Sergio Munguia

**MatLab Assignment 2 Report**

We take in a data->prepare data and then train the model , select a model - > train the model

* Using known data and known data responses ( cancer vs normal )

We then measure the acurracy of the model selected

Then use for prediction the model and new data for predicted responses.

By doing classification we can predict the best group a new observation belongs to.

Supervised learning , There is a tradeoff between the expressiveness of a hypothesis space and the complexity of finding a good hypothesis within that space.

K Fold – Cross validation is an honest assessment of the true accuracy.

In cross validation, we divide our data into a large training set and smaller validation set, then train on the training set and use the validation set to measure our accuracy.

1. Maximize both of the sets. – To get best observation

SVM - 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 operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM’s theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.

**Questions for additional reflection:**

1. *What happens to some key values (and decisions made upon those values), if you change line 13 in the code (settings for the random number generator)?*

The misclassification error changes & It is better to randomly select validation examples, rather than go on a set of examples specifically for validation, because you want the validation set to be diverse.

2*. What are the main differences between "feature selection by filtering" and "sequential filter selection"? Why was the former used as a preprocessing step for the latter?*

The goal is to make the size of the dimensions of the data smaller.

The sequential filter selection gives the smaller MCE % on the test set.

When we graph it shows number N features in a line of data plots, excluding the redundant.

Feature selection algorithm does not consider interaction between features; besides, features selected from the list based on their individual ranking may also contain redundant information

since the typical goal of classification is to minimize the MCE, the feature selection procedure performs a sequential search using the MCE of the learning algorithm

3. *Why did we not use manual / interactive feature selection (using the Classification Learner App) in this case (as we did for the Fisher Iris problem, for instance)?*

You can use Classification Learner to train models to classify data. Using this app, you can explore supervised machine learning using various classifiers. You can explore your data, select features, specify validation schemes, train models

4. *Why is the accuracy "per class" (computed in part 5.1) relevant?*

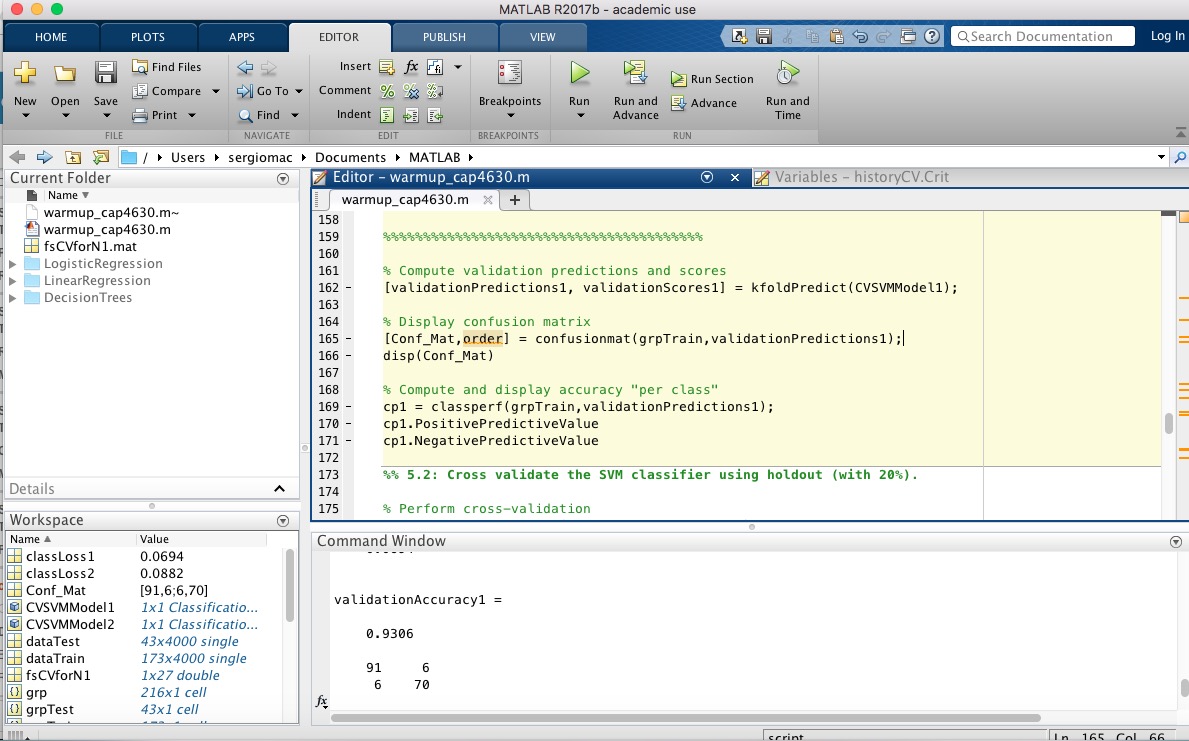
The mean of data accuracy per model can be different. It shows the difference between the models of K- Fold Cross validation and SVM.

