# Knowledge Reuse in Multiple Classifier Systems ⋆

Kurt Dewitt Bollacker and Joydeep Ghosh

*Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712*

**Abstract**

We introduce a framework for the reuse of knowledge from previously trained classifiers to improve performance in a current and possibly related classification task. The approach used is flexible in the type and relevance of reused classifiers and is also scalable. Experiments on public domain data sets demonstrate the usefulness of this approach when one is faced with very few training examples or very noisy training data.

*Key words:* Knowledge Transfer, Multiple Classifier, Mutual Information

## 1 Introduction

Artificial classifiers depend heavily on the set of training samples to make classification decisions. If the training set insufficiently represents the "essence" of a classification task, then creation of a well generalizing classifier for that task may not be possible. In the construction of artificial classifiers, the inclusion of previously learned knowledge embodied in existing classifiers is a potential approach to the problem of inadequate training data. However, both a suitable representation of the knowledge to be reused, and a mechanism for identification of pertinent knowledge and its incorporation using that representation must be designed. We use attributes of human knowledge reuse as a guide to this design.

One of the most impressive traits of human knowledge reuse is the ability to draw simultaneously from a large number of previous experiences quickly and

⋆ This work was supported by the Army Research Office contracts DAAH-94-G-0417 and DAAH 049510494 and The National Science Foundation.

easily. Each bit of learned knowledge may not help much, but as a whole, the knowledge gained from experience can paint a very clear picture of the problem domain. Analogously, a practical artificial knowledge reuse system should be able to have good performance scalability with the amount of knowledge reused. Human flexibility in knowledge reuse is also quite notable. Humans can use knowledge learned from a variety of types of experiences without considering how that knowledge was gained. Also, humans are capable of quickly and efficiently picking out learned knowledge that is relevant to the current classification task from their immense body of experience. A flexible knowledge reuse system should be able to take advantage of a diversity of knowledge sources for reuse and have a means to judge the relevance of such knowledge.

**Previous Work:** The most common approach to obtaining decent generalization given inadequate training sets is to severely constrain the solution space using prior domain knowledge. For example, in Bayesian approaches to classification, such knowledge exists in the form of prior distributions assumed for the model parameters and the choice of prior class probabilities. In the machine learning community, the design choice is called "inductive bias" of the classifier. For example, in a decision tree, the bias is indicated by the size of the tree and the variables (or combinations thereof) considered for making the branches. In feed-forward neural networks, the type and number of hidden units, amount and form of regularization (e.g. weight decay) serve to constrain the solution. In all of these approaches, knowledge reuse is indirect. More importantly, they work well only if the inductive bias is a good match to the current problem. This is often difficult to attain in practice.

Some recent work in knowledge reuse has focused on the automated extraction and reuse of knowledge from the data sets of other relevant classifiers, including reuse of the trained classifiers themselves. Under the belief that related classification tasks may benefit from common internal features, Caruana[Caruana, 1995] has created a multilayer perceptron (MLP) based multiple classifier system that is trained *simultaneously* to perform several related classification tasks. The first layer of the MLP is common to all tasks and the second layer is specific to individual tasks. The first layer is expected to learn common features that are useful to all of the related tasks. Baxter[Baxter, 1994] has developed a rigorous analysis of a similar type of architecture, showing that as the number of simultaneously trained tasks increases, the number of examples needed per task for good generalization decreases. Pratt[Pratt, 1994] has explored a similar knowledge reuse method in which some of the trained weights from one MLP network are used to initialize weights in an MLP to be trained for a later, related task. A different approach is taken by Thrun [Thrun and O'Sullivan, 1996], who proposed a method to estimate classifier relevance by measuring how much better a classifier performs with a reused scaling vector for nearest neighbor classifiers. Tasks with mutually helpful scaling vectors can be "clustered" into related groups.

Recently, popular approaches such as committees, ensembles, and mixture of experts also use multiple classifiers. However, since all these classifiers try to solve the same task (though they may specialize in different input regions), they are not germane to the work presented here.

