INCREMENTAL CONSTRUCTION OF COST-CONSCIOUS ENSEMBLES USING MULTIPLE LEARNERS AND REPRESENTATIONS IN MACHINE LEARNING

by

Mehmet Aydın Ulaş

B.S., in Computer Engineering, Boğaziçi University, 1999

M.S., in Computer Engineering, Boğaziçi University, 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2009

dedicated to my son Dağhan

"Hoş geldin oğlum"

ACKNOWLEDGEMENTS

I want to thank Prof. Ethem Alpaydın for his invaluable guidance and support in this thesis, and his contribution to my undergraduate and graduate education. I want to thank Prof. Lale Akarun and Prof. Günhan Dündar for their comments and feedback throughout the preparation of this thesis which greatly improved this work. I want to thank Prof. Muhittin Gökmen and Prof. Fikret Gürgen for their participation in my thesis jury and helpful comments; Mehmet Gönen, Murat Semerci and OT* for their technical contributions to this thesis and Berk Gökberk for providing me the face recognition data set. Without them, some part of this thesis would not be complete.

I am grateful to the inspiring atmosphere of our department and the joy of having dear friends; with their spiritual support, I had the motivation to complete this thesis. In particular, I want to thank Prof. Cem Ersoy, Oya Aran, Rabun Koşar, Burak Gürdağ, Atay Özgövde, İlker Demirkol, Itır Karaç, Burak Turhan, Ali Salah, Berk Gökberk, Gürkan Gür and Koray Balcı. I want to express my sincere gratitudes to Gülçin Gürdağ and Gamze Esen for their support for my wife during hard times, which is more difficult than completing this thesis.

I want to thank Jorge Cham for his excellent comics. I also want to thank the director, script writer and the cast of TV show Scrubs; especially Dr. Cox, TheTodd, Dr. Kelso, Jan Itor, Nervous Guy and Ted.

Finally, and most of all, I want to thank my parents Fadime and Nurettin, my brothers Aytekin and Sedat, my grandfather Yusuf, and my dear wife Özlem. Without their patience and support, this thesis would not be.

This work has been supported by Boğaziçi University Scientific Research Project 05HA101, Turkish Scientific Technical Research Council TÜBİTAK EEEAG 104E079 and the State Planning Organization of Turkey DPT/03K120250.

ABSTRACT

INCREMENTAL CONSTRUCTION OF COST-CONSCIOUS ENSEMBLES USING MULTIPLE LEARNERS AND REPRESENTATIONS IN MACHINE LEARNING

In this thesis, the main purpose is to combine multiple models to increase accuracy, while at the same time keeping a check on complexity. Towards this aim, we propose two methods, and these methods are tested by simulations using well-known classification algorithms on standard uni- and multi-representation data sets.

In the literature, methods have been proposed to create diverse classifiers. These methods change: (i) Algorithms used for training, (ii) Hyperparameters of the algorithms, (iii) Training set samples, (iv) Input feature subsets, and (v) Input representations. In this thesis, we show that these methods are not enough to decrease the correlations among base classifiers. Furthermore, we present the relation between error and correlation for fixed combination rules and a linear combiner, using three different cases. The cases are: (i) Independence, (ii) Equicorrelation, and (iii) Groups. We see that, the sum rule and the trained combiner show the most robust behavior to changes in correlation. Previous studies in the literature assume that the base classifiers are independent, the analysis in the presence of correlation, as presented in this thesis, is novel.

To remove the correlation between classifiers, we propose two algorithms to construct ensembles of multiple classifiers: (i) An incremental algorithm, named Icon which generates an ensemble of multiple models (representation/classifier pairs) to improve performance, taking into account both accuracy and the concomitant increase in cost, i.e., time and space complexity, and (ii) An algorithm which post-processes before fusing, using principal component analysis (Pca) and linear discriminant analysis

(LDA) to form uncorrelated metaclassifiers from a set of correlated experts.

ICON chooses a subset among correlated base classifiers. The algorithm has three dimensions: (i) Search direction (forward, backward, floating), (ii) Model evaluation criterion (accuracy, diversity and complexity), and (iii) Combination rule (fixed rules or a trained combiner). Our simulations using fourteen classifiers on thirty eight data sets show that, accuracy is the best model selection criteria and sum rule is the best combination rule. Other approaches create less preferred results compared to these two. There has been studies of subset selection in the literature, but the work in this thesis has a larger number of classifiers and data sets and its scope is wider. Using this method, we create ensembles which are more accurate than the single best algorithm and using all algorithms; and which are not worse than the optimal subset using smaller number of base classifiers. When applied to multi-representation data sets, we see that ICON automatically chooses classifiers which combine different representations and generates a set of complementary classifiers.

PCA which uses principal component analysis, and LDA which uses linear discriminant analysis create uncorrelated metaclassifiers from correlated base classifiers and these metaclassifiers are combined using a linear classifier. This method is successful with a small number of components and has the same accuracy as combining all classifiers. The work in this thesis allows generalization to multiple classifiers, combines multiple representations, allows knowledge extraction, and is novel in these respects. In this method, principal component analysis is more successful than linear discriminant analysis.

As the overall result, in comparing these two methods which get rid of correlation, we see that if the aim is to decrease complexity, then subset selection is better; if the aim is higher accuracy, we should prefer metaclassifiers which extract knowledge and has redundancy.

ÖZET

YAPAY ÖĞRENMEDE ÇOKLU ÖĞRENİCİ VE GÖSTERİMLERİ KULLANARAK MALİYET BİLİNÇLİ KÜMELERİN ARTIRIMLI OLUŞTURULMASI

Bu tezde, gözetimli öğrenmede birden çok modelin, sınıflandırma başarısını artıracak ve karmaşıklığı denetim altında tutacak bir şekilde birleştirilmesi amaçlanmıştır. Bunun için iki yöntem önerilmiş ve bilinen tek ve çok gösterimli veri kümeleri üzerinde, standart sınıflandırıcılar kullanılarak yapılan benzetimlerle bu yöntemler sınanmıştır.

Literatürde, birbirinden farklı sınıflandırıcılar üretmek için birçok yöntem önerilmiştir. Bunların arasında, (i) Farklı algoritmalar, (ii) Farklı üstparametreler, (iii) Farklı girdi altkümeleri, (iv) Farklı girdi gösterimleri ve (v) Öğrenme kümesinin farklı örneklemlerini sayabiliriz. Bu tezde, bu yöntemlerin sınıflandırıcılar arasındaki ilintiyi azaltmakta etkili olmadığını gösteriyoruz. Bunun yanında, ilinti ve hata arasındaki bağıntıyı ortaya koyarak, ilintinin üç değişik durumu için, sabit ve eğitilmiş birleştirme kurallarının hatalarının nasıl değiştiğini gösterdik. Bu durumlar: (i) Bağımsız sınıflandırıcılar, (ii) Eşilintili sınıflandırıcılar ve (iii) İlintili sınıflandırıcı gruplarıdır. Yapılan benzetimlerde, toplama kuralının ve eğitilmiş doğrusal birleştiricinin, ilintiye karşı en gürbüz davranışı gösterdiğini gözlemledik. Bu konuda yapılan önceki çalışmalarda sınıflandırıcıların bağımsız oldukları varsayılmıştır, ilintili olan durumdaki incelemeler bu çalışmaya özgündür.

Taban sınıflandırıcılar arasındaki ilintiyi kaldırmak için iki algoritma öneriyoruz. Bunlar: (i) Başarıyı artırırken aynı zamanda maliyeti, yani zaman ve bellek karmaşıklığını da göz önünde tutan, ICON isimli, artırımlı bir birleşik sınıflandırıcı oluşturma algoritması ve (ii) Birleştirmeden önce ana bileşenler analizi ya da doğrusal ayırtaç

analizi yardımıyla ardıl işlem yaparak ilintisiz üstsınıflandırıcılar üreten bir algoritmadır.

ICON algoritması ilintili sınıflandırıcılar arasından altküme seçmektedir. Algoritmanın üç boyutu vardır: (i) Arama yönü (ileri, geri, kayan), (ii) Model değerlendirme ölçütü (başarı, çeşitlilik ve model karmaşıklığı) ve (iii) Birleştirme kuralı (sabit kurallar, eğitilmiş doğrusal birleştirici). Otuz sekiz veri kümesi üzerinde, on dört sınıflandırıcı kullanılarak yapılan benzetimlerde, model seçme ölçütü olarak başarının ve birleştirme kuralı olarak da toplama kuralının en iyi olduğu sonucuna varılmıştır. Diğer yaklaşımlar bu iki seçeneğe göre daha az yeğlenir sonuçlar vermektedir. Bilimsel yazında daha önce de altküme seçme çalışmaları yapılmıştır, ama bu tezdeki çalışma diğer çalışmalara göre, kapsam, veri kümesi ve sınıflandırıcı sayısı açısından daha geniştir. Bu yöntem kullanılarak, en iyi taban sınıflandırıcıdan ve tüm sınıflandırıcıları kullanmaktan daha başarılı sonuçlara ulaşılmış, en iyi altkümeden ise daha kötü olmayan fakat daha basit olan birleşik sınıflandırıcılar üretilmiştir. Çok gösterimli veri kümelerine uygulandığında, ICON'un otomatik olarak farklı gösterimlerle eğitilmiş ve birbirini tamamlayan sınıflandırıcılar seçtiğini gözlemledik.

İlintili sınıflandırıcıların çıktılarını ilintisiz hale getirmek için temel bileşenler analizi kullanan PCA ve doğrusal ayırtaç analizi kullanan LDA algoritmaları ilintisiz üstsınıflandırıcılar oluşturmakta ve bu üstsınıflandırıcılar, doğrusal sınıflandırıcı kullanılarak birleştirilmektedirler. Az sayıda üstsınıflandırıcı, bu yöntemin başarılı olması için yeterli olmaktadır. Bu tezde yapılan çalışma, çok sayıda sınıfa genelleştirilebildiği, çok gösterimli veri kümelerine uygulanabildiği ve bilgi özütleyerek sonuçların yorumlanabilmesini sağladığı için yeni bir çalışmadır. Bu yöntemde, temel bileşenler analizi, doğrusal ayırtaç analizine göre daha başarılı olmuştur.

Genel sonuç olarak, ilintiyi ortadan kaldırmak için kullanılan bu iki yöntemin karşılaştırılmasında, eğer amaç karmaşıklığı azaltmak ise, altküme seçmenin daha iyi olduğu, başarının daha önemli olduğu durumlar içinse öznitelik çıkaran üstsınıflandırıcıların kullanılmasının daha öne çıktığı görülmüştür.

LIST OF SYMBOLS/ABBREVIATIONS

C_k	kth class
d	Number of free parameters
D	Input dimension
e	Error
E	Ensemble
K	Number of classes
l	Log-likelihood
L	Number of classifiers
M_i	Classifier i
N	Number of instances in the data set
$P(C_k\|\boldsymbol{x})$	Posterior probability of class C_k given input $\boldsymbol{x}$
$P(C_k\|\boldsymbol{x}, M)$	$P(C_k\|\boldsymbol{x})$ estimated by classifier M
Q	Q-statistics
S	Number of components of PCA
$\boldsymbol{x}$	Input vector
$\mathcal{X}$	Input data set
α	Confidence value
κ	A diversity measure
λ	Coefficient of the cost term
μ	Average error
ρ	Correlation coefficient

ACC	ICON variant using accuracy
AIC	Akaike's Information Criterion
BEST	Combination of the single classifiers having best accuracy
BIC	Bayesian Information Criterion
CORR	ICON variant using correlation coefficient
CV	ICON variant using cross-validation
FP	False Positive

ICON	Incremental Ensemble CONstruction
knn	k-nearest neighbor
LDA	Linear Discriminant Analysis
LDA	Metaclassifier constructed using Linear Discriminant Analysis
LP	Linear Perceptron
MAX	Max rule for classifier combination
MDL	Minimum Description Length
MDL	ICON variant using minimum description length
MDT	Meta Decision Tree
MED	Median rule for classifier combination
MIN	Min rule for classifier combination
mlp	Multilayer Perceptron
MLR	Multiresponse Linear Regression
OPT	Subset with optimum validation accuracy
PCA	Principal Component Analysis
PCA	Metaclassifier constructed using PCA
PRO	Product rule for classifier combination
QSTAT	ICON variant using Q-statistic
RND	Combination of classifiers selected randomly
SUM	Sum rule for classifier combination
svm	Support Vector Machine
TP	True Positive

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Given a D-dimensional input $\boldsymbol{x}$, the purpose of a classification algorithm is to assign this instance to one of K classes: $C_1, \ldots, C_K$. A classification algorithm first builds a classifier given N training instances, then tries to estimate the class of a newly presented example using this classifier. We can divide classification into two basic approaches: generative and discriminative. Generative models try to estimate posterior probability densities, $P(C_k|\boldsymbol{x})$, and the unknown class according to these probability estimates, whereas discriminative models learn a discriminant function from the data. A classification algorithm tries to find a decision function which minimizes the error on new data [1].

An example of a classification problem with two classes can be seen in Figure 1.1. This is a two class problem, and the classes are represented with three and two Gaussians respectively. The dotted black line shows the Bayes optimal discriminant that separates these classes. We show the discriminants of three example classifiers: (i) Linear perceptron, (ii) Support vector machine with radial kernel, and (iii) k-nearest neighbor with $k = 5$.