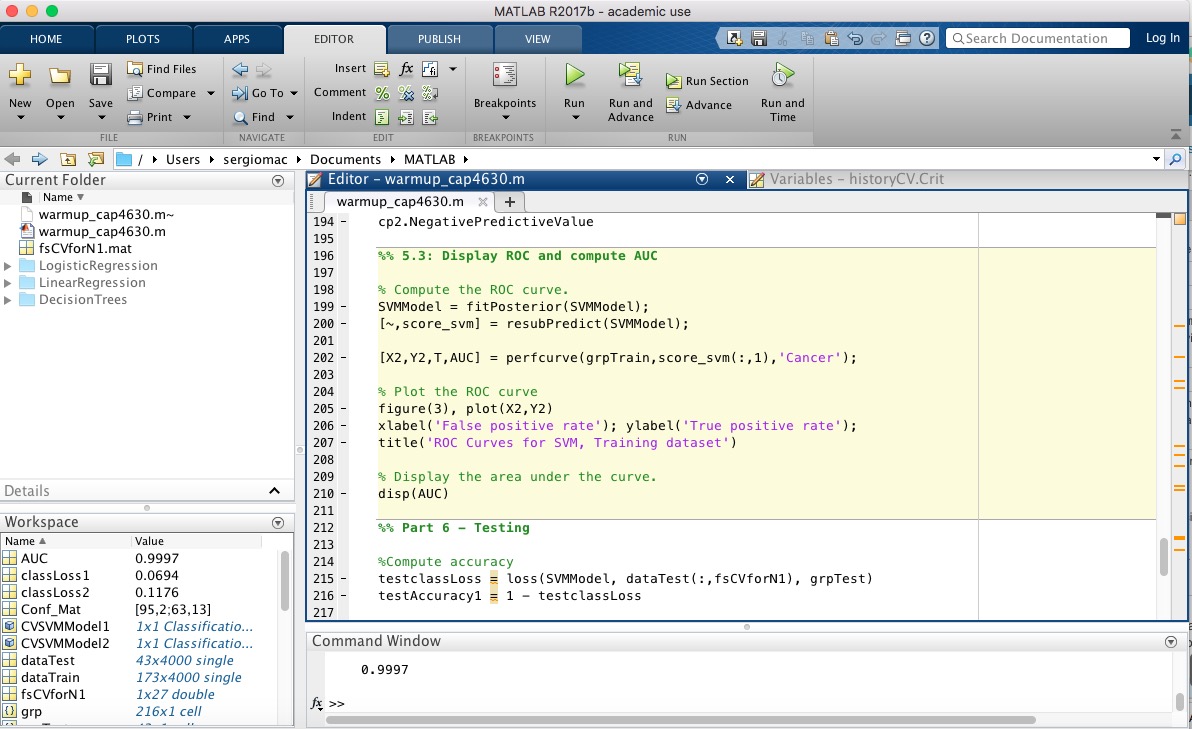
5. *The test results for SVM were basically perfect. Can you trust them completely? Why (not)?*

