A Novel Feature Selection and Extraction Technique for Classification

Kratarth Goel[1], Raunaq Vohra[2] and Ainesh Bakshi[3]

***Abstract*— This paper presents a versatile technique for the purpose of feature selection and extraction - Class Dependent Features (CDFs). We use CDFs to improve the accuracy of classification and at the same time control computational expense by tackling the curse of dimensionality. In order to demonstrate the generality of this technique, it is applied to handwritten digit recognition and text categorization.**

Keywords- *MNIST; USPS; WebKB; Reuters-21578*

I. Introduction

This paper proposes a novel and robust technique for feature selection and extraction which gives results comparable to the current state-of-the-art with the added advantage of being very fast and easy to implement on a range of devices. It suggests a better alternative to the Tf-Idf statistic, which can inadvertently decrease the statistic for words which occur frequently in a class label, and are innate to the entire class label, thereby resulting in poor features and lower accuracy for classification. The algorithm works by first selecting features relevant to their class label and extracts them accordingly. These extracted features are relevant to the entire class they are a part of, not just the individual data item they are extracted from. We call these features *Class Dependent Features* (CDFs). Moreover the entire learning problem is then broken down into smaller classification tasks by creating a SVM for each pair of class labels.

II. The Algorithm

A. Feature Selection

Consider a vector $P = \{P_1, P_2, ..., P_m\}$ representing the set of all class labels in the training dataset (m is the total number of class labels in the dataset). Each $P_c = \{p_1, p_2, ..., p_M\}, p_k \in \mathbb{R}^N, \forall k \in [1, M]$ further represents all data points in the c^{th} class label. Then the function $f(P_c) = \{a_{c_1}, a_{c_2}, ..., a_{c_N}\}$ representing the *summation function* for the c^{th} class label is given by

$$a_{ci} = \sum_{k=1}^{M} p_k(i) \, . \tag{1}$$

where N denotes the dimensionality of p_k, M is the cardinality of the c^{th} class label and $a_{ci} \in \mathbb{R}$. Here, for instance, the element $p_k(i)$ would be the pixel intensity value of the i^{th} pixel of the k^{th} image, belonging to the class label c (say, the digit 0) of the MNIST dataset. We then obtain the measure $T(P_c) = \{q_{c_1}, q_{c_2}, ...q_{c_N}\}$

$$q_{ci} = a_{ci}/M \tag{2}$$

$\forall i \in [1, N]$.

This measure *T* represents the probability distribution over the entire digit class.

Now that we have the measure *T* for each class label, we must establish a relation $\mathbf{R_{xy}}$ between each pair of class labels P_x and P_y. This is used to obtain the degree of relatedness between the two class labels P_x and P_y, which gives us the degree of similarity and dissimilarity between them. The experiments in this paper were conducted taking $\mathbf{R_{xy}}$ as

$$\mathbf{R_{xy}} = \{q_{xi}/q_{yi} \mid \forall q_{xi} \in T(P_x) \ and \ \forall q_{yi} \in T(P_y)\} \, . \tag{3}$$

We then take the mean of all the values of $q_{xi}/q_{yi} \in \mathbf{R_{xy}}$ as shown below.

$$\mu_{xy} = \frac{\sum_{i=1}^{N} (q_{xi}/q_{yi})}{N} \, . \tag{4}$$

We generate two thresholds τ and τ', given by

$$\begin{aligned} \tau &= b\mu_{xy} \, . \\ \tau' &= b'\mu_{yx} \, . \end{aligned} \tag{5}$$

where $b, b' \in \mathbb{R}$

Values of $q_{ci} \in T(P_c)$ greater than the thresholds τ or τ' will be selected as the feature locations for the c^{th} class label. Considering $T(P_c)$ as a vector of real values, only those indices *i.e.* i's in $T(P_c)$ for which q_{ci}'s have their values greater than either threshold are chosen as class dependent. Hence the values b and b' can be thought of as parameters controlling dimensionality of the input space for the given problem statement.

Thus we define a new dataset $P' = \{P'_1, P'_2, ..., P'_m\}$, where each P'_c is the set of modified data items in the c^{th} class label, given by

$$P'_c = \{p'_1, p'_2, ..., p'_M\} \, . \tag{6}$$

where each $p'_k, \forall k \in [1, M]$ is a probability distribution, formed as follows:

$$p'_k(i) = \begin{cases} p_k(i), & if \ q_{ci} > \tau \ or \ q_{ci} > \tau' \\ NULL, & otherwise \end{cases} \tag{7}$$

$\forall i \in [1, N]$.