It is well-known that there is no single classification algorithm that is always the most accurate and various methods have been proposed to combine classifiers based on different learning algorithms [2]. Each algorithm has a different inductive bias, that is, makes a different assumption about the data and errs on different instances and by suitable combination, the overall error can be decreased. Hence, research has focused on combining these algorithms for better classification performance. Model combination however is no panacea and models in the ensemble should be carefully chosen for error to decrease. In particular, model combination through averaging reduces variance [3], and hence error, but only if bias does not increase in the process, or if the concomitant increase in bias is small with respect to the decrease in variance. It is therefore essential that only those models that contribute to accuracy are added and the poorly performing ones are weeded out. Besides its effect on statistical accuracy,

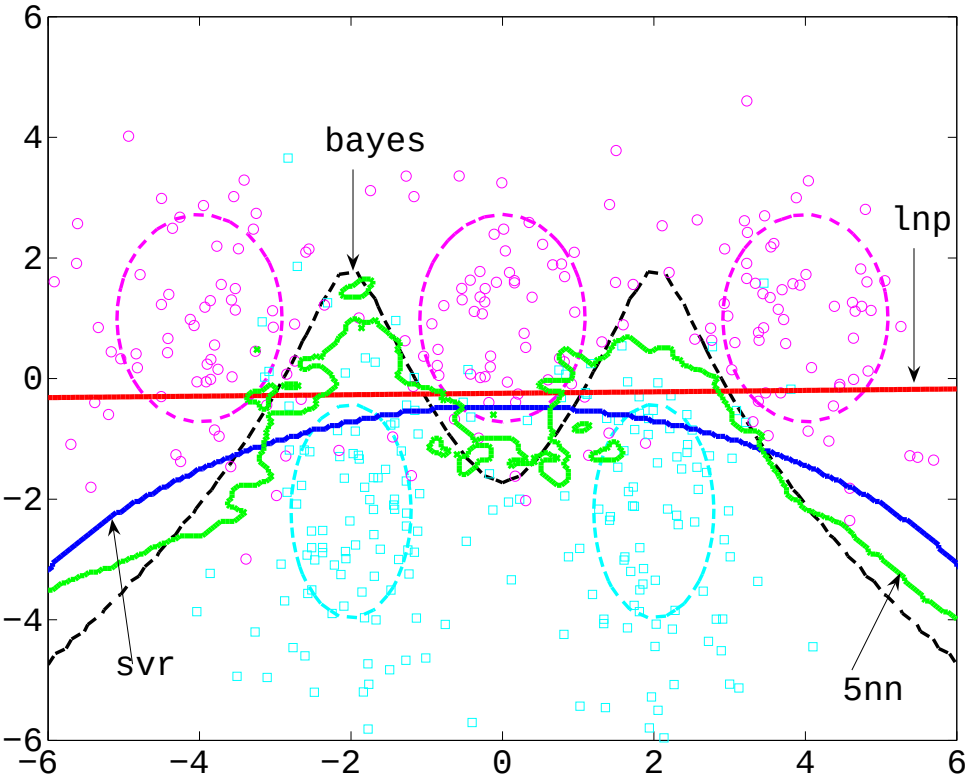


Figure 1.1. Two class problem with mixtures of Gaussians. Dotted line is the Bayes optimal discriminant. The other lines show the discriminants learned by a linear perceptron (*lnp*), 5-nearest neighbor (*5nn*) and support vector machine with a Gaussian kernel (*svr*)

each additional model increases the space and computational complexity. A new model may also be sensing/extracting a costly representation which can be saved if the model is considered redundant.

When one tries to fit a model on data, one should predict both the structure of the model and the parameters of the model. Most machine learning algorithms can calculate the parameters of a model given a structure, but selecting the model is carried out by the user (system designer) and so is subjective. In this thesis our aim is to combine multiple models to reach better accuracy, and at the same time, we consider the cost of the ensemble. It is also known that different representations (from different sensors) of the same object/event make different characteristics apparent and fusing these different representations improve accuracy [4]. Though combining multiple models—we define a *model* as a pair of *representation* and *classifier* induced by a learning algorithm—may improve accuracy, there is also the additional cost of processing, i.e., the time and space (memory) complexity of the classifier using the representation. Existing model combination algorithms aim to improve accuracy without worrying about the cost of using the ensemble.

In *Multiexpert* methods, learners work in parallel on some data and a model combination method combines the outcomes of these algorithms to come up with the final decision. *Multistage* methods use a serial approach, where each learner uses data formed after the previous learner has been trained. The overall classifier is called an *ensemble* classifier which is composed of component classifiers. It is possible to use crisp (0/1) outputs of base classifiers, or use soft outputs (it is safe to assume that these values are between 0 and 1) for classifier combination. In this thesis, we use the posterior probabilities of base classifiers for combination and evaluation. Throughout the thesis, the words "learner", "classifier" and "expert" are used interchangeably. In the following section, we first explore in brief the widely used combination strategies; we then continue with a chronological examination of them in more detail.

Besides ensemble construction methods, there is another type of ensemble forming strategy, which we do not discuss further in this thesis. These are selection methods

which choose a classifier (or a few classifiers) from a larger set *dynamically* for each input instance [5, 6]. These methods are akin to mixture of experts [7] and use a neighborhood measure to divide the input space into regions of expertise of the different base classifiers.

1.1. Classifier Combination

1.1.1. Notation

Let $P(C_k|\boldsymbol{x})$ denote the true posterior probability of class C_k given instance $\boldsymbol{x}$, and $P(C_k|\boldsymbol{x}, M)$ denote the posterior probability of class C_k estimated by classifier M. The Bayes optimal class is then calculated by $\text{argmax}_k\, P(C_k|\boldsymbol{x})$ and the estimated class for classifier M is calculated by $\text{argmax}_k\, P(C_k|\boldsymbol{x}, M)$.

1.1.2. Combination Methods

In the literature, there are several methods for calculating the overall output from the outputs of learners in an ensemble (committee). The simple method is to use voting [8] which corresponds to fixed rules. Suppose that the ensemble is denoted by E and we have L base classifiers $M_i, i = 1 \ldots L$. The fixed rules used in the literature are given in Table 1.1 [8]:

Table 1.1. Fixed rules and the trained linear rule in classifier combination

SUM: $P(C_k|\boldsymbol{x}, E) = \sum_{i=1}^{L} P(C_k|\boldsymbol{x}, M_i)$
MAX: $P(C_k|\boldsymbol{x}, E) = \max_i P(C_k|\boldsymbol{x}, M_i)$
MIN: $P(C_k|\boldsymbol{x}, E) = \min_i P(C_k|\boldsymbol{x}, M_i)$
MED: $P(C_k|\boldsymbol{x}, E) = \text{median}\{P(C_k|\boldsymbol{x}, M_1), \ldots, P(C_k|\boldsymbol{x}, M_L)\}$
PRO: $P(C_k|\boldsymbol{x}, E) = \prod_{i=1}^{L} P(C_k|\boldsymbol{x}, M_i)$
LIN: $P(C_k|\boldsymbol{x}, E) = \sum_{i=1}^{L} w_i P(C_k|\boldsymbol{x}, M_i) + w_0$

We can see the results of combination rules on our toy problem in Figure 1.2. Since MAX and MIN rules are the same on two class data sets, one of them is shown.

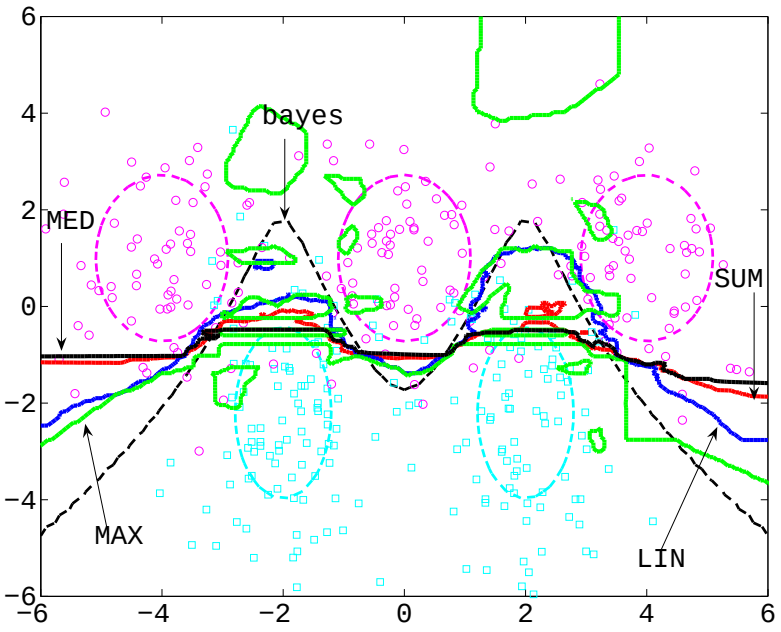


Figure 1.2. Discriminants found by the combination rules (Table 1.1) on the same problem of Figure 1.1

Voting or sum rule takes an average over multiple models and can be used to average over hyperparameters, for example, the initial weights of a multilayer perceptron (*mlp*). In stacking [9], the individual classifiers are combined using a second layer classifier which estimates the real output from the outputs of individual classifiers. In our notation, this corresponds to:

$$P(C_k|\boldsymbol{x}, E) = f(P(C_k|M_1, \boldsymbol{x}), \ldots, P(C_k|M_L, \boldsymbol{x})),$$

where f denotes another classification algorithm which is called L_1 in stacking (We use a linear combiner for stacking purposes in this thesis).

Methods based on resampling from a single data set, such as bagging [3] and AdaBoost [10] are not used to combine multiple representations. Bagging [3] is the abbreviation of **B***ootstrap* **Agg***regat***ing**. As can be deduced from the name, the basic idea is to generate L data sets from a given data set by sampling with replacement. Then a majority vote over the L learners is the final prediction. The vital element of the

algorithm is the instability of the learner. If perturbing the training set can change the outcome of the learner significantly, then bagging can improve accuracy. The simple idea of AdaBoost is to find instances that are incorrectly classified in one iteration and give them a higher probability to be selected in the next iteration. Freund and Schapire [10] compare performances of bagging and AdaBoost. A short introduction to boosting and its relationship with support vector machines is given in [11]. In a mixture of experts architecture, models are local and a separate gating network selects one of the local experts based on the input [7].

Breiman constructs another method called arcing in [12]. The methods work over unstable learners whose accuracy can change dramatically when perturbed. The basic idea of arcing and bagging is to reduce variance. According to Breiman [12], arcing is better than bagging; like bagging, arcing also constructs L classifiers but $(L'+1)$st classifier depends on the previously trained L' classifiers. In [13], Breiman defines a prediction game and defines arcing algorithms as finding good game strategies.

The advantage of using multiple representations is shown in [4, 14] where for handwritten digit recognition, combining two representations, leads to more accuracy than any of the alone. Gökberk [15] et al. show that using inputs from different face modalities achieves better accuracy. Alpaydın [16] presents methods for combining multiple representations and learners. *Uni-representation* and *Multi-Representation* of a data set has been considered. If uni-representation is used, different learners should be used. If multi-representation is used, either the sensors are different or different features have been extracted from the same data.

Cascade Generalization, a method for combining multiple classifiers is introduced in [17]. It is an iterative composition of classifiers. At each stage a new classifier is generated. The input space is extended by adding new attributes. The new attributes are obtained by the generated base classifier. Suppose that there are K classes. The second level training set is constructed using the original training set with K more attributes which are the posterior probabilities of each class, $P(C_i|\boldsymbol{x})$ according to the first classifier. This is a two step procedure but it can be extended sequentially and

in parallel as stated in [17]. In [18], Gama extends this idea locally, in other words applying the idea at each iteration of a *divide and conquer algorithm.* At each iteration of the algorithm, the input space is reconstructed by adding new attributes.

Alpaydın [19] trains multiple condensed nearest neighbors and takes a vote over them which has accuracy higher than running nearest neighbor algorithm on the whole, uncondensed data [19].

Bay proposes to use voting over neural network classifiers who operate on different subsets of features of data [20]. This method is similar to the random subspace method [21].

Merz et al. [22] propose a new combining algorithm called *SCANN* based on *Correspondance analysis* and stacking. The algorithm focuses on building a new representation from the outputs of base classifiers and combining them. The new representation is captured in a space of uncorrelated dimensions. The nearest mean algorithm is then used within the resulting representation.

Bauer and Kohavi [23] give an empirical comparison of ensemble algorithms (bagging, boosting, and variants). The paper works on the question as to when and why these algorithms, which use perturbation, reweighting and combination techniques affect classification error.

A multistage learner, built as a cascade of multi-layer perceptron (*mlp*) and *knn* (*k*-nearest neighbor) is proposed in [24]. *mlp* is a distributed algorithm which learns the "*rule*" (globally) and *knn* is a local algorithm which catches and learns the "*exceptions*" of *mlp*. In [25], Kaynak and Alpaydın present results with the cascading algorithm. The main advantage of cascading algorithm over other methods is that not all of the classifiers are trained with the whole data; this reduces complexity and cost. Since complex learners are trained with the exceptions only, the amount of data for training the next level learner decreases, which reduces complexity. In [26], the cascading algorithm is carried one step further in constructing a new algorithm called *REx* (Standing

for Rules and EXceptions). In the cascading algorithm, voting is used to combine learners. In REx, the combiner is also trained. Given any input $\boldsymbol{x}$, if the maximum posterior probability $P(C_i|\boldsymbol{x})$ is greater than a certain threshold θ $(0 < \theta < 1)$ then $\boldsymbol{x}$ is said to be covered by the rule, else $\boldsymbol{x}$ is called an exception. The relationship of *cascading* with voting, stacking, mixture of experts and boosting is also discussed.

An experimental comparison for constructing ensembles of decision trees is given in [27]. The methods inspected are bagging, boosting and randomization; see also [28] for a review of ensemble methods for combining classifiers.

Grading [29] is an algorithm which tries to correct the incorrect predictions of the base level classifiers. For each base level classifier, a meta level classifier is trained, whose job is to detect when the base level classifier will make an error. The final prediction is constructed by voting the results of base-level classifiers which are predicted to be correct by the meta classifier. Ženko et al. [30] use Meta Decision Trees (MDT) in a stacking framework. MDTs differ from ordinary decision trees in that the leaves of MDT specify the base level classifier to be used instead of the classification result. Rahman and Fairhurst's paper [31] is an excellent review on classifier combination.

1.2. Stacking

Among all the combination techniques mentioned above, stacking (stacked generalization) [9] deserves a detailed consideration since most of the other algorithms are based on constructing the ensemble, whereas stacking concentrates on combining previously trained base classifiers (which is also our focus in this thesis). Stacked generalization can be used as a technique for combining classifiers but it is also useful when you want to improve the accuracy of a single classifier. On the other hand, stacked generalization can also be viewed as a means of collectively using the ouputs of all classifiers to estimate their own generalizing biases with respect to a particular learning set, and then filter out those biases [9]. Figure 1.3 shows the structure of stacking.

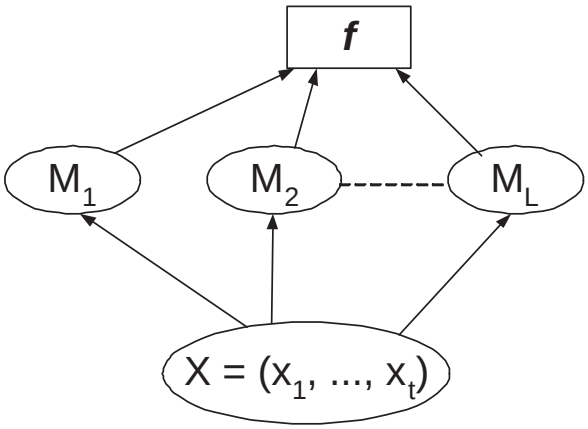


Figure 1.3. Stacking. $\boldsymbol{x}$ is the input, $M_1, M_2, \ldots, M_L$ are the base classifiers and f is the combination function

Above methods either use too many base classifiers, or concentrate on forming the ensemble, or use all base classifiers in the given ensemble. Furthermore, they do not consider the cost of the ensemble, and most of them are not suitable for combining multiple representations. It is infeasible to use bagging and boosting with complex base classifiers, and if there are too many base classifiers, stacking may overlearn. So research has focused on either selecting a subset of classifiers to reduce complexity (and in the same time increase accuracy), or using other means of measures (such as diversity) to select which classifiers should be included in the final ensemble. A detailed discussion of such methods can be found in Section 3.6.