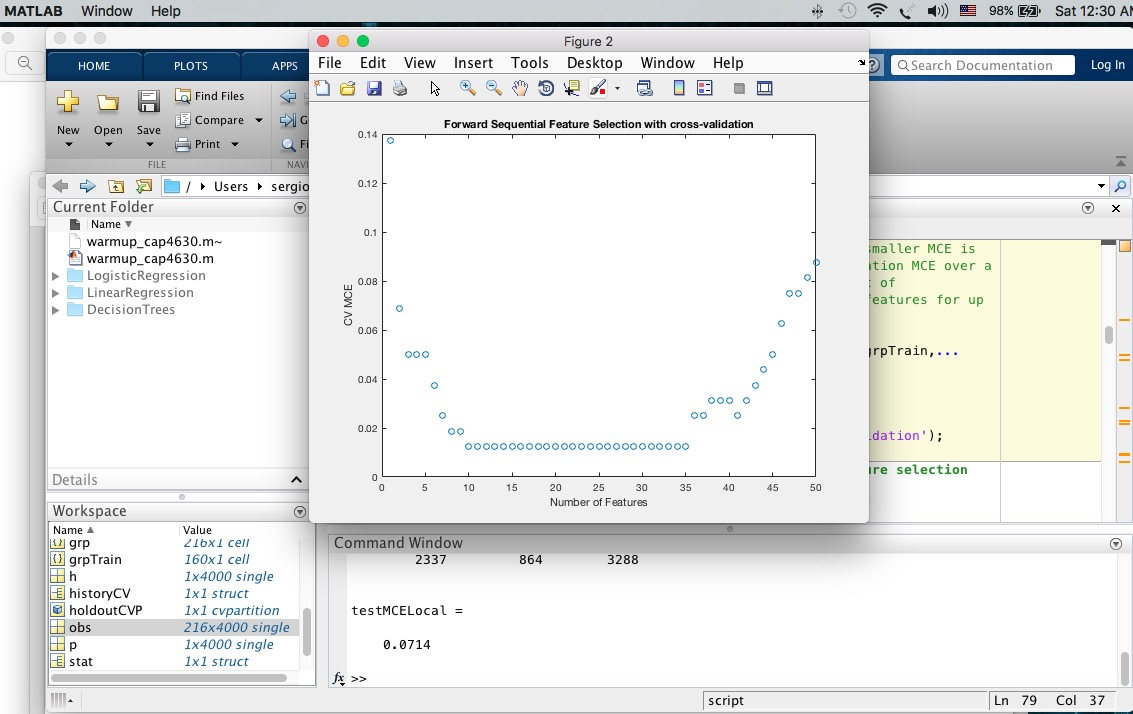
No because of the boundaries that we set on the decision line, it’s to make it generalized, and it always is still an approximate and there is always a chance for a classifier to be untrustworthy.