## 2 Methods

We describe here an architecture for knowledge reuse from previously trained classifiers. Classifiers trained for the current classification task are called target classifiers while classifiers previously trained to perform other classification tasks are termed support classifiers as indicated in Figure 1. Our reuse strategy

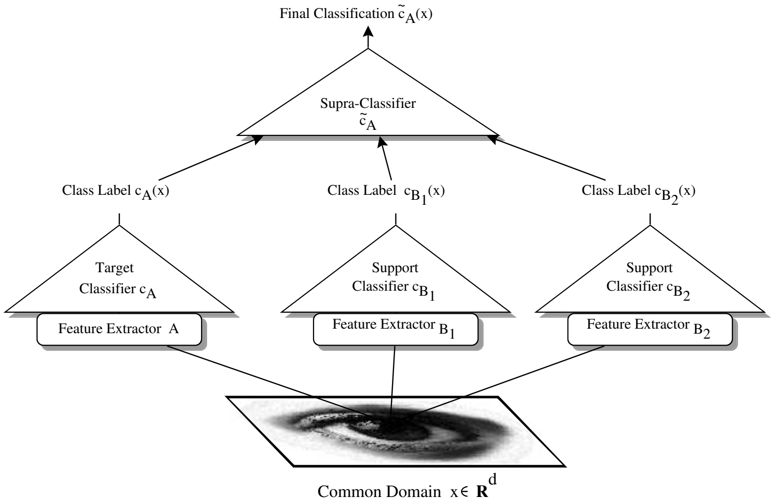


Fig. 1. A *Supra-Classifier* Reuse Architecture

is to apply the input values of each of the training samples available for the target task to all available *relevant* classifiers. The output class labels of the target and support classifiers are observed by a second stage *supra-classifier* which makes the ultimate classification ($\tilde{c}_A(\cdot)$ in the figure). Since no internal information is being used, the support classifiers can be of any type.

**2.1 A Few Definitions:** Let the target classification task be $A$, and let $A$ have discrete range $\mathcal{S}_A$ and $d$ dimensional input domain space $\mathcal{R}^d$. Let $\{x, y\}_A : x \in \mathcal{R}^d, y \in \mathcal{S}_A$ be the set of training examples for task $A$. We assume that $\{x, y\}_A$ is sampled from the true distribution for task $A$ with associated random variable $(X_A, Y_A) \in (\mathcal{R}^d, \mathcal{S}_A)$. Our goal is to find the most likely value of the conditional marginal $Y_A|(X_A = x)$ and define this maximum likelihood function to be $t_A(x) = \arg\max_y P(Y_A = y|X_A = x)$. Thus, $t_A(\cdot) : t_A(\cdot) \in \mathcal{S}_A$ is the target function that we would like to approximate using the information in $\{x, y\}_A$. Let $c_A(\cdot)$ be a function mapping $\mathcal{R}^d \mapsto \mathcal{S}_A$ which is designed to perform classification task $A$. Let $\mathcal{B}$ be a set of support classification tasks which have the same input domain space $\mathcal{R}^d$ as task $A$. Let $\{c_B(\cdot)\} : B \in \mathcal{B}$ be the corresponding set of classifiers where each $c_B(\cdot)$ maps $\mathcal{R}^d \mapsto \mathcal{S}_B : B \in \mathcal{B}$. Let $\hat{X}_A$ be the random variable associated with the input values of training sample set $\{x, y\}_A$. Let $T_A : T_A = t_A(\hat{X}_A)$ be defined as the random variable associated with the target function of $\hat{X}_A$. Similarly, let $C_A : C_A = c_A(\hat{X}_A)$ and $C_B : C_B = c_B(\hat{X}_A)$ be the random variables resulting from the application of $\hat{X}_A$ to classifiers $c_A(\cdot)$ and $c_B(\cdot)$ respectively.

