A Directed Inference Approach towards Multi-class Multi-model Fusion

Tianbao Yang, Lei Wu, and Piero P. Bonissone

GE Global Research Center
{tyang,wul,bonissone}@ge.com

Abstract. In this paper, we propose a directed inference approach for multi-class multi-model fusion. Different from traditional approaches that learn a model in training stage and apply the model to new data points in testing stage, directed inference approach constructs (one) general direction of inference in training stage, and constructs an individual (ad-hoc) rule for each given test point in testing stage. In the present work, we propose a framework for applying the directed inference approach to multiple model fusion problems that consists of three components: (i) learning of individual models on the training samples, (ii) nearest neighbour search for constructing individual rules of bias correction, and (iii) learning of an optimal combination weights of individual models for model fusion. For inference on a test sample, the prediction scores of individual models are first corrected with bias estimated from the nearest training data points, and then the corrected scores are combined using the learned optimal weights. We conduct extensive experiments and demonstrate the effectiveness of the proposed approach towards multi-class multiple model fusion.

1 Introduction

Big data has posed great challenges in applying machine learning technologies. First, the scale of the data is too big to feed into most single-node and batch-mode machine learning algorithms. Second, the model trained on a small subset of data usually subjects to high bias and high variance.

To meet the big data challenge, a common approach is to adopt a distributed learning framework, where data and learning are distributed to different nodes in a cloud based computational network. These computational nodes are usually categorized into two types: one master node and a set of slave nodes. Each slave node will train an independent model on a subset of training data with single-node solvable scale, and make temporary decisions based on each independent model. The master node will take charge of distributing data, collecting information from slave nodes, and making the final decision, also called *model fusion*.

There are two steps involved in distributed learning framework. The first step is the distributing of multiple models in different slave nodes. The simplest way is to train each model, e.g., support vector machine (SVM)[8], neural network

Z.-H. Zhou, F. Roli, and J. Kittler (Eds.): MCS 2013, LNCS 7872, pp. 352–363, 2013.

(NN) [13], decision tree [11], logistic regression (LR) [15], etc., independently on each node. The second step is to fuse multiple models and make the final decision, which has become a bottleneck problem in the distributed learning framework. There are several ways for model fusion. The simplest approach that combines the scores of multiple models with equal weights suffers from severe problems. First, each model may have substantially biased prediction and as a result adding them together may blow up the prediction bias on a test sample[1]. Second, each individual model may perform very differently since different models are learned based on different assumptions and objective functions, as a result the simple average would be very vulnerable to poorly performed models.

Although some other methods have been considering different weighting schemes to fuse multiple models, e.g., bagging, boosting, maximum margin of ensembles [10], and etc, they are studied in the traditional system on a single machine over all training samples and therefore they may not fit into the modern distributed system.

In this work, we seek an approach to directly combine multiple models with each trained on the same set (or different subset) of training samples. The proposed directed inference approach consists of three key components: (i) learning of individual models, which is same as traditional approach; (ii) nearest neighbour search for estimating the prediction bias on a test sample to correct the prediction scores of individual models; and (iii) learning of an optimal combination weights for model fusion. To make an inference on a test sample, the raw prediction scores are first computed for each model and then are corrected with estimated bias from the nearest neighbours retrieved using a distance metric and finally are added together using the learned optimal combination weights. The proposed approach can be also understood from the viewpoint of bias-variance trade-off. Combination of multiple models has shown to be effective in reducing the variance of prediction, however it could have adversary effect by increasing the bias. Therefore, the bias correction step in the proposed method helps to reduce the bias in individual models and the optimal weighting scheme further alleviate the impact of models with large bias.

We organize the remaining part of the paper as follows. In section 2, we review some related work from three angles, directed inference, bias and variance trade-off, and model fusion. In section 3 we present the proposed approach with three key components: learning of individual models, learning of a distance metric and learning of an optimal combination weights. In section 4, we present the experimental results and finally we conclude in section 5.

