Sergio Munguia

**MatLab Assignment 2 Report**

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.

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 exmples 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?*

For random feature variations , the sequential filtering was selected because it sets up between the bounderies for the features.

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?*

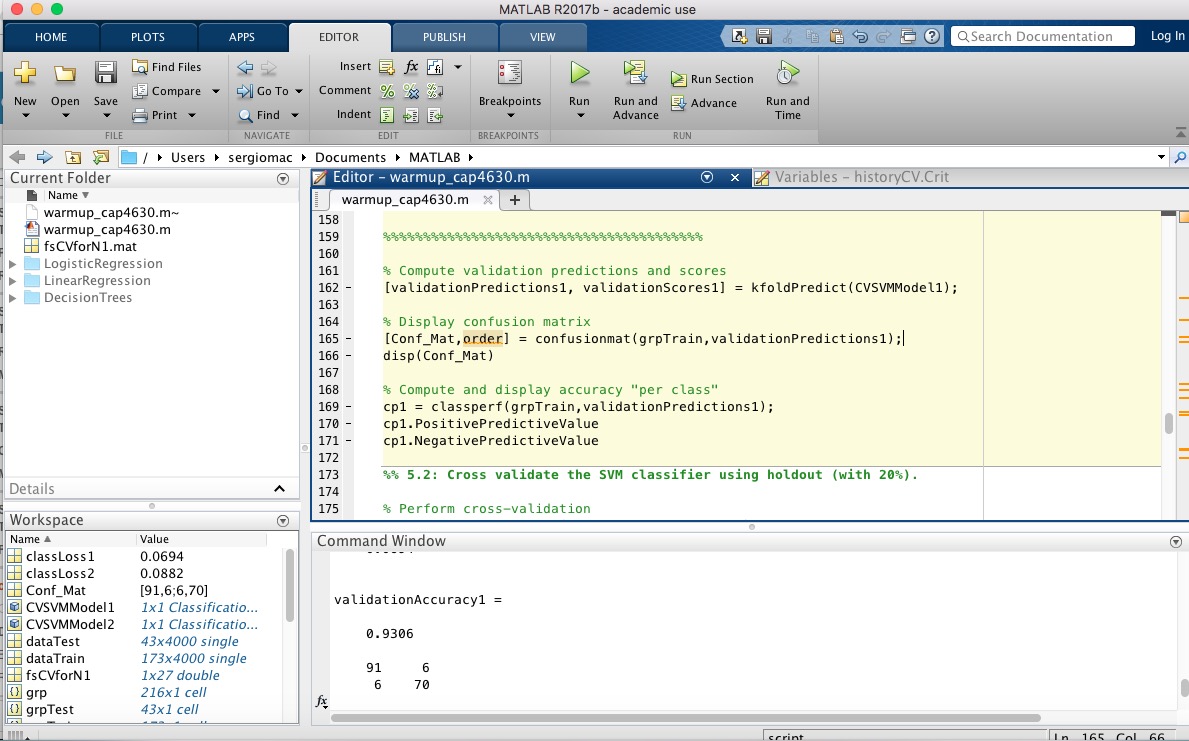
The mean of data accuracy per model can be different. It shows the difference between the models of K- Fold Cross validation and SVM is fairly.

