TRAINING ENSEMBLE OF DIVERSE CLASSIFIERS ON FEATURE SUBSETS

Rahul Gupta, Kartik Audhkhasi, and Shrikanth Narayanan

Signal Analysis and Interpretation Lab (SAIL), Department of Electrical Engineering
University of Southern California, Los Angeles, USA-90007

ABSTRACT

Ensembles of diverse classifiers often out-perform single classifiers as has been well-demonstrated across several applications. Existing training algorithms either learn a classifier ensemble on pre-defined feature sets or independently perform classifier training and feature selection. Neither of these schemes is optimal. We pose feature subset selection and training of diverse classifiers on selected subsets as a joint optimization problem. We propose a novel greedy algorithm to solve this problem. We sequentially learn an ensemble of classifiers where each subsequent classifier is encouraged to learn data instances misclassified by previous classifiers on a concurrently selected feature set. Our experiments on synthetic and real-world data sets show the effectiveness of our algorithm. We observe that ensembles trained by our algorithm performs better than both a single classifier and an ensemble of classifiers learnt on pre-defined feature sets. We also test our algorithm as a feature selector on a synthetic dataset to filter out irrelevant features.

Index Terms— Classifier ensemble, diversity, simulated annealing, loss function optimization

1. INTRODUCTION

Ensemble of multiple classifiers have outperformed single classifier systems in several applications (e.g. Netflix challenge [1], DARPA GALE [2], Intoxication sub-challenge of Interspeech 2011 [3]). Dietterich [4] noted that ensemble of classifiers can have potentially lower generalization error and can capture rather complex class boundaries using simple classifiers. We propose a novel algorithm to train an ensemble of classifiers, jointly performing class boundary optimization and feature selection for each classifier. Consider an ensemble of K classifiers $\{f_1, .., f_K\}$ to model a dataset $S = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_M, y_M)\}$ of size M from a probability distribution $\mathcal{P}$. The data-samples $\boldsymbol{x}_m$ lie in a D dimensional feature space such that $\boldsymbol{x}_m = \{x_{m1}, .., x_{mD}\}$. We aim to learn the target labels y_m using the ensemble, where y_m is drawn from a set of J classes $\{c_1, .., c_J\}$. Ensemble learning optimizes a set of loss functions $\{L_1, .., L_K\}$ over S, where L_k corresponds to $f_k (k = 1..K)$. The learning scheme optimizes the loss functions over a set of parameters $\{(\boldsymbol{\Theta}_1, \boldsymbol{\Lambda}_1), ..., (\boldsymbol{\Theta}_K, \boldsymbol{\Lambda}_K)\}$, where $\boldsymbol{\Theta}_k$ and $\boldsymbol{\Lambda}_k$ determine the class boundaries and features subset used by the k^{th} classifier, respectively. We represent $\boldsymbol{\Lambda}_k$ as a D-dimensional binary indicator vector $(\lambda_{k1}, \lambda_{k2}, ..., \lambda_{kD})$ where $\lambda_{kd} = 1$ indicates the use of d^{th} feature while training the k^{th} classifier and vice versa. Existing methods enforce diversity in the ensemble by using empirically-chosen diverse loss functions for each classifier or subjecting various constraints over $\boldsymbol{\Lambda}_k$ and/or $\boldsymbol{\Theta}_k$ while training f_k. We instead pose ensemble learning as a joint optimization problem over the parameters $\{(\boldsymbol{\Theta}_1, \boldsymbol{\Lambda}_1), ..., (\boldsymbol{\Theta}_K, \boldsymbol{\Lambda}_K)\}$ in (1).

$$\underset{\boldsymbol{\Lambda}_1,..,\boldsymbol{\Lambda}_K,\boldsymbol{\Theta}_1,..,\boldsymbol{\Theta}_K}{\text{minimize:}} \{L_1, .., L_K\} \tag{1}$$

As joint optimization of a set of functions is a multi-objective optimization over a large parameter space, it is not trivial. Hence current ensemble learning schemes impose several relaxations on the ensemble optimization problem. Existing techniques replace the joint optimization by a sequential optimization (2) or individual optimization (3) of classifiers. Typically in sequential optimization, L_k depends upon the performance of $\{f_1, .., f_{k-1}\}$, whereas in individual optimization, it is computed independent of other classifiers.