2 Related Work

2.1 Directed Inference

Directed (ad-hoc) inference (DAHI) approach is a new machine learning technique proposed by Vladimir Vapnik [18]. The key difference between DAHI and

[1] Throughout the paper, we use the terms of sample, example, instance and data point interchangeability.

traditional inductive/deductive or transductive learning is that in the testing stage, DAHI constructs a specific individual rule for each test example based on a principle concept learned in the training stage. The present work fits into the framework of DAHI by first learning multiple individual models in a single machine or in a distributed learning framework, a distance metric for retrieving a nearest neighbour and an optimal combination weights, then for each test sample by computing a bias corrected score for each individual model and then combing the multiple scores using the learned weights.

2.2 Bias and Variance Trade-Off

Bias and variance take-off is a common problem in model selection and model assessment. It has been shown that the mean square error of an estimator can be decomposed into a sum of the variance and the bias square of the estimator. Given multiple unbiased estimators, by simply averaging their prediction scores, one can obtain an estimator with dramatically reduced variance. However, if the individual models are biased, the trade-off between bias and variance may kick in, i.e. the variance of combined models may be reduced, while the bias may be blown up. One of the key motivations of the proposed approach is to reduce the bias of individual models. Given the bias and variance trade-off, it is however generally a difficult and even impossible task to construct a fixed estimator with both small bias and variance. Therefore, we resort to DAHI to construct individual rules with small bias and combine them to obtain a small variance.

Bias correction has been introduced to construct individual rules with small bias [4,2] and has shown to be effective in regression [4] and binary classification [2]. Bias correction works by subtracting an estimated bias value from the prediction score on any test example. The bias on a test sample is estimated by taking average of the bias values on training data points in the nearest neighbourhood. The underlying assumption is that in the small neighbourhood of a test example, the bias value is a constant. Previous works have used Euclidean distance or rectangle distance to retrieve a number of nearest neighbours. However, the Euclidean nearest neighbour may not share similar bias as the models may learned in a different space (e.g., kernel SVM is learned in a mapped high dimensional or infinite dimensional space).

2.3 Model Fusion

Model fusion is part of the ensemble learning process, by which multiple intelligent models are trained and combined for making a decision. Fusion is a major scheme for improving the performance by generating a more robust decision boundary based on multiple decision models. It can also be considered as a generalized model selection process, where instead of selecting the best model, fusion selects the best combination of models. The commonly used fusion methods include simple fusion, majority voting, Borda count, threshold voting, and heuristic decision rules [14,25,12], weighted average [27], fuzzy integral, fuzzy

templates, and Dempster-Shafer theory[20], dynamic model selection [22], neural network (NN) based NN combination [21], local fusion [26], fuzzy combination [3], bagging [6], boosting [16], and etc.

3 A Directed Inference Approach towards Multi-class Multi-model Fusion

In this section, we present a directed inference approach towards multi-class multi-model fusion. The proposed approach consists of three key components: (i) learning of individual models, (ii) nearest neighbour search for bias correction, and (iii) learning optimal combination weights.

3.1 Learning of Individual Models

Our goal is to classify a data point into one of the K classes, denoted by $\{C_1, \ldots, C_K\}$. A common approach for multi-class classification is to cast the problem into several binary classification problems, e.g., one vs all or one vs one. In what follows, we briefly describe several methods for multi-class classification. Throughout the paper, we let $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$ denote the feature vectors and $y_i \in \{1, \ldots, K\}, i = 1, \ldots, n$ denote their class labels. Without incurring confusion, we also use $y_i \in \{0, 1\}^K$ to denote a K-dimensional vector with only one entry equal to 1 indicating the class label.

Support Vector Machine (SVM) [8] constructs a hyperplane in the linear form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + c$ by maximizing the margin from the hyperplane to the nearest training data point. It categorizes any data point into one of the two classes by checking the sign of the prediction score $\mathbf{w}^\top \mathbf{x} + c$. In addition to linear classification, SVM can also perform non-linear classification by using the kernel trick, which is equivalent to mapping data points into high dimensional or infinite dimensional feature spaces. In the experiments, we choose LibSVM [7] to run kernel SVM with RBF kernel. To perform multi-class classification, it follows one vs one scheme by constructing $K(K-1)/2$ binary classifiers and finally outputs a vector of scores that sum up to one, with each element indicating the confidence of assigning the data point into the corresponding class.

