Voting Over Multiple Condensed Nearest Neighbors

Ethem Alpaydın
Department of Computer Engineering
Boğaziçi University
TR–80815 Istanbul Turkey
alpaydin@boun.edu.tr

December 10, 1995

Abstract

Lazy learning methods like the k-nearest neighbor classifier require storing the whole training set and may be too costly when this set is large. The condensed nearest neighbor classifier incrementally stores a subset of the sample, thus decreasing storage and computation requirements. We propose to train multiple such subsets and take a vote over them, thus combining predictions from a set of concept descriptions. We investigate two voting schemes: simple voting where voters have equal weight and weighted voting where weights depend on classifiers' confidences in their predictions. We consider ways to form such subsets for improved performance: When the training set is small, voting improves performance considerably. If the training set is not small, then voters converge to similar solutions and we do not gain anything by voting. To alleviate this, when the training set is of intermediate size, we use bootstrapping to generate smaller training sets over which we train the voters. When the training set is large, we partition it into smaller, mutually exclusive subsets and then train the voters. Simulation results on six datasets are reported with good results. We give a review of methods for combining multiple learners. The idea of taking a vote over multiple learners can be applied with any type of learning scheme.

Keywords— Lazy Learning, Nonparametric Estimation, K-Nearest Neighbor, Condensed Nearest Neighbor, Voting.

1 Introduction

Lazy learning methods like the k-nearest neighbor classifier estimate directly from a given labelled sample (Duda and Hart, 1973). In statistical estimation theory, such techniques are named *nonparametric*. This differs from the *parametric* approaches where a given model is assumed whose parameters are estimated from the given sample (e.g., using maximum likelihood). In many situations, no appropriate parametric model is known or the estimation of parameters may be too costly or not optimal. In such cases, one opts for the nonparametric approach where particular instances or examples are stored as opposed to abstracting general rules.

Methods where response is computed by interpolating from a table of stored patterns is called *instance-based* (Aha et al., 1991) or *memory-based* (Stanfill and Waltz, 1986) in the machine learning literature. In statistics, this approach is called *kernel-based density estimation* (Silverman, 1986) when one estimates the class-conditional densities for classification and *locally-weighted regression* (Härdle, 1990) when one estimates a continuous function. The assumption is that the real world is smooth and that similar inputs give similar outputs or belong to the same class. Thus when giving an output, closest patterns with known outputs are the ones that should be taken into account; see (Hastie and Tibshirani, 1990; Chapter 2) for a review of smoothing models.

In this paper, we concentrate on classification where a given input is to be assigned to one of several mutually exclusive classes; extension to continuous function approximation is straightforward. In Section 2, we explain the condensed nearest neighbor classifier that incrementally forms a "sufficient" subset of the training sample. Section 3 advocates forming multiple such subsets and taking a vote over their responses. This approach is empirically justified in Section 4 on six datasets with varying characteristics. In Section 5, a review of related literature on combining multiple learners is given. The final section concludes by pointing out the implications of the current work in a more general setting.

2 Condensed Nearest Neighbor Rule

A main disadvantage of a lazy learner like the k-nearest neighbor classifier is the large memory requirement to store the whole sample. When the sample is large, response time on a sequential computer is also large. To get rid of the redundancy and decrease the number of free parameters, an alternative to abstract, parametric models is an editing procedure that selectively discards the redundant part of the training set. Hart (1968) proposed to minimize the number of stored patterns by storing only a subset of the training set. The basic idea is that patterns in the training set may be very similar and some do not add extra information and thus may be discarded.

We want to find a subset of the sample $\mathcal{S}$ that is small and accurate. To choose the best subset $\mathcal{Z}^*$, out of the $2^{|\mathcal{S}|}$ possible subsets, we define the error measure given in Equation (1) using regularization theory:

$$
\begin{aligned}
\mathcal{Z}^* &= \arg\min_{\mathcal{Z}} E(\mathcal{Z}) \\
E(\mathcal{Z}) &= \sum_{x \in \mathcal{S}} L(x|\mathcal{Z}) + \gamma|\mathcal{Z}| \qquad (1) \\
L(x|\mathcal{Z}) &= \begin{cases} 1, & \text{if } \mathcal{D}(z_c, x) = \min_j \mathcal{D}(z_j, x) \text{ and} \\ & \text{class}(x) \neq \text{class}(z_c); \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
$$

$z_c \in \mathcal{Z}$ is the closest stored pattern to $x \in \mathcal{S}$ using the distance measure $\mathcal{D}(\cdot)$. $L(x|\mathcal{Z})$ is non-zero when the labels of x and z_c do not match.

According to regularization theory (Vapnik, 1995), the solution to a problem can be obtained by combining the data and prior smoothness information. The first term in Equation (1) measures the data-misfit that is due to the error in classification using the 1-NN rule. The second term measures the size of the subset stored and as such defines the smoothness of the class boundary. The nearest neighbor classifier divides the input space in the form of a Voronoi tesselation and the class boundaries are piecewise linear (Preparata and Shamos, 1985). As we have more examples, this boundary becomes more ragged and less smooth. The smoothest case is when we have only one example per class where we have linear boundaries between classes. γ is the regularization parameter that indicates the trade-off between these two terms. Equivalently, this is a Bayesian approach that attributes higher prior probabilities to simpler models.