$$\begin{aligned}&\text{Sequential optimization of classifiers:}\\ &\text{For } k = 1 \text{ to } K\text{: } \underset{\boldsymbol{\Lambda}_k,\boldsymbol{\Theta}_k}{\textit{minimize:}}\ L_k\end{aligned} \tag{2}$$

$$\begin{aligned}&\text{Individual optimization of classifiers:}\\ &\underset{\boldsymbol{\Lambda}_k,\boldsymbol{\Theta}_k}{\text{minimize:}}\ L_k \quad k = \{1, .., K\}\end{aligned} \tag{3}$$

While problems stated in (2) and (3) cover a majority of ensemble learning schemes, existing techniques use several ad-hoc ways to determine the set of parameters $\{\boldsymbol{\Theta}_k, \boldsymbol{\Lambda}_k\}$. Bagging [5] performs individual optimization of classifiers solely on class boundary parameters $\boldsymbol{\Theta}_k$ after introducing diversity through data sub-sampling. This scheme optimizes over $\boldsymbol{\Theta}_k$ on the entire feature set by setting $\boldsymbol{\Lambda}_k$ to $\mathbf{1}_D = \{1, ..(D\text{-times}).., 1\}$. Boosting [6] performs sequential optimization over the parameters $\boldsymbol{\Theta}_k$ with $\boldsymbol{\Lambda}_k = \mathbf{1}_D$ after implementing diversity by setting different target values $y_m (m = 1..M)$ for each classifier. Similarly, Audhkhasi et al. [7] perform sequential optimization over $\boldsymbol{\Theta}_k$ with $\boldsymbol{\Lambda}_k = \mathbf{1}_D$ in an ensemble of maximum entropy models after introducing diversity measures based on Kullback-Leibler divergence and posterior cross correlation. The DECORATE algorithm [8] introduces diversity by optimizing $\boldsymbol{\Theta}_k$ using a mix of natural and artificial data, with all the features. Methods like multiview learning [9] partition the features into disjoint sets and perform individual optimization over $\{\boldsymbol{\Theta}_1, .., \boldsymbol{\Theta}_K\}$ on $\{\boldsymbol{\Lambda}_1, .., \boldsymbol{\Lambda}_K\}$ as determined by the partition. Random forests [10] samples $\boldsymbol{\Lambda}_k$ from a D-dimensional multinomial distribution and then individually optimizes $\boldsymbol{\Theta}_k$ on the sampled $\boldsymbol{\Lambda}_k$. All these schemes thus pre-determine $\{\boldsymbol{\Lambda}_1, .., \boldsymbol{\Lambda}_K\}$ and then evaluate $\{\boldsymbol{\Theta}_1, .., \boldsymbol{\Theta}_K\}$ using either the sequential or the individual optimization method, which is not optimal.

Several schemes in [11–13] address the above problem by sequentially optimizing the classifiers over $\boldsymbol{\Lambda}_k$ and $\boldsymbol{\Theta}_k$. These schemes sequentially optimize each L_k over $\boldsymbol{\Theta}_k$ on several candidate values of $\boldsymbol{\Lambda}_k$ and retain the feature subset with the best performance. However this is both suboptimal and computationally expensive since the entire optimization procedure is re-run for every candidate value of $\boldsymbol{\Lambda}_k$. We propose an algorithm that jointly optimizes L_k over the parameters $\boldsymbol{\Lambda}_k, \boldsymbol{\Theta}_k$ to address these issues. We fuse the decisions from classifiers using two fusion schemes and

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

compare the performance of our algorithm against a single classifier and an ensemble learnt by optimizing only $\mathbf{\Theta}_k$ with a preset $\mathbf{\Lambda}_k = \mathbf{1}_D$.

Apart from classification experiments, we also test our algorithm as a feature selector. Prominent wrapper feature selection schemes (such as forward feature selection) [14] are usually computationally expensive where as filter schemes (such as feature selection through mutual information) use a selection criterion which is usually different from the loss function. As our algorithm performs coupled feature selection and class boundary optimization, we avoid any extra computations as introduced by shuttling optimization between $\mathbf{\Lambda}_k$ and $\mathbf{\Theta}_k$ in the filter schemes. Furthermore, as our algorithm optimizes $\mathbf{\Lambda}_k$ over L_k, the selected features are directly relevant to the loss function. We present an application to our algorithm as a feature selector and present results on a synthetic dataset with dummy features.