Neural Network (NN) [13] models the relationship between input and output in a structured information processing network, consisting of hidden layers of nodes between input and output. The learning process is actually adapting the model to the training data by changing the structure of the network. To adopt the NN for muti-class classification, we build K feed-forward neural networks with a hidden layer of 25 neurons. The k-th neural network NN_k is trained by regressing the input features $\mathbf{x}_i$ to the indicator variable $I(y_i = k)$ on the training data. The decision on a test point is made by $C(\mathbf{x}) = \arg\max_k \mathrm{NN}_k(\mathbf{x})$, where $\mathrm{NN}_k(\mathbf{x})$ gives the prediction value on $\mathbf{x}$.

Decision Tree [5] is a widely used non-linear model for both regression or classification. A decision tree could be either a classification tree or a regression tree depending on the type of the target variable and it is built upon the training

data by recursively splitting the feature space with one feature and a splitting criterion that minimizes error in the two resulting sub-spaces. To classify a data point into one of K classes, we construct K regression trees with each tree T_k built on the training data $\mathbf{x}_i, i = 1, \ldots, n$ with binary indicator variables $I(y_l = k)$, and predict the class of a test point by $C(\mathbf{x}) = \arg\max_k T_k(\mathbf{x})$. In this work, we choose the most well-known implementation of decision tree, CART [5].

Logistic Regression (LR) [15] is a discriminative model for classification. We consider linear logistic regression model for multi-class classification, which defines the class conditional probability by $\Pr(y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{l=1}^K \exp(\mathbf{w}_l^\top \mathbf{x})}$ and learns the K weights $\mathbf{w}_1, \ldots, \mathbf{w}_k$ by maximizing the log-likelihood on the training data. To avoid over-fitting, a regularization term $(\lambda/2) \sum_{k=1}^K \|\mathbf{w}_k\|_2^2$ is added to the objective.

3.2 Nearest Neighbour Search

Given multiple models denoted by $f_1, \ldots, f_m$ learned from the training data, the remaining question is to combine them into a single model for achieving a better performance. In this and next section, we address the question by nearest neighbour search using a distance metric for bias correction and learning an optimal combination weights for model fusion.

The raw prediction scores of model f_j on a given test example X are generated by $f_j(\mathrm{X}) \in \mathbb{R}^K$. The motivation of bias correction is to reduce the bias of individual models in predicting test data points. If we can accurately estimate the bias $b_j(\mathrm{X}) = f_j(\mathrm{X}) - \mathrm{Y}$, where Y is the unknown class label of the given example X, we can subtract the estimated bias $\widehat{b}_j(\mathrm{X})$ from the raw prediction scores $f_j(\mathrm{X})$ and obtain a more accurate classification decision based on $f_j(\mathrm{X}) - \widehat{b}_j(\mathrm{X})$. The question reduced to accurately estimation of the bias $\widehat{b}_j(\mathbf{X})$ for a given test point X. We take a non-parametric method, i.e., nearest neighbour estimation. A non-parametric method fits into the framework of directed inference [19], which is useful for constructing individual rules for test examples.

Let $\mathcal{N}(\mathrm{X})$ denote a small neighbourhood of X that contains the nearest training data points, which we assume shares the similar bias as the test data point X, then the bias of X can be estimated by

$$\widehat{b}_j(\mathrm{X}) = \frac{1}{|\mathcal{N}(\mathrm{X})|} \sum_{\mathbf{x}_i \in \mathcal{N}(\mathrm{X})} (f_j(\mathbf{x}_i) - y_i) \tag{1}$$

It still remains a problem how to retrieve a nearest neighbourhood of the test point X. A simple method is to define a nearest neighbourhood by using the Euclidean distance metric $\|\mathbf{x}_i - \mathrm{X}\|_2$. However, in some cases it may not reflect the underlying manifold of the bias function $b_j(\mathbf{x}) = f_j(\mathbf{x}) - y$, which depends on the model prediction $f(\mathbf{x})$ and the ground-truth y. A simple example that provides a negative evidence of using the Euclidean distance is given in Figure 1, where for the green test point, the bias of the nearest training data points (in the

