Empirical Analysis Of ML Algorithms

Group-6, CS-771A Project

24 Nov 2018

Outline

What we did

- Five standard text datasets from TREC (Text REtrieval Conference[1]) have been taken.
- Bag of Words and Tf-Idf Feature extraction is used. Tf-Idf Scheme produces better results in most of the algorithms, hence shown here.
- Hyperparamters are tuned using `grid_search` and the best model is used.
- Among Deep Learning Algorithms, only Multi-Layer Perceptron has been tried.
- Evaluation Metric : Micro Averaged F1-score.

[1] https://trec.nist.gov/

Performance Comparison

Dataset	SVM	RF	Softmax	NB	*k*NN	MLP
TR11	0.90	0.89	**0.91**	0.66	0.87	0.89
TR12	0.85	0.84	**0.87**	0.55	0.84	0.84
TR23	0.87	**0.92**	0.90	0.63	0.85	0.90
TR41	**0.96**	0.95	**0.96**	0.85	0.91	0.94
TR45	**0.93**	**0.93**	**0.93**	0.71	0.87	0.91
20NewsGroup	0.73	0.63	0.73	0.73	0.59	85.9

Table: Micro-averaged F1-Scores of different SOTA classical ML Algorithms and MLP on some standard text datasets

Learnings and Next Steps

- As for Multi-Layer Perceptron, it is performing worse than every other classifier taken on every datasets.
- Overfitting on every dataset. Training error is 0 everytime.
- RNNs and LSTMs are yet to be tried.
- Also yet to record performances by *fasttext*.
- To be tested on datasets with even higher dimensionality.

What we did

Dataset description: Fraud Detection data
No. of non fraud examples: 300
No. of fraud examples: 30

Modifying the data-

Method	Prototype		Naive Bayes		Decision tree		kNN		Logistic		SVM	
Metric	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Original	0.98	0.60	0.97	0.60	0.96	0.77	0.93	0.59	0.98	0.79	0.95	0.36
Undersampling	0.98	0.59	0.97	0.67	0.90	0.87	0.95	0.83	0.96	0.83	0.92	0.72
Oversampling	0.98	0.59	0.97	0.67	0.98	0.77	0.94	0.85	0.98	0.85	0.98	0.58
k-means	0.98	0.60	0.95	0.70	0.86	0.90	0.93	0.82	0.95	0.90	0.98	0.53
Cluster os.	0.98	0.57	0.97	0.66	0.98	0.77	0.93	0.79	0.99	0.76	0.99	0.69
Smote	0.98	0.47	0.97	0.66	0.97	0.80	0.94	0.80	0.97	0.85	0.98	0.74

What we did

Reweighting examples-

Method	Decision Tree		Logistic		SVM	
Metric	Precision	Recall	Precision	Recall	Precision	Recall
Original	0.96	0.77	0.98	0.79	0.95	0.36
Reweighted	0.96	0.77	0.97	0.87	0.98	0.80

Specific algorithms for handling imbalances-

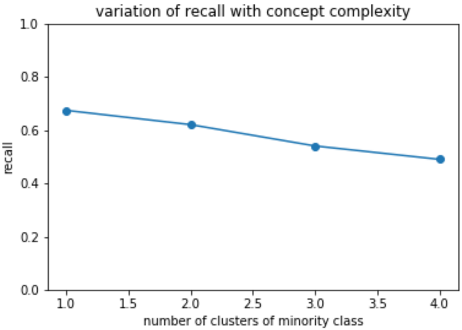
Method	Bagging		Random forest		EasyEnsemble		Gradient Boosting	
Metric	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Value	0.87	0.75	0.93	0.75	0.84	0.71	0.84	0.72

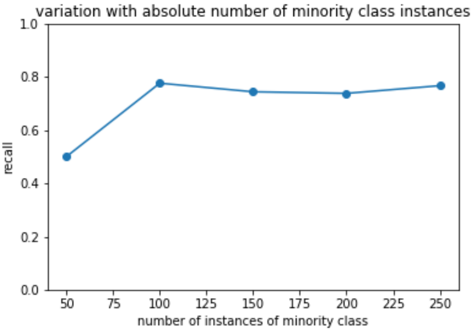
Learnings

- Improvements noted in Naive bayes, kNN, Logistic regression and SVM
- Decision tree and kNN classifies well separable imbalaced datasets well but here data is not well separable
- SVM showed a drastic improvement after reweighting
- Needed to balance test data as well to calculate precision
- Need to be careful while using distance based metrics in high dimensions
- Clustering based oversampling also helps overcome within-class imbalances

Effect of variation in data properties

- We generated 2D data sets - for visualizational ease - and varied specific properties to see how recall varies with them.
- Relative imbalance = 0.01; Classification algorithm: random forest.