Minimizing Equation (1) is a combinatorial problem and no known polynomial time procedure can solve it to optimality. Though no formal proof is known, we believe this problem to be NP-complete. A computationally simple local search method has been proposed first as Condensed Nearest Neighbor (CNN) (Hart, 1968) and then later independently as IB2 (Aha et al., 1991) and Grow and Learn (GAL) (Alpaydın, 1990). The *consistent subset*, $\mathcal{Z}$, of a sample $\mathcal{S}$, is the set of patterns that can classify all the elements of $\mathcal{S}$ correctly using the 1-NN rule. There are usually many consistent subsets—one trivial consistent subset is the set itself. One normally is interested in the *minimal* consistent subset (i.e., the subset with the minimum cardinality) to minimize the cost of storage and computation.

The CNN algorithm is given in Table 1. Starting from an empty stored subset, we pass one by one over the patterns and add a pattern to the subset if it cannot be classified correctly with the already stored subset. This implies that error is more important than size (i.e., $\lambda < 1$ in Equation 1). This method, being a local search, does not guarantee finding the minimal subset and furthermore, different subsets are found when the training set order is changed. Generally passing over the sample a few times is sufficient until no more additions are made when the stored subset classifies all the patterns in the training set. At each pass, on the average, classification accuracy increases and fewer patterns are added to the subset. To make learning even faster, one can stop if the number of added patterns in the last pass is fewer than say, 5%, of the training set.

The classification rule is 1-NN but other nonparametric variants are also possible at the expense of more computation. If the sample is noisy, there is also the possibility of using k-CNN but this may be costlier (Gates, 1972). If $k = 2i + 1$, then for the correct classification of a new pattern, at least $i + 1$ of its nearest neighbors must be from the correct pattern class and if this is not true, in the worst case, it would have to be added $i + 1$ times to $\mathcal{Z}$.

Gates (1972) proposed the reduced nearest neighbor (RNN) rule that aims to further reduce the stored subset. With RNN, after having applied CNN, for each element of the subset we check if its removal causes an error in the training set and we accept the removal if it does not. GAL (Alpaydın, 1990) also has a "sleep" mode where an element of the stored subset is removed if the closest stored element is also of the same class. In both

Table 1: The Condensed Nearest Neighbor Algorithm (CNN).

```
PROCEDURE CNN(𝒮,𝒟,𝒵)
BEGIN
𝒵 := {};
REPEAT
          additions:=FALSE;
          FOR all patterns in the training set DO
                Randomly pick x from training set, 𝒮
                Find z_c ∈ 𝒵 such that 𝒟(x, z_c) = min_j 𝒟(x, z_j)
                IF class(x) ≠ class(z_c) THEN
                   𝒵 := 𝒵 ∪ x;
                   additions:=TRUE
                END IF
          END FOR
UNTIL NOT(additions);
END CNN;
```

approaches, it is concluded that the small saving in memory is not worth the extra computation.

3 Voting over Multiple Learners

The problem with this approach to determining a consistent subset is that when the order of the training set is changed, the search trajectory changes and one can converge to a different local minimum storing a different subset. These subsets, although they all classify all the patterns in the training set correctly, perform differently on the test set. A method proposed previously (Alpaydın, 1991) is to train multiple such subsets and then take a vote over their responses. This method is not limited to lazy learning methods but can be generalized to any kind of learning scheme. A review of methods for combining multiple learners is given in Section 5.

Let us say we have m classifiers and n classes. We denote by d_{ji}, the estimate of classifier j for class i, computed for example using a lazy learning method. In voting, we combine votes for classes through a weighted sum and then assign the input to the class c receiving the maximum vote:

$$r_i = \sum_{j=1}^{m} d_{ji}\beta_j \tag{2}$$

$$c = \arg\max_{i=1}^{n} [r_i] \tag{3}$$

The weights are nonnegative and sum up to unity: $\beta_j \geq 0, \quad \sum_{j=1}^{m} \beta_j = 1$. Unless there is any a priori reason to favor one voter over another, weights are taken as equal: $\beta_j = 1/m$. This is *simple voting.*

There is a method that can be used to determine the weights β_j easily for *weighted voting*: The outputs of most classifiers can be converted to probabilities. Then if, for classifier j, e is the most probable class and f is the next most probable, one can compute:

$$\alpha_j = p_j(e|x) - p_j(f|x) \tag{4}$$

which, after normalization can be used as weights: $\beta_j = \alpha_j / \sum_{l=1}^{m} \alpha_l$. If $\alpha_j \approx 1$, classifier j is very "certain"; as x gets closer to the boundary then $p_j(e|x) \approx p_j(f|x)$ and the difference gets closer to zero and classifier j is less certain. Thus certain classifiers will have more say in the final output than the classifiers who are not that sure.

In the case of CNN, we use a simple method to compute α_j by just looking at two neighbors. If $z_{j,[1]}$ is the closest pattern to x in $\mathcal{Z}_j$ and $z_{j,[2]}$ is the second closest, we have:

$$\alpha_j = \begin{cases} 1, & \text{if } \mathrm{class}(x_{j,[1]}) = \mathrm{class}(x_{j,[2]}); \\ \frac{\mathcal{D}(x,z_{j,[2]}) - \mathcal{D}(x,z_{j,[1]})}{\mathcal{D}(x,z_{j,[2]})}, & \text{otherwise.} \end{cases} \tag{5}$$

where $\mathcal{D}(\cdot)$ is the distance measure. If the two neighbors both belong to the same class, the classifier is very certain. Otherwise its certainty decreases as the input gets closer to the boundary which is halfway between them.