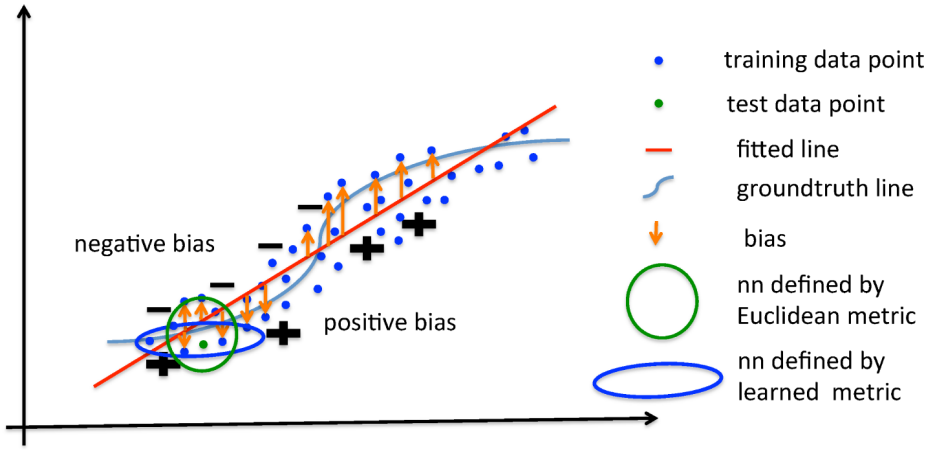

Fig. 1. An Illustration of nearest neighbourhood (nn) defined by the Euclidean metric and the learned metric for estimating the bias on a test point (green dot)

green circle) defined by Euclidean distance metric are mixed with positive values and negative values. As a consequence, by averaging the biases of the nearest training data points may yield a poor estimation of bias on the test data point. In contrast, if we define a nearest neighbourhood by a distance metric (e.g. the blue elliptical circle) that is consistent with the ground truth, i.e. data points with the same class labels have small distances and data points with different labels have large distances, then the estimation of bias can be improved. There exist many methods to formulate the distance metric learning [23,28,24]. In our empirical study, we choose a simple and effective method, relevant component analysis (RCA) [17,1], which is briefly described below.

RCA is originally proposed for learning a distance metric from partially labelled similar data points. Let $C_1, \ldots, C_K$ denote a set of K chunklets, where a chunklet is defined as a set of data points that share the same class labels. In our settings, each chunklet corresponds to one class. Then a positive semidefinite distance metric $A \in \mathbb{R}^{d \times d}$ is learned by minimizing the within class distances, i.e.,

$$\min_{A \in \mathbb{S}_+^{d \times d}} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{y_i = k} (\mathbf{x}_i - \mathbf{c}_k)^\top A (\mathbf{x}_i - \mathbf{c}_k) - \log\det A \tag{2}$$

where $\mathbb{S}_+^{m \times m} \subseteq \mathbb{R}^{m \times m}$ denotes a PSD cone, $\mathbf{c}_k$ is the center of the kth chunklet and n_k is the number of data points in C_k. The negative log-determinant term is added to avoid a trivial solution, which also has an information theoretic and Bayesian interpretation [1]. Finally, one can easily show that the optimal solution to (2) can be computed as

$$A = \left(\sum_{k=1}^{K} \frac{1}{n_k} \sum_{y_i=k} (\mathbf{x}_i - \mathbf{c}_k)(\mathbf{x}_i - \mathbf{c}_k)^\top \right)^{-1}$$

Equipped with a distance metric A, either the Euclidean metric or the learned metric, we can retrieve k nearest neighbors of the test sample X with k shortest distance $(\mathbf{x}_i - \mathrm{X})^\top A(\mathbf{x}_i - \mathrm{X})$ to form $\mathcal{N}(\mathrm{X})$.