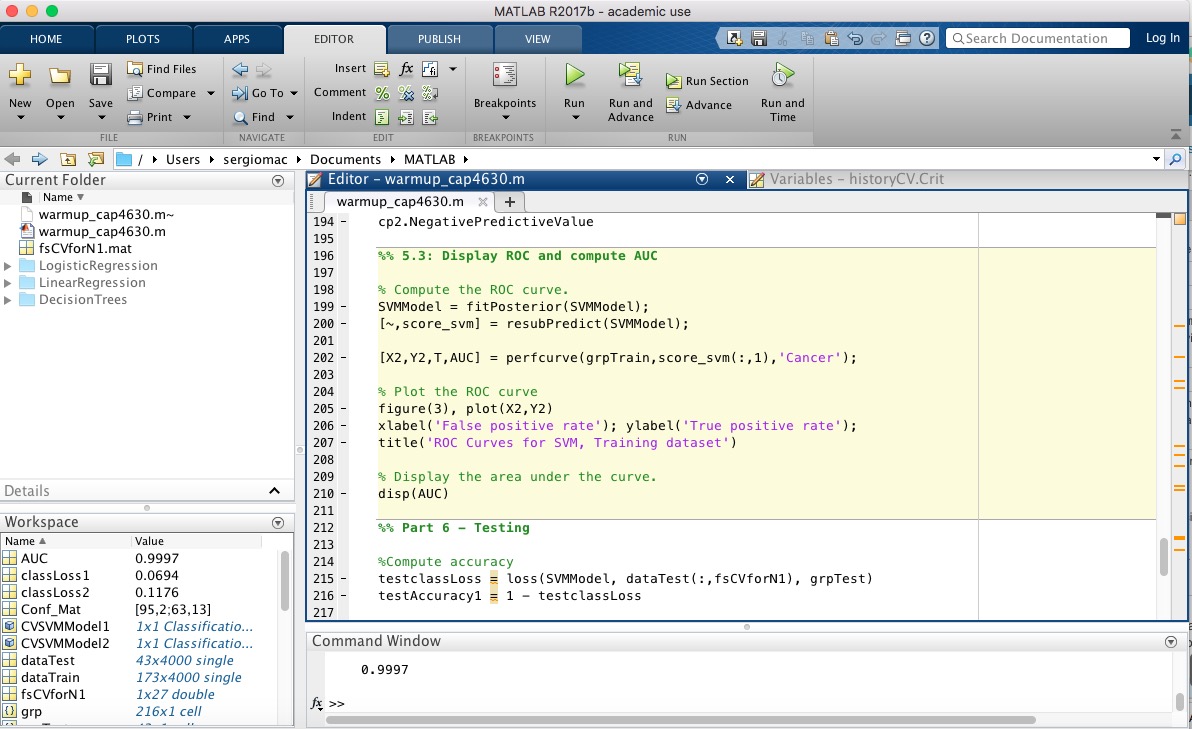
5. *The test results for SVM were basically perfect. Can you trust them completely? Why (not)?*

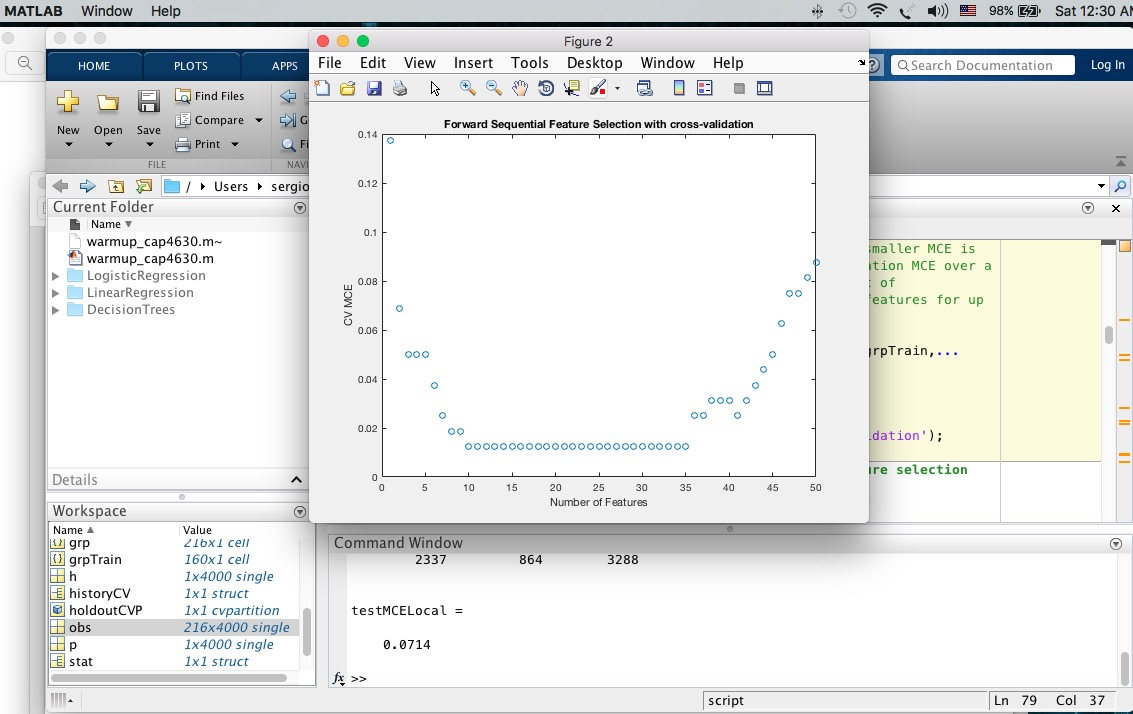
No because of the boundaries that we set on the decision line, it’s to make it generalized, and it always is still an approximate and there is always a chance for a classifier to be untrustworthy.