We show in the Appendix that error decreases through voting with an increasing number of uncorrelated voters. Intuitively what is happening is that each particular subset is generating a different noisy approximation to the real class boundary and the averaging process is smoothing over the noise

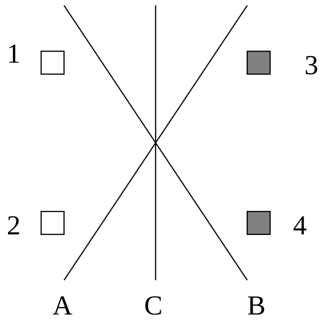

Figure 1: Patterns 1 and 2 belong to one class and 3 and 4 to the second one. One run of CNN can choose 1 and 4 which define the discriminant A and another CNN can choose 2 and 3 that define the discriminant B. Both are minimal consistent subsets. C is the average of A and B and is the right discriminant. Adapted from (Perrone, 1993).

(Figure 1). Each particular learner falls into a different (local) minimum and they perform poorly in different regions of the input space and their error terms will not be strongly correlated. As the number of voters increase, this assumption of independence breaks down and voters become more similar and there is nothing to be gained by combining learners unless measures are taken to force them to be dissimilar.

4 Simulation Results

4.1 Datasets

To validate the advantage of voting over multiple CNN, we have compared it on six datasets with CNN and 1-NN. The properties of the datasets are given in Table 2. OCR is a handwritten digit database (Guyon et al., 1989). Others are available from the UCI Repository (Murphy, 1994). In the OCR, VOWEL, and THYROID datasets, the training and test sets are separated. In others, we chose the training set small for not to have too large accuracy with NN thus leaving space for improvement. Euclidean distance is used as the distance metric; the WINE and THYROID databases are z-transformed before training (i.e., to normalize the numeric attributes such that all have zero mean and unit variance). 1-NN and 1-CNN are used in all datasets

Table 2: Properties of the datasets used.

	Number of Features	Number of Classes	Number of Training Examples	Number of Test Examples
OCR	256	10	600	600
IRIS	4	3	15	135
WINE	13	3	100	78
VOWEL	10	11	528	462
THYROID	21	3	3,772	3,428
GLASS	9	7	100	114

except VOWEL, where 7-NN and 7-CNN are used. This optimal k is chosen by 2-fold cross validation from among $\{1, 3, 5, 7, 9\}$. At each run of the CNN, the training set is scanned in a different order to get a new subset. Reported values are average and standard deviations of ten independent runs.

4.2 First Comparison

We applied nearest neighbor (NN), condensed nearest neighbor (CNN) and voting to all six datasets. Both simple and weighted voting are used with various number of voters. Another way to combine the subsets found by multiple CNN is by applying NN to the union of subsets chosen by multiple CNN. Comparing the results of voting with the performance on the union allows us to highlight whether it is the voting process or the number of instances that increase the performance. However, this union approach can still be considered a multiple learner scheme because we run CNN multiple times as opposed to just once. This can only be used with a method like k-NN where effects are local; one cannot for example train multiple multi-layer perceptrons and combine all hidden units together as one big hidden layer and accept any improvement.

By taking a vote over as few as three CNN subsets, in three out of six datasets, although one collectively stores a *subset* of the training set, one achieves higher classification accuracy than the nearest neighbor, where the whole training set is stored (Table 3). In GLASS and THYROID, with three

Table 3: Accuracy results of applying nearest neighbor (NN), condensed nearest neighbor (CNN) and simple and weighted voting over three voters are given. Percentage of training stored by CNN is also given to show the gain in memory.

	NN	CNN (% of training set stored by CNN)	Voting Simple	Voting Weighted	NN on Union
OCR	93.17	90.08 (0.20)	91.95	93.67	92.42
IRIS	91.85	91.41 (0.29)	92.67	94.00	92.22
WINE	94.87	93.21 (0.14)	93.85	95.00	93.97
VOWEL	60.17	51.62 (0.67)	56.56	55.97	57.14
THYROID	93.14	90.95 (0.17)	91.63	92.23	92.55
GLASS	71.94	69.82 (0.57)	70.00	71.67	71.40

voters we also get better results by a partitioning the dataset, as described in Section 4.3. We also note that voting is better than using NN on the union, showing that the performance benefit is not due to the larger number of stored patterns but rather to *the way predictions are combined* from a set of concept descriptions. Generally, weighted voting is better than simple voting.

As can be seen in Figure 2 for the case of weighted voting on the OCR database, when the number of voters increase, not only the average classification accuracy goes higher but the variance also decreases. This indicates better generalization and is the clear advantage of voting. Complete results are given in Table 4. Results for the IRIS and WINE datasets are similar and are omitted.

When one increases the number of voting subsets, after a certain number, new subsets do not contribute much. Whether an additional subset pays off the additional complexity and memory is a trade-off that needs to be resolved depending on the particular application at hand.

