BIOINFORMATICS HAL APPLICATION

by

James Duin

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major:

Under the Supervision of

Lincoln, Nebraska

,

Bio HAL Application

Training and Testing Coarse Grain and Fine Grain Classifiers

The first step is to examine the dataset.

Classes	All
0	19136
1	13
2	185
3	324
4	190
5	11
6	104
7	59
8	76
Total	20098
Shape	450

Table 1: This is what the dataset looks like there are 20098 instances total with 450 features each.

Next the dataset is partitioned into 10 folds, each fold contains a representative proportion of each of the classes, the instances are added to each partition at random. The total partitioning looks like Table **??**

All	Folds	0	1	2	3	4	5	6	7	8
1	2010	1914	1	19	32	19	1	11	6	7
2	2010	1914	1	19	32	19	1	11	6	7
3	2010	1914	1	19	32	19	1	11	5	8
4	2010	1914	1	19	32	19	1	10	6	8
5	2010	1914	1	18	33	19	1	10	6	8
6	2010	1914	1	18	33	19	1	10	6	8
7	2010	1913	2	18	33	19	1	10	6	8
8	2010	1913	2	18	33	19	1	10	6	8
9	2009	1913	2	18	32	19	2	10	6	7
10	2009	1913	1	19	32	19	1	11	6	7
Total	20098	19136	13	185	324	190	11	104	59	76

Table 2: This is what the folds of the dataset look like.

Then 9 of the folds are compressed into the test set and the fold held out is the test set the totals of each class in the train and test set for fold 1 is shown in Table **??**.

train	0	1	2	3	4	5	6	7	8	
Total	18088	17222	12	166	292	171	10	93	53	69
test	0	1	2	3	4	5	6	7	8	
Total	2010	1914	1	19	32	19	1	11	6	7

Table 3: This is what the train and test set look like.

How did I run coarse and fine classifiers.

coarse just used marked all positives as 1 and ran a binary classifer.

fine grained classifiers trained 8 separate classifiers for the 8 fine grained classes, so for fine grain classifier for 1, all other fine grained classes are marked as 0 in addition to all the coarse instances being marked as 0.

Also the Pr auc and Roc auc are the primary metrics for determining the performance of the classifier.

The next step is to determine what classifier can be applied to 'learn' the classes of this dataset.

because the experiment will involve running multiple rounds with increasing the instances to be trained on iteratively I tested each classifier against the full dataset and then a reduced dataset with one fifth of the negative instances.

I tried using SVM. Throughout this project I used the python library sci-kit learn [**?**]. The support vector machine implemented by this library has the following default parameters. SVC C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache-size=200, class-weight=None, verbose=False, max-iter=-1, decision-function-shape=None, random-state=None.

I tried different scaling methods (min max scaler, std scaler), I settled on std scaler.

I tried different feature select measures, I settled on 75 perc feature select.

I tried different different C costs, kernels, decision function shape, gamma, tolerance settled on classif = svm.SVC(C=1.0, kernel='rbf',decisionfunctionshape='ovo',gamma=0.0025, tol=0.00001)

I left the class weight as balanced for this part, the results did not show much advantage for using the fine grained classifier.

next I tried using a Logistic regression classifier.

I tried different scaling methods, min max scaler, std scaler, I settled on min max scaler.

I tried different feature select measures, decided to use all of the features.

I tried different C costs, tolerances, and class weights. I settled on C=0.1, tol = 0.00001, and weight equal to the balanced, adjusted via a scaling line.

I also further tuned the fine grained classifiers starting from the initial scaling from the line an then multiplying that by a ratio. I got this vector of ratios [0.87, 0.4, 0.78, 0.65, 3.48, 0.78, 1.74, 0.87]

This shows the advantage to fine grained labels to justify the experiment.

coarse-pr	fine-pr	coarse-roc	fine-roc	coarse-acc	fine-acc	coarse-f1	fine-f1
0.898	0.901	0.905	0.896	0.767	0.945	0.259	0.474
0.870	0.869	0.847	0.846	0.803	0.944	0.272	0.456
0.897	0.907	0.895	0.901	0.792	0.947	0.287	0.500
0.864	0.866	0.852	0.848	0.778	0.943	0.256	0.430
0.855	0.865	0.859	0.859	0.795	0.947	0.269	0.451
0.867	0.869	0.874	0.865	0.785	0.939	0.263	0.417
0.871	0.887	0.873	0.881	0.784	0.940	0.269	0.442
0.835	0.845	0.843	0.842	0.794	0.940	0.258	0.388
0.870	0.878	0.869	0.871	0.784	0.939	0.262	0.417
0.873	0.873	0.891	0.890	0.786	0.933	0.279	0.368
avg 0.870	avg 0.876	avg 0.871	avg 0.870	avg 0.787	avg 0.942	avg 0.268	avg 0.434

Table 4: Here are the results for the logistic regression passive 10 folds.

Passive SVM Rbf kernel vs Logistic Reg

add text'example cite'[**?**]