Homework Assignment 2
CSE 291 Spring 2011
Data Mining with Support Vector Machines

Jihoon Kim
UCSD Division of Biomedical Informatics
j5kim@ucsd.edu

Hari Damineni
UCSD Computer Enginering & Science
hdaminen@cs.ucsd.edu

André Christoffer Andersen
NTNU Industrial Economics
andrecan@stud.ntnu.no

April 19, 2011

1 Introduction

In this assignment, we build a Support Vector Machine (SVM) to estimate the willingness to donate to a national fund raiser campaign. It builds on some of the data pre-processing and feature selection of Homework Assignment 1. The extension in this assignment contains the following phases

1. Further data pre-processing and feature selection
2. Parameter search using cross-validation

R is used to pre-process data and to reduce the feature space. The data processed by dummification, imputation, normalisation, elimination and extraction. More on this in section 2.

RapidMiner is then used to model both a linear and RBF (Radial Basis Function) SVM where parameter selection is done through a grid search with cross validation. We obtained best accuracies of 65.63% and 62.96% from linear kernel and RBF kernel based SVMs respectively.

2 Data Preparation

We use a data set of 14,059 instances and 197 features in model building and evaluation. We conduct a preprocessing in following steps:

1. *Create*: creates an initial data set of 14,529 rows and 481 attributes from the raw data in `cup98LRN.txt` with 95,412 instances. This retrieves all positive `TARGET_B = 1` instances ($n = 4,843$) and samples twice-many negative `TARGET_B = 0` instances ($n = 9,216$).

2. *Add*: adds a binary missingness feature (1 if missing, otherwise 0) for each existing feature that has at least one missing value.

3. *Dummify*: introduces new k - 1 binary features for each k-valued categorical feature

4. *Impute*: replaces a missing value with the column mean for numeric feature and the mode for binary feature.

5. *Normalize*: applies min-max scaling to numeric feature to have the range $[0, 1]$. This ensures SVM model does not get biased and falsely dominated by a feature with extreme values

6. *Discard*: removes a instance whose outcome value is missing.

7. *Extract*: selects a predictive feature for outcome `TARGET_B` using t-test for a numerical feature and Fishers association test for a binary feature. Bonferroni correction is applied to deal with multiple testing problems.

3 Support Vector Machine Modeling

Our pre-processing stage employs data cleansing, feature reduction and selection rules and identifies the final principal features that are used in modeling.

The pre-processing implementation yields 14059 examples of 197 features numerically encoded, and 1 binary label. Our pre-processing stage is complete in itself which enables building models efficiently without any data tranformations built into modeling steps. RapidMiner (RM) is used to build linear and Radial Basis Function (RBF) kernel SVM models. RM operators for cross validation , SVM modeling, model application and binomial accuracy measurement operators are networked approrpriately to build models. The parameter settings of SVM modeling operator are used to set the kernel function (linear or RBF), box constraint (C) and γ. We used 3-fold cross-validation with stratified sampling and used a grid search mechanism for SVM parameters (C and γ) to build an array of SVM models. C and γ values are itertiavely and combinatorically searched from $\{..., 2^{-3}, 2^{-2}, 2^{-1}, 1, 2, 2^2, 2^3, ...\}^2$. Our RM process can be found in Figures 1 and 2.

3.1 Vector Machine Modeling Procedure

```
DO_SVM_MODELING(pre_process_method, dataset, model_type)
  if(pre_process_method == pre_process_complete)
    pre_process_data(dataset)
  end
  performance_tracker = init_performance_tracker()
  grid_C = [2^-3, 2^-2, 2^-1, 1, 2^1, 2^2, 2^3];
  grid_Gamma = [2^-3, 2^-2, 2^-1, 1, 2^1, 2^2, 2^3];
  for folds = 1:10
    test = get_test_fold(dataset, fold);
    train = get_train_fold(dataset, fold);
    if(pre_process_method == pre_process_each_fold)
      pre_process_data(test);
      pre_process_data(train);
    end
    for each C in grid_C
       if(model_type == linear)
         svmModel = svmtrain(train_data, train_labels,
          'kernel_function', 'linear','BoxConstraint',C);
        prediction = svmclassify(svmModel, test_data);
        track_performance(performance_tracker, prediction,
          test_labels);
          end
       if(model_type == rbf)
        for each g in grid_Gamma
          svmModel = svmtrain(train_data, train_labels,
            'kernel_function', 'rbf','BoxConstraint',C,
              'rbf_sigma', g);
          prediction = svmclassify(svmModel, test_data);
          track_performance(performance_tracker, prediction,
            test_labels);
        end
      end
    end
  end
end
```

C	accuracy	precision	recall	f-score
1/2	0.6561	0.5571	0.0080	0.0079
1*	0.6563	0.5753	0.0087	0.0086
2	0.6560	0.5480	0.0083	0.0081

Table 1: Linear SVM accuracy

C	γ	accuracy	precision	recall	f-score
1	1/2	62.75	30.29	6.24	0.0520
	1	62.88	30.96	6.30	0.0520
	2	62.77	30.30	6.19	0.0510
2*	1/2	62.94	31.48	6.44	0.0530
	1*	62.96	31.60	6.48	0.0540
	2	62.95	31.34	6.40	0.0503
4	1/2	62.88	30.77	6.19	0.0515
	1	62.89	30.99	6.30	0.0523
	2	62.86	30.64	6.19	0.0510

Table 2: RBF SVM accuracy

4 Results and Analysis

Tables 1 and 2 illustrats the accuracy, precision, recall and f-score values we obtained for different linear and RBF kernel SVM models.