What we did

1. Learn 'k' classifiers per cluster Vs Single Classifier
2. Will help to capture non-linearity in data
3. Classifiers experimented: DTree,SVM,Logistic Regression
4. Metrics: Accuracy,Precision F-Score.

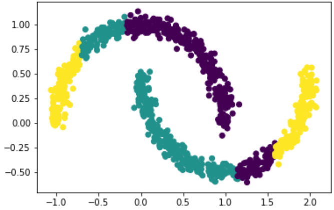


Figure: Non-Linear make moon dataset

Learnings

1. Prediction result is improving and able to handle non linear data.
2. Classification perform well in similar structured data generated by clustering.
3. Clustering will identify distinct distributions of the data generated from.

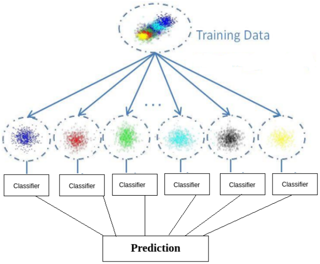


Figure: Algorithm Overview

Accuracy			
Clustering	DTree	Logistic Reg	SVM
No Cluster	0.8134	0.9107	0.84
K-Means	0.8454	0.9260	0.923
F1 Score			
Clustering	DTree	Logistic Reg	SVM
No Cluster	0.8052	0.9105	0.8316
K-Means	0.8435	0.9254	0.9277
Precision			
Clustering	DTree	Logistic Reg	SVM
No Cluster	0.8102	0.9123	0.8416
K-Means	0.8448	0.9237	0.9210

Table: MNIST Digit Recognizer

Kernel Approximation Methods

- Kernel methods widely used in non-linear learning: Computationally expensive !
- Kernel Approximation Methods : Make kernel methods scalable
  - For a given kernel $K(\cdot\,,\cdot) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ find a non-linear transformation $\phi(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^{'}$
  - Such that for any $x, y \in R^d$: $K(x, y) \approx \hat{K}(x, y) = \phi(x)^T \phi(y)$
  - Prominent Approaches: Random Fourier Features (RFF) , Orthogonal Random Features(ORF) proposed in NIPS-2016

RFF & ORF

- RFF
  - Shift-invariant kernels : $k(x, x') = \hat{k}(x - x') = \phi(x^T)\phi(x)$
  - $\phi(x) = \sqrt{1/D}[sin(w_1^T x), \cdots, sin(w_D^T x), cos(w_1^T x), \cdots, cos(w_D^T x),]$
  - w_i sampled i.i.d from PDF: $p(w)$, $W = [w_1, \cdots, w_D]^T$
  - **Wx** compute and store cost: $\mathcal{O}(Dd)$, typically $D > d$ for low approx. error
- ORF
  - Make $\mathbf{W}'s$ row orthogonal: reduces approx. error, $\mathcal{O}(d^3)$ time, $\mathcal{O}(d^2)$ space
  - **Structured ORF** : Special structured matrices, $\mathcal{O}(D\, log\, d)$ compute, in-place implementation

Experiments & Results

Table: MNIST DataSet

Details	Tr.Ti(s)	Pr.Ti/Ex.[μs]	Mean Accu.%
R-Dimen:256, W=128, K-SVM	135.72	3907.83	90.31
RFF	61.033	11.66	93.09
ORF	58.7	11.3	93.32
Normal,K-SVM	393.38	13351.69	75.27
W=128, RFF	63.31	8.33	92.25
W=128,ORF	62.9	8.39	92.01
W=512,W-Red=256,RFF	88.75	34.22	94.46
ORF	96.83	36.6	94.65
W=512,W-Red=128,RFF	60.7	26.21	92.11
W=512,W-Red=64,RFF	46.14	26.21	92.11

Learnings & Pending Tasks

- Kernel Garabage collector is giving better results
- Proofs not very clear to me, need to dig deeper
- Replicating numbers in paper is not trivial
- Still an active area of research despite DNN's
- Have to test on other datasets and different parameters

Thank You ! Questions!