In three datasets, VOWEL, THYROID, and GLASS, we do not seem to gain anything by voting. The VOWEL database defines a quite difficult problem and is very noisy; the optimal k is 7. The result with 7-NN is the highest achieved and is better than all reported in the UCI Repository

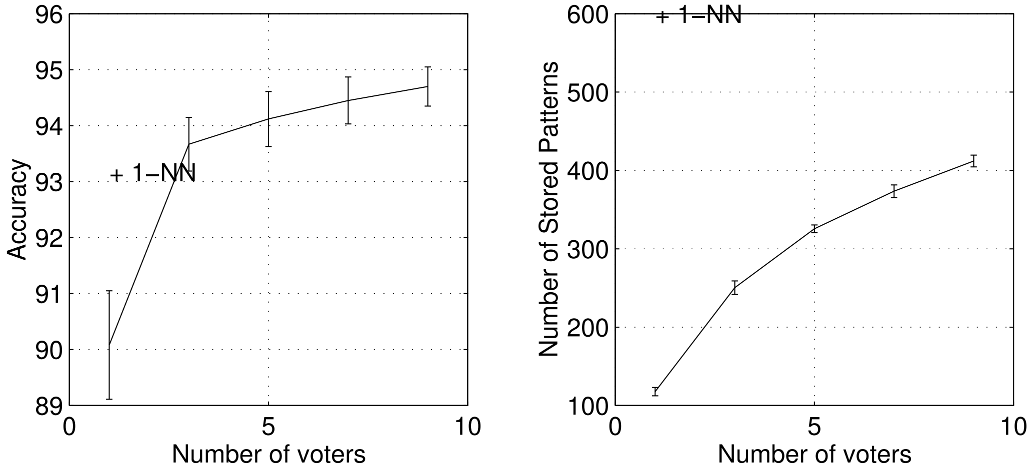

Figure 2: Average classification accuracy on the OCR dataset with weighted voting and the number of stored patterns stored as a function of the number of voters. For comparison, classification accuracy and the number of stored patterns are marked when 1-NN is used over the whole training set.

Table 4: Results on the OCR dataset with nearest neighbor classifier (NN) and condensed nearest neighbor classifier (CNN). Values are average, standard deviation over ten independent runs. Results with simple and weighted voting over multiple condensed 2-nearest neighbor classifiers are also given. Results on the IRIS and WINE datasets are similar.

Method	Accuracy	Patterns	Epochs
NN	93.17, 0.00	600.00, 0.00	1.00, 0.00
CNN	90.08, 0.97	117.70, 5.25	2.80, 0.63

		Accuracy		
Number of voters	Simple	Weighted	Union	Patterns
3	91.95, 0.78	93.67, 0.48	92.42, 0.64	250.40, 8.54
5	93.23, 0.55	94.12, 0.49	92.77, 0.53	325.50, 5.02
7	93.38, 0.50	94.45, 0.42	93.02, 0.30	373.50, 8.07
9	93.53, 0.53	94.70, 0.35	93.05, 0.16	412.10, 7.48

for that dataset (e.g., the multi layer perceptron gives 51.0). This is most probably due to the fact that each vowel sound is coded using only ten attributes and much information is lost in the process.

4.3 Multiple Training Subsets

In the case of the THYROID and GLASS datasets, the problem seems to be the large training set. It has been pointed out before that voting averages over noisy discriminants. As the training set gets larger, the variance of an estimator decreases and the voters become more similar. This has been shown empirically for the case of linear regression by Meir (1994). To test this, we ran CNN on mutually exclusive subsets of the THYROID dataset while always testing on the same test set. As can be seen in Figure 3, as the training set gets larger, although classification accuracy on the test set increases, variance decreases indicating that the subsets become more similar.

Having multiple learners only make sense when they differ so that they fail under *different* circumstances. So with m voters, the training set is *partitioned* into m and a separate CNN is trained on each part. Subset sizes are as given in Figure 3. Voting over these we get better results than what we get when all are trained on the same large training set. The large difference is noticeable in the comparitive results given in Figure 4. The complete results with partitioning are given in Table 5.

We also use this approach on the GLASS dataset where the training set contains 100 patterns. When we use three voters, each voter has 33 patterns to be trained on and, with simple voting, we get 72.2%, which is higher than that achieved by the previous scheme and 1-NN on the whole training set. However if we partition the training set for example into five, each subset contains only 20 patterns and classification accuracy is low and gets lower as the number of voters is increased.

If the training set is not large enough to allow partitioning, one can use *bootstrap* which involves generating new datasets from one original dataset by sampling randomly *with* replacement (Perrone, 1993) (LeBlanc and Tibshirani, 1994). These new datasets can then be used to generate multiple CNN subsets. With the GLASS dataset, we used bootstrapping to generate samples of size 50 from a training set of 100 patterns. For more than three voters, bootstrapping worked better than partitioning. Results are given in Table 6. Especially with 7 and 9 voters, these results are higher than those achieved