An *Ideal Supra-Classifier* $c^*_A(x)$ will always choose the most likely class $y \in \mathcal{S}_A$ given the class labels $c_A(x)$ and $\{c_B(x)\} : B \in \mathcal{B}$. More specifically, For any given $k_A : k_A \in \mathcal{S}_A$, $\{k_B : k_B \in \mathcal{S}_B\} : B \in \mathcal{B}$ we can define the maximum probability function $m(\cdot)$ as $m(k_A, \{k_B\} : b \in \mathcal{B}, A, \mathcal{B}) = \arg\max_y P(T_A = y | C_A = k_A, \{C_B = k_B\} : B \in \mathcal{B})$. We can then define an ideal classifier based on this maximum probability function as

$$c^*_A(x) = m(c_A(x), \{c_B(x)\} : B \in \mathcal{B}, A, \mathcal{B}). \tag{1}$$

where $c^*_A(\cdot)$ has associated random variable $C^*_A : C^*_A = c^*_A(\hat{X}_A)$. In practice if the number of support classifiers is quite large, Equation 1 becomes impractical due to the *curse of dimensionality*[Friedman, 1994]. Therefore, we introduce two approximating approaches to Equation 1 in Section 2.3 below.

**2.2 Classifier Relevance Measure:** A measure of relevance of each support classifier to the target classification task would be helpful in the construction of a supra-classifier. We have chosen to use mutual information $I(\cdot;\cdot)$ (a measure of the amount of shared information between two random variables) with the target distribution as a classifier's relevance to that classification task. If $I(T_A; C_{B_1}) > I(T_A; C_{B_2})$, then we say that $c_{B_1}$ "knows" more about $t_A(\cdot)$ than does $c_{B_2}(\cdot)$. We have empirically demonstrated that mutual information can be used effectively as a relevance measure in our knowledge reuse framework[Bollacker and Ghosh, 1997].

**2.3 Practical Supra-Classifier Methods:** The problem of designing a practical supra-classifier can be thought of as designing a classifier for a task with a large number of discrete features, many or most of which may only be barely useful. We introduce two supra-classifier approaches that are designed to scale linearly in their computational requirements with the number of reused support classifiers in order to satisfy our design goal of scalability.

**2.3.1 Cascaded Maximum Posterior Probabilities:** Let us relabel the support classifiers $\{c_B(\cdot)\} : B \in \mathcal{B}$ in an ordered fashion in the form $\{c_{B_i}(\cdot)\} : i = 0 \ldots |\mathcal{B}| - 1$ and then revisit Equation 1 considering only $c_A(\cdot)$ and the first support classifier $c_{B_0}(\cdot)$ to compute $\hat{c}^1_A(x) : \hat{c}^1_A(x) = m(c_A(x), c_{B_0}(x), a, B_0)$. We then progressively update our approximation of Equation 1 by adding each of the remaining support classifiers in $\mathcal{B}$ one at a time using the form $\hat{c}^{n+1}_A(x) = m(\hat{c}^n_A(x), c_{B_n}(x), a, B_n)$. (We define $c_A(\cdot) = \hat{c}^0_A(\cdot)$ for consistency.)

As $n$ increases, classification performance on the training example set is strictly non-decreasing (a simple proof omitted for brevity). In a variation to CMAP, we order the cascade of support classifiers by decreasing relevance as a heuristic based on the belief that it would be beneficial to have the most relevant classifiers be earlier in the cascade.

**2.3.2 Hamming Nearest Neighbor:** If $\delta(\cdot)$ is the indicator function, then the distance measure between two samples $x_{train}$ and $x_{test}$ can be calculated as $D_{Hamming}(x_{train}, x_{test}) = \sum_{B_i:i=0\ldots|\mathcal{B}|-1} \delta(c_{B_i}(x_{train}) \neq c_{B_i}(x_{test}))$. For each test example, the Hamming Nearest Neighbor (HNN) supra-classifier will choose the class label of the training example with the smallest Hamming distance from it. In a weighted variation of this supra-classifier method (WHNN), the distance contribution of each support classifier (0 or 1) to the total Hamming distance is multiplied by its relevance (mutual information).