3.3 Learning of an Optimal Combination Weights

In the previous section, we describe a nearest neighbour search for estimating the bias on a given test point X. Given the estimated bias, the prediction of each model is corrected by $f_j(\mathrm{X}) - \widehat{b}_j(\mathrm{X})$, and the corrected score will be combined by a weighted summation. In this section, we present a convex approach for learning a globally optimal combination weights. Let $\omega_1, \ldots, \omega_m$ denote the weights to be learned, the combined prediction is computed by

$$\widehat{f}(\mathrm{X}) = \sum_{j=1}^{m} \omega_j \left(f_j(\mathrm{X}) - \widehat{b}_j(\mathrm{X}) \right) \tag{3}$$

The combination weights are global in the sense that all test points share the same weights. The optimal combination weights $\omega = (\omega_1, \ldots, \omega_m)^\top$ are learned following the spirit of cross-validation. To this end, we let $(\mathbf{x}_i^v, y_i^v), i = 1, \ldots, N$ denote a separate set of N validation data points sampled from the same distribution of the training data points, and then we optimize the following objective

$$\min_{\omega \in \Delta_+} \sum_{i=1}^{N} \ell \left(\sum_{j=1}^{m} \omega_j (f_j(\mathbf{x}_i^v) - \widehat{b}_j(\mathbf{x}_i^v)), y_i^v \right) \tag{4}$$

where $\Delta_+ = \{\omega : \omega \geq 0, \sum_{j=1}^{m} \omega_j = 1\}$ is a simplex, $\widehat{b}_j(\mathbf{x}_i^v)$ is the estimated bias from the nearest neighbors and $\ell(\mathbf{z}, y)$ is a hinge loss for multi-class defined as

$$\ell(\mathbf{z}, y) = \max_{k \neq y} ([\mathbf{z}]_k - [\mathbf{z}]_y + b)_+$$

where b is a specified margin parameter and $[s]_+ = \max(0, s)$

To optimize the objective in (4), we can employ the widely adopted gradient descent method that iteratively updates $\omega_t = \omega_{t-1} - \eta \nabla L(\omega_{t-1})$, where η is a step size. However, the standard gradient decent method suffers from a low convergence rate of $O(1/\sqrt{T})$ for the non-smooth hinge loss function, i.e., $L(\widehat{\omega}_T) \leq \min_{\omega \in \Delta_+} L(\omega) + O(1/\sqrt{T}))$, where $\widehat{\omega}_T = \sum_{t=1}^{T} \omega_t / T$. In this paper, we extend the primal dual prox method proposed in [29] to optimize $L(\omega)$ that enjoys a convergence rate of $O(1/T)$. To this end, we write the objective in (4) into a min-max formulation:

$$\min_{\omega \in \Delta_+} \max_{\alpha \in \Omega_+^N} \underbrace{\frac{1}{N} \sum_{i=1}^{N} \sum_{k \neq y} \alpha_k^i \left([\widehat{f}(\mathbf{x}_i^v)]_k - [\widehat{f}(\mathbf{x}_i^v)]_{y_i^v} + b \right)}_{F(\omega, \alpha)}$$

Algorithm 1. Pdprox algorithm for optimizing *s*tructured *h*inge loss over a *s*implex (Pdprox-shs)

1: **Input**: step size γ
2: **Initialization**: $\theta_0 = \mathbf{1}/m, \alpha_0 = \mathbf{0}$
3: **for** $t = 1, 2, \ldots$ **do**
4: $\quad \omega_t = P_{\theta_{t-1}}(\gamma \nabla_\omega(\theta_{t-1}, \alpha_{t-1})) = \dfrac{\theta_{t-1} \circ \exp(-\gamma \nabla_\omega(\theta_{t-1}, \alpha_{t-1}))}{\sum_{j=1}^m [\theta_{t-1} \circ \exp(-\gamma \nabla_\omega(\theta_{t-1}, \alpha_{t-1}))]_j}$
5: $\quad \alpha_t = \mathit{\Pi}_{\Omega_+^N} [\alpha_{t-1} + \gamma \nabla_\alpha(\omega_t, \alpha_{t-1})]$
6: $\quad \theta_t = P_{\theta_{t-1}}(\gamma \nabla_\omega(\omega_t, \alpha_t)) = \dfrac{\theta_{t-1} \circ \exp(-\gamma \nabla_\omega(\omega_t, \alpha_t))}{\sum_{j=1}^m [\theta_{t-1} \circ \exp(-\gamma \nabla_\omega(\omega_t, \alpha_t))]_j}$
7: **end for**
8: **Output** $\widehat{\omega}_T = \sum_{t=1}^T \omega_t / T$ and $\widehat{\alpha}_T = \sum_{t=1}^T \alpha_t / T$.