Overall, the best accuracy we obtained was in the Linear SVM model with accuracy of 65.63

From table 3, one can observe that the precision and recall are 57.53% and 0.87% respectively. In other words, positive predictions of the model

	truth 1	truth 0	precision
prediction 1	42	31	0.575342
prediction 0	4801	9185	0.656728
recall	0.008672	0.996636	

Table 3: Confusion matrix for Linear SVM with $C = 1$

	truth 1	truth 0	precision
prediction 1	314	679	0.316213
prediction 0	4529	8537	0.653375
recall	0.064836	0.926324	

Table 4: Confusion matrix for RBF SVM with $[C, \gamma] = [2, 1]$

are true almost 58% of the time, while the positive predictions give that the examples are truly positive is 0.87

The best accuracy we obtained for the RBF kernel based SVM is 62.96% occurring at $[C, \gamma] = [2, 1]$. For this result, the confusion matrix can be seen in table 4.

At the outset, one can see that there is an increase in the number of both true-positives and false-positives when compared with the linear SVM Model. Compared with the linear SVM model, this model seems to have a more positive prediction bias which is not necessarily a good one as the precision dropped to 32%. Applying similar evaluation as done for the linear SVM above, the recall for the RBF SVM improved to 6.5% owing to the increase in true-positives and the specificity relatively decreased to around 93%. The f-score of RBF kernel SVM is 0.054 which is around 6X of that of the linear VSM model. Based on the f-score indicator, we feel that the RBF kernel based SVM is a better model.

5 Conclusion and Limitations

5.1 Leakage

In our implementation, leakage could be an issue as the test and datasets influence each other as our pre-processing implementation treats the dataset as a whole and does not pre-process separately for training and test folds. In our work, we tried to quantify the real effect of this leakage. In this process, we pre-processed the training and test datasets independently in order to avoid leakage during each fold of cross-validation. The Linear SVM model built in this way with C=1 yielded an accuracy of 64.19

5.2 Confusion Matrices

The precision scores for positive class obtained for the linear and RBF SVM models are low. The number of true-positives are significantly low which makes the models fare poorly in terms of classification. If our objective is to increase the true-positives and reduce the false-negatives and thereby reduce type-I and type-II errors, then the results should be used to feed-in to our data pre-processing implementation in order to improve our pre-processing rules. This shows a need to develop more robust data pre-processing implementation which transforms feature set more effectively.