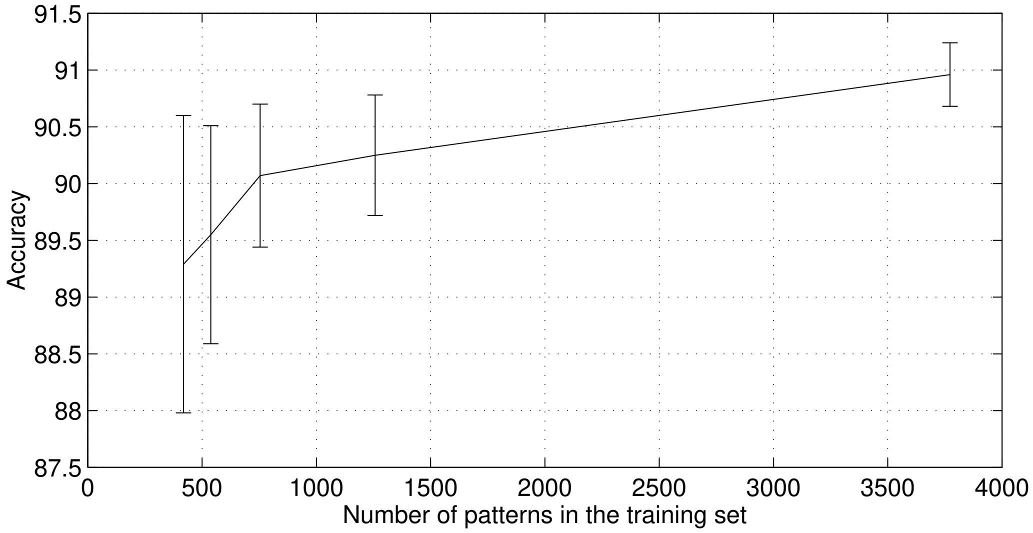

Figure 3: Average classification accuracy on the THYROID dataset with CNN as a function of the number of the training samples. The partitioning is into 9, 7, 5, 3, and 1.

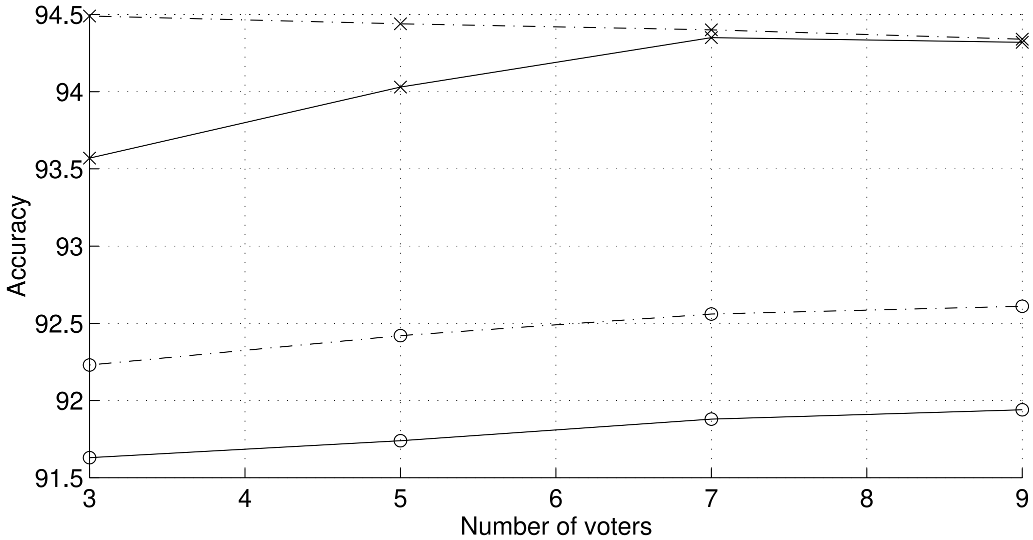

Figure 4: Comparison of averages on the THYROID dataset when the voters are trained on the entire training set ('o') vs. its partitions ('x') with simple (continuous) and weighted (dashed) voting.

Table 5: Complete results on the THYROID dataset. Partitioning is used when taking a vote.

Method	Accuracy	Patterns	Epochs
NN	93.14, 0.00	3,772.00, 0.00	1.00, 0.00
CNN	90.95, 0.24	625.30, 7.69	3.60, 0.70

	Accuracy			
Number of voters	Simple	Weighted	Union	Patterns
3	93.57, 0.19	94.49, 0.13	90.25, 0.53	708.70, 14.36
5	94.03, 0.29	94.44, 0.09	89.48, 0.44	738.50, 10.84
7	94.35, 0.15	94.40, 0.11	89.11, 0.28	776.30, 14.88
9	94.32, 0.14	94.34, 0.07	89.13, 0.36	792.50, 13.85

by applying CNN on the entire training set and also on its partitions.

We thus conclude that if the training set is small, the voters converge to sufficiently different solutions, then voting helps. When the training set is of intermediate size, one can use bootstrapping to generate smaller training sets. When the training set is large, each voter can use a separate training set. What makes a training set "small" or "large" is the difficulty of the underlying task depending on several factors (e.g., the variance of noise, the dimensionality of the input, number of classes, etc.).

5 Combining Multiple Learners

The simplest way to combine multiple learners is by *voting*, which corresponds to taking a linear combination of the learners. If each voter just indicates the class it thinks the input belongs to, then one can only have *simple voting* where all voters have equal weight. If the voters can also supply information on how much they vote for each class, then these can be converted to certainties and used as weights in a *weighted voting* scheme.

