CS 412/512
Machine Learning
Fall 2015

Sentiment Analysis - Turkish Tweets

Sentiment analysis is a problem that can be approached with several supervised learning models. We obtained the best results with using the regression version of support vector machines. Training sets were given to Matlab and Weka in order to conclude which supervised learning model performs best on this specific task.

We started the implementation of a basic neural network in Matlab. We used hyperbolic tangent function (Tanh) as the activation function since the output had to be interpolated between 1 and -1. The accuracy of the network could not get higher than 30% although we tried with different hidden layer combinations, different activation functions and neuron numbers.

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$

TanH function, f(x) in (-1,1)

$$f(x) = \frac{1}{1+e^{-x}}$$

Sigmoid function, f(x) in [0,1]

Therefore we decided to change direction in a way that the regression problem could be turned into a classification problem. The interpolated values of output could be mapped into three classes where the classes determine if the tweet is positive (label >= 0.25), negative (label <= -0.25) or objective (0.25 > label > -0.25). In addition to the 3-class approach, we also tried 21 classes, where each class denotes one decimal precision (-1.0, A; -.9, B; -.8, C, etc.). With these approaches, we explored the performance of the other models such as Naïve Bayes, Decision Tree and their combinations. When we evaluated the 3-class problem with Decision Trees and Naive Bayes algorithms, we realized the challenging part of our problem, i.e. regression. In classification case, since the target space is divided into discrete sets, also the error measure is discrete. However, in regression, each instance participates in the final error rate since none of the estimates perfectly matches the desired labels. Since our problem is a regression problem, we did not spend more time on this classification topic, but it definitely helped us to get an insight about our problem.

After such an exploration, we decided to use regression version of support vector machines due to a couple of reasons. One of them is that it can alter the dimensions intrinsically to solve problem linearly using kernel function. The other reason is that when used with decently large dataset, the overfitting should not have been much of a problem. To be able to attack the problem, we used LibSVM library in Weka and SVR (Support

Vector Regression) Trainer in Matlab, and we tested several configurations in order to obtain minimum mean absolute error on the ***training set***.

Activation function, tolerance of the termination criterion (epsilon) and cost parameter were the most significant parameters for maximizing the accuracy.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|f_i - y_i| = \frac{1}{N}\sum_{i=1}^{N}|e_i|$$

Mean Absolute Error that is used to evaluate every algorithm

$$K(x,y) = e^{-\lambda|x-y|^2}$$

Radial Basis, or Gaussian, Kernel of SVR

	Lambda	*C*	*Epsilon*	*Mean Absolute Error*
MATLAB	**0.5**	**4000**	**0.35**	**0.0272**
	0.5	**400**	**0.25**	**0.0316**
WEKA	**0.1**	**400**	**0.35**	**0.264**
	0.1	**400**	**0.25**	**0.278**

Training errors on different parameter configurations and platforms.

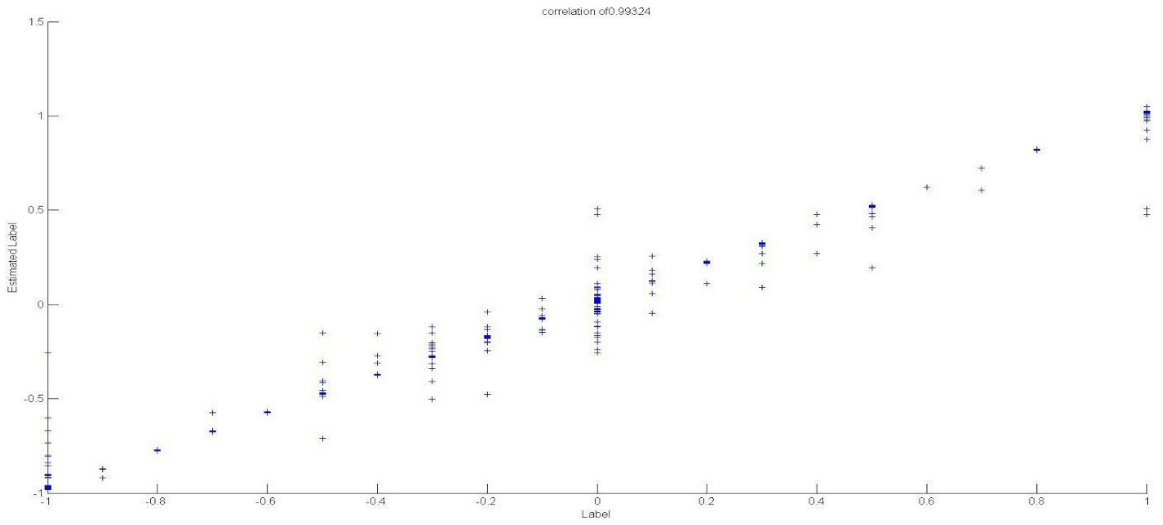
Two different SVM type were offered within the LibSVM wrapper for the regression problems, nu-SVR and epsilon-SVR. Firstly we experimented with changing the types and mainly the kernel types and it was concluded that Epsilon-SVR with radial basis function shown above produced the best results in terms of MAE.

Data normalization helps improving the accuracy since it minimizes the redundancy hence we also thought that with reducing the number of dimensions, we could obtain better results since some of the classes may not have an impact on the overall labelling. We ran the tests with removing F1,F3,F4,F5,F14,F15,F16,F17,F21 in Weka and just with F1,F6,F8,F9,F18,F19,F20,F22(the label) in Matlab. Both of the two different reductive approach did not work well as can be seen in the figures below.

	Lambda	*C*	*Epsilon*	*Mean Absolute Error*
MATLAB	**0.5**	**400**	**0.25**	**0.0418**
WEKA	**0.1**	**400**	**0.35**	**0.2931**

Reduced Dimension Results

Another measure to pay attention is the ***correlation*** between estimated and true labels. Below you can find the correlation graph from MATLAB with $\lambda = 0.5,\ C = 4000, \varepsilon = 0.35$. Since the error rate is to low, the correlation is calculated as 0.993.



We could say that there is a positive correlation between the labels which means the label values have positive proportion in-between as can be seen in figure above. It does not imply that they cause a related change between themselves. The factor causing this correlation may be unknown hence the correlation factor in our case simply refers to the fact that they vary together.

Conclusion

After spending several hours on finding the most suitable parameters we could get the error rate down to %2 when tested with the training set. It was not an expected result since the the accuracy of %98 in this case would almost be the perfect model therefore we looked into the possibility of ***overfitting.*** When tested with a 5-fold cross validation it appeared to have 60% error rate but when the parameters are further tweaked ($\lambda = 0.005,\ C = 450, \varepsilon = 0.3$) the overall mean absolute error went down to 35%.