In this thesis, first, we analyze theoretically the accuracy of fixed fusion rules, where we assume that L classifier outputs are being drawn from a L-variate Gaussian. We see that when we have positive correlation between experts in the ensemble, the overall ensemble accuracy decreases. Highest accuracy is achieved when the experts are negatively correlated or there is a group of classifiers which are negatively correlated with the others. We then investigate the effect of five factors on the correlation using 14 classifiers and 38 data sets, and we see that whatever we do, we still have positive correlation.

To solve this problem, we first propose a new incremental algorithm that chooses a subset of models from a large number of possible models (representation and classifier

pair), to maximize a combined performance measure of accuracy and complexity; we therefore check for diversity and cost at the same time. This is therefore a model selection problem where models are added to the ensemble only if their additional complexity is justified by their contribution to accuracy. We investigate various versions of the algorithm using different model selection criteria such as CV, ACC, CORR, QSTAT, and MDL [32].

Second, we use PCA and LDA on the outputs of the experts in the ensemble to construct new aggregate dimensions that are linear combinations of the original features, which we call metaclassifiers. Our subset selection algorithm ICON, on 38 uni-representation and 2 multi-representation data sets, show that combining a few complementary base classifiers is better than the single best algorithm and using all classifiers in the ensemble, and is not worse than the optimum subset. Also we see that combining classifiers from different representations leads to better generalization accuracy. PCA, which post-processes expert outputs is able to extract knowledge and achieve accuracy as high as combining all using a small number of metaclassifiers.

The thesis is organized as follows: We analyze the correlations between experts in Chapter 2, we introduce our proposed algorithm ICON in Chapter 3 and show our experiments on 38 data sets using 14 classifiers. In Chapter 4, we present experiments on two multi-representation data sets, and show the advantage of using base classifiers from different representations. We show our results using PCA in Chapter 5. We conclude and discuss future work in Chapter 6.

2. ANALYSIS OF CORRELATION BETWEEN EXPERTS IN AN ENSEMBLE

In this chapter, we will analyse the error of fixed rules theoretically and show how this analysis applies to real world data sets.

2.1. Analysis of Fusion Rules

Let us say that we have L experts (learners, base classifiers), with their outputs $d_j, j = 1, \ldots, L$, estimating some unknown parameter θ, for example, the posterior probability of a class for input x in a classification problem: $P(C_i|x)$[1] . Let us also say that our combined estimator d to θ is the simple average:

$$d = \frac{\sum_{j=1}^{L} d_j}{L}$$

Most fusion methods are variants of simple averaging, for example, with unequal weights, so the following derivation is general. It is known that the mean squared error of d in estimating θ can be written as the sum of squared bias and variance:

$$\begin{aligned} E[(d-\theta)^2] &= (E[d]-\theta)^2 + E[(d-E[d])^2] \\ &= \text{Bias}^2(d) + \text{Var}(d) \end{aligned}$$

In the case of a simple average, bias is just the average bias:

$$E[d] - \theta = \frac{\sum_{j=1}^{L}(E[d_j]-\theta)}{L}$$

[1]The work explained in this chapter is joint work with Murat Semerci.

In stacking [9], a second layer learner is trained to combine the outputs of the given classifiers and therefore also corrects for their bias, but in ensemble methods, the decrease in error is mostly due to the decrease in variance.

We can write the variance as:

$$\mathrm{Var}(d) = \frac{1}{L^2}\sum_j \mathrm{Var}(d_j) + \frac{1}{L^2}\sum_i \sum_{j\neq i} \mathrm{Cov}(d_i, d_j) \tag{2.1}$$

If d_j are independent, then their correlation is 0 and the second, covariance term in Equation 2.1 disappears. In such a case, variance decreases as L is increased. Indeed, most combination methods aim to generate uncorrelated experts, and it has been proposed [2] to use different (i) learning algorithms, (ii) hyperparameters, (iii) input features, (iv) training sets, (v) different representations. For example, bagging [3] uses bootstrapping to generate slightly different training sets and takes an average for fusion. Random subspace method [21] trains different experts with different subsets of a given feature set. Different representations of the same input make different characteristics explicit and therefore accuracy may be improved by combination [4, 14]. Fusion rules (i.e, average, vote, product, min, max, median) have mostly been analyzed under the assumption that the expert decisions are independent [8, 33, 34].

Normally, when we have a number of experts trained on the same data, we expect them to be positively correlated, i.e., they will be correct on the same instances and fail on the same difficult (noisy) instances. Looking at Equation 2.1, we expect then the variance (and hence the error) to increase as we increase the number of experts. It is therefore critical that any positive correlation between the experts should be found and taken care of.

Examining Equation 2.1, we see that in minimizing variance, even better than the case of uncorrelated experts would be the case when we have negatively correlated experts [35]. Note however that negative correlation may cause an increase in bias; for example, for the case of mixture of experts, it has been shown that experts which are

localized in different parts of the input space are negatively correlated but biased [36]. Comparing AdaBoost [10] with bagging, we can say that experts trained on previous expert's errors help in constructing negatively correlated experts.

Given a set of positively correlated experts, one line of research is in the direction of finding a minimal subset. To help us in finding those experts which are redundant, "diversity" measures have been proposed [37, 38] and one possibility is to have an incremental, forward search where we add a classifier to an ensemble if it is diverse or adds to accuracy [39, 14, 40], or another possibility is to have a decremental, backward search where a classifier is removed or pruned if it is not diverse enough or if its removal does not increase error [41, 42].

The fusion rules can be seen as a function operating on a vector of L classifier estimates: $\hat{P}_1(\boldsymbol{x}) = \mathcal{F}(\hat{\boldsymbol{p}})$

where $\boldsymbol{p} = [\hat{p_1}, \hat{p_2}, \ldots, \hat{p_L}]^T$ and $\hat{p_i} \equiv \hat{P}_1^i(\boldsymbol{x})$. In this section, we are going to model the L classifier outputs as being drawn from a L-variate Gaussian: $\hat{\boldsymbol{p}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with

$$\boldsymbol{\mu} = [p_0, p_0, \ldots, p_0]^T$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \ldots & \sigma_{1L} \\ \sigma_{21} & \sigma_2^2 & \ldots & \sigma_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{L1} & \sigma_{L2} & \ldots & \sigma_L^2 \end{pmatrix} \tag{2.2}$$

This corresponds to assuming that the classifier outputs are distorted versions of the ideal posterior p_0 where σ_i^2 is the variance of this distortion for classifier i and σ_{ij} is the covariance between classifiers i and j. The case of independent Gaussian errors used in previous studies [33, 43] correspond to a special case where the off-diagonals of $\boldsymbol{\Sigma}$ are 0 and the diagonals are equal: When the dimensions are independent, the multivariate reduces to a product of univariates.

2.1.1. Fusion Rules on Equicorrelated Classifier Ensembles

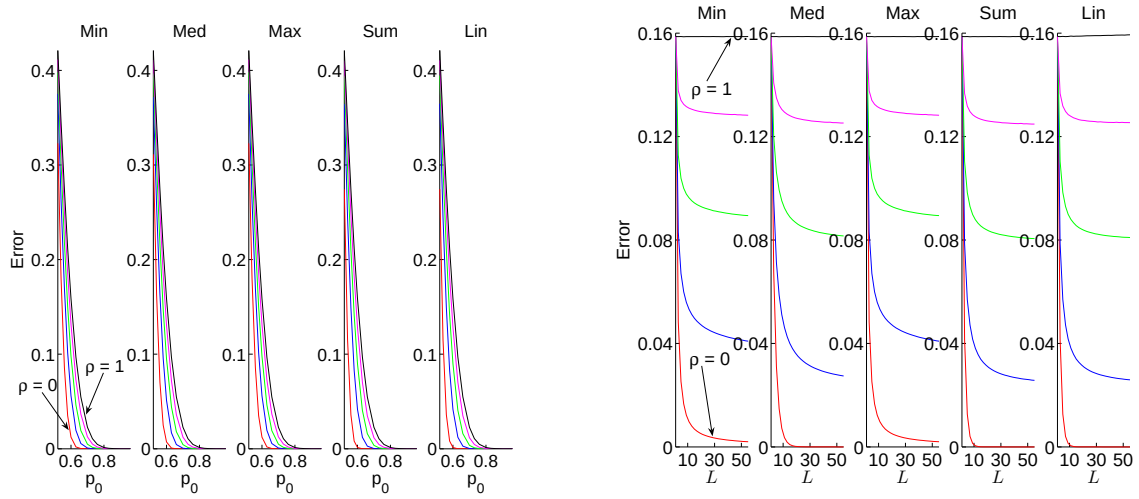
Let us compare the error rates of fusion rules starting with the case of equicorrelated experts: Given the correlation matrix $\boldsymbol{R}$, $\boldsymbol{R}_{ii} = 1, \forall i$ and $\boldsymbol{R}_{ij} = \rho, \forall i \neq j$. That is, all experts are dependent in the same way. An example of this would be when we train the same learning algorithm over bootstraps of a training set: p_0 would be the average accuracy and ρ would depend on the "stability" of the learning algorithm (see Section 2.3.1.3).

We know that error increases as correlation increases. The reason is that the classifiers tend to resemble each other more with increased correlation, and when the correlation becomes 1, all the classifiers are the same and we converge to the average error rate of a single classifier.

Figure 2.1(a) depicts the effect of the base posterior value p_0. Of course, the higher it is (away from 0.5), the less is the error. We see that at lower correlation, average rule is the best, the median is the second and the minimum and maximum rules perform the worst. We see that all the rules show the same error characteristic curve when correlation is 1.

In Figure 2.1(b), we observe that as long as the correlation is below 1, adding a new classifier to the ensemble improves the ensemble performance. If the classifiers are almost independent ($\rho \approx 0$), it is possible to have 0 error with a moderate size ensemble. But the errors of the minimum and maximum rules do not converge to 0, because their tails do not shrink much as new classifiers are added. It can be seen that even with considerable correlation between classifiers, the average and median rules perform better than the minimum and maximum rules. It can also be seen that the error rate decreases and then levels off; given a certain amount of correlation, new experts do not bring much new information.

In Figure 2.2, we see that using negatively correlated classifiers (if possible) improves the ensemble accuracy further. It must be kept in mind that ρ must satisfy



(a) Base posterior p_0 (b) Ensemble size L

Figure 2.1. Effect of correlation on different fusion rules as a function of (a) p_0 ($L = 9$) and (b) L ($p_0 = 0.6$), for the case of equicorrelated experts, for $\rho = 0, 0.25, 0.5, 0.75$ and 1

$\rho \geq -\frac{1}{L-1}$ for semi-definiteness. The case of negative correlation (keeping the same average error) means that the classifiers tend to give estimates opposite of each other, err on distinct instances, which can be interpreted as a sign of diversity. Still, whenever one or more classifiers give inaccurate decisions, the remaining ones (if they are in majority) can correct the ensemble decision.

2.1.2. Effect of Multiple Groups

We now consider how the performance of the ensemble is influenced if the classifiers in the ensemble can be grouped into N separate groups. This is what one would expect if in the ensemble we have variants of N algorithms which have certain intragroup correlation and which also have some intergroup correlation with variants of other algorithms. For example, one group can be support vector machine variants for different kernels, one group may be multilayer perceptron variants for different number of hidden units and we also expect to see some correlation between any support vector machine and multilayer perceptron variant (see Table 2.2 in Section 2.3.1).

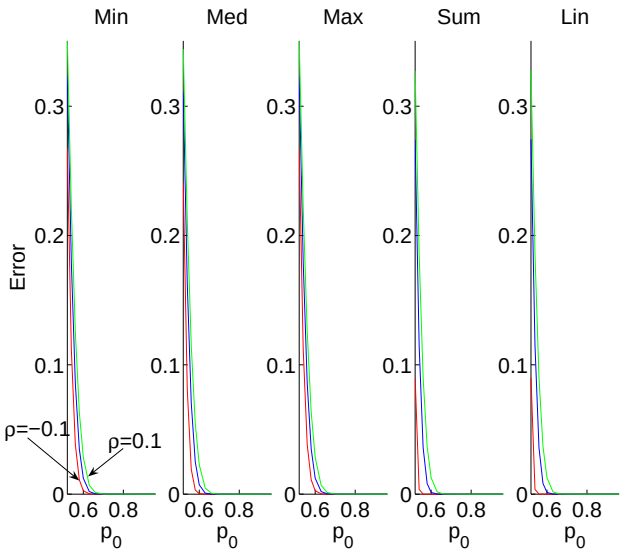


Figure 2.2. Effect of correlation on different fusion rules as a function of p_0 for the case of equicorrelated experts, for $\rho = -0.1, 0.0$ and 0.1, with $L = 9, \sigma = 0.1$

$$\boldsymbol{R} = \left(\begin{array}{c|c|c|c} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} & \dots & \boldsymbol{R}_{1N} \\ \hline \boldsymbol{R}_{21} & \boldsymbol{R}_{22} & \dots & \boldsymbol{R}_{2N} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \boldsymbol{R}_{N1} & \boldsymbol{R}_{N2} & \dots & \boldsymbol{R}_{NN} \end{array}\right)$$

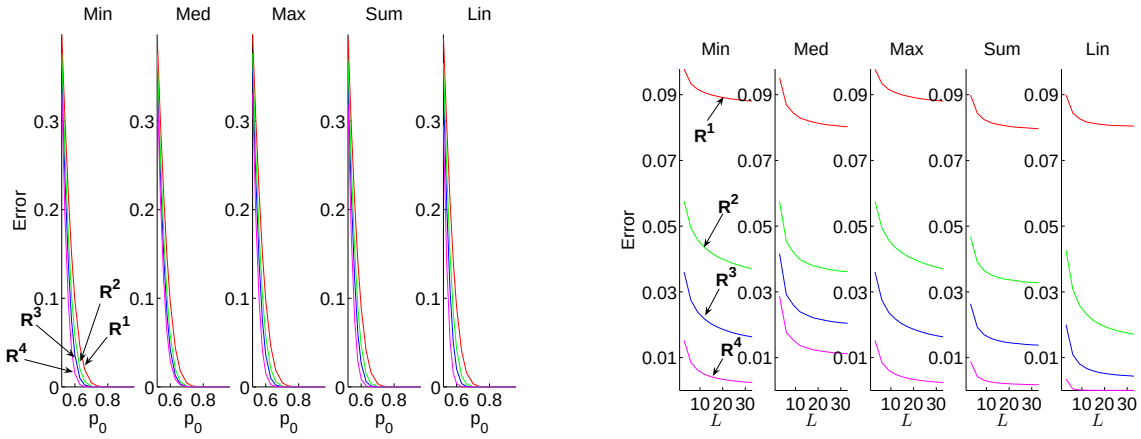
Suppose that we have N groups having L_i $(i = 1 \dots N)$ classifiers for each group where $\boldsymbol{R}_{ii}$ is the intragroup correlations matrix between elements of group i and $\boldsymbol{R}_{ij}$ is the intergroup correlation matrix between groups i and j. As a case study, we will present our results for three groups on four example cases; for simplicity we assume a single value for these matrices.

$$\boldsymbol{R}^1 = \left(\begin{array}{c|c|c} 0.5 & 0.5 & 0.5 \\ \hline 0.5 & 0.5 & 0.5 \\ \hline 0.5 & 0.5 & 0.5 \end{array}\right) \boldsymbol{R}^2 = \left(\begin{array}{c|c|c} 0.5 & 0.2 & 0.3 \\ \hline 0.2 & 0.2 & 0.2 \\ \hline 0.3 & 0.2 & 0.5 \end{array}\right)$$

$$\boldsymbol{R}^3 = \left(\begin{array}{c|c|c} 0.5 & 0 & 0.3 \\ \hline 0 & 0.2 & 0 \\ \hline 0.3 & 0 & 0.5 \end{array}\right) \boldsymbol{R}^4 = \left(\begin{array}{c|c|c} 0.5 & -0.2 & 0.3 \\ \hline -0.2 & 0.2 & -0.2 \\ \hline 0.3 & -0.2 & 0.5 \end{array}\right)$$

The first case, $\boldsymbol{R}^1$, consists of three groups with equal correlations; this is the case of a single group of equicorrelated classifiers we discussed above, which we include for comparison. For the remaining three cases, $\boldsymbol{R}^2$, $\boldsymbol{R}^3$, and $\boldsymbol{R}^4$, we have three groups with different intergroup correlations between group 2 and the other groups as positive, zero, and negative. In the first experiment, we have three classifiers from each group (total of nine classifiers) and we change the base posterior probability p_0 to see the effect of the correlations between groups of classifiers. We see in Figure 2.3(a) that when we have negative correlation among groups, the error rate is the smallest, and it decreases rapidly. We have more error when there is no intergroup correlation and even more error when this correlation is positive.