2. ENSEMBLE CREATION AND FUSION

Training an ensemble can be broken down into training a diverse set of classifiers followed by fusing their decisions. We present a novel algorithm for training diverse classifiers in section 2.1 and describe the fusion strategies in section 2.2.

2.1. Training diverse classifiers

We first propose an algorithm to sequentially optimize each loss function L_k over $\{\mathbf{\Lambda}_k, \mathbf{\Theta}_k\}$, thereby jointly determining the feature subset and the class boundaries. We focus on designing an ensemble of discriminative probabilistic classifiers of the form shown in (4). Such classification models are popular across several applications [15, 16].

$$f_k(\boldsymbol{x}_m) = f_k(\boldsymbol{x}_m/\mathbf{\Lambda}_k, \mathbf{\Theta}_k) = \underset{j\in\{1,..,J\}}{\arg\max}\, p_k(c_j/\boldsymbol{x}_m, \mathbf{\Lambda}_k, \mathbf{\Theta}_k) \quad (4)$$

We estimates the posterior probability of each $c_j (j = 1..J)$ given $\boldsymbol{x}_m$ based on a soft-max function (5) and assign the class with the maximum probability. The class assignment parameter $\mathbf{\Theta}_k$ determines linear class boundaries based on J D-dimensional vectors $\boldsymbol{\theta}_{kj} = \{\theta^1_{kj}, .., \theta^D_{kj}\}, j = 1..J$.

$$p_k(c_j/\boldsymbol{x}_m) = p_k(c_j/\boldsymbol{x}_m, \mathbf{\Lambda}_k, \mathbf{\Theta}_k) = \frac{exp(a_{mkj})}{\sum_{j'=1}^{J} exp(a_{mkj'})} \quad (5)$$

$$\text{where } a_{mkj} = \boldsymbol{x}_m \times \text{diag}(\mathbf{\Lambda}_k) \times \boldsymbol{\theta}^T_{kj} \quad (6)$$

We define the loss function L_k as negative of weighted accuracy function A_k defined in (7).

$$A_k(\mathbf{\Lambda}_k, \mathbf{\Theta}_k, \boldsymbol{W}_k) = \frac{1}{M}\sum_{m=1}^{M} w_{km} \times \delta(f_k(\boldsymbol{x}_m) - y_m)$$
$$\text{where } \delta(f_k(\boldsymbol{x}_m) - y_m) = \begin{cases} 1 \text{ if } f_k(\boldsymbol{x}_m) = y_m \\ 0 \text{ otherwise} \end{cases} \quad (7)$$

We introduce diversity by assigning different weights to the data sample while learning each classifier. The sample weights $\boldsymbol{W}_k = \{w_{k1}, .., w_{kM}\}$ while training the k^{th} classifier are heuristically set as shown in equation (8). This scheme assigns a higher weight w_{km} to the instance $\boldsymbol{x}_m$, if the probability of the true class y_m in the previously trained classifiers is low. Hence the return is higher if the k^{th} classifier correctly classifies $\boldsymbol{x}_m$. This encourages each subsequent classifier to learn data instances misclassified by previously trained classifiers.

Line	Variable	Description
1	η	gradient ascent scaling constant
	thr	threshold for convergence
	p_λ	probability to flip any λ_{kd}
3	$\mathbf{\Lambda}_k$=$\mathbf{1}_D$	Initial training using all dimensions
	$\mathbf{\Theta}^i_k$	Random initialization for $\mathbf{\Theta}_k$
6	$\{U_1, .., U_D\}$;	D uniform random variables; $U_d \in [0, 1]$
	$\{u_1, .., u_D\}$	A realization of $\{U_1, .., U_D\}$
7	$\boldsymbol{U}_{bin}$	Binary variable where the d^{th} element
		$\boldsymbol{U}_{bin}(d) = 1$ if $u_d < p_\lambda$
8	$\mathbf{\Lambda}'_k$	An alternate dimension subset where
		$\lambda'_{kd} = NOT(\lambda_{kd})$ with probability p_λ
9,10	$\mathbf{\Theta}^{new}_k$, $\mathbf{\Theta}^{new'}_k$	$\mathbf{\Theta}_k$ after gradient ascent on
		$\mathbf{\Lambda}_m$, $\mathbf{\Lambda}'_m$ respectively
11,12	A^{old}, A^{new},	Weighted accuracies using: $(\mathbf{\Lambda}_k, \mathbf{\Theta}_k)$,
13	$A^{new'}$	$(\mathbf{\Lambda}_k, \mathbf{\Theta}^{new}_k)$, $(\mathbf{\Lambda}'_k, \mathbf{\Theta}^{new'}_k)$ respectively

Table 1. Intermediate variables in Algorithm 1 and their description.