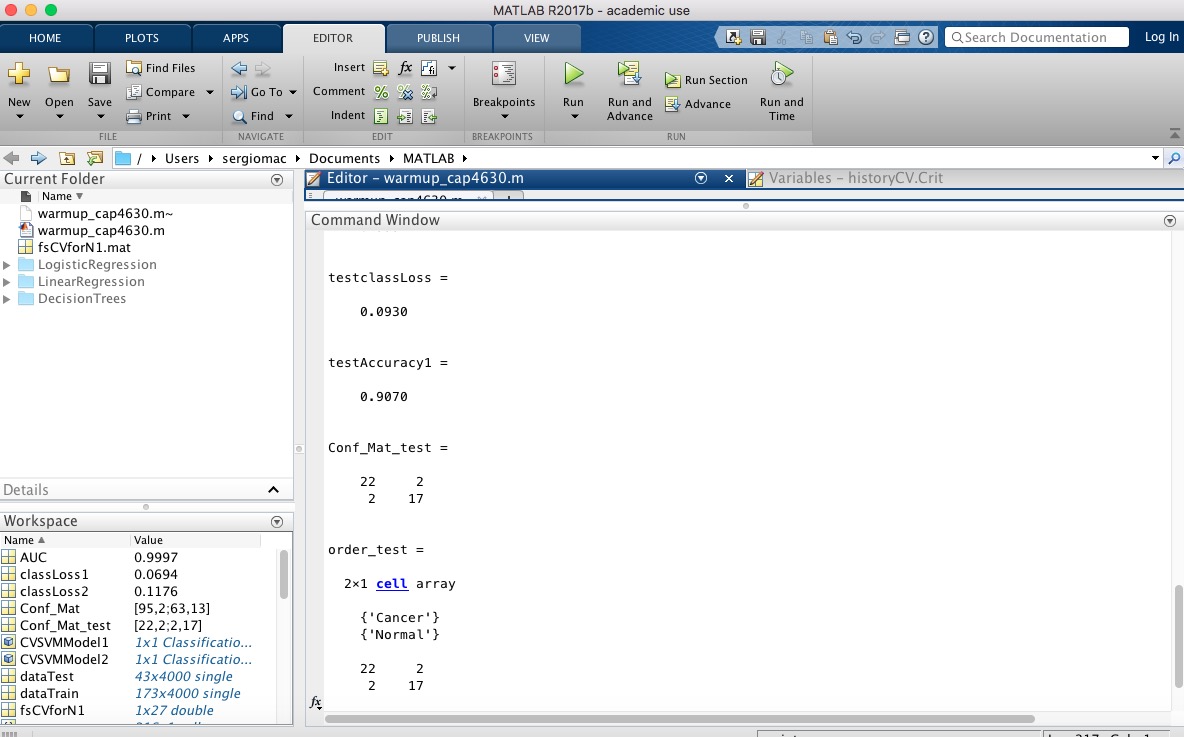
In practice, the reason that SVMs tend to be resistant to over-fitting, even in cases where the number of attributes is greater than the number of observations, is that it uses regularization.

The SVM is an approximate implementation of a bound on the generalization error, that depends on the margin (essentially the distance from the decision boundary to the nearest pattern from each class).









