Analysis of Resource-efficient Predictive Models for Natural Language Processing

Raj Ratn Pranesh
Birla Institute of Technology,
Mesra
raj.ratn18@gmail.com

Ambesh Shekhar
Birla Institute of Technology,
Mesra
ambesh.sinha@gmail.com

Abstract

In this paper, we presented an analyses of the resource efficient predictive models, namely Bonsai, Binary Neighbor Compression(BNC), ProtoNN, Random Forest, Naive Bayes and Support vector machine(SVM), in the machine learning field for resource constraint devices. These models try to minimize resource requirements like RAM and storage without hurting the accuracy much. We utilized these models on multiple benchmark natural language processing tasks, which were sentimental analysis, spam message detection, emotion analysis and fake news classification. The experiment results shows that the tree-based algorithm, Bonsai, surpassed the rest of the machine learning algorithms by achieve higher accuracy scores while having significantly lower memory usage.

1 Introduction

In last few years, large pretrained language models have gained a lot of popularity. These models were able to achieve state-of-the-art performance on various natural language processing tasks. But due to the higher resource requirement such as time and computation power, researchers have shifted their focus on developing more efficient and sustainable language models.

Devices like Arduino board, ATmega328 etc. which are essential in IoT infrastructures like health care, smart grids, wearables, etc, have very limited computational resources. Therefore, mostly these devices transfer data to cloud to extract some information and are dependent on them. These devices need models that can run without depending on cloud computing since cloud connectivity is not present everywhere, in a network data security can be compromised, it takes time to compute and transfer data which might not be good for real time analysis. So we need models that can run locally and need limited resources as well as do not hamper the accuracy of the task.

There have been many advances in this field like reducing the prediction cost of KNN with prototype based methods like ProtoNN (Gupta et al., 2017) and BNC(Binary Neighbour Compression) (Zhong et al., 2017) where you learn prototypes to reduce model size, SNC(Stochastic Neighbour compression) (Kusner et al., 2014) where you learn very small synthetic dataset to perform KNN, Tree based methods like bonsai tree where under constraints model learns non-linear decision rules at each node (Kumar et al., 2017), pruning the random forests based on resource constraints (Pal, 2005).

In this paper, we present an analysis of various resource efficient machine learning algorithm for performing NLP tasks, such as, sentiment and emotion classification, fake news and spam message detection. We used six models, namely, Bonsai, Binary Neighbor Compression(BNC), ProtoNN, Random Forest, Naive Bayes and Support vector machine(SVM) and reported their performance accuracy and memory usage for each task. We observed that the Bonsai model performed the best by achieving significantly higher accuracy scores than other models at the cost of minimum memory usage. We believe that our generated insights would be very useful in designing and developing IoT for NLP-based application purposes.

2 Methods

In this section, we discussed about the various models used in our analysis.

2.1 Naive Bayes

Naive Bayes (Rish et al., 2001) is based on supervised machine learning methods that uses the primitive or naive approach by applying Bayes' theorem between pair of features present in a text data point.

Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pages 136–140
Online, November 20, 2020.

Naive Bayes states the conditional probabilty between pair of words in a given sentence. Based on naive conditional independence assumption for x feature with y labels given, therefore for all i, this relationship is simplified to

$$P(y|x_1, ..., x_n) = \frac{P(y)\Pi_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

Since $P(y|x_1, ..., x_n)$ is the given input to the model, for classification process the predicted is

$$\hat{y} = \underset{y}{argmax} P(y)\Pi_{i=1}^{n} P(x_i|y)$$

Therefore for each text data point we have in our dataset, this machine learning algorithm calculates the conditional propbabilty for pairs of words and therefore based on the domain specific training it quantifies each data point to its respective classes.

2.2 Support Vector Machine

Support Vector Machine (Suykens and Vandewalle, 1999) is also a supervised learning methods majorly used in classification and outliers detection. Due to their performance with high dimensional space or higher features handling irrespective of the dimension of samples and available kernels functions for specified functions makes this model best in handling text data. If we pass a sparse matrix generated using TF-IDF function to a SVM classifier, it maximizes the decision boundary by minimizing $|||w||$ to find an optimal hyperplane for all the classification tasks:

$$minf : \frac{1}{2}||w||^2$$

$$\hat{y}^(i) = (w^t x^{(i)} + b) \geq 1, i = 1, ..., m \quad (1)$$

where w is the weight vector, for all i, x is input features matrix with b as the bias, with a resulting $\hat{y}$.