by observing that $\ell(\mathbf{z}, y) = \max_{\alpha \in \Omega_+} \sum_{k \neq y} \alpha_k ([\mathbf{z}]_k - [\mathbf{z}]_y + b)$, where $\Omega_+ = \{\alpha \in \mathbb{R}^{K-1} : \alpha \geq 0, \sum_k \alpha_k \leq 1\}$. To present the algorithm, we let $\nabla_\alpha(\omega, \alpha)$ denote the partial gradient of $F(\omega, \alpha)$ in terms of ω, $\nabla_\alpha(\omega, \alpha)$ denote the partial gradient of $F(\omega, \alpha)$ in terms of α, and let $[\mathbf{u}]_j$ denote the jth element in $\mathbf{u}$. The detailed steps for updating the primal variable ω and the dual variables α are presented in Algorithm 1, which is a variant of Algorithm 2 proposed in [29]. The updating rule for the primal variable ω and the auxiliary primal variable θ is due to a proximal mapping $P_\theta(g) = \arg\min_{\omega \in \Delta_+} g^\top(\omega - \theta) + V(\omega, \theta)$, where $V(\omega, \theta) = \sum_j \omega_j \log(\omega_j / \theta_j)$ is the entropy distance function. The updating rule for the dual variables α is due to a projection $\mathit{\Pi}_{\alpha \in \Omega_+^N}[\widehat{\alpha}] = \arg\min_{\alpha \in \Omega_+^N} \|\alpha - \widehat{\alpha}\|_F^2$, which can be efficiently computed using the algorithm in [9]. Finally, we present the following theorem that states the convergence rate of Algorithm 1 for optimizing the structured hinge loss over a simplex. The proof can be easily duplicated following the analysis in [29].

Theorem 1. *Assuming* $\|[\widehat{f}(\mathbf{x})]\|_\infty \leq R$ *and setting* $\gamma = \sqrt{\frac{N}{8mR^2}}$, *by running Algorithm 1 with* T *steps, we have*

$$L(\widehat{\omega}_T) \leq \min_{\omega \in \Delta_+} L(\omega) + \frac{\log m + N}{2\gamma T}$$

4 Experiments

In this section, we present some preliminary experimental results. The data sets we choose for study include open benchmarks, *DNA, letter, pendigits, protein, satimage,* in **UCI** data repositories. We also adopted a jet engine fault classification data, which contains a total of 19,635 instances. Each instance corresponding to a case of engine has 11 attributes from sensors and also is labelled to one of seven classes which indicates one of the seven fault types including normal. We refer to the data as aircraft engine fault diagnosis (AEFD) data.

More details can be found in [27]. The data is split into a training set of 15,708 instances and a testing set of 3,927 instances. Table 1 summarizes the statistics of the chosen datasets.

Table 1. Statistics of datasets

Name	instances	features	source	class	type
dna	3,186	180	statlog	3	multi-class
letter	20,000	16	Statlog	26	multi-class
segment	2,310	19	Statlog	7	mult-class
protein	24,387	357	JYW02a	3	multi-class
satimage	6,435	36	Statlog	6	multi-class
AEFD	19,635	11	GE	7	multi-class

Table 2. Prediction performance of individual models with/without bias correction, where nbs and bs indicate performances without and with bias correction, respectively. The reported results of bias correction is using the Euclidean distance metric.