Algorithm 1 Sequential optimization of weighted accuracy A_k over class boundary parameter $\mathbf{\Theta}_k$ and feature subset indicator vector $\mathbf{\Lambda}_k$

1: Define: $S = \langle(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_M, y_M)\rangle$, p_λ, η, thr
2: **for** $k = 1$ .. K **do** ▷ Sequential optimization
3: $\mathbf{\Theta}_k = \mathbf{\Theta}^i_k$; $\mathbf{\Lambda}_k = \mathbf{1}_D$
4: Evaluate w_{km} as in equation (8)
5: **while** 1 **do**
6: $\{u_1, ..., u_D\} \sim \{U_1, ..., U_D\}$
7: $\boldsymbol{U}_{bin} = \{(u_1 < p_\lambda), ..., (u_D < p_\lambda)\}$
8: $\mathbf{\Lambda}'_k = \text{XOR}(\mathbf{\Lambda}_k, U_{bin})$
9: $\mathbf{\Theta}^{new}_k = \mathbf{\Theta}_k + \eta \frac{\partial O_k(\mathbf{\Theta}_k, \mathbf{\Lambda}_k)}{\partial \mathbf{\Theta}_k}$
10: $\mathbf{\Theta}^{new'}_k = \mathbf{\Theta}_k + \eta \frac{\partial O_k(\mathbf{\Theta}_k, \mathbf{\Lambda}'_k)}{\partial \mathbf{\Theta}_k}$
11: $A^{old} \leftarrow A_k(\mathbf{\Lambda}_k, \mathbf{\Theta}_k, W_k)$
12: $A^{new} \leftarrow A_k(\mathbf{\Lambda}_k, \mathbf{\Theta}^{new}_k, W_k)$
13: $A^{new'} \leftarrow A_k(\mathbf{\Lambda}'_k, \mathbf{\Theta}^{new'}_k, W_k)$
14: $\Delta A = \max(A^{new}, A^{new'}) - A^{old}$
15: **if** $\Delta A < thr$ **then**
16: break
17: **else if** $A^{new} < A^{new'}$ **then**
18: $\mathbf{\Theta}_k = \mathbf{\Theta}^{new'}_k$; $\mathbf{\Lambda}_k = \mathbf{\Lambda}'_k$
19: **else**
20: $\mathbf{\Theta}_k = \mathbf{\Theta}^{new}_k$
21: **end if**
22: **end while**
23: **end for**

$$w_{km} = 1 - \left(\frac{1}{k-1}\sum_{k'=1}^{k-1} p_{k'}(y_m/\boldsymbol{x}_m)\right) \quad (8)$$

$$O_k(\mathbf{\Theta}_k, \mathbf{\Lambda}_k, \boldsymbol{W}_k) = \sum_{m=1}^{M} w_{km} \times \log(p_k(y_m/x_m)) \quad (9)$$

We state the optimization procedure in Algorithm 1. We evaluate the binary indicator vector $\mathbf{\Lambda}_k$ by performing simulated annealing [17] based binary integer programming to maximize A_k. As A_k is not differentiable, we evaluate $\mathbf{\Theta}_k$ by performing gradient ascent on the weighted data log-likelihood reward function O_k as a proxy for A_k (Note we maximize A_k, O_k as it is negative of loss function). We list the set of intermediate variables and their description in Table

1. The inner loop performs optimization on A_k, O_k over two candidate feature subsets $\mathbf{\Lambda}_k$ and $\mathbf{\Lambda}'_k$. The candidate feature subset $\mathbf{\Lambda}'_k$ is derived from $\mathbf{\Lambda}_k$ where we flip λ_{kd} with a flipping probability p_λ. We retain the dimension subset with a higher A_k after performing gradient ascent to determine $\mathbf{\Theta}_k$.