2.3 ProtoNN

ProtoNN (Gupta et al., 2017) is kNN based model that uses compressed model and prototypes for prediction. Prototypes are learned from the the data along with the estimation of projection matrix jointly, due to which it avoids pruning after the model is learnt to fit the model in desired memory. Prototypes are points that represent the entire data. The projection matrix is sparse matrix estimated by performing SGD and iterative hard-thresholding. Since number of prototypes are far less than number of inputs and number of features are less the model is comparatively small. ProtoNN gives nearly the same accuracy as most popular models that take a huge amount of RAM with very small amount of memory used , which makes it fit for our use in resource constrained IoT devices. ProtoNN tries to optimize the following loss function:

$$\mathcal{L}_i(Z, B, W) = \mathcal{L}(y_i, \sum_{j=1}^{m} z_j K_\gamma(b_j, W x_i)) \quad (2)$$

This is for each data point i. B is prototypes and Z is its corresponding score vector. W is low dimensional projection matrix. K_γ is RBF similarity kernel function used in the paper, any other kernel function can be used as well. The optimizing problem they obtained is non-convex but alternating optimization works in this case. Each of the parameters (B, Z, W) are learnt alternately using the algorithm provided with sparsity constraints.

2.4 BNC

Another simple model which is similar to the one described above is Binary Neighbor Compression (BNC) (Zhong et al., 2017). Here a KMeans clustering is performed on each class and number of clusters from each class are given by $k_y = \beta N_y$ where $\beta \in (0, 1)$ and N_y is the number of points that belong to class y. In this way we create a matrix of prototypes C of size m $\times$ d, where m is the total number of prototypes, and d is the dimension of the data. Then we initialize a matrix W of size d $\times$ r randomly, where r is the new dimension of data. Using this we convert the Prototypes to lower dimension representation and also binary form as: $B = sign(CW)$. Hence B will be of the size m $\times$ r. After this we learn B and W alternately, using the loss function below which is similar to multi-class hinge loss:

$$\underset{W,B}{min} \frac{1}{N} \sum_{i=1}^{N} [\alpha - \underset{j:z_j=y_i}{max} (tanh(\gamma W^T x_i)^T b_j) + \underset{j:z_j \neq y_i}{max} (tanh(\gamma W^T x_i)^T b_j)] + \lambda \sum_{k \in [r]} (||w_k||^2 - 1)^2$$

where tanh is used instead of sign function, as sign is not differentiable. So as $\gamma \rightarrow \infty, tanh(\gamma W^T x_i) \rightarrow (W^T x_i)$. Here $[x]_+ = max\{0, x\}$ and α is a hyper-parameter. Also a regularization is applied on W. Prediction is made by computing the similarity of projected test point

Dataset	#Train	#Test	#Features	#Classes
SMS-Span Collection	4218	1032	1226	2
Fake and Real News	1600	400	756	2
Sentiment-140	3200	800	1362	3
The Emotion in Text	3116	780	1542	14

Table 1: Dataset Statistics

with all the prototypes, and the label of the prototype which has highest similarity is assigned to the test point.

2.5 Bonsai Tree

Based on the paper (Kumar et al., 2017) unlike normal trees which learn axis aligned decision rules bonsai learns a non linear decision rule at every node. It first projects the data into low dimensional space(can be done in streamlined fashion), then projected features are traversed through the tree with each node scoring the output in their own way and sum of all scores is used as net score.

Scoring Function Bonsai learns a single, shallow sparse tree whose predictions for a point x is given by :

$$y(x) = \sum_k I_k(x) W_k^T Z x \odot tanh(\sigma V_k^T Z x) \quad (3)$$

where $\odot$ denotes the element wise Hadamard product, σ is a hyper-parameter, Z is a sparse projection matrix and $I_k(x)$ is an indicator function taking the value 1 if node k lies along the path traversed by x and 0 otherwise and W_k and V_k are sparse scoring vectors learnt at node k.