	DNA				letter				segment			
	SVM	NN	CART	LR	SVM	NN	CART	LR	SVM	NN	CART	LR
nbs	0.9625	0.9475	0.9740	0.9765	0.8158	0.9350	0.8250	0.7532	0.9450	0.9850	0.9750	0.9350
bs	0.9740	0.9645	0.9800	0.9765	0.9436	0.9596	0.9148	0.8234	0.9750	0.9850	0.9800	0.9500
	protein				satimage				AEFD			
	SVM	NN	CART	LR	SVM	NN	CART	LR	SVM	NN	CART	LR
nbs	0.6709	0.6849	0.4948	0.6892	0.8575	0.8885	0.8320	0.8170	0.7675	0.8296	0.7056	0.7833
bs	0.6324	0.6130	0.5369	0.6801	0.9040	0.8995	0.8845	0.8170	0.8273	0.8442	0.7904	0.8182

Table 3. Prediction performance of multiple model fusion with bias correction using equal weights and optimal combination weights

DNA		letter		segment		protein		satimage		AEFD	
average	opt	average	opt	average	opt	average	opt	average	opt	average	opt
0.9795	0.9850	0.9126	0.9680	0.9950	0.9850	0.6730	0.6683	0.8528	0.8622	0.8180	0.8745

We use the default splitting of training, validation and testing if there exists a validation data, otherwise we manually generate a validation data set by sampling from the training data with the same size of the testing data set. For the purpose of demonstration, we train 4 classification models (SVM, NN, CART, LR) on all training data points, and report the metric of overall accuracy computed based on the confusion table [27]. The parameters in models are tuned on the validation data set. The number of nearest neighbours for estimating the bias is set to 5. The margin parameter in the structured hinge loss is set to 0.5. Both the bias correction and the model fusion are done on the previously listed dataset.

We first demonstrate the effectiveness of bias correction on individual models. The results are summarized in Table 2. From the results, we can observe

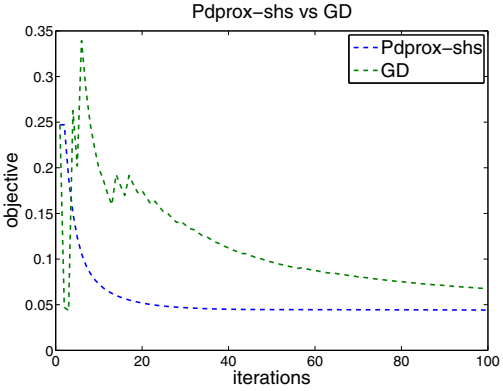

Fig. 2. Comparison of Pdprox-shs vs Gradient Descent (GD) method for optimizing the structured hinge loss on dna data set.

that bias correction can improve the prediction substantially. Furthermore, we compare the performance by using the Euclidean distance and a learned distance metric by RCA. We observed on several data sets that the distance metric learned from the ground truth can improve the performance of Euclidean distance metric, e.g. on AEFD data sets the performances of the four models are improved to $(0.8533, 0.8630, 0.8024, 0.8215)$, on letter data set the performances are improved to $(0.9546, 0.9656, 0.9216, 0.8302)$. On other data sets, the learned distance metric by RCA is comparable to the Euclidean distance metric.

Next, we demonstrate the effectiveness of model fusion. We compare the proposed convex approach for learning an optimal combination weights to the equal weighting fusion. The results are reported in Table 3. Among the six selected benchmark datasets for multi-class classification, the proposed optimal fusion approach significantly outperforms the equal weighting method on four datasets, and performs almost the same on the segment and protein data. By checking these two data sets, we find all individual classifiers perform almost equally. Thus, we draw a conclusion that the proposed fusion approach significantly outperforms simple fusion method when outputs of individual classifiers are diverse.

Finally, we show the efficiency of Pdprox-shs algorithm compared to gradient descent (GD) method for optimizing the structured hinge loss over a simplex. Both the initial step size of GD and the step size of Pdprox-shs are set to the same value 100. We plot the objective value versus the number of iterations on DNA data in Figure 2. It clearly shows that Pdprox-shs performs better than GD, which verifies our theoretical analysis on the convergence rate.

5 Conclusion

In this paper, we propose a directed inference approach for multi-class multi-model fusion. Different from traditional approaches, directed inference approach constructs a principle concept in training stage and individual (ad-hoc) rules for

classifying test samples. The presented approach consists of three key components: (i) learning of individual models, (ii) nearest neighbour search for estimating the bias of a given test sample, and (iii) learning of an optimal combination weights for fusing the bias corrected scores of multiple models. We demonstrate the effectiveness of the proposed approach on extensive data sets. In the future work, we plan to extend the work to other tasks (e.g. regression and binary classification) and conduct the experiments in real distributed learning framework.