In this setup, we also test the effect of number of classifiers per group. We see in Figure 2.3(b) the same behavior: $\boldsymbol{R}^4$ with negative intergroup correlation has the lowest error with increasing ensemble size. We observe another interesting fact in this figure: Though a trained linear combiner was no better than the fixed average or median rules in the previous cases, for these cases of grouped ensembles with different intergroup correlations, a linear combiner achieves lowest error using smaller ensembles.



(a) Base posterior p_0 (b) Ensemble size L

Figure 2.3. Effect of intragroup correlation on the fusion rules for the four cases of $\boldsymbol{R}^1$, $\boldsymbol{R}^2$, $\boldsymbol{R}^3$, and $\boldsymbol{R}^4$

2.2. Experimental Details

In this section, we define the experimental methodology; namely the classifiers used, the data sets used and measures of significance used to compare ensemble methods over multiple data sets.

2.2.1. Data Sets

We use 38 data sets from the UCI machine learning repository [44], Delve [45]. Table 2.1 shows the properties of data sets.

2.2.2. Base Classifiers

We use fourteen base classifiers which we have chosen to span as much as possible the wide spectrum of possible machine learning algorithms[2] :

- (1–3) *knn*: k-nearest neighbor with $k = 1, 3, 5$.
- (4–8) *mlp*: Multilayer perceptron where with D inputs and K classes, the number

[2]Base classifiers have been trained using ISELL machine learning toolbox by Olcay Taner Yıldız.

Table 2.1. Properties of the data sets

name	instances	inputs	classes	source
zoo	101	16	7	uci
iris	150	4	3	uci
tae	151	5	3	uci
hepatitis	155	19	2	uci
wine	178	13	3	uci
flags	194	26	8	uci
glass	214	9	6	uci
heart	270	13	2	uci
haberman	306	3	2	uci
flare	323	10	3	uci
ecoli	336	7	8	uci
bupa	345	6	2	uci
ionosphere	351	34	2	uci
dermatology	366	34	6	uci
horse	368	26	2	uci
monks	432	6	2	uci
vote	435	16	2	uci
cylinder	540	45	2	uci
balance	625	4	3	uci
australian	690	14	2	uci
credit	690	15	2	uci
breast	699	9	2	uci
pima	768	8	2	uci
tictactoe	958	9	2	uci
cmc	1473	9	3	uci
yeast	1484	8	10	uci
car	1728	6	4	uci
titanic	2201	3	2	delve
segment	2310	19	7	uci
thyroid	2800	27	4	uci
optdigits	3823	64	10	uci
spambase	4601	57	2	uci
pageblock	5473	10	5	uci
ringnorm	7400	20	2	delve
twonorm	7400	20	2	delve
pendigits	7494	16	10	uci
mushroom	8124	22	2	uci
nursery	12960	8	4	uci

of hidden units is taken as D (*ml1*), K (*ml2*), $(D+K)/2$ (*ml3*), $D+K$ (*ml4*), $2(D+K)$ (*ml5*).

- (9) *lnp*: Linear perceptron with softmax outputs trained by gradient-descent to minimize cross-entropy.
- (10) *c45*: The most widely-used C4.5 decision tree algorithm.
- (11) *mdt*: This is a multivariate tree where unlike C4.5 which uses univariate and axis-orthogonal splits uses splits that are arbitrary hyperplanes using all inputs [46].
- (12–14) *svm*: Support vector machines with a linear kernel (*svl*), polynomial kernel of degree 2 (*sv2*), and a radial (Gaussian) kernel (*svr*). We use the LIBSVM 2.82 library that implements pairwise *svm*s [47].

2.2.3. Division of Training, Validation, and Test Sets

Our methodology is as follows: A given data set is first divided into two parts, with 1/3 as the test set, *test*, and 2/3 as the training set, *train-all*. The training set, *train-all*, is then resampled using 5×2 cross-validation (cv) [48] where 2-fold cv is done five times (with stratification) and the roles swapped at each fold to generate ten training and validation folds, tra_i, val_i, $i = 1, \ldots, 10$. tra_i are used to train the base classifiers. val_i are divided into two randomly as *val-*A_i and *val-*B_i, where *val-*A_i are used to train the combiner and *val-*B_i are used for model selection (in choosing the optimal subset or to choose the size of generated subsets). These ten trained models (base classifiers and combiner) are tested on the same *test* and we have ten $test_i$ accuracy results. This processed data of base classifier outputs are publicly available [49].

To compare the accuracies of different ensemble construction methods for statistically significant difference, we use two different methodologies. First, for each data set, we use the 5×2 cv F test [50] ($\alpha = 0.05$) which is a parametric test to compare the methods for each data set; we then use the sign test to check if the numbers of wins/losses over all 38 data sets is significant. Second, we use Friedman's test which is a nonparameteric test using the rankings, and if it rejects, we use the Nemenyi test as a post-hoc test to check for significant difference between methods [51].

2.2.4. Pen-Based Digit Recognition

The *pendigits* data set [4, 14] is for pen-based handwritten digit recognition. There are 7,494 training examples and 3,498 test examples (half of which were taken from people not included in the training and validation sets) (kept in a "vault" and never used during training or validation). There are two main representations:

- The *dynamic* (*dyn*) representation keeps track of the (x, y) coordinates of the pen tip as the digit is written on a pressure-sensitive tablet and keeps the temporal information.
- The *static* (*s16*) representation is the bitmap image formed after the digit is written, which we create by connecting the points in the dynamic representation by straight lines.

Note that because the same image can be written in different ways, or similar hand movements may cause different images, it is useful to combine these two representations. These representations have been normalized to constant length; dynamic representation contains a sequence of 8 (x^t, y^t) pairs and the static image is 16×16. We also form downsampled 8×8 and 4×4 versions of the static representation (Fig. 2.4)[3] .

The different representations of an example digit is shown in Figure 2.4. The advantage of combining multiple representations on this data set is shown in [4] and in [14], an incremental method that constructs a subset of classifiers using different representations is discussed. This data set is again divided as explained in Section 2.2.3.

2.3. Correlation Analysis on Real Data Sets

In the next subsection, we check for the effect of five factors on the correlation between the expert outputs:

[3]The multiple representations of *pendigits* is publicly available [49].

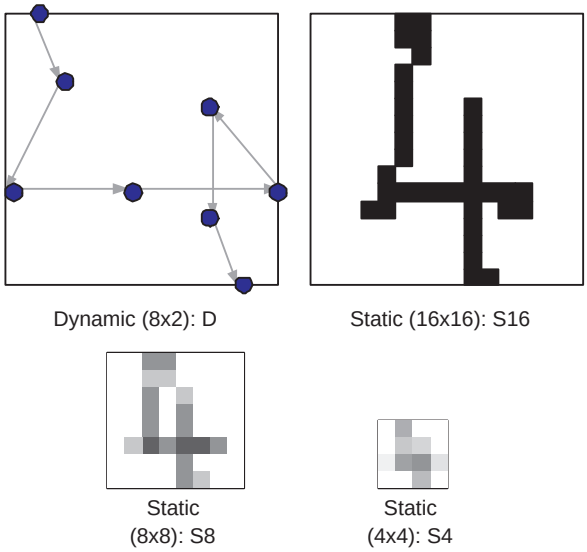


Figure 2.4. Pendigits representations for digit four

- Learning algorithms used to train the experts,
- Hyperparameters of the learning algorithms,
- Resampling due to folding,
- Subset of input features,
- Input representations.

2.3.1. Estimating the Correlations of Classifiers

We train all fourteen algorithms on a training fold, generate their posterior probabilities for the correct class on the test set and calculate correlations between classifier outputs for the correct class; the reason we use only the output for the correct class and not all classes is to (i) be able to average correlations over data sets with different number of classes, and (ii) keep the analysis simpler. We do this ten times on the ten training folds and calculate the average test correlation for a data set. We then do this for all 38 data sets and take another average to give a general, data-independent view. This overall correlation matrix is given in Table 2.2. We believe that a correlation value over 0.6 indicates a strong correlation and such entries are shown in boldface.

2.3.1.1. Correlations due to Hyperparameters. Almost all learning algorithms have hyperparameters which affect the model complexity and we check for the effect of

Table 2.2. Average correlations over all data sets

	kn1	*kn3*	*kn5*	*ml1*	*ml2*	*ml3*	*ml4*	*ml5*	*lnp*	*mdt*	*c45*	*svl*	*sv2*	*svr*
kn1	1.00	**0.71**	**0.64**	0.37	0.37	0.37	0.38	0.37	0.38	0.35	0.30	0.39	0.34	0.44
kn3	**0.71**	1.00	**0.88**	0.51	0.50	0.51	0.51	0.51	0.51	0.45	0.41	0.53	0.45	0.58
kn5	**0.64**	**0.88**	1.00	0.57	0.56	0.57	0.57	0.57	0.55	0.49	0.45	0.59	0.49	**0.64**
ml1	0.37	0.51	0.57	1.00	**0.79**	**0.81**	**0.81**	**0.79**	**0.67**	0.59	0.52	**0.75**	0.53	**0.69**
ml2	0.37	0.50	0.56	**0.79**	1.00	**0.81**	**0.79**	**0.77**	**0.66**	**0.62**	0.54	**0.76**	0.52	**0.69**
ml3	0.37	0.51	0.57	**0.81**	**0.81**	1.00	**0.81**	**0.81**	**0.67**	**0.61**	0.53	**0.75**	0.53	**0.70**
ml4	0.38	0.51	0.57	**0.81**	**0.79**	**0.81**	1.00	**0.80**	**0.67**	**0.61**	0.53	**0.75**	0.52	**0.69**
ml5	0.37	0.51	0.57	**0.79**	**0.77**	**0.81**	**0.80**	1.00	**0.67**	**0.60**	0.52	**0.75**	0.53	**0.69**
lnp	0.38	0.51	0.55	**0.67**	**0.66**	**0.67**	**0.67**	**0.67**	1.00	0.57	0.48	**0.71**	0.45	**0.63**
mdt	0.35	0.45	0.49	0.59	**0.62**	**0.61**	**0.61**	**0.60**	0.57	1.00	0.50	**0.64**	0.45	**0.60**
c45	0.30	0.41	0.45	0.52	0.54	0.53	0.53	0.52	0.48	0.50	1.00	0.54	0.43	0.54
svl	0.39	0.53	0.59	**0.75**	**0.76**	**0.75**	**0.75**	**0.75**	**0.71**	**0.64**	0.54	1.00	0.57	**0.74**
sv2	0.34	0.45	0.49	0.53	0.52	0.53	0.52	0.53	0.45	0.45	0.43	0.57	1.00	**0.65**
svr	0.44	0.58	**0.64**	**0.69**	**0.69**	**0.70**	**0.69**	**0.69**	**0.63**	**0.60**	0.54	**0.74**	**0.65**	1.00

these on correlation. Looking at the top-left corner of Table 2.2, we see the overall correlation matrix achieved by varying k of *knn*. We notice that varying k has a small effect on removing this intragroup correlation. This indicates that if you already have *3nn* in your ensemble, adding *5nn* is not a good idea; because they are highly correlated, addition would not increase accuracy significantly. As other examples, we see a similar behavior when we vary the number of hidden units of *mlp* or the degree of polynomial kernel of *svm*, though in this latter case, we see that the classifiers are less correlated when compared with *knn* variants.

2.3.1.2. Correlations due to Algorithms. There is also correlation depending on how similar the algorithms are: We see that the perceptron variants (*lnp* and *mlp*), linear models (*lnp*, *svl*), and the *svm* variants (*sv2*, *svl*, *svr*) are correlated. We see a clear case of grouping here: The variants of the same algorithm are grouped with high intragroup correlation and we also see lower but still positive intergroup correlation. For example, there is correlation between *mlp* and *svm* variants, both nonparametric estimators. The correlation between classifier groups decrease as they are less similar in terms of the models they use, the criteria they optimize, or the method they use for optimization.

2.3.1.3. Correlations due to Sampling. In order to figure out the correlations due to resampling, we calculate the correlation of the fourteen classifiers on the test set, trained on different training folds and average over them (over $10{\cdot}9/2$ fold pairs), and average once more over 38 data sets. The boxplots of correlations for the fourteen algorithms are given in Figure 2.5. We see that trees, *c45* and *mdt*, have low correlations and *mlp*, *knn* and *svm* variants have high correlations (except *1nn*); this shows that it makes sense to bag (fuse) trees but not *knn*, which is indeed actual practice. Breiman [3] mentions this when he defines the concept of stable algorithms and he says that it makes sense to bag unstable algorithms such as trees but not stable algorithms such as *knn*[4] . In general, the correlation seems to decrease with increasing training set size (as training set gets larger, different folds become more similar), though not as much as expected. Examples are given for *lnp* and *sv2* in Figure 2.6.