Branching Function Bonsai tree computes I_k by learning a sparse vector θ_k at each internal node such that the sign of $\theta_k^T Z x$ determines whether data object x should be branched to the left or right child. Optimizing the I_k is hard problem so it is relaxed as follows:

$$I_{k>1} = 0.5 \times I_j(x)(1 + (-1)^{k-2j} tanh(\sigma_I \theta_j^T Z x)) \quad (4)$$

where, j^{th} node is parent of k^{th} node.

Optimization Problem The optimization problem can be formulated with any empirical loss function (e.g categorical cross entropy loss), as follows :

$$\min_{\Theta} \{\mathcal{L}(y, x, \Theta) + \frac{\lambda_\theta}{2} Tr(\theta^T \theta) + \frac{\lambda_W}{2} Tr(W^T W) + \frac{\lambda_V}{2} Tr(V^T V) + \frac{\lambda_Z}{2} Tr(Z^T Z)\}$$

All the parameters are simultaneously optimized in alternating fashion. This model now can be trained using gradient descent approach, newton method etc. the original implementation used gradient descent with IHT(iterative hard thresholding) constraints over the parameters.

2.6 Random Forest

Random Forest is designed as an ensemble learning based classifier that combines different decision tree classifiers to perform class prediction (Injadat et al., 2016). The model is consists of multiple decision trees and the training of each of the decision tree is done using random subsets of features. In the Random Forest model, the final prediction is given through the majority voting of generated predictions from all the trees in the forest. As described by the author in (Malik et al., 2011), the Random Forest algorithm can be formulated as following:
(i) T number of trees are selected
(ii) For dividing each node m number of variables are selected, m<<M, where M represents total number of input variables.
(iii) Tree is populated by using the following methods:

- Given N training samples, a sample of size N is created while growing and replacing a tree from the produced sample.
- To obtain finest split, randomly choose m variable from m while populating each node in the tree.
- The tree is left for growing without any hindrance.

(iv) For the classification of node X, majority voting is utilized to predict the class label.

3 Dataset

We have used four textual datasets(see table 1) for different natural language processing classification task, namely, Sentiment140 dataset (Go et al., 2009) for sentiment analysis, the SMS Spam Collection dataset[1] for spam classification, the The Emotion in Text (Mohammad and Bravo-Marquez, 2017) dataset for emotion analysis and the Fake and Real news[2] dataset for fake new classification task. These datasets hold clean textual data with

[1]https://archive.ics.uci.edu/ml/datasets
[2]https://www.kaggle.com/c/nlp-getting-started

Datasets / Models	Sentiment-140	SMS-Span-Collection	Fake-News	The Emotion in Text
Naive-Bayes	59.66(2kB)	86.85(2kB)	78.16(2kB)	53.88(3kB)
SVM-Linear	59.83(137kB)	88.19(15.89kB)	89.66(18.37kB)	56.15(51.6kB)
SVM-poly	60.63(131kB)	88.64(64kB)	81.00(121kB)	55.52(144kB)
SVM-rbf	60.46(115kB)	88.64(18kB)	89.33(43.27kB)	57.22(125kB)
Random Forest	56.08(34kB)	88.19(1.61MB)	89.50(549kB)	60.74(812kB)
BNC	60.71(5.8kB)	90.71(3.9kB)	92.85(3.8kB)	62.41(5.0kB)
ProtoNN	62.45(3.5kB)	90.87(3.5kB)	94.17(3.5kB)	**68.97(3.5kB)**
Bonsai	**64.38(2kB)**	**94.91(2kB)**	**97.29(2kB)**	67.20(2kB)

Table 2: Models performance on datasets. For each model, accuracy(%) along with Memory usage is provided

their corresponding labels for supervised learning tasks.

Sentiment140: This contains 1.6M tweets text data extracted using twitter API. Each tweet has been annotated with labels neutral, negative, and positive to express their sentiment.