2.2. Fusing diverse classifiers

Given an ensemble of classifiers, we define a fusion function over the outputs of the classifiers $\{f_1(\boldsymbol{x}_m), .., f_K(\boldsymbol{x}_m)\}$ and an importance variable $\boldsymbol{I} = \{i_1(\boldsymbol{x}_m), .., i_K(\boldsymbol{x}_m)\}$ as shown in (10). We state two methods to determine $\boldsymbol{I}$ after (10).

$$g(\boldsymbol{x}_m) = \arg\max_{j \in \{1,..,J\}} \prod_{k=1}^{K} p_k(c_j/\boldsymbol{x}_m)^{i_k(\boldsymbol{x}_m)} \tag{10}$$

- **Oracle fusion function** (g_{orc}): This function uses a 1-in-K encoding scheme over the importance variable $\boldsymbol{I}$. Given there exists at least one classifier that correctly classifies $\boldsymbol{x}_m$, g_{orc} deterministically sets an $i_k(\boldsymbol{x}_m)$ to 1 if $f_k(\boldsymbol{x}_m) = y_m$. In case none of the classifiers output the correct class, this function assigns an incorrect class. Note that this scheme is not practically feasible as we are using the true labels in determining $\boldsymbol{I}$.
- **Equal weighting scheme** (g_{EW}): This is a proxy for above scheme where each $i_k(\boldsymbol{x}_m)$ is assigned a value $1/K$. g_{EW} assumes that each classifier in the ensemble in equally important.

In the next section, we test our ensemble learning algorithm and the fusion schemes on several datasets.

3. EXPERIMENTAL RESULTS AND DISCUSSION

We test our algorithm on a synthetic dataset followed by several real world datasets. We also present an application to our algorithm as a feature selector and test it on the synthetic dataset.

3.1. Synthetic dataset

We design a two-dimensional dataset with $\boldsymbol{x}_m = (x_{m1}, x_{m2}); (0 < x_{m1} < 1, 0 < x_{m2} < 1)$ and classes $\{c_1, c_2\}$. As our algorithm trains linear classifiers, we chose the class assignment boundary to be quadratic (11) to mimic a practical scenario where the true class boundary may not be perfectly modeled by the classifier. We conduct two sets of experiments on this synthetic dataset. The first experiment shows that the trained ensemble is a better classifier whereas the second experiment illustrates our algorithm as a feature selector.

$$y_i = \begin{cases} c_1 & \text{if } (x_{i1})^2 + (x_{i2})^2 < 1 \\ c_2 & \text{otherwise} \end{cases} \tag{11}$$

3.1.1. As a better classifier

We train the ensemble with an equal number of samples from both the classes, with varying dataset size. A large test set size of 50,000 data-samples per class ensures small sampling bias while calculating the classification accuracy. We choose a single classifier trained on all the features $\{K = 1; \mathbf{\Lambda}_1 = \mathbf{1}_D = \mathbf{1}_2\}$ as our baseline. We train two kinds of ensembles. In the first case, we solely optimize over $\mathbf{\Theta}_k$ and set $\mathbf{\Lambda}_k = \mathbf{1}_2$. This is analogous to several previously described ensemble learning schemes which fix $\mathbf{\Lambda}_k$ and optimize over $\mathbf{\Theta}_k$. This is achieved by setting the flipping probability $p_\lambda = 0$ in the algorithm. In the second case, we optimize over both set of parameters $\mathbf{\Lambda}_k, \mathbf{\Theta}_k$ and p_λ is empirically set to 0.01. We compare the accuracies obtained by these ensembles to study the effect of including $\mathbf{\Lambda}_k$ in optimization. K is tuned on the train set itself for maximum accuracy. η, thr are empirically set. Figure 1 reports mean relative improvements from models trained on 100 different datasets for each dataset size.