In Section 3, we discussed one way to compute these weights in the case of lazy learning based on distances. A *belief* measure (Xu et al., 1992) or Dempster-Schafer theory can be used for the same purpose (Xu et al., 1992;

Table 6: Results on the GLASS dataset.

Method	Accuracy	Patterns	Epochs
NN	71.93, 0.00	100.00, 0.00	1.00, 0.00
CNN	69.82, 0.94	57.10, 3.11	2.30, 0.48

	Accuracy			
Number of voters	Simple	Weighted	Union	Patterns
3 (partition)	72.19, 1.31	70.53, 2.16	70.00, 1.93	64.30, 1.49
5 (bootstrap)	72.02, 1.96	72.11, 2.44	71.84, 1.52	74.60, 2.12
7 (bootstrap)	74.56, 1.89	73.25, 2.04	71.32, 1.24	84.50, 1.72
9 (bootstrap)	74.12, 1.39	74.30, 1.66	71.67, 0.59	88.60, 1.90

Rogova, 1994). Lincoln and Skrzypek (1990) propose a way to learn the weights in a voting scheme. Model complexities can also be taken into account in a Bayesian framework to make sure that complex models are not given very large weights (Alpaydın, 1993). Perrone (1993) gives a number of didactic examples that depict the advantage of voting. He also shows that for minimum square error, when the learners are unbiased and uncorrelated, weights should be inversely proportional to variances (see Appendix). Benediktsson and Swain (1992) propose to use a voting scheme for multisource sensing where data from different sources are integrated based on consensus theory.

It has been shown by Hansen and Salamon (1990) that given independent classifiers with classification accuracy probability higher than 1/2, by taking a majority vote, classification accuracy increases as the number of voting classifiers increase. Mani (1991) has shown that in the case of simple voting, variance decreases as the number of voters increase (See Appendix).

Adapting what Perrone (1993) stated for regression to classification, if we view each learner as a random noise function added to the true class discriminant function and if these noise functions are uncorrelated with zero mean, then the averaging of the individual estimates is like averaging over the noise. In this sense, the voting method is smoothing in the functional space and can be thought of as a regularizer with a smoothness assumption on the true discriminant function.

Wolpert (1992) proposed a method called *stacked generalization* that extends voting. In stacking, the output of the learners is combined through a combiner system which is also trained and is not restricted to be linear. The learners are called level 0 generalizers and the combiner is the level 1 generalizer. The level 1 generalizer learns what the correct output is when level 0 generalizers give a certain output combination. Thus level 1 needs be trained on data unused in training the level 0 generalizers. Wolpert proposes to use leave-one-out though this is too costly and n-fold cross validation seems to be better. Zhang et al. (1992) use stacking for protein secondary structure prediction with significant improvement in accuracy. In their study, the level 0 generalizers are a statistical model, a lazy learner and a one hidden layer neural network. The level 1 generalizer is another neural network with one hidden layer. Breiman (1992) discusses stacking from a statistical perspective.

Boosting (Drucker et al., 1993) trains the learners serially. After having trained the first, they train the second learner with the data on which the first fails (and as many data on which the first succeeds) thus making sure that that the two learners complement one another. Then a third learner is trained with the data on which the two learners disagree. During testing, if the first two learners agree then that is taken as the output, otherwise the third learner is consulted. Large training samples are required for boosting as a learner is trained only with the data on which the previous learners fail.

The boosting approach makes sense because when we have a second learner, we do not care about its overall performance but we just want it to perform well on cases where the first one fails and an intelligent switch to choose between the two. In the *adaptive mixtures of local experts* (Jacobs et al., 1991), there are a set of local experts that partition the input space among themselves and a separate gating expert that, given an input, decides which expert to use. The local experts and the gating expert are trained all in parallel as opposed to the serial approach taken by boosting.

Krogh and Vedelsby (1995) measure "ambiguity" as the variation of the output of voters averaged over unlabelled data to quantify the disagreement among the voters. They define:

$$E = \overline{E} - \overline{A}$$

where E is the error after voting, $\overline{E}$ is the average of the generalization errors of the individual voters and $\overline{A}$ is the average of ambiguities. Thus for minimum error, we need to maximize the ambiguity. They show that if the voters are strongly biased, the ambiguity will be small because the voters implement very similar functions and thus agree on inputs even outside the training set. If on the other hand there is a large variation, the ambiguity is high and in this case the generalization error will be smaller than the average generalization error. They also note that one way to increase the ambiguity is to train the voters on different datasets. Our results given in Section 4 are in accordance with theirs.

Tresp and Taniguchi (1995) propose to use a "competence" measure to determine the weights in a voting scheme. This uses $\hat{p}(x|j)$, an estimate of the distribution of input data used to train voter j.

Although much work has been done on how to combine learners, the question of what to choose as the learners is an open problem. Wolpert (1992) stated that one wants level 0 generalizers to "span the space of generalizers" and be "mutually orthogonal." Generally this is done by biasing the learners in different ways (e.g., by using different learning methods or different initial conditions, etc.). For example because CNN is a local search method, we get different subsetsby reordering the training set. A better idea seems to have rather different learners as opposed to variants of one method.