1. %% CAP 4630 - Intro to AI - FAU - Dr. Marques - Fall 2016
2. %% Final Project - Warmup exercise (2-**class** classifier)
3. %% Part 1 - Loading the data
5. load ovariancancer;
7. %% Part 2 - Feature selection
8. %% 2.1: Prepare the data
10. % Set the random number generator to a known state.
11. % Otherwise, your results may differ.
12. rng(2000,'twister');
14. % Partitioning the dataset (**for** feature selection): 160 points **for** the
15. % training set and the remaining 56 **for** testing.
16. holdoutCVP = cvpartition(grp,'holdout',56);
17. dataTrain = obs(holdoutCVP.training,:);
18. grpTrain = grp(holdoutCVP.training);
20. %% 2.2: Feature selection (step 1): using a simple filter approach
22. % Filters are usually used as a pre-processing step in feature selection,
23. % due to their simplicity and speed.
24. % A widely-used filter method **for** bioinformatics data is to apply
25. % a statistical test separately on each feature, assuming that there is
26. % no interaction between features.
28. % For example, we might apply the \_t\_-test on each feature and compare
29. % \_p\_-value (or the absolute values of \_t\_-statistics) **for** each
30. % feature as a measure of how effective it is at separating groups.
31. dataTrainG1 = dataTrain(grp2idx(grpTrain)==1,:);
32. dataTrainG2 = dataTrain(grp2idx(grpTrain)==2,:);
33. [h,p,ci,stat] = ttest2(dataTrainG1,dataTrainG2,'Vartype','unequal');
35. % In order to get a general idea of how well-separated the two groups are
36. % by each feature, we plot the empirical cumulative distribution function
37. % (CDF) of the \_p\_-values:
38. figure(1), ecdf(p);
39. xlabel('P value');
40. ylabel('CDF value')
42. %% 2.3: Feature selection (step 2): using sequential feature selection
44. % Use the filter results from the previous section as a
45. % pre-processing step to select features: sort the features according
46. % to their p values and select the top 150 features.
47. [~,featureIdxSortbyP] = sort(p,2);
48. fs1 = featureIdxSortbyP(1:150);
50. % Generate a stratified 10-fold partition **for** the training set:
51. tenfoldCVP = cvpartition(grpTrain,'kfold',10);
53. % Apply forward sequential feature selection on these 150 features.
54. % The function |sequentialfs| provides a simple way (the **default** option) to
55. % decide how many features are needed. It stops when the first local
56. % minimum of the cross-validation MCE (misclassification error) is found.
57. fun = @(xtrain,ytrain,xtest,ytest) ...
58. sum(~strcmp(ytest,classify(xtest,xtrain,ytrain,'quadratic')));
59. fsLocal = sequentialfs(fun,dataTrain(:,fs1),grpTrain,'cv',tenfoldCVP);
61. % The selected features are the following:
62. fs1(fsLocal)
64. % To evaluate the performance of the selected model with these four features,
65. % we compute the MCE on the 56 test samples.
66. testMCELocal = crossval(fun,obs(:,fs1(fsLocal)),grp,'partition',...
67. holdoutCVP)/holdoutCVP.TestSize
69. %% 2.4: Feature selection (step 3): improving sequential feature selection
71. % The algorithm may have stopped prematurely. Sometimes a smaller MCE is
72. % achievable by looking **for** the minimum of the cross-validation MCE over a
73. % reasonable range of number of features. Let's draw a plot of
74. % the cross-validation MCE as a function of the number of features **for** up
75. % to 50 features.
77. [fsCVfor50,historyCV] = sequentialfs(fun,dataTrain(:,fs1),grpTrain,...
78. 'cv',tenfoldCVP,'Nf',50);
79. figure(2), plot(historyCV.Crit,'o');
80. xlabel('Number of Features');
81. ylabel('CV MCE');
82. title('Forward Sequential Feature Selection with cross-validation');
84. %% 2.5: Feature selection (step 4): performing actual feature selection
86. %%%%% ENTER THE VALUE OF N1 HERE!!! %%%%
87. %
88. N1 = 27
89. %
90. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92. fsCVforN1 = fs1(historyCV.In(N1,:));
94. % Save selected features **for** later use
95. save('fsCVforN1.mat', 'fsCVforN1');
97. % To show these N1 features in the order in which they are selected in the
98. % sequential forward procedure, we find the row in which they first become
99. % **true** in the |historyCV| output:
100. [orderlist,ignore] = find( [historyCV.In(1,:); diff(historyCV.In(1:N1,:) )]' );
101. fs1(orderlist);
103. % To evaluate these N1 features, we compute their MCE **for** QDA on the test
104. % set. We get the smallest MCE value so far:
105. testMCECVforN1 = crossval(fun,obs(:,fsCVforN1),grp,'partition',...
106. holdoutCVP)/holdoutCVP.TestSize
108. %% Part 3 - Starting fresh (with only the selected features)
110. close all; clear all; clc
112. load ovariancancer;
113. load fsCVforN1;
115. %% 3.1: Partition dataset into 3 groups
117. % 80% **for** training and cross validation
118. % 20% **for** testing
120. %%%%% ENTER YOUR CODE HERE!!! %%%%
121. holdoutCVP = cvpartition(grp,'holdout',43);
122. dataTrain = obs(holdoutCVP.training,:);
123. grpTrain = grp(holdoutCVP.training,:);
124. dataTest = obs(holdoutCVP.test,:);
125. grpTest = grp(holdoutCVP.test,:);
126. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
128. %% Part 4 - Building a model (SVM)
130. X = dataTrain(:,fsCVforN1);
131. Y = grpTrain;
133. % Train an SVM classifier using the radial basis kernel.  Let the software
134. % find a scale value **for** the kernel function.  It is good
135. % practice to standardize the predictors.
137. SVMModel = fitcsvm(X,Y,'Standardize',**true**,'KernelFunction','RBF',...
138. 'KernelScale','auto', 'ClassNames', {'Cancer','Normal'});
140. %% Part 5 - Evaluating the model
142. %% 5.1: Cross validate the SVM classifier using 10-fold cross validation.
144. % Perform cross-validation
145. %%%%% ENTER YOUR CODE HERE!!! %%%%
146. CVSVMModel1 = crossval(SVMModel,'kfold',10);
147. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149. % Estimate the out-of-sample misclassification rate.
150. %%%%% ENTER YOUR CODE HERE!!! %%%%
151. classLoss1 = kfoldLoss(CVSVMModel1)
153. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
155. % Compute validation accuracy
156. %%%%% ENTER YOUR CODE HERE!!! %%%%
157. validationAccuracy1 = 1 - classLoss1
159. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
161. % Compute validation predictions and scores
162. [validationPredictions1, validationScores1] = kfoldPredict(CVSVMModel1);
164. % Display confusion matrix
165. [Conf\_Mat,order] = confusionmat(grpTrain,validationPredictions1);
166. disp(Conf\_Mat)
168. % Compute and display accuracy "per class"
169. cp1 = classperf(grpTrain,validationPredictions1);
170. cp1.PositivePredictiveValue
171. cp1.NegativePredictiveValue
173. %% 5.2: Cross validate the SVM classifier using holdout (with 20%).
175. % Perform cross-validation
176. CVSVMModel2 = crossval(SVMModel,'holdout',0.2);
178. % Estimate the out-of-sample misclassification rate.
179. classLoss2 = kfoldLoss(CVSVMModel2)
181. % Compute validation accuracy
182. validationAccuracy2 = 1 - classLoss2
184. % Compute validation predictions and scores
185. [validationPredictions2, validationScores2] = kfoldPredict(CVSVMModel2);
187. % Display confusion matrix
188. [Conf\_Mat,order] = confusionmat(grpTrain,validationPredictions2);
189. disp(Conf\_Mat)
191. % Compute and display accuracy "per class"
192. cp2 = classperf(grpTrain,validationPredictions1);
193. cp2.PositivePredictiveValue
194. cp2.NegativePredictiveValue
196. %% 5.3: Display ROC and compute AUC
198. % Compute the ROC curve.
199. SVMModel = fitPosterior(SVMModel);
200. [~,score\_svm] = resubPredict(SVMModel);
202. [X2,Y2,T,AUC] = perfcurve(grpTrain,score\_svm(:,1),'Cancer');
204. % Plot the ROC curve
205. figure(3), plot(X2,Y2)
206. xlabel('False positive rate'); ylabel('True positive rate');
207. title('ROC Curves for SVM, Training dataset')
209. % Display the area under the curve.
210. disp(AUC)
212. %% Part 6 - Testing
214. %Compute accuracy
215. testclassLoss = loss(SVMModel, dataTest(:,fsCVforN1), grpTest)
216. testAccuracy1 = 1 - testclassLoss
218. % Label the test sample observations.
219. % Display the results **for** the observations in the test sample.
220. [label\_test,score\_test] = predict(SVMModel,dataTest(:,fsCVforN1));
222. % 3.2: Display confusion matrix
223. % PASTE YOUR CODE HERE!
224. [Conf\_Mat\_test,order\_test] = confusionmat(grpTest,label\_test)
225. disp(Conf\_Mat\_test)
227. % Compute and display accuracy "per class"
228. cp3 = classperf(grpTest,label\_test);
229. cp3.PositivePredictiveValue
230. cp3.NegativePredictiveValue
232. % Display ROC and compute AUC
233. % Compute the ROC curve.
234. [X\_test,Y\_test,T\_test,AUC\_test] = perfcurve(grpTest,score\_test(:,1),'Cancer');
236. % Plot the ROC curve
237. figure(4), plot(X\_test,Y\_test)
238. xlabel('False positive rate')
239. ylabel('True positive rate')
240. title('ROC for Classification by SVM, Test Data Set')
242. disp(AUC\_test)