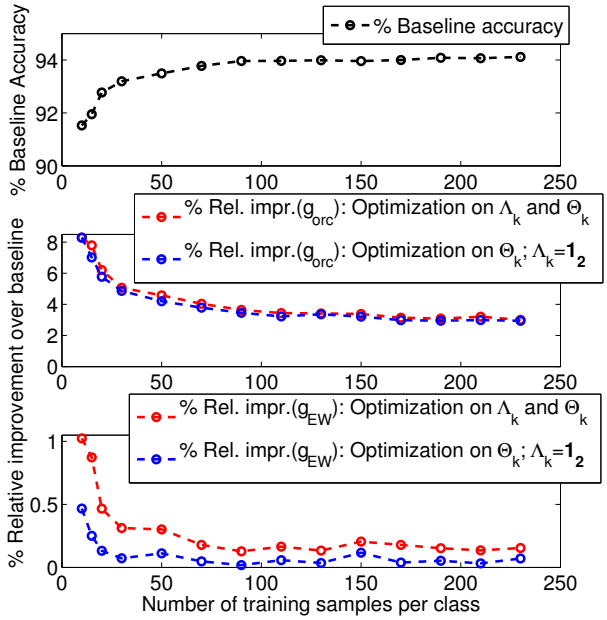

Fig. 1. Baseline classifier accuracy and relative improvements over baseline accuracy using the two fusion schemes g_{orc}, g_{EW} on the synthetic dataset.

We observe that while an ensemble of classifier beats the baseline classifier every time, the relative improvement in accuracy varies with the dataset size. g_{orc} performs substantially better than g_{EW}. The ensembles provide a higher relative improvement for a smaller dataset size for both g_{orc}, g_{EW}. This suggests that our algorithm works better particularly in the case of sparse distribution of datapoints. We also observe that optimization over both $\{\mathbf{\Lambda}_k, \mathbf{\Theta}_k\}$ performs better than optimization on just $\mathbf{\Theta}_k$ with a pre-assigned value to $\mathbf{\Lambda}_k$. This supports our case on optimization over both the parameters $\{\mathbf{\Lambda}_k, \mathbf{\Theta}_k\}$.

3.1.2. As a feature selector

We run our algorithm on an extended synthetic dataset obtained after appending 6 dummy features $(x_{m3}, .., x_{m8}); (0 < x_{md} < 1, d = 3..8)$ to (x_{m1}, x_{m2}) in the above synthetic dataset. The dummy features play no role in determining the class. We define a symmetric feature co-selection matrix F on $\mathbf{\Lambda}_k$ obtained after running the algorithm. The $(p,q)^{th}$ entry $F_{(p,q)}$ in the matrix is defined as shown in (12). $F_{(p,q)}$ for $p \neq q$ determines the proportion of times the p^{th} and the q^{th} features are used together in the K classifiers. $F_{(p,q)}$ for $p = q$ gives an individual feature's selection frequency. We binarize F based on two thresholds T_{cross}, T_{indiv} to obtain the most frequently selected pair of features and most frequently selected features. This matrix F^{bin} is shown in (13).

$$F_{(p,q)} = \frac{1}{K} \sum_{k=1}^{K} \lambda_{kp} \times \lambda_{kq} \tag{12}$$

$$F^{bin}_{(p,q)} = \begin{cases} F_{(p,q)} > T_{cross} & \text{if } p \neq q \\ F_{(p,q)} > T_{indiv} & \text{if } p = q \end{cases} \tag{13}$$

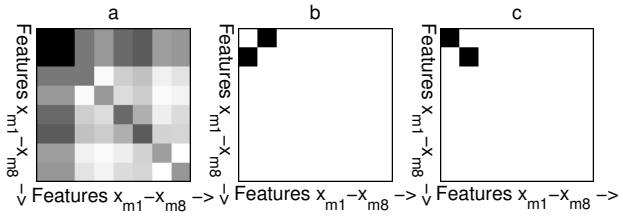

Fig. 2. (a) Feature co-selection matrix F (b) Binarized feature co-selection matrix F_{bin} depicting top off-diagonal element (c) F_{bin} depicting top two diagonal elements on extended synthetic dataset.

We run the algorithm on the extended synthetic dataset and show the matrix F as an image in Figure 2(a). Each element is represented as a block in the image and darker color implies a higher co-selection rate. We observe that the top left corner corresponding to x_{i1}, x_{i2} has a higher co-selection rate than rest of the matrix. We show F^{bin} separated into diagonal and off-diagonal portions in Figure 2(b) and 2(c) respectively. We tune T_{cross}, T_{indiv} to obtain top two diagonal entries and top off-diagonal entries shown as black blocks (F is symmetric so two blocks are shaded in 2(b)). We observe that binarization filters out x_{i1}, x_{i2} as top two individual features; at the same time choosing (x_{i1}, x_{i2}) as the most frequently selected feature pair.