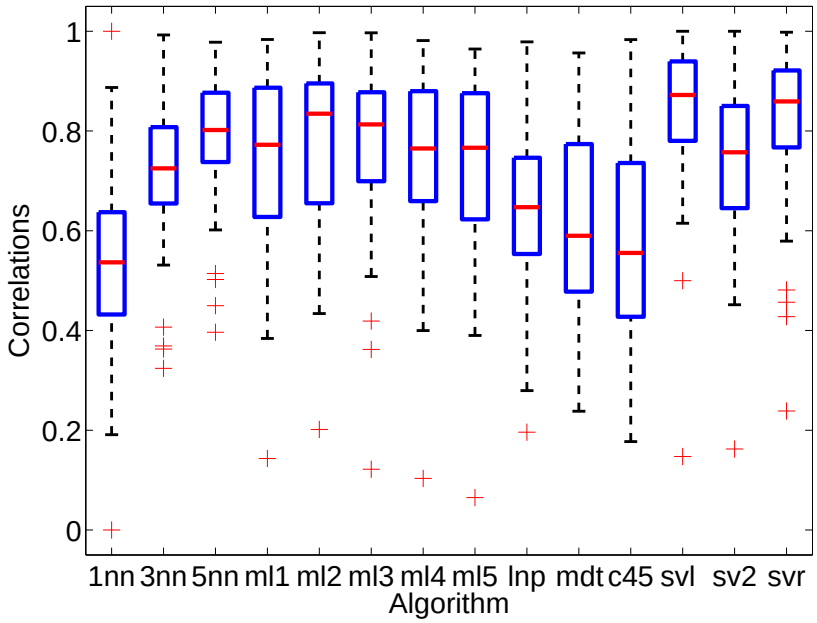
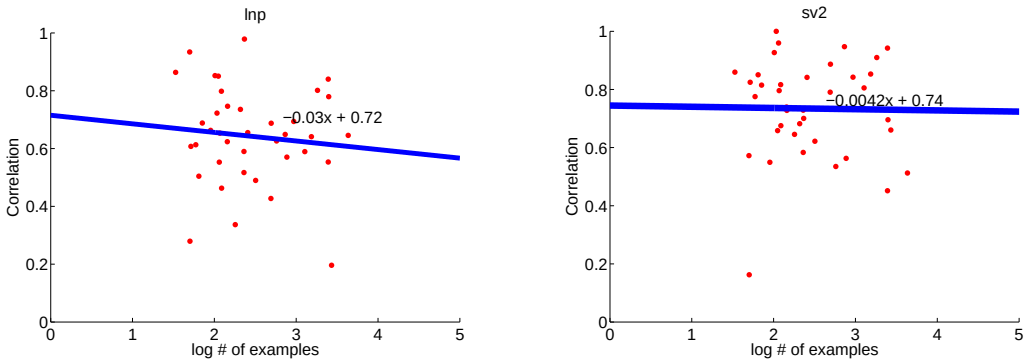


Figure 2.5. The boxplot of correlations between folds of the fourteen algorithms averaged over test sets of all data sets

2.3.1.4. Correlations due to Shared Input Features. In order to figure out the correlations due to input features used, we calculate the correlation of the fourteen classifiers trained on randomly chosen half of the original features, but on the whole training set (without folding). Doing this ten times each time choosing a different subset, we

[4] *knn* can be made unstable by condensing [19].



(a) *lnp* (b) *sv2*

Figure 2.6. Average correlations (over folds) of two algorithms, *lnp* and *sv2*, on the test set, as a function of training set size. Each point corresponds to one of 38 data sets. Correlation seems to decrease with increasing training set size, though not as much as one would expect

average over the $10 \cdot 9/2$ pairs on a data set, and average once more over 38 data sets. The boxplots of correlations for the fourteen algorithms are given in Figure 2.7. We can see that there is not much difference between the algorithms and that the correlation can be anywhere between 0.0 and 1.0, mostly around 0.5. We cannot say much that is general from these results; this random subspace method [21] can be an effective method (more effective than resampling or varying the hyperparameter) for generating less correlated experts but there is no guarantee. We have checked to see if the correlation depends on the input dimensionality; it does not seem to decrease with increasing input dimensionality, as would be expected. Examples are given for *3nn* and *ml1* in Figure 2.8.

2.3.1.5. Correlations due to Different Representations of the Same Input. To examine the correlations due to using different input representations, we use the *pendigits* data set [4, 14]:

We use the same methodology here as we do in other data sets, with 5×2 cross-validation and a separate test set. The base classifiers we use for this case are *svr*, support vector machines with radial kernels. The average correlation matrix is given in Table 2.3 where entries over 0.6 are shown in boldface.

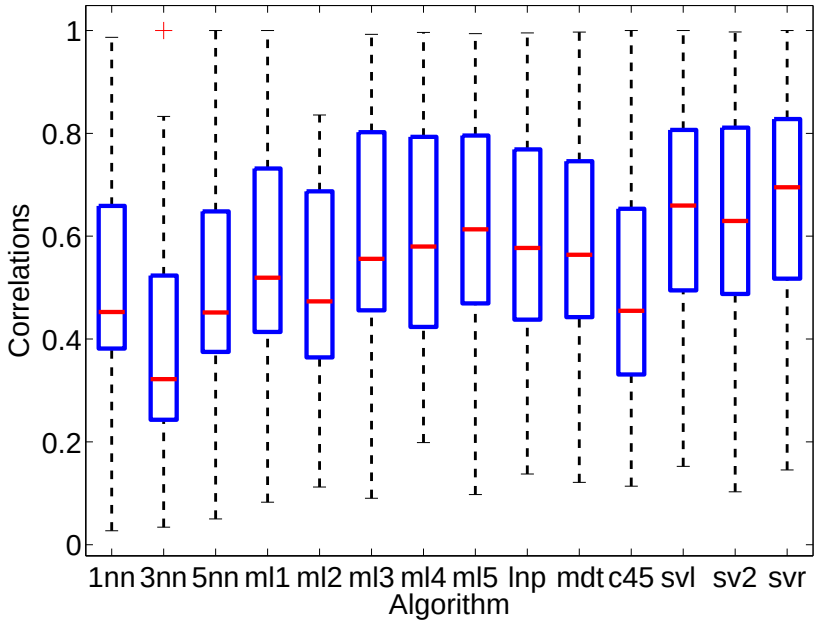
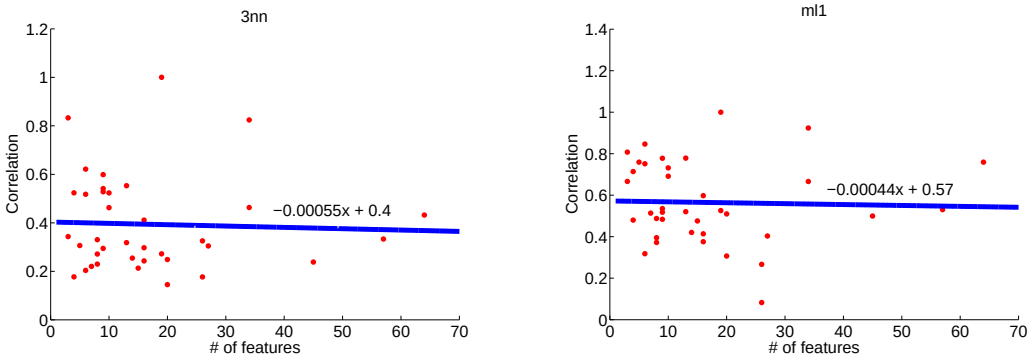


Figure 2.7. The boxplot of correlations between random input feature subsets of the fourteen algorithms averaged over test sets of all data sets



(a) *3nn* (b) *ml1*

Figure 2.8. Average correlations (over different randomly chosen input subsets) of two algorithms, *3nn* and *ml1*, on the test set, as a function of input dimensionality. Each point corresponds to one of 38 data sets. Correlation does not decrease with increasing dimensionality, as one would expect

Table 2.3. Average correlation matrix between classifiers trained with the four representations calculated on the test set

	dyn	*s4*	*s8*	*s16*
dyn	1.00	0.32	0.41	0.38
s4	0.32	1.00	**0.66**	0.55
s8	0.41	**0.66**	1.00	**0.82**
s16	0.38	0.55	**0.82**	1.00

We see that different basic representations (*s16* and *dyn*) have low correlation. The downsampled versions of the same basic representation have high correlation if the sizes are similar (i.e. *s16* is highly correlated with *s8*, *s8* is highly correlated with *s4*). This shows us that it is better to use different basic representations to achieve more diverse base classifiers, rather than preprocessing the same representation (e.g., by downsampling). Here, we would like to make a distinction between a *representation* and a *modality*: In our case, we expect to have some correlation between the two basic representations because one is calculated from the other; we would expect less or no correlation if inputs come from different modalities; for example, we expect no correlation between a person's face image and his/her signature.

3. INCREMENTAL CONSTRUCTION OF CLASSIFIER ENSEMBLES

Most of the algorithms we summarized in Chapter 1 are methods for forming ensembles, or are not used with multiple representations. Instead of focusing on the formation of an ensemble, or selecting what to include in an ensemble beforehand, what we want to achieve is to select amongst a given set of classifiers, or post-process the given classifiers to achieve better accuracy.

Methods have been proposed to choose a small subset from a large set of candidate models. Since there are 2^L-1 possible subsets of L models one cannot try for all possible subsets unless L is small, and various methods have been proposed to get a reasonable subset of size $m < L$ in reasonable time.

Ensemble construction methods also differ in the criterion they optimize. Additional to methods which directly optimize ensemble accuracy, heuristics have also been proposed as measures of "diversity" in pinpointing models which best complement each other, to allow diverse ones to be added and similar ones to be deemed redundant and pruned.

Ensemble construction can be viewed as an optimization problem and methods proposed in the literature correspond to different search strategies in optimization: There are greedy "forward" algorithms which are incremental and add one model at a time if the addition improves the criterion to be optimized. There are "backward" search methods which prune from a large set if the removal is not harmful. There are also "floating" methods which do both, as well as ones that use genetic algorithms whose operators allow both addition and deletion.

A chronological review of major ensemble construction methods in more detail is deferred to Section 3.6. In this chapter, we discuss and evaluate two ensemble construction approaches:

- We incrementally construct an *ensemble of classifiers* as in the methods discussed above. On 38 data sets using 14 different base classifiers, we test the effect of (i) The criterion to be optimized (accuracy, statistically significant improvement and two diversity measures, correlation and Q statistics), (ii) The search direction (forward, backward, floating), and (iii) The combiner (fixed voting, trained linear combiner).
- We incrementally construct an *ensemble of discriminants* where a classifier may be used for some of the classes but not for others [52, 40].

Let us say we have L possible models we can combine. If we take a vote over all L without any selection, then among these, there will be models which do not contribute to accuracy (and which may even decrease accuracy). There will also be the cost of all the models. Our proposed algorithm ICON to choose m out of L base classifiers is greedy in that it starts with the empty set and Incrementally CONstructs an ensemble where at each iteration, it chooses among all possible classifiers the one that best improves the performance when added to the current ensemble. Our approach is to select a subset from L to optimize a combined performance criterion of accuracy and cost. The performance is measured using the particular model selection method. The algorithm stops when there is no further improvement. Of course, this does not guarantee finding the best subset but this algorithm has polynomial complexity, $O(L^2)$, whereas exhaustive searching all possible subsets, $O(2^L)$, is of exponential complexity. The pseudocode of the algorithm is given in Figure 3.1. We start with $E^{(0)} = \emptyset$. At iteration t of the algorithm, we have ensemble $E^{(t)}$ containing t models (representation and the classifier using that representation). Given the set of remaining $L-t$ candidate models, $M_k \notin E^{(t)}$, we have new candidate ensembles for iteration $t+1$ as $S_k^{(t+1)} \equiv E^{(t)} \cup M_k, k = 1, \ldots, L-t$. Among these, we choose the one that is preferred to all the other candidates and is also preferred to the current ensemble:

$$E^{(t+1)} \leftarrow S_j^{(t+1)} \quad \text{if} \quad S_j^{(t+1)} \prec S_k^{(t+1)}, \forall k \neq j \text{ and } S_j^{(t+1)} \prec E^{(t)}$$

```
1  function icon(P)
2  E^0 ← ∅
3  for t = 0 to L − 1
4      S_k^(t+1) ← E^(t) ∪ M_k, ∀M_k ∈ P where M_k ∉ E^(t)
5      if ∃S_j^(t+1) such that S_j^(t+1) ≺ S_k^(t+1), ∀k ≠ j
           and S_j^(t+1) ≺ E^(t)
6          then E^(t+1) ← S_j^(t+1), t ← t + 1
7          else break
8  end for
9  return E^(t)
```

Figure 3.1. Pseudocode of the forward searching ICON algorithm

$E_i \prec E_j$ denotes the binary relation of "preference" comparing two ensembles and holds if E_i is preferred to E_j according to the model selection criterion used (which we will discuss next). If none of the candidates is preferred to $E^{(t)}$, the algorithm stops and $E^{(t)}$ is taken as the final ensemble.

3.1. Search Direction

The algorithm discussed above implements forward search. The backward version of ICON begins with all the models in the ensemble and prunes them. The idea is the same: If the complex ensemble is better than the simpler one (in terms of the model selection criterion we use) then keep the complex ensemble and stop, else prune the model from the ensemble and continue. Also it is possible to use a hybrid of both (a floating algorithm), which deletes a classifier from the ensemble if possible, else tries to add another classifier to the ensemble. In the forward version (.F), we choose the simplest model to be added to the ensemble in terms of the model selection criterion we use. In its backward version (.B), we delete the model which is the most complex. The floating version (.L) does both.