6 Conclusions

Many lazy learning techniques require storing the complete sample. They generalize well indicating that they are serious competitors to more complex, gradient-descent methods. Much of the functionality of neural networks (i.e., parallel structures for pattern recognition) can be obtained from closely related but simpler techniques (e.g., distance-based classifiers) without needing complex network structures, learning rules nor precise weights (Alpaydın and Gürgen, 1995). Actually the distinction between a neural network and a lazy learning method is quite hazy. Omohundro (1987) shows how simple lazy learners can be implemented as a neural network. The neural network implementation of CNN is given in (Alpaydın, 1990). Most neural networks whose hidden units implement a local activation function like the gaussian can be recast as a lazy learner and vice versa.

To decrease storage requirements and speed-up processing, one can incrementally select a subset of the sample by making just a few passes over the sample. Most of the time, the gain in memory is worth the cost of this extra training time, which is much smaller than the time it takes to do gradient-descent. A method like CNN is promising when very rapid adaptation is necessary (e.g., in real-time systems such as robotics) and when memory space is a premium. Examples of its usage in robotics domain is given in (Reignier et al., 1995) and in industrial measurement in (Hines et al., 1993).

Generalization can be improved by training a number of such subsets and combining their predictions. It is generally accepted that there is not one optimal learning method to do anything (Schaffer, 1994). If we do not know a priori which learning method is the best, we can train a number of different learners and combine their predictions (e.g., by voting). This approach of voting is not limited to lazy learners but can be generalized to any estimation method. Recent advances in parallel processing technology allow separate voters to be implemented on separate processors thus leading to considerable increases in computational speed.

Acknowledgments

This work is supported by Grant EEEAG–143 from Tübitak, Turkish Scientific and Technical Research Council. The handwritten digit dataset (OCR) is provided by Isabelle Guyon of AT&T Bell Labs. The other datasets are from the UC Irvine Repository maintained by P. Murphy. Thanks to the two anonymous reviewers for constructive comments. Special thanks to David Aha for prompt feedback and many suggestions during the revision that significantly improved the content and the presentation of the paper.

Appendix. Effect of Voting on Estimation Error

Let us denote the output of learner j as d_j and the output after voting as r where

$$r = \sum_{j=1}^{m} d_j \beta_j$$

where $\beta_j \geq 0$ and $\sum_{j=1}^m \beta_j = 1$. We assume that $d_j, j = 1 \ldots m$ are independent and identically distributed. When b is an estimator of the parameter θ,

$$b_\theta(d) = E[d] - \theta$$

is the *bias* of d as an estimator of θ. If $b_\theta(d) = 0$ for all θ, then d is an *unbiased* estimator. In other words, an estimator is unbiased if its expected value equals the value of the parameter it is attempting to estimate. If we compute the expected value of r, we see that it is equal to the expected value of d_j:

$$\begin{aligned} E[r] &= E\left[\sum_{j=1}^m d_j \beta_j\right] = \sum_j \beta_j E[d_j] = E[d_j] \sum_j \beta_j \\ &= E[d_j] \end{aligned} \tag{6}$$

Thus voting does not change the bias; if the voters are unbiased so is the combined estimator. The *variance* of estimator d measures how much d deviates from its expected value:

$$\mathrm{Var}(d) = E[(d - E[d])^2]$$

Let us compute the variance of r in the case of simple voting where $\beta_j = 1/m$. We see that it decreases as the number of voters m increases:

$$\begin{aligned} \mathrm{Var}(r) &= \mathrm{Var}\left(\sum_{j=1}^m d_j \beta_j\right) = \mathrm{Var}\left(\sum_j \frac{d_j}{m}\right) = \frac{1}{m^2} \sum_j \mathrm{Var}(d_j) \\ &= \frac{1}{m} \mathrm{Var}(d_j) \end{aligned} \tag{7}$$

The mean square error of an estimator is equal to the sum of its variance and the square of its bias:

$$r(d, \theta) = \mathrm{Var}(d) + (b_\theta(d))^2$$

Thus variance and mean square error decreases with increasing m. Note that this assumes independence of d_j which may not always be true. It can also be shown that for minimum variance, the optimal weight to give an estimator is inversely proportional to its variance. We want to minimize:

$$E(\beta) = \mathrm{Var}\left(\sum_j d_j \beta_j\right) = \sum_j \beta_j^2 \mathrm{Var}(d_j)$$

Taking $\sigma_j^2 \equiv \mathrm{Var}(d_j)$ and using the method of Lagrange multipliers, we look for σ_j^2 that minimize:

$$E(\beta) = \sum_j \beta_j^2 \sigma_j^2 + \lambda \left(1 - \sum_j \beta_j\right)$$

Find β_j and λ that satisfy:

$$\frac{\partial E}{\partial \beta_j} = 2\beta_j \sigma_j^2 - \lambda = 0 \text{ and } \sum_j \beta_j = 1$$

We find:

$$\beta_j = \frac{1/\sigma_j^2}{\sum_l 1/\sigma_l^2} \tag{8}$$

References

[1] Aha, D. W., Kibler, D., Albert, M. K. (1991) "Instance-Based Learning Methods," *Machine Learning*, **6**, 37–66.

[2] Alpaydın, E. (1990) *Neural Models of Incremental Supervised and Unsupervised Learning*, PhD dissertation, No 869, Department d'Informatique, Ecole Polytechnique Fédérale de Lausanne, Switzerland, 1990.