These sets of experiments establish the efficacy of our algorithm in training better ensemble of classifiers as well as serving as a feature selector. Next, we evaluate our algorithm on a set of real-world datasets and demonstrate better classification accuracies.

3.2. Real-world datasets

We test our algorithm on three real world datasets listed in Table 2. We randomly split the data into training (80%), development (10%) and test (10%) sets. We set our baseline as a single classifier trained on all the features. We train two sets of ensembles. In the first ensemble training, we optimize on $\mathbf{\Theta}_k$ and set $\mathbf{\Lambda}_k = \mathbf{1}_D$ and in the second case we optimize on both the parameters, with $p_\lambda = .01$. The development set is used to tune K. We report the mean accuracies on the baseline classifier and the mean relative improvement over the baseline accuracy using the ensembles over 100 iterations of random splitting in Table 2.

Dataset	$D, J,$ M	Baseline accuracy	Relative Improvement			
			g_{orc}		g_{EW}	
			$\{\mathbf{\Theta}\}$	$\{\mathbf{\Theta}, \mathbf{\Lambda}\}$	$\{\mathbf{\Theta}\}$	$\{\mathbf{\Theta}, \mathbf{\Lambda}\}$
Abalone [18]	8,3, 4177	54.2	36.5	35.2	0.9	1.5
Image Segmentation [18]	19,7, 2310	91.3	3.9	5.4	1.0	1.4
IEMOCAP [19]	200,4, 5498	54.9	43.2	50.3	2.9	5.1

[1] $\mathbf{\Theta}$ indicates optimization over $\{\mathbf{\Theta}_k\}$ with $\mathbf{\Lambda}_k = \mathbf{1_D}$ ($p_\lambda = 0$), $\{\mathbf{\Theta}, \mathbf{\Lambda}\}$ indicates optimization over $\mathbf{\Theta}_k, \mathbf{\Lambda}_k$ ($p_\lambda = 0.01$)

Table 2. Mean accuracies on the test set over 100 iterations of cross validation on various datasets (D: Number of features, J: Number of classes, M: Number of data-samples in the dataset).

3.3. Discussion

We observe that the ensemble learning techniques consistently outperform the single classifier case. g_{orc} achieves substantial gains over the baseline result. We summarize this in Theorem 1 which proves that given an ensemble of classifier and the oracle fusion function g_{orc}, we will always achieve a lower generalization error than all classifiers in the ensemble. We define generalization error as the probability (Pr) of a classifier f assigning an incorrect class to an instance x drawn from the data distribution $\mathcal{P}$ in (14). y is the correct class corresponding to x.

$$err_{\mathcal{P}}(f) = \underset{(x\sim\mathcal{P})}{Pr}[f(x) \neq y] \tag{14}$$

Lemma 1. *Probability of intersection of events $E_1, .., E_K$ is less than probability of any single event $E_k; k = 1..K$ (***Frechet inequality** *[20]).*

$$Pr[E_1 \wedge E_2 \wedge .. \wedge E_K] \leq Pr[E_k]; \vee k = 1..K \tag{15}$$

Theorem 1. $err_{\mathcal{P}}(g_{orc}) \leq err_{\mathcal{P}}(f_k); \vee k = 1..K$

Proof. $err_{\mathcal{P}}(g_{orc}) = \underset{(x\sim\mathcal{P})}{Pr}[g_{orc}(x) \neq y]$

= $\underset{(x\sim\mathcal{P})}{Pr}[(f_1(x) \neq y) \wedge (f_2(x) \neq y) \wedge .. \wedge (f_K(x) \neq y)]$

(Since $g_{orc}(x)$ is incorrect if all the classifiers are incorrect.)