[1] Kratarth Goel is with the Department of Computer Science, BITS Pilani KK Birla Goa Campus, Goa, India kratarthgoel@gmail.com

[2] Raunaq Vohra is with the Department of Mathematics, BITS Pilani KK Birla Goa Campus, Goa, India ronvohra@gmail.com

[3] Ainesh Bakshi is with the Department of Computer Science, Rutgers University, New Brunswick, NJ aineshbakshi@gmail.com

Thus we have effectively reduced the dimensionality of the dataset by keeping only the class dependent values of these features *i.e* the values of the features greater than the thresholds τ or τ'. This can be interpreted as the non-NULL values represented in equ. (7).

B. Feature Extraction

With our thresholds τ and τ' calculated, we can now proceed with the extraction of class dependent features (CDFs) for the pair of class labels x and y. In order to do so, we use the concept of the *Kullback-Leibler (KL) divergence*. The feature vector $\mathbf{F_{xy}}$ used for the purpose of classification between the pair of class labels x and y is calculated as follows:

$$\mathbf{F_{xy}}(k) = D_{KL}(p'_k \,||\, T(P_x)) \; . \tag{8}$$

Also a set of labels is created to be given to the classifier as follows:

$$\mathbf{L_{xy}}(k) = \begin{cases} 1 & p'_k \in P'_x \; . \\ -1 & p'_k \in P'_y \; . \end{cases} \tag{9}$$

We have now obtained our CDFs for each class label comparison. These features will be passed to the classifier for training on the dataset. This process is repeated for all pairs of class labels x and y to cover the entire dataset.

III. Experimental Results

In order to test our technique and prove its generality and versatility, we used it on two fundamentally different problem statements - handwritten digit recognition and text categorization. For the former, we used the well-known MNIST and USPS datasets and used the WebKB and Reuters-21578 datasets for the latter. We used SVMs as the classifier for both problem statements and the optimum parameters were determined by using n-fold cross-validation.

A. Handwritten Digit Recognition

Techniques	**Error**
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, L3	2.83
Products of boosted stumps	1.26
40 PCA + Quadratic classifier	3.3
SVM, Gaussian kernel	1.4
3-layer NN, 500+150 hidden units	2.95
2-layer NN, 800 HU, Cross-Entropy	1.53
Deep Belief Net	1.0
Large Convolutional Net (no distortions)	0.62
CDFs, SVM, 2-degree poly kernel	**1.25**

TABLE I: Comparison of various generic techniques with error rates (%) on the MNIST dataset.

As can be observed from Table I - which shows the error rates for various technique on the MNIST dataset - our technique produces the best reported error rate among generic classification algorithms (no preprocessing was carried out on the data) after Hinton and Salakhutdinov's (2007) deep belief nets (1.00%) and Deng and Yu's deep convex nets (0.83%). It outperforms Kegl and Busa-Fekete's products of boosting stumps (1.26%).

B. Text Categorization

The macro and micro averaged F measure on the Reuters-21578 dataset are tabulated in Table II. It is evident that CDFs outperform all other techniques and are a significant improvement upon them. The macro and micro F scores (89.28% and 96.32%) exceed Gini index, the second best, by 20% and 6.5% respectively.

These experiments were carried out on a 2$^{\text{nd}}$ Generation Intel Core i5-2410M processor running Ubuntu 13.04 and the code was implemented using OpenCV 2.4.6.1 [2]. The training and testing on the MNIST dataset takes 5.77 minutes. In comparison, while running the MNIST task using Stacked Denoising Autoencoders, pre-training takes 585.01 minutes. Fine-tuning is completed after 36 epochs in 444.2 minutes. The final testing score obtained is 1.3%, which our technique outperforms by 0.05%. These results were obtained using Theano 0.6rc3 [1] on a machine with an Intel Xeon E5430 @ 2.66GHz CPU, with a single-threaded GotoBLAS. This serves to illustrate the accuracy our technique brings with its computation time being a fraction of that taken by Stacked Denoising Autoencoders.

Feature Weight Function	**Macro F**	**Micro F**
TFxIDF	62.63	84.73
TFxGINI	69.82	89.79
TFxIG	63.88	84.64
TFxCROSS-ENTROPY	66.63	86.55
TFxX2	60.88	83.71
TFxMUTUAL-INFO	68.15	87.59
TFxODDS-RATIO	69.16	88.05
TFxWEIGHT OF EVID	64.24	85.22
CDF	**89.29**	**96.32**

TABLE II: Benchmarking CDFs with contemporary techniques on Reuters-21578 using SVM classifiers.

IV. Conclusions and Future Work

We have proposed a technique for feature selection and extraction using *Class Dependent Features* (CDFs). Our technique has been tested on the MNIST and USPS datasets for handwritten digit recognition as well as on the Reuters-21578 and WebKB datasets for text categorization. We have obtained strong competitive results using an SVM with a degree-2 polynomial kernel and these results along with those of contemporary techniques have been tabulated. Our results are comparable to the current state-of-the-art, and on par with the current best generic algorithms.

References

[1] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.