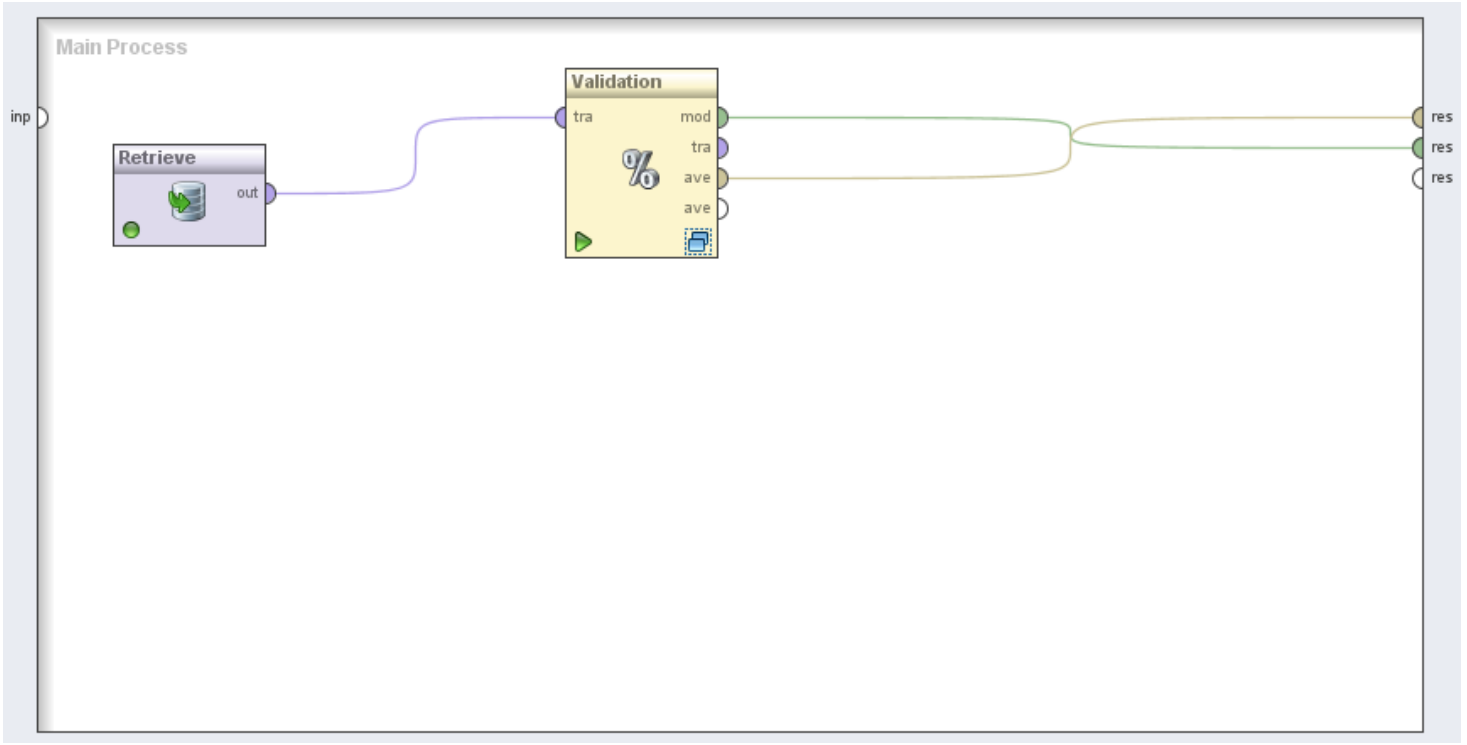

Figure 1: RapidMiner SVM process tree

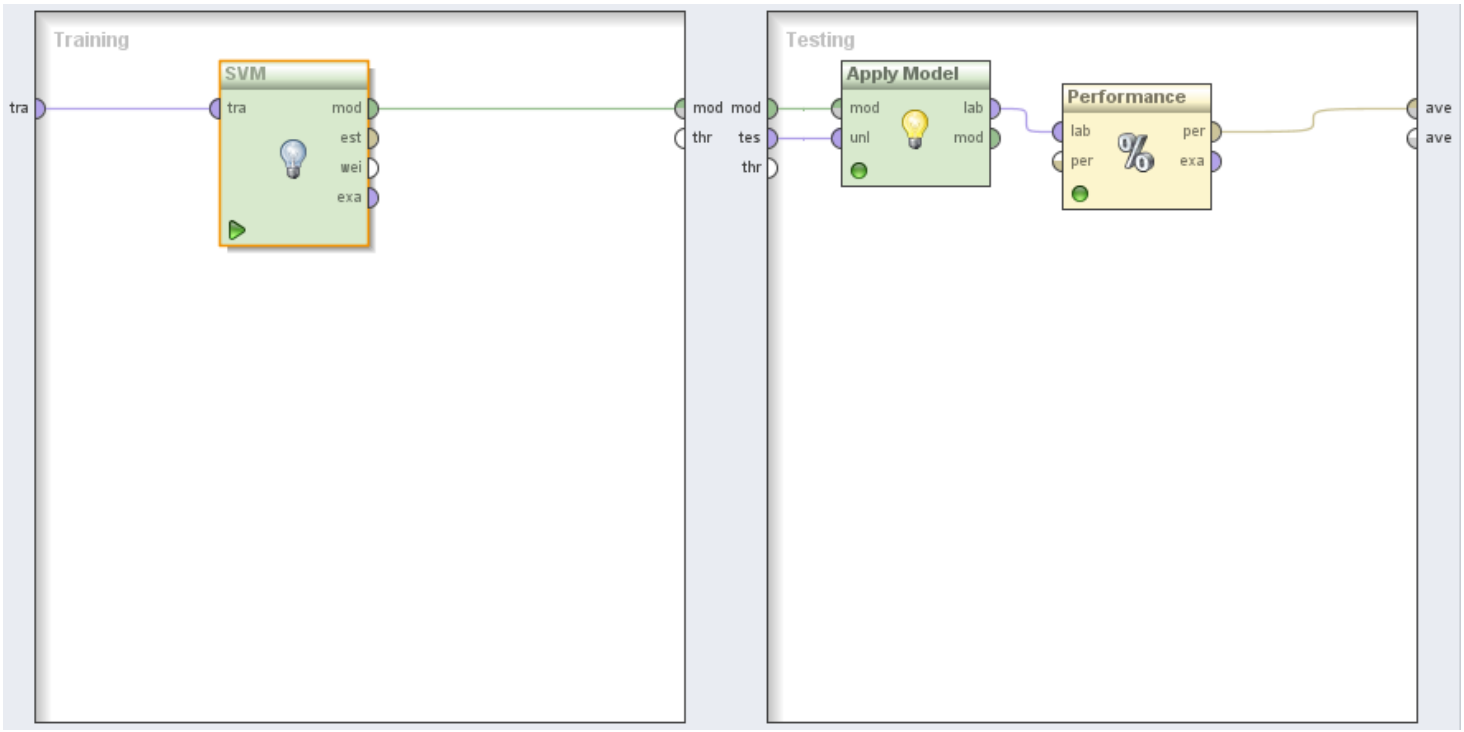

Figure 2: RapidMiner training process sub–tree