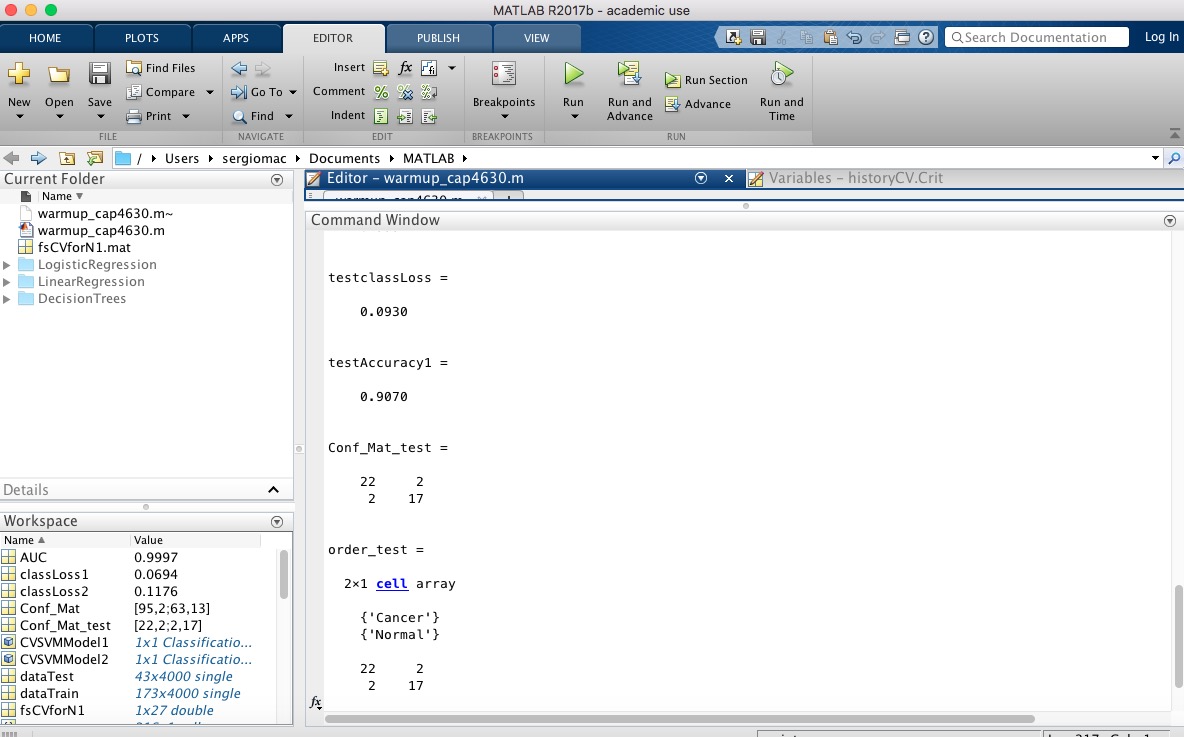
In practice, the reason that SVMs tend to be resistant to over-fitting, even in cases where the number of attributes is greater than the number of observations, is that it uses regularization.