3.2. Ensemble Evaluation in ICON

For input $\boldsymbol{x}$, the posterior probability of class C_k calculated by ensemble E_i is denoted as $P(C_k|\boldsymbol{x}, E_i)$. Given a data set $\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^N$ (where $r_k^t = 1$ if $\boldsymbol{x}^t \in C_k$ and 0 otherwise), this can be used to calculate a misclassification error rate, $e(E_i)$, or log-likelihood, $l(E_i)$, for ensemble E_i:

$$\begin{aligned} e(E_i) &= \sum_t 1(\operatorname*{argmax}_k P(C_k|\boldsymbol{x}, E_i) \neq \operatorname*{argmax}_k r_k^t) \\ l(E_i) &= \sum_t \log P(C_{\operatorname{argmax}_k r_k^t}|\boldsymbol{x}, E_i) \ . \end{aligned} \tag{3.1}$$

$\operatorname{argmax}_k P(C_k|\boldsymbol{x}, E_i)$ returns the index of the class having the highest posterior, $\operatorname{argmax}_k r_k^t$ returns the index of the desired class and $1(a)$ is 1 if a is true and 0 if a is false. In our algorithm, the decision of an ensemble is calculated by simple voting (a fixed rule), or a linear combiner. If $P(C_k|\boldsymbol{x}, M_j)$ denotes the posterior probability of class C_k by model M_j for input $\boldsymbol{x}$, the overall probability for C_k is

$$P(C_k|\boldsymbol{x}, E) = \frac{1}{|E|} \sum_{M_j \in E} P(C_k|\boldsymbol{x}, M_j) \ ,$$

where $|E|$ denotes the number of models in ensemble E. Voting is the simplest way to combine the decisions of multiple models and it works quite well in practice. Though a method like stacking [9] may have lower bias, in such a case, there would also be the cost of the combiner model (called L_1 model in stacking). There are different types of costs in machine learning [53] such as the cost of sensing the features, the cost of misclassification errors, and the cost of computation. This third is the cost we use in this thesis and it corresponds to the cost of computation and space that is used by a model during testing. The total number of free parameters in the ensemble, $d(E_i)$, is the sum of the free parameters of the models in the ensemble (if we use a trained learner we add the cost of the learner also) because voting does not add any parameters:

$$d(E_i) = \sum_{M_j \in E_i} d(M_j) \ .$$

Given two ensembles E_i and E_j, a model selection method prefers one based on their accuracy and complexity.

3.2.1. Cross-Validation (CV)

We use k-fold cross-validation and use the pairwise one-sided k-fold paired t-test or 5×2 cross-validation with paired 5×2 cv t-test to compare the expected error rates of the two ensembles and check whether the more costly ensemble is statistically significantly more accurate than the simpler one [1]. If the test accepts, we prefer the more costly ensemble (the additional cost is justified), otherwise we prefer the simpler one (either because it is more accurate and cheaper, or they have the same expected accuracy and we prefer the cheaper). For two ensembles E_i and E_j where $d(E_i) < d(E_j)$, we calculate their error rates on the k validation folds and test the null hypothesis

$$H_0 : \mu_i \leq \mu_j \text{ vs. } H_1 : \mu_i > \mu_j$$

where μ_i is the average error of ensemble E_i. If the test accepts, we prefer E_i; if the test rejects, we prefer E_j.

3.2.2. Minimum Description Length (MDL)

Minimum Description Length (MDL), takes a weighted sum and the cost penalty term is interpreted as a suitable penalty term giving more probability to simpler models:

$$MDL(E_i) = -l(E_i) + \lambda d(E_i)$$

Equivalently, this can be interpreted as an augmented utility measure combining model accuracy on data and model complexity. By playing with λ, we can trade-off cost with accuracy and see which models will be preferred [32].

Akaike's Information Criterion (AIC) [54] and Bayesian Information Criterion (BIC), also known as the Schwartz Criterion [55], both are special cases of MDL and take a weighted sum of log-likelihood and cost (as measured by the number of free parameters):

$$AIC(E_i) = -l(E_i) + d(E_i)$$
$$BIC(E_i) = -l(E_i) + \frac{1}{2}\ln N \cdot d(E_i)$$

Note that MDL (and its variants), unlike cross-validation, do not require leaving out part of the data for validation and need not be run many times.

3.2.3. Accuracy (ACC)

This is basically MDL with $\lambda = 0$. We just check if the newly added classifier leads to higher average accuracy.

3.2.4. Q-Statistic (QSTAT)

As a diversity measure, we use the Q statistic [37]. First, we select the two classifiers which form the most diverse ensemble, and at each iteration, we add another classifier if diversity increases; else, we stop. The Q statistic for two classifiers is calculated as:

$$Q_{i,j} = \frac{N_{11}N_{00} - N_{10}N_{01}}{N_{11}N_{00} + N_{10}N_{01}}$$

where N_{00}, N_{01}, N_{10}, and N_{11} are defined as in Table 3.1. The average diversity for an ensemble is calculated as

$$Q_{av} = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{k=i+1}^{m} Q_{i,k}$$

3.2.5. Correlation Coefficient (CORR)

As another frequently used diversity measure, we use the average correlation coefficient [37]:

$$\rho_{i,j} = \frac{N_{11}N_{00} - N_{10}N_{01}}{\sqrt{(N_{11} + N_{10})(N_{01} + N_{00})(N_{11} + N_{01})(N_{10} + N_{00})}}$$

The average correlation for the ensemble is calculated by averaging over all pairs, as done in QSTAT.

Table 3.1. Contingency table used by the measures of diversity

	M_j correct	M_j wrong
M_i correct	N_{11}	N_{10}
M_i wrong	N_{01}	N_{00}

3.3. Model Combination

Given an ensemble of base classifiers, the easiest way to calculate the overall output is by taking a sum, which corresponds to taking a vote. Our base classifiers generate posterior probabilities so there is no need for scaling or other type of normalization or transformation [56, 57].

There are other fixed rules, i.e., median, product, minimum, or maximum [8] (which we will investigate in our experiments), but the sum rule we use is known to work best in practice. Alkoot and Kittler [43] investigate fixed rules and see that the sum and median rules are more robust to noise. Kuncheva [33] discusses six fixed rules for classifier combination (minimum, maximum, sum, median, majority vote, and the oracle) on two-class problems with Gaussian and uniform error, under the assumption that the outputs are independent. She concludes that the min/max rule (they are the

same for two-class problems) find the best ensemble when uniform error is the case. For Gaussian distributed errors, the combination rules behave similarly. Tax et al. [58] show that in multi-class problems product rule may be superior to sum rule when the independent data representation assumption is met. On the other hand, sum rule is more robust to noise. Especially in the cases when one of the classifiers is an outlier, product rule "acts as a veto" and this decreases the combination performance. In two class problems, though, there is no difference between them. Cabrera [34] analyzes average, median and maximum rules when the number of classifiers becomes large. He finds that average is the best for normal error, maximum is the best for uniform error and median is the best for Cauchy error. The analysis in these studies is under the assumption that classifiers are independent, though in practice they are correlated [59].

The sum rule gives equal weight to all base classifiers. We also try a trained linear combiner; this is called stacking [9, 60]. We do not constrain weights to be nonnegative or sum to 1, and there is also a constant intercept. A data set separate from the one used to train the base classifiers is used to train this linear combiner.

A trained rule may have lower bias, but fixed rules are generally favored for a number of reasons: (i) There is not the extra cost of storing/processing for the cost of the combiner model (called L_1 model in stacking). (ii) We save from the time needed to train the combiner model, and (iii) There is no need to leave out a part of the training set to train the combiner, and all data can be used to train the base classifiers. Duin and Tax [61] compare various trained and fixed combination rules. According to their results there is no winning fixed combination rule; median, mean and majority rules work best on correlated data while the product rule works better in the case of independent errors.

If $P(C_k|\boldsymbol{x}, M_j)$ denotes the posterior probability of class C_k by model $M_j \subset E$ for input $\boldsymbol{x}$, the fused probability for C_k by ensemble E is

$$P(C_k|\boldsymbol{x}, E) = f\Big(P(C_k|\boldsymbol{x}, M_1), P(C_k|\boldsymbol{x}, M_2), \ldots, P(C_k|\boldsymbol{x}, M_L)\Big)$$

where $f(P(C_k|\boldsymbol{x}, M_1), P(C_k|\boldsymbol{x}, M_2), \ldots, P(C_k|\boldsymbol{x}, M_L))$ denotes the rule used to fuse the output of base models to calculate the posterior probability.

In our experiments we checked the effect of fixed rules given in Table 1.1 and a trained combiner (linear perceptron .LIN).

Therefore, in choosing an ICON variant, there are three factors: (1) Model selection criterion: MDL/ACC/CV/QSTAT/CORR, (2) Rule type: SUM/MAX/MIN/ MED/PRO/LIN, and (3) Search direction: F/B/L. We use the following notation: ACC.MIN.F stands for the ICON variant using ACC criterion, rule MIN, and doing a forward search.

3.4. Ensemble of Classifiers

We investigate the effect of various factors in ensemble construction using a wide variety of learning algorithms, data sets and evaluation criteria.

3.4.1. Compared Ensembles

The following ensembles are generated and compared:

- BEST: We order the base classifiers in terms of accuracy and use the first 1, 3, 5, 7, 9 of them. This has two variants: BEST.SUM uses the fixed sum rule and BEST.LIN uses the trained linear combiner.
- RND: We randomly choose 1, 3, 5, 7, 9 base classifiers, with SUM and LIN options.
- ALL: All the available base classifiers are combined without selection, with SUM and LIN options.
- OPT: We try all possible subsets (there are $2^{14}-1$) and choose the most accurate. It has SUM and LIN options.
- ICON: The classifier ensembles generated by ICON variants (ACC, CV, QSTAT, and CORR) are used. They all have SUM and LIN options.

- FSS: This is the discriminant ensemble which uses forward subset selection and a linear combiner.
- DT: This is the discriminant ensemble which uses a decision tree.
- DT.LIN: This is the discriminant ensemble which uses a decision tree for feature selection and a linear combiner.

FSS is an algorithm similar to ICON, but instead of adding base classifiers, the algorithm incrementally adds a new discriminant and selects a subset of discriminants. DT creates a decision tree over the outputs of all base classifiers, and acts as discriminant selector and L_1 combiner. DT.LIN is similar to DT in it that, it constructs a decision tree on all outputs of the base classifiers, this decision tree does not act as the L_1 classifier, but it selects the discriminants only, and a post-hoc linear classifier is trained on the selected discriminants for the final classification. For details about these three algorithms, see [52, 40].

3.5. Experimental Results

First, we check for the effect of search direction in ICON. In the next subsection, we discuss and compare the results of our proposed methods on two data sets in detail, before moving on to an overall comparison on all data sets.

3.5.1. Comparison of Search Methods

As we have mentioned in the introduction of this chapter, it is possible to search in the forward direction adding one at a time, backward direction removing one at a time, or a floating search where we try to remove a previously added base classifier before adding another one. We can hence consider three ICON variants implementing forward (.F), backward (.B) or floating (.L) search. The comparison of the accuracies of the three search directions according to the four criteria used by ICON variants can be seen in Table 3.2. These entries are the number of data sets on which there is a statistically significant win/loss of the method in the row over the method in the column (using 5×2 cv F test); $38 - \text{wins} - \text{losses}$ gives the number of ties; the entry

is bold if the number of wins/losses is significant in 38 trials (using Sign test). The average and standard deviation of the number of base classifiers in the found ensemble and the number of visited states during search are given in Tables 3.3 and 3.4.

Table 3.2. The number of statistically significant accuracy wins/losses of .F (Forward), .B (Backward), and .L (Floating) over 38 data sets according to the criterion used by ICON. The bold face entries show statistically significantly difference using Sign test

ACC	.F	.B	.L	CV	.F	.B	.L
.F	0/0	1/0	1/0	.F	0/0	1/5	0/0
.B	0/1	0/0	0/1	.B	5/1	0/0	5/1
.L	0/1	1/0	0/0	.L	0/0	1/5	0/0
QSTAT	.F	.B	.L	CORR	.F	.B	.L
.F	0/0	2/5	2/0	.F	0/0	**1/8**	4/0
.B	5/2	0/0	8/2	.B	**8/1**	0/0	**11/1**
.L	0/2	2/8	0/0	.L	0/4	**1/11**	0/0

Table 3.3. Average ± standard deviation of the number of classifiers in ensembles found by each search direction and optimization criterion

	ACC	CV	QSTAT	CORR
.F	2.39±1.5	1.05±0.2	8.16±4.9	5.74±5.1
.B	7.79±3.2	2.08±2.1	11.55±3.7	11.39±3.7
.L	2.26±1.3	1.05±0.2	6.29±4.4	3.32±2.3

We see that in all four criteria of ACC, CV, QSTAT, and CORR, .F and .L give similar results; this is because most of the time small ensembles are enough and there is not much to prune back after few additions. We also see that in terms of the ensemble sizes and search time, .L and .F stand out as the best. With the diversity measure CORR, backward search is significantly more accurate and faster because CORR needs larger ensembles, and that is why with the diversity-based measures, forward and floating search takes more steps.

Table 3.4. Average ± standard deviation of the number of search steps visited by each search direction and optimization criterion

	Acc	Cv	Qstat	Corr
.F	42.37±15.1	27.63±2.7	143.05±28.6	125.84±30.6
.B	73.47±22.9	101.68±8.0	37.26±29.0	38.95±29.0
.L	712.47±345.4	415.58±40.6	172.13±61.2	131.68±40.6

We can see that .B and .L increase both the number of search steps and the ensemble size for Acc and Cv. We believe that it is not beneficial to use floating search because it finds the same ensembles as forward search does but takes more steps. In diversity-based measures, backward search is more accurate but this increases the ensemble size and we have no benefit over using the whole ensemble. Aiming high accuracy and small ensembles, we therefore adopt forward search in the rest of the chapter.

3.5.2. Initial Results

We start by discussing in detail our results on two data sets, *optdigits* and *nursery*, as two example cases. We present our overall results on all data sets in the next subsection.

3.5.2.1. Optdigits Data Set. The accuracies of base classifiers (sorted in increasing accuracy) and the ensembles on *test* data are given in Table 3.5, together with the number of classifiers, the number of discriminants, and the chosen ensembles. The plot of accuracies vs the number of base classifiers is shown in Figure 3.2. On this data set (as many others), using a linear combiner .Lin does not increase the accuracy significantly, and therefore only .Sum results are given to keep the table and figures simpler.