Acknowledgement. We sincerely thank Dr. Shengzhuo Zhu for pointing out the connection to DAHI.

References

1. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: Proc. of ICML, pp. 11–18 (2003)
2. Bonissone, P.P.: Lazy meta-learning: creating customized model ensembles on demand. In: Liu, J., Alippi, C., Bouchon-Meunier, B., Greenwood, G.W., Abbass, H.A. (eds.) WCCI 2012. LNCS, vol. 7311, pp. 1–23. Springer, Heidelberg (2012)
3. Bonissone, P.P., Varma, A., Aggour, K.S., Xue, F.: Design of local fuzzy models using evolutionary algorithms. Comput. Stat. Data Anal. 51(1), 398–416 (2006)
4. Bonissone, P.P., Xue, F., Subbu, R.: Fast meta-models for local fusion of multiple predictive models. Appl. Soft Comput. 11(2), 1529–1539 (2011)
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey (1984)
6. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
7. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning, 273–297 (1995)
9. Duchi, J., Shalev-Shwartz, S., Singer, Y., Chandra, T.: Efficient projections onto the l1-ball for learning in high dimensions. In: Proc. of ICML, pp. 272–279 (2008)
10. Grove, A.J., Schuurmans, D.: Boosting in the limit: Maximizing the margin of learned ensembles (1998)
11. Ho, T.K.: Random decision forest. In: Proc. of the 3rd International Conference on Document Analysis and Recognition, pp. 278–282. IEEE (1995)
12. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE TPAMI 16(1), 66–75 (1994)
13. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Pro. of the National Academy of Sciences 79(8), 2554–2558 (1982)
14. Lam, L., Suen, S.Y.: Application of majority voting to pattern recognition: an analysis of its behavior and performance. Trans. Sys. Man Cyber. Part A 27(5), 553–568 (1997)
15. Mayers, J.H., Forgy, E.W.: The development of numerical credit evaluation systems. Journal of the American Statistical Association, 799–806 (1963)
16. Schapire, R.: The boosting approach to machine learning: An overview (2003)
17. Shental, N., Hertz, T., Weinshall, D., Pavel, M.: Adjustment learning and relevant component analysis. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 776–790. Springer, Heidelberg (2002)

18. Vapnik, V.: Problems of empirical inference in machine learning and philosophy of science. Invited Talk at Tenth International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Regina, Saskatchewan (2005)
19. Vapnik, V.: Estimation of Dependences Based on Empirical Data: Springer Series in Statistics. Springer-Verlag New York, Inc., Secaucus (1982)
20. Verikas, A., Lipnickas, A., Malmqvist, K., Bacauskiene, M., Gelzinis, A.: Soft combination of neural classifiers: A comparative study. Pattern Recognition Letters 20(4), 429–444 (1999)
21. Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241–259 (1992)
22. Woods, K., Philip Kegelmeyer Jr., W., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. IEEE TPAMI 19(4), 405–410 (1997)
23. Wu, L., Hoi, S.C.H., Jin, R., Zhu, J., Yu, N.: Learning bregman distance functions for semi-supervised clustering. IEEE TKDE 24(3), 478–491 (2012)
24. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Proc. of NIPS, pp. 505–512. MIT Press (2002)
25. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics 22(3), 418–435 (1992)
26. Xue, F., Subbu, R., Bonissone, P.P.: Locally weighted fusion of multiple predictive models. In: Proc. of IJCNN, pp. 2137–2143. IEEE (2006)
27. Yan, W., Xue, F.: Jet engine gas path fault diagnosis using dynamic fusion of multiple classifiers. In: Proc. of IJCNN, pp. 1585–1591. IEEE (2008)
28. Yang, T., Jin, R., Jain, A.K.: Learning from noisy side information by generalized maximum entropy model. In: Proc. of ICML (2010)
29. Yang, T., Mahdavi, M., Jin, R., Zhu, S.: An efficient primal-dual prox method for non-smooth optimization, arxiv (2012)