## 3 Experiments

If there are too few training examples, or if the examples are too noisy, then good generalization may not be possible with the information from the target problem's training examples alone. It is these two cases that we have investigated. In order to test and compare the supra-classifier methods with unaided target classifiers, we took two public domain data sets from the U.C Irvine Machine Learning database and partitioned the examples into two disjoint and unequal sized subsets *based on their class labels*. The subset with fewer classes became the target task. The other subset was used to create several two-class problems using all combinations of two classes. First, a 20000 example capital English letter data set (LR) donated by David Slate was divided into the target data set consisting of the five classes "H", "L", "O", "R", and "S" and 210 support classifiers consisting of two-class classifiers of the other 21 classes. Second, a spoken vowel data set (VOW) contributed by Peter Turney consisted of 990 examples evenly distributed among 11 spoken vowels. The two classes "hud" and "hed" were chosen to form the target classifier task and examples from the remaining 9 classes were used to construct 36 support classifiers.

**Case of Few Examples:** The LR data set of 20000 training examples was randomly partitioned into equal sized "base" training and test sets. Both target and support classifier training and test sets were created by taking only examples of the target or support classes respectively from the base training and test sets. The target training set was used to create MLP, single nearest neighbor (1-NN), and C4.5 target classifiers for each target problem. The 210 LR support classifiers were trained MLP's. In order to consider the case of few available target training examples, only a fraction of the available training examples was actually used for training of the target classifier and supra-classifiers. Target training sets of sizes 5, 20, 40, 80, 160, 320, and 480 examples were applied to the MLP and 1-NN target classifiers and all of the support classifiers. The outputs of these target and support classifiers were then used as the input vector for each of the supra-classifiers. Average results over 20 trials can be seen in Figure 2. The WHNN followed by the unweighted

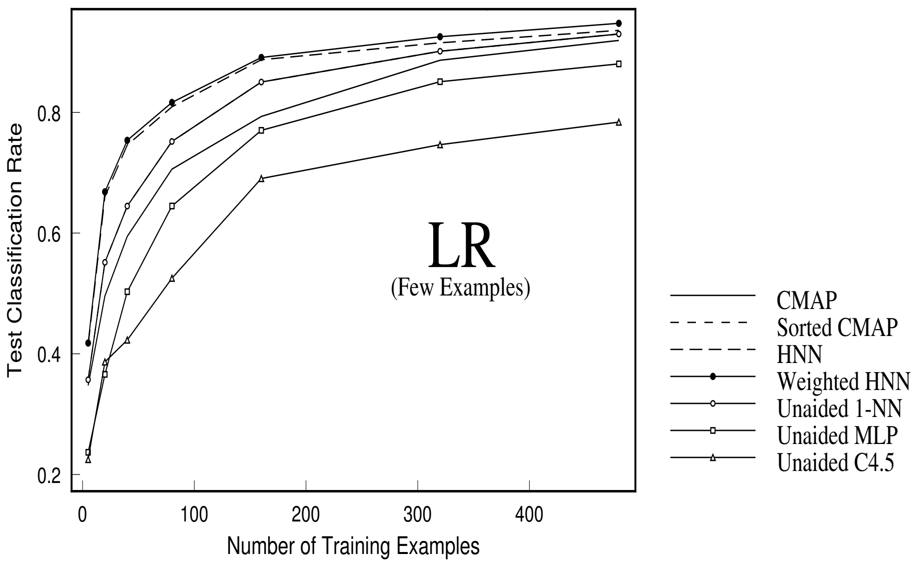


Fig. 2. Classification performance of supra-classifiers and unaided classifiers vs. number of training examples on the LR data set.

HNN supra-classifiers showed better classification performance than the unaided MLP, 1-NN and C4.5 classifiers, especially when the number of training examples was very low. The sorted CMAP supra-classifier performed identically to the unaided 1-NN and the unsorted CMAP performed worse. This gives evidence that the information provided by the support classifiers can compensate somewhat for a lack of sufficient training set size.