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. %% Final Project - Warmup exercise (2-**class** classifier)
2. %% Part 1 - Loading the data
4. load ovariancancer;
6. %% Part 2 - Feature selection
7. %% 2.1: Prepare the data
9. % Set the random number generator to a known state.
10. % Otherwise, your results may differ.
11. rng(2000,'twister');
13. % Partitioning the dataset (**for** feature selection): 160 points **for** the
14. % training set and the remaining 56 **for** testing.
15. holdoutCVP = cvpartition(grp,'holdout',56);
16. dataTrain = obs(holdoutCVP.training,:);
17. grpTrain = grp(holdoutCVP.training);
19. %% 2.2: Feature selection (step 1): using a simple filter approach
21. % Filters are usually used as a pre-processing step in feature selection,
22. % due to their simplicity and speed.
23. % A widely-used filter method **for** bioinformatics data is to apply
24. % a statistical test separately on each feature, assuming that there is
25. % no interaction between features.
27. % For example, we might apply the \_t\_-test on each feature and compare
28. % \_p\_-value (or the absolute values of \_t\_-statistics) **for** each
29. % feature as a measure of how effective it is at separating groups.
30. dataTrainG1 = dataTrain(grp2idx(grpTrain)==1,:);
31. dataTrainG2 = dataTrain(grp2idx(grpTrain)==2,:);
32. [h,p,ci,stat] = ttest2(dataTrainG1,dataTrainG2,'Vartype','unequal');
34. % In order to get a general idea of how well-separated the two groups are
35. % by each feature, we plot the empirical cumulative distribution function
36. % (CDF) of the \_p\_-values:
37. figure(1), ecdf(p);
38. xlabel('P value');
39. ylabel('CDF value')
41. %% 2.3: Feature selection (step 2): using sequential feature selection
43. % Use the filter results from the previous section as a
44. % pre-processing step to select features: sort the features according
45. % to their p values and select the top 150 features.
46. [~,featureIdxSortbyP] = sort(p,2);
47. fs1 = featureIdxSortbyP(1:150);
49. % Generate a stratified 10-fold partition **for** the training set:
50. tenfoldCVP = cvpartition(grpTrain,'kfold',10);
52. % Apply forward sequential feature selection on these 150 features.
53. % The function |sequentialfs| provides a simple way (the **default** option) to
54. % decide how many features are needed. It stops when the first local
55. % minimum of the cross-validation MCE (misclassification error) is found.
56. fun = @(xtrain,ytrain,xtest,ytest) ...
57. sum(~strcmp(ytest,classify(xtest,xtrain,ytrain,'quadratic')));
58. fsLocal = sequentialfs(fun,dataTrain(:,fs1),grpTrain,'cv',tenfoldCVP);
60. % The selected features are the following:
61. fs1(fsLocal)
63. % To evaluate the performance of the selected model with these four features,
64. % we compute the MCE on the 56 test samples.
65. testMCELocal = crossval(fun,obs(:,fs1(fsLocal)),grp,'partition',...
66. holdoutCVP)/holdoutCVP.TestSize
68. %% 2.4: Feature selection (step 3): improving sequential feature selection
70. % The algorithm may have stopped prematurely. Sometimes a smaller MCE is
71. % achievable by looking **for** the minimum of the cross-validation MCE over a
72. % reasonable range of number of features. Let's draw a plot of
73. % the cross-validation MCE as a function of the number of features **for** up
74. % to 50 features.
76. [fsCVfor50,historyCV] = sequentialfs(fun,dataTrain(:,fs1),grpTrain,...
77. 'cv',tenfoldCVP,'Nf',50);
78. figure(2), plot(historyCV.Crit,'o');
79. xlabel('Number of Features');
80. ylabel('CV MCE');
81. title('Forward Sequential Feature Selection with cross-validation');
83. %% 2.5: Feature selection (step 4): performing actual feature selection
85. %%%%% ENTER THE VALUE OF N1 HERE!!! %%%%
86. %
87. N1 = 27
88. %
89. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91. fsCVforN1 = fs1(historyCV.In(N1,:));
93. % Save selected features **for** later use
94. save('fsCVforN1.mat', 'fsCVforN1');
96. % To show these N1 features in the order in which they are selected in the
97. % sequential forward procedure, we find the row in which they first become
98. % **true** in the |historyCV| output:
99. [orderlist,ignore] = find( [historyCV.In(1,:); diff(historyCV.In(1:N1,:) )]' );
100. fs1(orderlist);
102. % To evaluate these N1 features, we compute their MCE **for** QDA on the test
103. % set. We get the smallest MCE value so far:
104. testMCECVforN1 = crossval(fun,obs(:,fsCVforN1),grp,'partition',...
105. holdoutCVP)/holdoutCVP.TestSize
107. %% Part 3 - Starting fresh (with only the selected features)
109. close all; clear all; clc
111. load ovariancancer;
112. load fsCVforN1;
114. %% 3.1: Partition dataset into 3 groups
116. % 80% **for** training and cross validation
117. % 20% **for** testing
119. %%%%% ENTER YOUR CODE HERE!!! %%%%
120. holdoutCVP = cvpartition(grp,'holdout',43);
121. dataTrain = obs(holdoutCVP.training,:);
122. grpTrain = grp(holdoutCVP.training,:);
123. dataTest = obs(holdoutCVP.test,:);
124. grpTest = grp(holdoutCVP.test,:);
125. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127. %% Part 4 - Building a model (SVM)
129. X = dataTrain(:,fsCVforN1);
130. Y = grpTrain;
132. % Train an SVM classifier using the radial basis kernel.  Let the software
133. % find a scale value **for** the kernel function.  It is good
134. % practice to standardize the predictors.
136. SVMModel = fitcsvm(X,Y,'Standardize',**true**,'KernelFunction','RBF',...
137. 'KernelScale','auto', 'ClassNames', {'Cancer','Normal'});
139. %% Part 5 - Evaluating the model
141. %% 5.1: Cross validate the SVM classifier using 10-fold cross validation.
143. % Perform cross-validation
144. %%%%% ENTER YOUR CODE HERE!!! %%%%
145. CVSVMModel1 = crossval(SVMModel,'kfold',10);
146. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
148. % Estimate the out-of-sample misclassification rate.
149. %%%%% ENTER YOUR CODE HERE!!! %%%%
150. classLoss1 = kfoldLoss(CVSVMModel1)
152. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154. % Compute validation accuracy
155. %%%%% ENTER YOUR CODE HERE!!! %%%%
156. validationAccuracy1 = 1 - classLoss1
158. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
160. % Compute validation predictions and scores
161. [validationPredictions1, validationScores1] = kfoldPredict(CVSVMModel1);
163. % Display confusion matrix
164. [Conf\_Mat,order] = confusionmat(grpTrain,validationPredictions1);
165. disp(Conf\_Mat)
167. % Compute and display accuracy "per class"
168. cp1 = classperf(grpTrain,validationPredictions1);
169. cp1.PositivePredictiveValue
170. cp1.NegativePredictiveValue
172. %% 5.2: Cross validate the SVM classifier using holdout (with 20%).
174. % Perform cross-validation
175. CVSVMModel2 = crossval(SVMModel,'holdout',0.2);
177. % Estimate the out-of-sample misclassification rate.
178. classLoss2 = kfoldLoss(CVSVMModel2)
180. % Compute validation accuracy
181. validationAccuracy2 = 1 - classLoss2
183. % Compute validation predictions and scores
184. [validationPredictions2, validationScores2] = kfoldPredict(CVSVMModel2);
186. % Display confusion matrix
187. [Conf\_Mat,order] = confusionmat(grpTrain,validationPredictions2);
188. disp(Conf\_Mat)
190. % Compute and display accuracy "per class"
191. cp2 = classperf(grpTrain,validationPredictions1);
192. cp2.PositivePredictiveValue
193. cp2.NegativePredictiveValue
195. %% 5.3: Display ROC and compute AUC
197. % Compute the ROC curve.
198. SVMModel = fitPosterior(SVMModel);
199. [~,score\_svm] = resubPredict(SVMModel);
201. [X2,Y2,T,AUC] = perfcurve(grpTrain,score\_svm(:,1),'Cancer');
203. % Plot the ROC curve
204. figure(3), plot(X2,Y2)
205. xlabel('False positive rate'); ylabel('True positive rate');
206. title('ROC Curves for SVM, Training dataset')
208. % Display the area under the curve.
209. disp(AUC)
211. %% Part 6 - Testing
213. %Compute accuracy
214. testclassLoss = loss(SVMModel, dataTest(:,fsCVforN1), grpTest)
215. testAccuracy1 = 1 - testclassLoss
217. % Label the test sample observations.
218. % Display the results **for** the observations in the test sample.
219. [label\_test,score\_test] = predict(SVMModel,dataTest(:,fsCVforN1));
221. % 3.2: Display confusion matrix
222. % PASTE YOUR CODE HERE!
223. [Conf\_Mat\_test,order\_test] = confusionmat(grpTest,label\_test)
224. disp(Conf\_Mat\_test)
226. % Compute and display accuracy "per class"
227. cp3 = classperf(grpTest,label\_test);
228. cp3.PositivePredictiveValue
229. cp3.NegativePredictiveValue
231. % Display ROC and compute AUC
232. % Compute the ROC curve.
233. [X\_test,Y\_test,T\_test,AUC\_test] = perfcurve(grpTest,score\_test(:,1),'Cancer');
235. % Plot the ROC curve
236. figure(4), plot(X\_test,Y\_test)
237. xlabel('False positive rate')
238. ylabel('True positive rate')
239. title('ROC for Classification by SVM, Test Data Set')
241. disp(AUC\_test)