$\leq \underset{(x\sim\mathcal{P})}{Pr}[f_k(x) \neq y] = err_{\mathcal{P}}(f_k); \vee k = 1..K$ (from Lemma 1)

□

We observe that even though the proxy fusion scheme g_{EW} beats the baseline classifier it performs worse that g_{orc}. This indicates a severe shortcoming in our fusion proxy. We also observe that ensemble of classifiers optimized on both $\mathbf{\Theta}_k, \mathbf{\Lambda}_k$ outperforms those optimized over $\mathbf{\Theta}_k$ with $\mathbf{\Lambda}_k = \mathbf{1_D}$ in every case except for the oracle fusion for Abalone dataset. This supports our previous premise that joint optimization should provide us with a better ensemble over pre-assigning $\mathbf{\Lambda}_k$ and then optimizing over $\mathbf{\Theta}_k$. We also observe that our algorithm gives the best relative improvement in case of IEMOCAP dataset, where we have relatively smaller number of data-samples given the feature dimensionality. This is consistent with our observation on the synthetic dataset where we obtained a higher relative improvement when the data is sparsely distributed.

4. CONCLUSION

We present a novel algorithm for learning an ensemble of classifiers that performs joint optimization to determine class boundaries and the feature set for each classifier in the ensemble. We introduce a loss function that introduces data-driven diversity. We sequentially optimize the loss function for each classifier in the ensemble. We use an oracle fusion function and an equal weighting function to obtain the final decision from the ensemble. We present our results on several datasets and observe that not only does our algorithm trains better classifier ensembles, it can also filter out features unrelated to class assignments.

Even though we achieve better classification, there is a room for several improvements. In particular, one can suggest other fusion proxies to better approximate the oracle fusion scheme. We also aim at providing theoretical bounds for ensemble performance given the fusion function and the set of classifiers. As a further experimentation, one can also test the joint parameter optimization on other ensemble learning techniques with different loss functions. Further, we need better investigation over the performance of our algorithm and relate it to data sparsity and the number of features. One can also work on the feature selection scheme and suggest further improvements to the selection procedure.

5. ACKNOWLEDGMENT

This research is supported by NSF, NIH and DARPA.

6. REFERENCES

[1] James Bennett and Stan Lanning, "The netflix prize," in *Proceedings of KDD cup and workshop*, 2007, vol. 2007, p. 35.

[2] George Saon, Hagen Soltau, Upendra Chaudhari, Stephen Chu, Brian Kingsbury, Hong-Kwang Kuo, Lidia Mangu, and Daniel Povey, "The ibm 2008 gale arabic speech transcription system," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4378–4381.

[3] Daniel Bone, Ming Li, Matthew P Black, and Shrikanth S Narayanan, "Intoxicated speech detection: A fusion framework with speaker-normalized hierarchical functionals and gmm supervectors," *Computer Speech & Language*, 2012.

[4] Thomas G Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*, pp. 1–15. Springer, 2000.

[5] Leo Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[6] Jerome H Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[7] Kartik Audhkhasi, Abhinav Sethy, Bhuvana Ramabhadran, and Shrikanth S Narayanan, "Creating ensemble of diverse maximum entropy models," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4845–4848.

[8] Prem Melville and Raymond J Mooney, "Constructing diverse classifier ensembles using artificial training examples," in *IJCAI*. Citeseer, 2003, vol. 3, pp. 505–510.

[9] Ion Muslea, Steven Minton, and Craig A Knoblock, "Active+ semi-supervised learning= robust multi-view learning," in *ICML*, 2002, vol. 2, pp. 435–442.

[10] Leo Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] Gabriele Zenobi and Padraig Cunningham, "Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error," in *Machine Learning: ECML 2001*, pp. 576–587. Springer, 2001.

[12] Padraig Cunningham and John Carney, "Diversity versus quality in classification ensembles based on feature selection," in *Machine Learning: ECML 2000*, pp. 109–116. Springer, 2000.

[13] César Guerra-Salcedo and Darrell Whitley, "Feature selection mechanisms for ensemble creation: A genetic search perspective," in *Research Directions–Papers from the AAAI Workshop*, 1999.

[14] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[15] Adwait Ratnaparkhi et al., "A maximum entropy model for part-of-speech tagging," in *Proceedings of the conference on empirical methods in natural language processing*, 1996, vol. 1, pp. 133–142.

[16] Christopher Manning and Dan Klein, "Optimization, maxent models, and conditional estimation without magic," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials-Volume 5*. Association for Computational Linguistics, 2003, pp. 8–8.

[17] Peter JM Van Laarhoven and Emile HL Aarts, *Simulated annealing*, Springer, 1987.

[18] Kevin Bache and Moshe Lichman, "UCI machine learning repository," 2013.

[19] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[20] Ludger Rüschendorf, *Fréchet-bounds and their applications*, Springer, 1991.