Table 3.5. Results on *optdigits*

Alg	*test*	# cla	# disc	Chosen
c45	81.76±1.3	1	10	*c45*
mdt	92.98±1.0	1	10	*mdt*
lnp	94.28±0.8	1	10	*lnp*
ml2	94.86±0.5	1	10	*ml2*
ml4	96.03±0.9	1	10	*ml4*
ml5	96.21±0.4	1	10	*ml5*
ml1	96.02±0.6	1	10	*ml1*
3nn	95.99±0.4	1	10	*3nn*
1nn	96.67±0.4	1	10	*1nn*
5nn	95.86±0.3	1	10	*5nn*
ml3	96.34±0.3	1	10	*ml3*
svl	97.48±0.2	1	10	*svl*
svr	97.67±0.2	1	10	*svr*
sv2	97.49±0.3	1	10	*sv2*
Best.3.Sum	98.22±0.2	3	30	*sv2 svr svl*
Best.5.Sum	98.19±0.2	5	50	*sv2 svr svl ml3 5nn*
Best.7.Sum	97.90±0.2	7	70	*sv2 svr svl ml3 5nn 1nn 3nn*
Best.9.Sum	98.01±0.3	9	90	*sv2 svr svl ml3 5nn 1nn 3nn ml1 ml5*
Rnd.1.Sum	97.67±0.2	1	10	*svr*
Rnd.3.Sum	97.60±0.3	3	30	*c45 svl 1nn*
Rnd.5.Sum	97.67±0.2	5	50	*ml3 ml1 ml5 sv2 1nn*
Rnd.7.Sum	97.82±0.2	7	70	*ml3 ml4* mdt *ml5 svr svl 1nn*
Rnd.9.Sum	97.61±0.2	9	90	*ml2 c45 ml3 lnp* mdt *ml5 svr sv2 3nn*
All.Sum	97.85±0.2	14	140	
Opt.Sum	98.22±0.2	3	30	*svr sv2 svl*
Acc.Sum	98.22±0.2	3	30	*sv2 svr svl*
Cv.Sum	96.02±0.6	1	10	*ml1*
QStat.Sum	97.85±0.2	14	140	All
Corr.Sum	97.85±0.2	14	140	All
Fss	98.83±0.1	10	25	*c45*(1,4) *1nn*(3) *3nn*(3,5) *5nn*(7) *ml1*(9) *ml2*(4) *ml3*(1) *svl*(2,3,5,6,7) *sv2*(1,2,4,7,8,9) *svr*(0,3,4,6,8)
Dt	97.65±0.4	6.6	11.8	*1nn*(6) *5nn*(5) *lnp*(0) *svl*(7) *sv2*(1,2,3,4) *svr*(8,9)
Dt.Lin	98.22±0.3	6.6	11.8	*1nn*(6) *5nn*(5) *lnp*(0) *svl*(7) *sv2*(1,2,3,4) *svr*(8,9)

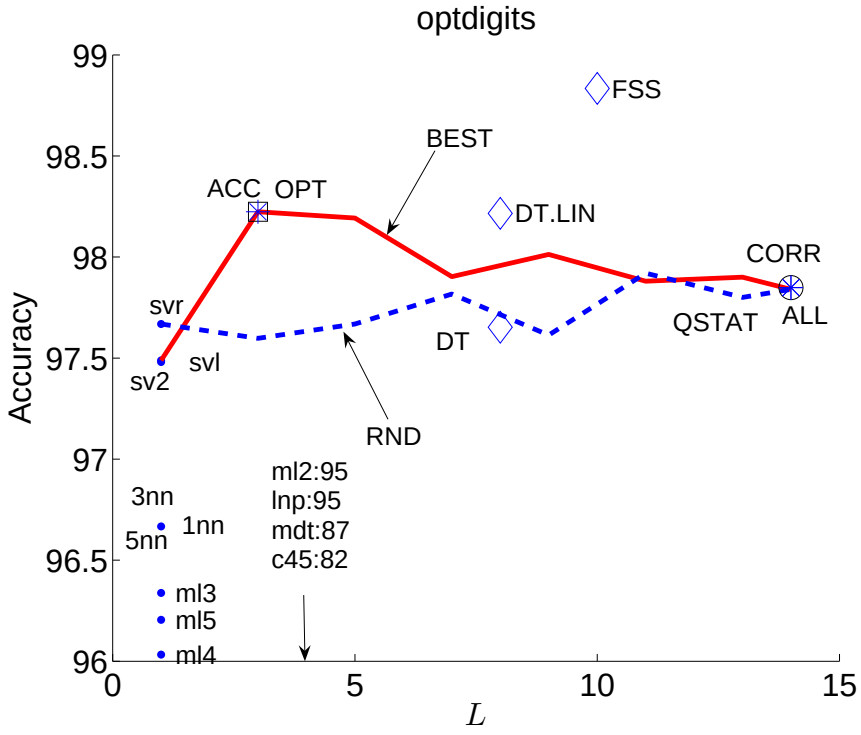


Figure 3.2. Accuracy vs the number of classifiers on *optdigits test*

We see that the optimal subset OPT chooses only three out of fourteen and is significantly as accurate as ALL that uses all fourteen. Ordering base classifiers in terms of accuracy and choosing the best m, BEST, is also significantly better than ALL. Note that the accuracy of BEST starts to decrease when too many base classifiers are added; in such a case the increase in bias is higher than the decrease in variance. Choosing a random subset, RND, does not work as well and requires more base classifiers.

In terms of the ensembles of classifiers generated by ICON, we see that ACC also finds the optimal subset. The optimal subset generally contains a few base classifiers and searching in the forward direction, as ICON variant ACC does, greedily returns a quite good subset in polynomial time.

The ICON variants that use diversity measures, QSTAT and CORR, use too many base classifiers, showing that it is best to use accuracy as the ensemble evaluation criterion directly rather than an intermediate diversity measure; or that diversity should not be used alone but in some combination with accuracy.

CV uses only one base classifier as it finds that adding a second does not increase accuracy significantly. The behavior of CV depends on the statistical test and the confidence level. We show in Figure 3.3 how the ensemble changes at different confidence levels. When the confidence level is lower than 0.95, the test uses a smaller confidence interval, is more likely to reject and generates larger ensembles: CV tends to behave similar to ACC as the confidence level decreases. As the confidence level increases, the test gets more and more conservative and generates smaller ensembles, until it chooses only one.

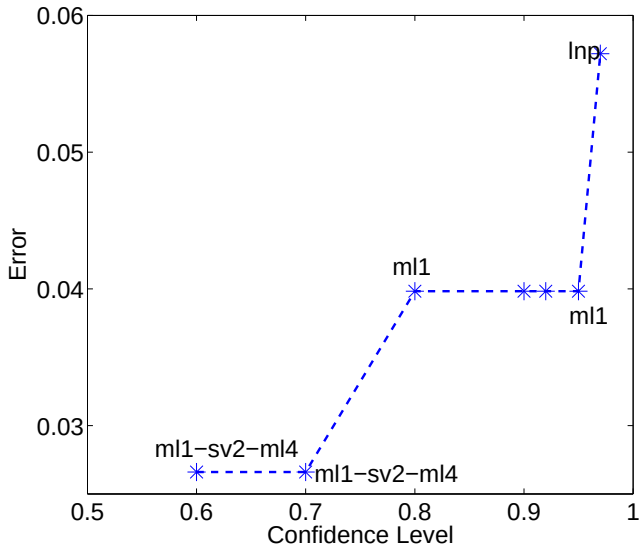


Figure 3.3. The ensemble found by CV changes as the confidence level changes on *optdigits*

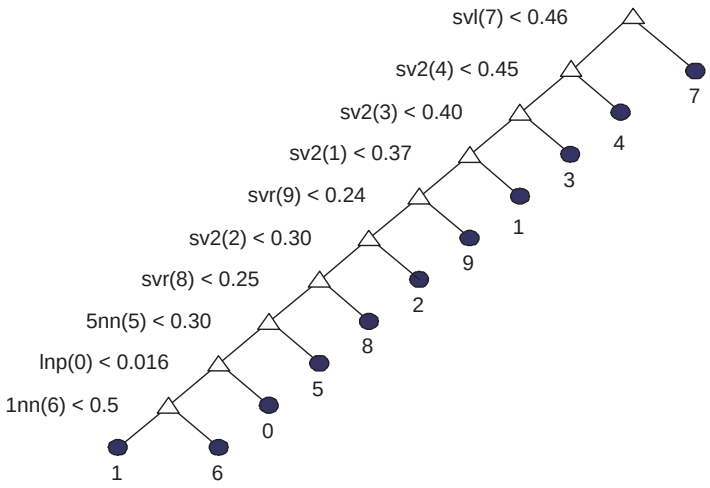


Figure 3.4. One of the decision trees learned by DT on *optdigits*

As for the ensembles of discriminants, we see that FSS uses more discriminants than DT, but is significantly more accurate than all other methods. On the average, DT chooses 11.8 discriminants (the smallest possible with ten classes is 9) from 6.6 classifiers. Example tree (one of ten) learned by DT is given in Figure 3.4 and its discriminants are given in Table 3.5: It starts by looking at the output of *svl* for class '7' and chooses '7' if this value is higher than 0.46. Note that we only evaluate the discriminants in our path; for example we see here that the complex *5nn* is only evaluated to distinguish '0', '1' and '6' from others. DT.LIN improves over DT, but not significantly ($p = 0.90$). Most classes are identifiable by looking at a single discriminant; only '1' requires two (*sv2* and *1nn*). The example tree DT chooses only ten discriminants, choosing discriminants from six base classifiers; this set includes the optimal subset found by OPT (which has $3 \cdot 10 = 30$ discriminants), and uses three more discriminants from three classifiers (*5nn*, *1nn* and *lnp*).

3.5.2.2. Nursery Data Set. The results on *nursery* is similar except that .LIN improves accuracy over .SUM, for some ensemble methods. The results for the single base classifiers and the combination methods are given in Table 3.6. In Figure 3.5, we compare the .SUM and .LIN variants.

On this data set, the accuracies of the base classifiers range from 76.85 to 99.41 and we see that using .SUM (Fig. 3.5(a)), ALL has low accuracy and the accuracy of BEST decreases as more classifiers are added; we do not see this as strongly with .LIN (Fig. 3.5(b)). That is, a trained combiner is more robust to addition of erroneous classifiers. The trained combiner is able to weight the accurate ones more and effectively ignores those that are not as accurate. We also see that ICON, as a subset selection method shows the same robust behavior; that is, because it does not add and use the inaccurate classifiers, its accuracy is not degraded. Note that this is valid even if .SUM is used (Fig. 3.5(a)). We therefore see that in the presence of inaccurate base classifiers in the ensemble, either one should use a trained combiner which learns to ignore them or use a subset selection method which does not include them.

This is a data set with four classes and the tree learned by DT for this fold (Fig. 3.6) has only three decision nodes, is very simple, and has 100.00 *test* accuracy, which is higher than what we get when we use all outputs of all base classifiers for that fold (99.81). FSS uses more discriminants and is as accurate. On this data set, ACC, OPT, FSS and BEST are significantly more accurate than ALL and RND.

Table 3.6. Results on *nursery* data set

Alg	*test*	# cla	# disc	Chosen
1nn	76.86±0.5	1	4	*1nn*
3nn	85.18±0.5	1	4	*3nn*
5nn	89.93±0.4	1	4	*5nn*
lnp	90.86±0.7	1	4	*lnp*
svl	92.43±0.3	1	4	*svl*
mdt	92.74±0.4	1	4	*mdt*
c45	92.72±0.5	1	4	*c45*
ml2	94.97±0.9	1	4	*ml2*
svr	95.50±0.4	1	4	*svr*
sv2	98.70±0.6	1	4	*sv2*
ml3	99.13±0.4	1	4	*ml3*
ml5	99.38±0.3	1	4	*ml5*
ml4	99.42±0.3	1	4	*ml4*
ml1	99.41±0.2	1	4	*ml1*
BEST.3.LIN	99.58±0.2	3	12	*ml1 ml4 ml3*
BEST.5.LIN	99.81±0.1	5	20	*ml1 ml4 ml3 ml5 sv2*
BEST.7.LIN	99.78±0.1	7	28	*ml1 ml4 ml3 ml5 sv2 svr ml2*
BEST.9.LIN	99.77±0.1	9	36	*ml1 ml4 ml3 ml5 sv2 svr ml2 c45* mdt
RND.1.LIN	91.48±0.3	1	4	*5nn*
RND.3.LIN	99.40±0.2	3	12	*ml1 3nn svr*
RND.5.LIN	99.51±0.3	5	20	*lnp ml4 ml5 1nn svr*
RND.7.LIN	99.60±0.2	7	28	*ml2 ml3 ml1 ml4* mdt *ml5 1nn*
RND.9.LIN	99.75±0.1	9	36	*ml2 ml3 lnp ml1 ml4 c45 5nn svl sv2*
ALL.LIN	99.76±0.1	14	56	
OPT.LIN	99.81±0.1	5	20	*ml3 ml1 ml5 5nn sv2*
ACC.LIN	99.82±0.1	4	16	*ml1 sv2 ml4 3nn*
CV.LIN	99.83±0.1	2	8	*ml1 sv2*
QSTAT.LIN	93.13±0.5	2	8	*c45 1nn*
CORR.LIN	98.91±0.4	2	8	*ml3 1nn*
FSS	99.95±0.0	5	8	*ml1*(0) *ml3*(0) *ml4*(0,3) mdt(0) *sv2*(0,2,3)
DT	99.95±0.1	2.7	3	*c45*(1) *sv2*(2,3)
DT.LIN	98.85±1.1	2	3	*c45*(1) *sv2*(2,3)

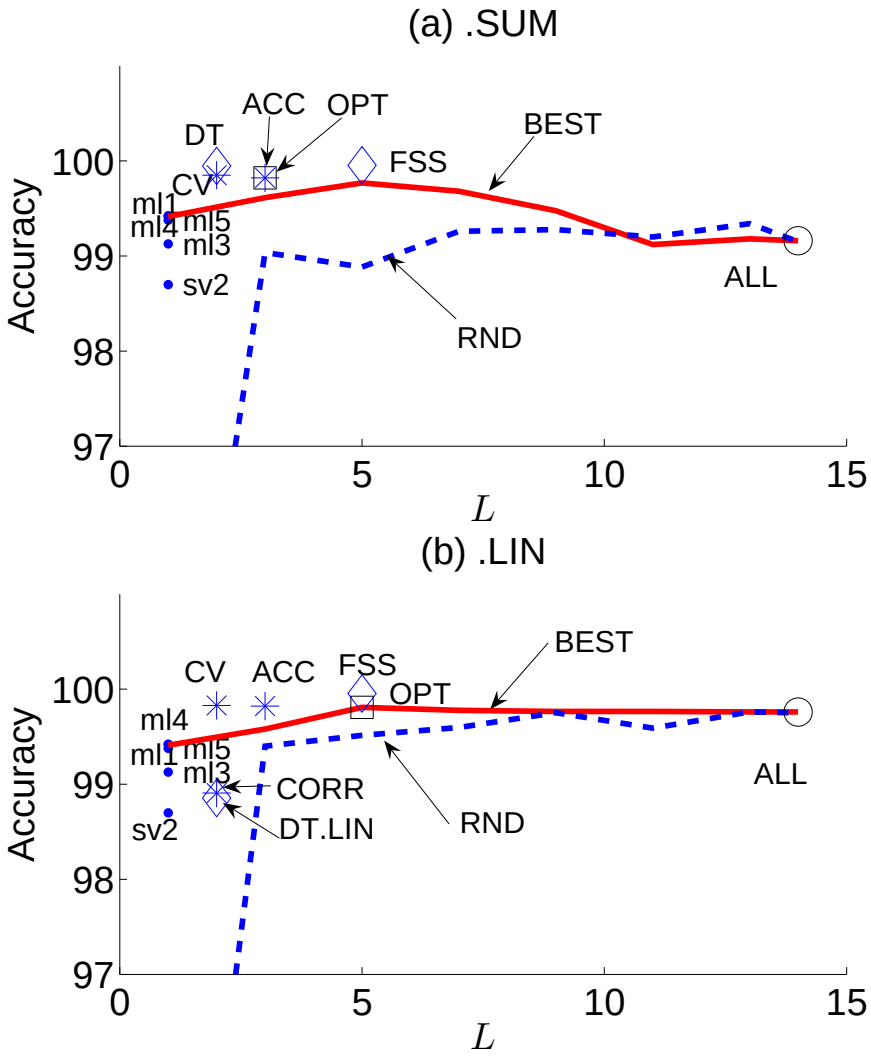


Figure 3.5. Comparison of fixed .SUM and trained .LIN on *nursery test*

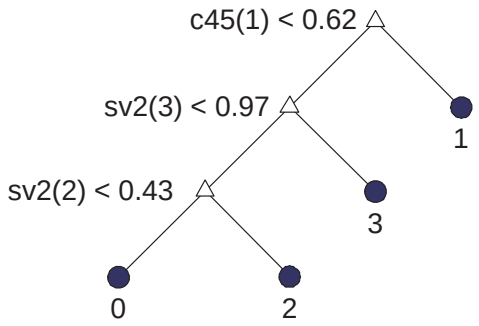


Figure 3.6. One of the decision trees learned by DT on *nursery*

3.5.3. General Results

We compare accuracies of all ensemble methods in a pairwise manner on *test* in Table 3.7. These are the number of significant wins and losses of method in the row over the method in the column. The sum of wins and losses subtracted from 38 gives the number of ties. If the entry is bold, this means that the number of wins/losses over 38 is statistically significant using the Sign test. We do further statistical analysis with nonparametric tests using the average ranks of the six ensemble methods on 38 data sets (Table 3.8). Friedman's test rejects the hypothesis that the six methods have equal ranks. Doing Nemenyi's post-hoc test for pairwise comparison, we get the results in Figure 3.7.

