Domain and Data

Prepared for the Neural Information Processing Symposium 2003 Feature Extraction Workshop

http://clopinet.com/isabelle/Projects/NIPS2003 (http://clopinet.com/isabelle/Projects/NIPS2003)

Data

MADELON is an artificial dataset, which was part of the NIPS 2003 feature selection challenge. This is a two-class classification problem with continuous input variables. The difficulty is that the problem is multivariate and highly non-linear.

MADELON is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the +-1 labels). We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized.

Problem Statement

The NIPS 2003 challenge in feature selection is to find feature selection algorithms that significantly outperform methods using all features in performing a binary classification task.

Solution Statement

We will develop a binary classification model and attempt to augment its performance using automatic feature selection techniques.

Metric

We will use accuracy score for comparing models.

Benchmark

We will use as a benchmark our mean accuracy across five random train test splits using a K Nearest Neighbors model with an optimal value for number of `n_neighbors`. This model had a 73.8% accuracy.

Result from Step 1 from Benchmarking

With the naive LogisticRegression it appears that model worked very well on train set however performed poorly on the test data set. **The train score was perfectly 1.0 and test data score dcid poorly 0.544** . This suggest we need to further tune the model with more penaly or even performing LogisticRegression using Lasso method.

Result from Step 2 Identify Salient Features

We ran 6 models with LogisticRegression. **With penalty Lasso and C value as 0.027 we found the highest test score with 8 features.** We will use this model and compare that against other models e.g. SelectKBest, KNN and GridSearchCV

	feature	coef_
475	feat_475	0.308325
48	feat_048	0.135008
307	feat_307	0.036766
46	feat_046	0.029403
378	feat_378	0.027127
424	feat_424	0.016196
329	feat_329	0.013342
282	feat_282	0.009831
116	feat_116	0.003299
338	feat_338	0.000000

Results from Step 3 Build the model

I built multiple models using GridSearch classifier with Logistic Regression and KNN models. First I construct pipeline to run on GridSearch with Logistic Regression

1. Build a new pipeline for LogisticRegression.
2. Ran model on L1 and L2
3. With Regularization 0.01, .1, .2, 0.03
4. Run 5 Fold Grid Serach
5. Review the Score

Then I Construct a pipeline that uses SelectKBest, KNN and ran on GridSearch classifer

1. Build a new pipeline for SelectKBest and KNN.
2. Set the neighbors between 11 and 21
3. Select the max of 10 features using SelectKBest
4. Run 5 Fold Grid Serach
5. Review the Score
6. Compare with LogisticRegression Score
7. Review features

KNN/SelectKBest is significantly higher and we will use that for the model. See below

Logistic Regression Best Model (Total 8 models Ran)

```
{'penalty': 'l1', 'C': 0.03}
0.626
LogisticRegression(C=0.03, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

KNN Best Model (Total 6 models Ran)

```
{'n_neighbors': 11}
0.816
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=11, p=2,
           weights='uniform')
```

Recommendations: I would have liked to graph on the features to visually see which features might have better weight so that would be something I would like to explore, also with KNN I would like to continue to predict and see the Preceision, recall, F1 and support score for further tuning.