[3] Alpaydın, E. (1991) *GAL: Networks that Grow When They Learn and Shrink When They Forget*, International Computer Science Institute, Berkeley: CA, TR-91-032.

[4] Alpaydın, E. (1993) "Multiple Networks for Function Learning," *IEEE International Conference on Neural Networks*, March, San Francisco: CA, **1**, 9–14.

[5] Alpaydın, E., Gürgen, F. (1995) "Comparison of Kernel Estimators, Perceptrons and Radial-Basis Functions for OCR and Speech Classification," *Neural Computing and Applications*, **3**, 38–49.

[6] Benediktsson, J. A., Swain, P. H. (1992) "Consensus Theoretic Classification Methods," *IEEE Transactions on Systems, Man, and Cybernetics*, **22**, 688–704.

[7] Breiman, L. (1992) *Stacked Regressions*, TR-367, Department of Statistics, University of California, Berkeley.

[8] Drucker, H., Schapire, R., Simard, P. (1993) "Improving Performance in Neural Networks Using a Boosting Algorithm," in *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. Cowan, L. Giles (Eds.), 42–49, Morgan Kaufmann.

[9] Duda, R. O., Hart, P. E. (1973) *Pattern Classification and Scene Analysis*, Wiley and Sons.

[10] Gates, G. W. (1972) "The Reduced Nearest Neighbor Rule," *IEEE Transactions on Information Theory*, **18**, 431–433.

[11] Guyon, I., Poujoud, I., Personnaz, L., Dreyfus, G., Denker, J., le Cun, Y. (1989) "Comparing Different Neural Architectures for Classifying Handwritten Digits," *International Joint Conference on Neural Networks*, Washington USA.

[12] Hansen, L. K., Salamon, P. (1990) "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 993–1001.

[13] Härdle, W. (1990) *Applied Nonparametric Regression*, Econometric Society Monographs, Cambridge University Press.

[14] Hart, P. E. (1968) "The Condensed Nearest Neighbor Rule," *IEEE Transactions on Information Theory*, **14**, 515–516.

[15] Hastie, T., Tibshirani, R. (1990) *Generalized Additive Models*, Chapman Hall.

[16] Hines, E. L., Gianna, C. C., Gardner, J. W. (1993) "Neural Network Based Electronic Nose Using Constructive Algorithms," in M. Taylor, P. Lisboa (Eds.) *Techniques and Application of Neural Networks*, Ellis Horwood, 135–154.

[17] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., Hinton, G. E. (1991) "Adaptive Mixtures of Local Experts," *Neural Computation*, **3**, 79–87.

[18] Krogh, A., Vedelsby, J. (1995) "Neural Network Ensembles, Cross Validation, and Active Learning," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, T. K. Leen (Eds.), MIT Press.

[19] LeBlanc, M., Tibshirani, R. (1994) *Combining Estimates in Regression and Classification*, Department of Statistics, University of Toronto.

[20] Lincoln, W. P., Skrzypek, J. (1990) "Synergy of Clustering Multiple Back Propagation Networks," *Advances in Neural Information Processing Systems 2*, D. Touretzky (Ed.), Morgan Kaufmann, 650–657.

[21] Mani, G. (1991) "Lowering Variance of Decisions by using Artificial Neural Network Ensembles," *Neural Computation*, **3**, 484–486.

[22] Meir, R. (1994) *Bias, Variance and the Combination of Estimators: The Case of Linear Least Squares*, Department of Electrical Engineering, Technion.

[23] Murphy, P. M. (1994) *UCI Repository of Machine Learning Databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[24] Omohundro, S. M. (1987) "Efficient Algorithms with Neural Network Behaviour," *Complex Systems*, **1**, 273–347.

[25] Perrone, M. P. (1993) *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*, PhD Thesis, Department of Physics, Brown University.

[26] Preparata, F. P., Shamos, M. I. (1985) *Computational Geometry*, Springer.

[27] Reignier, P., Hansen, V., Crowley, J. (1995) "Incremental Supervised Learning for Mobile Robot Reactive Control," *Intelligent Autonomous Systems*, U. Rembold et al. (Eds.), IOS Press, 287–294.

[28] Rogova, G. (1994) "Combining the Results of Several Neural Network Classifiers," *Neural Networks*, **7**, 777-781.

[29] Schaffer, C. (1994) "A conservation law for generalization performance," in *Proceedings of the Eleventh International Conference on Machine Learning*, 259–265, New Brunswick, NJ:Morgan Kaufmann.

[30] Silverman, B. W. (1986) *Density Estimation for Statistics and Data Analysis*, Chapman & Hall.

[31] Stanfill, C., Waltz, D. (1986) "Toward Memory-Based Reasoning," *Communications of the ACM*, **29**, 1213–1228.

[32] Tresp, V., Taniguchi, M. (1995) "Combining Estimators Using Non-Constant Weighting Functions," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, T. K. Leen (Eds.), MIT Press.

[33] Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer.

[34] Wolpert, D. H. (1992) "Stacked Generalization," *Neural Networks*, **5**, 241–259.

[35] Xu, L., Krzyżak, A., Suen, C. Y. (1992) "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, **22**, 418–435.

[36] Zhang, X., Mesirov, J. P., Waltz, D. L. (1992) "Hybrid System for Protein Secondary Structure Prediction," *Journal of Molecular Biology*, **225**, 1049–1063.