00010	0.819	0.835	0.761	0.661	0.662
0.00100	0.868	0.884	0.892	0.891	0.887
0.01000	0.884	0.904	0.937	0.905	0.913
0.10000	0.890	0.904	0.953	0.919	0.938

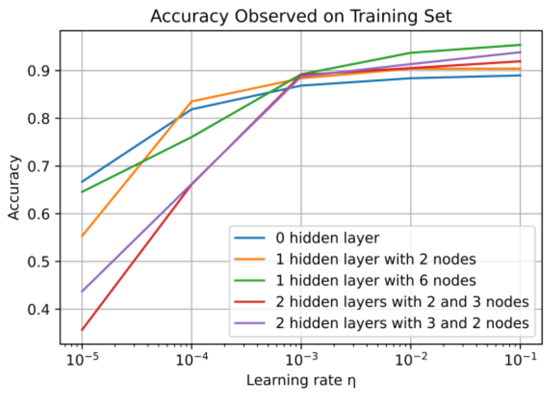

Fig 4.1 Plot of accuracy observed over training set

It can be observed from Table 4.2 and Fig. 4.1 that learning rate plays a crucial role in determining the accuracy of the model. For low learning rates, the global minima is not reached in desired epochs and hence low accuracy is observed in this case. It can be deduced that a more complex model will give high training accuracy, as it tends to overfit the examples.

Table 4.3 Accuracy observed on test set for different models for different learning rates

Learning Rate	**M1**	**M2**	**M3**	**M4**	**M5**
0.00001	0.645	0.573	0.611	0.355	0.346
0.00010	0.782	0.820	0.706	0.664	0.664
0.00100	0.791	0.820	0.867	0.829	0.848
0.01000	0.853	0.877	0.882	0.872	0.815
0.10000	0.858	0.886	0.829	0.825	0.839

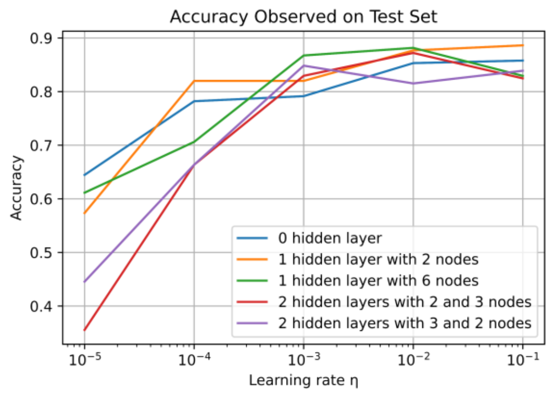

Fig 4.2 Plot of accuracy observed over test set

From Fig 4.2, we can see that the best model is M2 which gives test accuracy of 0.886 with train accuracy of 0.904 for learning rate 0.1 and *Stochastic Gradient Descent* as its optimizer. It can be justified as follows: Complex models tend to overfit the training examples and hence perform badly on test set. Simpler models generalize well on test set. This is what we call *Occam's razor.*

5 Conclusion

For predicting the class of chemicals, we have used two different approaches: support vector machines and multilayer perceptron classifier. Both are supervised machine learning classifiers. MLP is a parametric classifier that uses hyper-parameters tuning during the training phase. An SVM is a non-parametric classifier that finds a linear vector (if a linear kernel is used) to separate classes. Actually, in terms of the model performance, SVMs are sometimes equivalent to a shallow neural network architecture. Generally, an ANN will outperform an SVM when there is a large number of training instances, however, neither outperforms the other over the full range of problems. In our task, using SVM, the best test accuracy was observed to be 0.896, and that of MLP was 0.886. Therefore, in our case SVM has outperformed the MLP classifier. However, one advantage of using MLP is that the model size is fixed, which in case of SVM, depends on number of support vectors, which can be equal to number of training examples in worst case. Therefore, for tasks involving very large number of training examples, MLP is preferred over SVM.