**Noisy Examples:** A similar experimental setup to above was made but for the VOW data set in the case of noisy examples. For the target classification problems, Gaussian noise was added to each input vector of the target training set. (The noise covariance matrix was $\sigma^2 I$.) An MLP target classifier and 36 1-NN support classifiers were used in 100 experimental trials performed over a range ($\sigma^2 = 0$ to 16) of noise variances. Average results for the vowel problem are shown in Figure 3. The performance boost from knowledge reuse in the HNN is quite prominent, but as expected, the advantage disappears as the noise level is lowered.

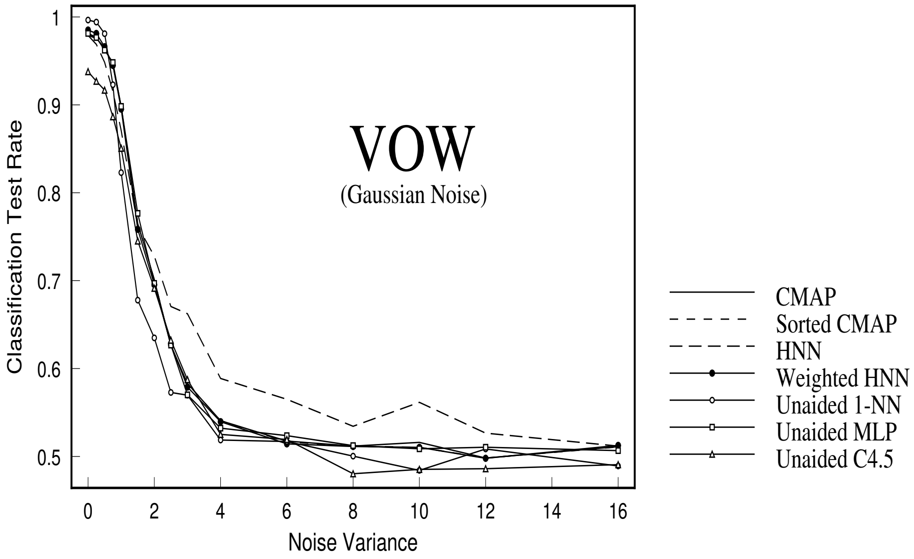


Fig. 3. Classification performance of supra-classifiers and unaided classifiers vs. Gaussian noise variance on the vowel data set.

## 4 Conclusions

In both the case of high noise and of few training examples, knowledge reuse from relevant classifiers via an appropriate supra-classifier improved performance while adhering to the flexibility and scalability design constraints. This happened even though the previously trained support classifiers *had no output classes in common* with the target classifier. Thus, we have evidence that the knowledge reuse framework presented here can be a practical means for the reuse of knowledge from classifiers that are diverse in form and purpose. We also used a mutual information based relevance measure to guide the construction of some of the supra-classifier methods. This had mixed results for both the CMAP and HNN supra-classifiers, indicating that a relevance measure may help if used carefully.

Although we have shown some encouraging empirical results, there are several directions in which this work can be extended. One of the most important extensions to this work will be application to a truly complex problem domain. We envision the eventual construction of a powerful and broadly applicable "warehouse" of previously constructed reusable classifiers for a large domain of interest (e.g. image databases), where the set of support classifiers will serve as an efficient representation of the problem domain knowledge.

## References

[Baxter, 1994] Baxter, J. (1994). *Learning Internal Representations.* PhD thesis, The Flinders University of South Australia.

[Bollacker and Ghosh, 1997] Bollacker, K. D. and Ghosh, J. (1997). A scalable method for classifier knowledge reuse. In *Proceedings of the 1997 International Conference on Neural Networks (Accepted).*

[Caruana, 1995] Caruana, R. (1995). Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664.

[Friedman, 1994] Friedman, J. H. (1994). An overview of predictive learning and function approximation. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*, pages 1–61. Springer Verlag.

[Pratt, 1994] Pratt, L. Y. (1994). Experiments on the transfer of knowledge between neural networks. In Hanson, S., Drastal, G., and Rivest, R., editors, *Computational Learning Theory and Natural Learning Systems, Constraints and Prospects*, chapter 19, pages 523–560. MIT Press.

[Thrun and O'Sullivan, 1996] Thrun, S. and O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC alogorithm. In *The 13th International Conference on Machine Learning.*