SMS Spam Collection: A collection of 5,574 English non-encoded messages annotated spam or ham(legitimate). The dataset contains 425 SMS manually extracted from the Grumbletext website, and a subset 3375 SMS randomly chosen legitimate messages of the NUS SMS corpus.

The Emotion in Text: It is a collection of 40,000 manually labelled tweets dataset for emotion detection and classification tasks. The dataset has 14 emotion categories.

Fake and Real news dataset: This contains 38,729 English news text data annotated with fake and true labels denoting whether they are fake or not. development in NLP tasks.

4 Experiment

We experimented with six machine learning models on four benchmark natural language dataset for different tasks. We used Tf-Idf as our text-features conversions, where we randomly selected the number of features to be considered while conversion. For each of the dataset, as seen in the figure 1, we used a specific number of features. We fixed the number of features so that we can evenly compare and evaluate the machine learning models. For SMS-Spam dataset, we used a subset consisting of 5120 instances of dataset with 1226 features. For Fake-News data, we used 2000 instances of dataset with 756 features. For Sentiment-140 dataset, we used 4000 instances of the total dataset with 1362 features. For The Emotion in Text dataset, we used 3896 instances of dataset with 1542 feature count. Each dataset was splitted into train and validation dataset with a ration of 80/20. The hyperparameter setting of all the models was done based on their best performance on the validation dataset. We used ADAM (Kingma and Ba, 2014) and Gradient Descent optimization for the models.

5 Result and Discussion

We have reported the model performance in the table 2. We can clearly see that the Bonsai model was able to outperform other model in majority of the tasks which makes it suitable for IoT based applications. Bonsai achieved an accuracy of 97.29, 64.38 and 68.97 over the classification task in fake-news, sentiment-140 and SMS-spam dataset with 2kB of memory usage. The ProtoNN model was able to beat Bonsai in the The Emotion in Text dataset task by gaining an improvement of 2.56% on the Bonsai model. Being said that, even with lesser accuracy, the Bonsai model performed the classification task in just 2kB memory while ProtoNN took 4.2kB. On an average, these two light weight models, ProtoNN and Bonsai Tree, outperformed other models on average by 4.83% and 7.06% respectively. Out of Naive-Bayes, SVM and Random Forest, with significantly higher memory consumption, Random Forest surpassed other models with better performance. On the other hand, Naive-Bayes's accuracy to memory conception ratio was higher than SVM and Random Forest. This suggest that the Naive-Bayes is capable of getting accuracy with lesser memory consumption.

6 Conclusion

In this paper, we presented a comparative analysis of various machine learning model for performing NLP tasks. We investigated the models performance based on accuracy achieved and memory required for performing a NLP task. We found that Bonsai model was able to surpass other models

with higher accuracy and lesser memory consumption. We also conclude that these models can be trained on a laptop and can be transferred to IoT devices satisfying resource requirements of model. Through our work, we aim at contributing towards the goal of sustainable NLP by developing more resource efficient NLP methods.

References

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. 2017. Protonn: Compressed and accurate knn for resource-scarce devices. In *International Conference on Machine Learning*, pages 1331–1340.

MohammadNoor Injadat, Fadi Salo, and Ali Bou Nassif. 2016. Data mining techniques in social media: A survey. *Neurocomputing*, 214:654–670.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ashish Kumar, Saurabh Goyal, and Manik Varma. 2017. Resource-efficient machine learning in 2 kb ram for the internet of things. In *International Conference on Machine Learning*, pages 1935–1944.

Matt Kusner, Stephen Tyree, Kilian Weinberger, and Kunal Agrawal. 2014. Stochastic neighbor compression. In *International Conference on Machine Learning*, pages 622–630.

Arif Jamal Malik, Waseem Shahzad, and Farrukh Aslam Khan. 2011. Binary pso and random forests algorithm for probe attacks detection in a network. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 662–668. IEEE.

Saif M Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. *arXiv preprint arXiv:1708.03696*.

Mahesh Pal. 2005. Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222.

Irina Rish et al. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Kai Zhong, Ruiqi Guo, Sanjiv Kumar, Bowei Yan, David Simcha, and Inderjit Dhillon. 2017. Fast classification with binary prototypes. In *Artificial Intelligence and Statistics*, pages 1255–1263.