OR	AUC		TOP10		TOP20		nullp		
	SVM	Normal	SVM	Normal	SVM	Normal	SVM	Normal	#SV
1	0.455	0.486	0.500	0.375	0.750	0.750	0.418	0.513	125.875
1.25	0.906	0.908	5.125	5.375	6.500	6.875	0.399	0.456	79.500
1.5	0.976	0.982	7.625	7.875	9.125	9.500	0.439	0.477	51.625
1.75	0.998	0.998	9.000	9.250	10.000	10.000	0.455	0.466	16.125
2	0.999	0.999	9.375	9.500	10.000	10.000	0.478	0.484	6.875

OR2	AUC		TOP10		TOP20		nullp		
	SVM	Normal	SVM	Normal	SVM	Normal	SVM	Normal	#SV
1.1-1.5	0.865	0.886	5.625	5.125	6.250	6.75	0.428	0.483	76.875
1.1-2	0.896	0.899	6.750	6.750	7.625	8.25	0.455	0.484	37.125
1.5-2	0.996	0.996	9.000	8.875	9.750	9.75	0.468	0.481	19.125
2-2.5	0.998	0.998	9.125	9.125	10.000	10.00	0.472	0.474	3.625

Preprocessing with SVM

Ryan Del Bel

February 27, 2014

1 Simulations with uncorrelated features

We first consider the simple case with uncorrelated features. We let n=500, simulate 10 truly associated features and 190 noisy features. The effect of the features on the outcome was achieved by setting the odds ratio in the underlying logistic model. For simplicity and efficiency when using the linear SVM we simply set the tuning parameter C to be one. Since the number of support vectors tends to be very large we only remove the support vectors that have maximal weights (i.e. with absolute value 1). For each feature we obtain the p-value from the t-test on the full sample and the sample with the support vectors with maximal weight removed. We then calculate the AUC, number of truly associated features that are in the top 10, 20 smallest p-values, and the number of support vectors removed. We replicate these simulations 100 times and report the average of each of the above statistics.

We first simulate data with the 10 associated features having the same OR.
Next we simulate 10 associated features with OR evenly spaced between a specified minimum and maximum OR.
In both cases the results are slightly worse when using SVM as a preprocessing step. We notice that even when only removing the support vectors with maximal weight, we still usually remove a very large number of support vectors. We will now limit the number of support vectors we remove to k. If there are more than k support vectors with maximal weight we then randomly remove $k/2$ support vectors with weight 1 and $k/2$ support vectors with weight -1.
Even when limiting the number of support vectors we remove, the normal approach still performs better.

OR3	**AUC**		**TOP10**		**TOP20**		**nullp**		
	SVM	Normal	SVM	Normal	SVM	Normal	SVM	Normal	#SV
1	0.528	0.532	0.625	0.500	0.875	1.00	0.431	0.507	113.625
1.25	0.823	0.834	3.750	4.250	5.125	5.25	0.422	0.481	91.500
1.5	0.963	0.964	6.375	6.625	7.875	8.25	0.438	0.485	71.500
1.75	0.973	0.976	7.375	7.750	8.875	9.00	0.435	0.478	56.000
2	0.990	0.988	8.625	8.375	9.500	9.50	0.442	0.469	35.500

OR5	**AUC**		**TOP10**		**TOP20**		**nullp**		
	SVM	Normal	SVM	Normal	SVM	Normal	SVM	Normal	#SV
1	0.522	0.524	0.250	0.375	0.750	0.750	0.490	0.496	10
1.25	0.814	0.817	4.125	4.000	5.625	5.625	0.477	0.485	10
1.5	0.951	0.953	6.500	6.500	8.000	7.875	0.480	0.488	10
1.75	0.982	0.984	7.500	7.750	9.000	9.250	0.457	0.465	10
2	0.981	0.980	8.500	8.375	9.375	9.250	0.467	0.475	10

OR6	**AUC**		**TOP10**		**TOP20**		**nullp**		
	SVM	Normal	SVM	Normal	SVM	Normal	SVM	Normal	#SV
1.1-1.5	0.844	0.850	4.500	4.500	5.750	6.00	0.468	0.482	10.000
1.1-2	0.874	0.872	6.000	6.000	7.000	6.75	0.476	0.486	10.000
1.5-2	0.961	0.965	7.375	7.375	8.750	8.75	0.467	0.479	10.000
2-2.5	0.991	0.991	8.625	8.625	9.625	9.75	0.464	0.473	8.875