**Intro**: Initially our intuition for using SVM was to see if we can identify any one class easily by running the original binary classification problem for each class against the others. Subsequently, when these results turned out poorly, and in the process of exploring the various SVM models and packages in R, we came across specifications that are adaptable to multi-class problems.

**Support Vector Machines - Main**

Another classification method we took advantage of was classification support vector machine class of models with various specifications. Several different specifications of the SVM models were used and the thought process as well as the results relating to the selection process, as well as parameter specification is presented in this section.

Model description

The SVM method for classification of data into classes is commonly been used in practice and discussed in the course. This is a supervised learning method which is appropriate for our current problem, given that it is supervised learning we are dealing with – we have labels attached to the observations in the training set. The SVM model is a non-probabilistic binary linear classifier that uses Kernel methods to implicitly map otherwise non-linearly separable data into a higher dimensional space where we can use linear separation.

Thought process for SVM

As mentioned previously, the initial idea behind using SVM for the given task was to identify any class that stands out significantly when compared to all others. We thought using SVM was relevant considering that it is unlikely that the data is simply linearly separable to run a perceptron algorithm. Either way, even if the data was linearly separable SVM would be able to deal with this. The idea of SVM to maximise the margin between classes also seemed appealing.

A brief note should be made about the methodology to find the ideal parameters in the SVM models considered. The initial approach was to estimate and perform diagnostics on the model trained on the entire training set. However, upon initial tries of doing so (even with default parameter settings), the computation time was over 4 hours in R. For this reason, tuning of the SVM parameters was done on the training set containing only 20% of the training set data provided. The assumption was that similar results, in respect to performance of different models, would be observed. This approach has allowed a greater set of models to be tested.

Data and model “tricks”

All of the types of SVM models considered were run in two different versions, considering the transformation of the continuous variables. These were scaled according to subtracting the mean and dividing by standard deviation of the training sample. The rationale is that this would make the numerical optimisation routine more stable. As with other models, soil type 15 was removed from the feature space.

It needs to be mentioned that some experimentation with different kernels was done in the training of the SVM. The performing kernels were the RBF and the Laplace kernel; the other kernels were severely underperforming in comparison.

One-against-all SVM

Initial set of models tested considered was the one against all SVM classification for each class. So given 7 classes, 7 different model were run. The performance of these is evaluated using both in-sample validation error and cross validation error. The results are presented graphically below.

Insert picture – one vs all classification error for one against all classification

As evident both of the error measures are incredibly high and thus these models are discarded instantly. The results were almost equally as bad for both versions of input, scaled and non-scaled.

Multi-class SVM

Although in the given problem we do not have only two classes, but 7 classes, and therefore standard SVM framework needs to be adapted if we to try a different approach than “One-against-all”. A classic approach is to use a so called ‘one-against-one’ method where we train k(k-1)/2 binary classifiers. Here, k represents the number of classes, so in our cases it would be k=7. The appropriate class is determined using a voting scheme. Once again, the RBF kernel was used for training, as experimentation with other kernels yielded inferior error rates.

It is worth pointing out that scaling the continuous variables resulted in an improvement of 0.0024 in the error rate.

We found that increasing the C parameter in model training increases accuracy significantly. The relationship of the C parameter and the error rates can be seen in the graph below. However, increasing the C parameter also hugely increased the training time. This makes sense because given that C is a regularisation parameter, assigning higher penalty to misclassification the SVM optimisation routine looks for a “harder” boundary the greater the C. This imposed strictness causes the computational difficulties resulting in higher run time. From the below graph we can see that marginal improvement from increasing C decreases greatly as C increases. We chose arbitrarily C=5000 as a reasonable parameter in accuracy/run time pay off. The final in sample misclassification error was 0.1072. The 10 fold cross validation however was much greater

Conclusion

SVM models proved to be inefficient with regard to run time and accuracy when compared to other models we considered. Thus we have discarded this class of models as suitable for the given problem.