Table 3.7. Pairwise comparison of accuracies (wins/losses over 38) of all methods using 5×2 cv F-test

	Best	Rnd	All	Opt	Acc	Cv	Qstat	Corr	Fss	Dt	Dt.Lin
Best	0/0	3/1	**6/0**	0/0	0/1	**9/1**	**7/0**	**8/0**	10/4	6/1	**9/1**
Rnd	1/3	0/0	4/2	**0/7**	1/7	**9/1**	8/2	**10/1**	7/6	7/3	10/4
All	**0/6**	2/4	0/0	**0/6**	**0/6**	9/3	5/2	8/2	6/9	6/3	8/6
Opt	0/0	**7/0**	**6/0**	0/0	0/0	**12/0**	**9/0**	**9/0**	9/6	**6/0**	9/2
Acc	1/0	7/1	**6/0**	0/0	0/0	**14/2**	**10/1**	**9/1**	10/3	**6/0**	**10/1**
Cv	**1/9**	**1/9**	3/9	**0/12**	**2/14**	0/0	6/7	6/7	3/9	4/1	5/6
Qstat	**0/7**	2/8	2/5	**0/9**	**1/10**	7/6	0/0	5/1	5/12	5/5	8/8
Corr	**0/8**	**1/10**	2/8	**0/9**	**1/9**	7/6	1/5	0/0	7/12	5/5	5/8
Fss	4/10	6/7	9/6	6/9	3/10	9/3	12/5	12/7	0/0	7/1	**12/3**
Dt	1/6	3/7	3/6	**0/6**	**0/6**	1/4	5/5	5/5	1/7	0/0	1/4
Dt.Lin	**1/9**	4/10	6/8	2/9	**1/10**	6/5	8/8	8/5	**3/12**	4/1	0/0

Table 3.8. Average ranks of compared methods

Best	Rnd	All	Opt	Acc	Cv	Qstat	Corr	Fss	Dt	Dt.Lin
4.55	5.68	5.76	3.89	4.08	8.01	6.97	7.17	4.82	8.62	6.43

Table 3.9 shows the average number of base classifiers (/discriminants) that ensembles constructed using different methods contain. The discriminant values are normalized by dividing with the number of classes.

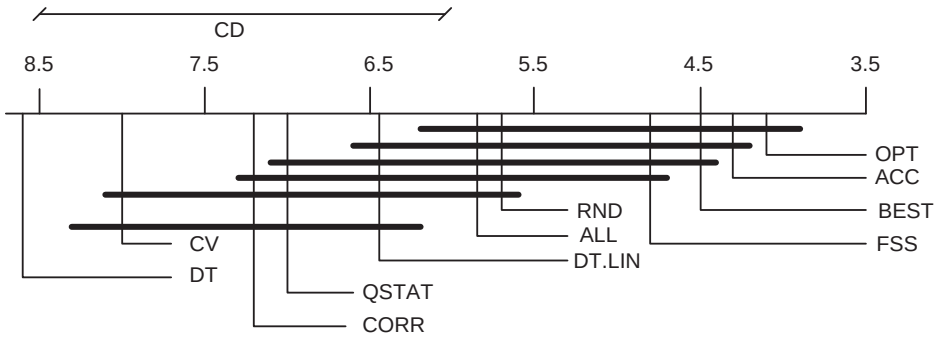


Figure 3.7. Graphical representation of post-hoc Nemenyi test results of compared methods with ranks given in Table 3.8, as proposed in [51]. The numbers on the line represent the average ranks, CD is the critical difference for statistical significance, and bold lines connect the methods which have no significant difference

Table 3.9. Average number of base classifiers (/discriminants) contained in different ensembles

	Best	Rnd	All	Opt	Acc	Cv	Qstat	Corr	Fss	Dt
Sum	2.79	4.89	14.00	4.53	2.39	1.05	8.16	5.74	4.58/1.83	5.60/2.76
Lin	5.84	6.58	14.00	6.08	3.95	1.08	8.16	5.74		5.60/2.76

Additional to their accuracies and complexities, we also check how similar are the ensembles found by different methods. Given two ensembles E_i and E_j, we define the similarity between them as the number of shared base classifiers (or discriminants):

$$\text{Sim}(E_i, E_j) = \frac{|E_i \cap E_j|}{|E_i \cup E_j|}$$

If the two ensembles share the same base classifiers (discriminants), the similarity is 1; if there is no intersection, the similarity is 0. The average similarity between ensembles found by different methods is given in Table 3.10.

Analyzing these tables, our general results are as follows:

- Overall, using a trained linear combiner does not increase the ensemble accuracy significantly. In Table 3.11, we report a comparison of accuracies using .SUM vs .LIN; we see that except with CV, there is no significant difference between them, and on CV, it is .SUM which is more accurate.

  We believe this is because we have sufficiently well-trained, capable base classifiers which are able to approximate posterior probabilities quite well so that no further bias correction (which the linear combiner effectively is there for) is required. It may also be the case that on small data sets, there are not enough data to train the linear combiner sufficiently. It is possible to increase the number of folds and for example use 20×1 cross-validation which will have the effect of increasing the training set both for base classifiers and the combiner at the expense of doubling the running time. In the extreme case, one can use leave-one-out (as proposed in the original stacking paper of Wolpert [9]) but this cannot be afforded (nor is it necessary) unless the data set is small.

  Using a fixed rule decreases time/space complexity of training/testing, does not require data to train the combiner allowing more data to train base classifiers, and is simpler to interpret. Therefore throughout the rest of this section, we consider .SUM results only.

Table 3.10. Average similarity of base classifiers (discriminants) between ensembles found by different methods

	Best	All	Opt	Acc	Cv	Qstat	Corr	Fss	Dt
Best	1.00	0.20	0.44	0.61	0.11	0.13	0.14	0.32	0.20
								0.20	0.13
All	0.20	1.00	0.32	0.17	0.08	0.58	0.41	0.33	0.40
								0.13	0.20
Opt	0.44	0.32	1.00	0.48	0.14	0.23	0.20	0.37	0.28
								0.21	0.17
Acc	0.61	0.17	0.48	1.00	0.09	0.13	0.15	0.29	0.19
								0.19	0.14
Cv	0.11	0.08	0.14	0.09	1.00	0.06	0.09	0.16	0.07
								0.12	0.05
Qstat	0.13	0.58	0.23	0.13	0.06	1.00	0.66	0.23	0.32
								0.11	0.17
Corr	0.14	0.41	0.20	0.15	0.09	0.66	1.00	0.24	0.24
								0.14	0.14
Fss	0.32	0.33	0.37	0.29	0.16	0.23	0.24	1.00	0.26
	0.20	0.13	0.21	0.19	0.12	0.11	0.14	1.00	0.12
Dt	0.20	0.40	0.28	0.19	0.07	0.32	0.24	0.26	1.00
	0.13	0.20	0.17	0.14	0.05	0.17	0.14	0.12	1.00

Table 3.11. Comparison of accuracies (wins/losses over 38) of .Sum vs .Lin

	Best	Rnd	All	Opt	Acc	Cv	Qstat	Corr
test	2/1	2/3	0/5	0/0	1/0	**6/0**	3/6	5/5

- According to Nemenyi's test given in Figure 3.7, there is no significant difference in accuracy between BEST, OPT, RND, ACC, FSS. Among those, as we see in Table 3.9, ACC uses the least number of classifiers, and FSS uses the least number of discriminants. These results show that our proposed methods for ensembles of classifiers and discriminants construct ensembles which are simple and as accurate as optimal subset or the whole.
- Among ICON variants, ACC is the most accurate. It is also more accurate than ALL on six data sets out of 38 on *test* (which is significant using Sign test). ACC does not lose to OPT on any data set. The similarity of the ensembles they find is 0.48; this is the fourth highest value in the table—the highest similarity is between QSTAT and CORR with 0.66. This shows that a greedy, forward, incremental accuracy-based ensemble construction method works quite well. CV rarely uses more than a single base classifier and the diversity based ICON variants are significantly less accurate than ACC.
- BEST is also as accurate as OPT but needs more base classifiers than ACC does. The similarity between ACC and BEST ensembles is 0.61 (which is the second highest value in the table.), showing that the greedy algorithm selects from the best m but sometimes it chooses differently. ACC ensembles contain fewer base classifiers than those of BEST, indicating that instead of choosing the first best m classifiers it is better to do some choosing for those which complement each other and then we can do with less.
- Even RND works quite well but needs a larger ensemble, requiring more than two times more base classifiers than ACC does.
- According to the average number of base classifiers they choose to include in their ensembles, we have the following ordering: CV < ACC < BEST < OPT < FSS < RND < DT < CORR < QSTAT < ALL.
- ACC and CV use different base classifiers because ACC selects the classifier with the highest accuracy, whereas CV also takes complexity into account and chooses the classifier which is the least complex among the most accurate. Because CV ensembles contain a single base classifier, they are generally less accurate than other ensembles.

- In 32 of the 38 data sets, OPT contains BEST.1, which means that an incremental algorithm beginning with the best individual algorithm has higher probability of finding OPT than an incremental algorithm beginning with a random classifier. On the remaining six data sets, OPT contains the second best algorithm (and on five of them OPT contains the same algorithm as BEST.1 with a different hyperparameter.).
- CV requires statistically significant difference for a classifier to be added and rarely adds more than one classifier (average number of base classifiers found by this method is 1.05), but ACC, which looks at accuracy only, may achieve statistically significant improvement after more than one step of classifier addition.
- Diversity based methods, QSTAT and CORR, do not work well. Most of the other methods are significantly more accurate than them. The similarity between the ensembles found by these two is 0.66, indicating that they tend to find similar (but not very similar) ensembles, but their similarity to OPT is around $0.20-0.25$, showing that they do not choose good base classifiers. Their similarity to other accuracy based methods (other than ALL) is also around 0.2. These results indicate that rather than using such diversity measures alone, it is best to combine them with accuracy in some general goodness measure.
- Although the classifier and discriminant ensembles seem to contain a comparable number of base classifiers, in terms of the discriminants, the discriminant ensembles tend to need around half as many base discriminants, cutting by half the space and time complexity of training/testing. On some data sets, this can be as low as one-fifth.

3.6. Related Work

Since choosing from a subset of classifiers is of exponential complexity, heuristic methods have been developed which are similar to the ones we have evaluated in our study. Below, we give a chronological survey of work on subset selection methods for ensemble construction (Table 3.12 gives a summary) and also related work on variants of stacking.

Table 3.12. Work similar to ICON analyzed in five dimensions: (1) Aim: SS: subset selection, EP: ensemble pruning, EC: ensemble construction, OWC: optimal weight. (2) Optimization criterion: A: accuracy, D: diversity, TP-FP: TP-FP spread, RCM: realistic cost, TC: test cost, AHR: average hit rate. (3) Base classifiers. (4) Optimization method: M: best m, G: genetic algorithm, F: forward, B: backward, R: random, L: floating, tabu search, X: exhaustive, E: early stop, O: optimization algorithms (integer programming etc.), RLO: random linear oracle, C: clustering, DP: tree pruning, BC: best for class, MSE: squared error, MCE: classification error, MDM: margin minimization, CM: complementariness, CAL: calibration

Reference	Aim	Criteria	Base Classifiers	Method	# of data sets
Partridge and Yates (1996)	SS	A	*rbf*, *mlp*	M, G, L	3
Margineantu and Dietterich (1997)	EP	A, D	*dt*	L, F, E	10
Tamon and Xiang (2000)	EP	A	*dt*	B, O	8
Prodromidis and Stolfo (2001)	SS	A, TP-FP, RCM	*bayes*, *ripper*, *dt*,	DP, B	2
Roli et al. (2001)	SS	A, D	*mlp*, *rbf*, *pnn*, *knn*	F, M, BC, B, L, C	1
Zhou et al. (2002)	SS	A	*ann*	G	20
Kim et al. (2002)	EC	A, D	*ann*	G	17
Bakker and Heskes (2003)	SS	A	*ann*	C	2
Caruana et al. (2004)	SS	A, CAL	*svm*, *ann*, *knn*, *dt*	F	7
Goebel and Yan (2004)	SS	D	*ann*	F	1
Ruta and Gabrys (2005)	SS	A, D	*nmc*, *ann*, *knn*, *parzen*, *lp*, *rbf*	F, B, G	27
Demir and Alpaydın (2005)	SS	A, TC	*mlp*, *lp*	F	1
Rokach et al. (2006)	SS	AHR, QRecall, PEM	*dt*	M	1
Zhang et al. (2006)	EP	A	*dt*	O	16
Kuncheva and Rodriguez (2007)	EC	A	*dt*	RLO	42
Martínez-Muñoz and Suárez (2007)	SS	A	*dt*	M, E	18
Yang et al. (2007)	SS, EC	A, TC	SPODE	F, B, R	58
This work	SS	A, D	*knn*, *mlp*, *svm*, *mdt*, *dt*	F, B, L	38

Partridge and Yates [41] use the concept of "methodological diversity" to come up with diverse classifiers and use three different methods for combining *rbf* or *mlp* neural networks. They use a subset selection technique which first trains multiple base classifiers and chooses amongst them. The evaluations are carried out on three data sets, and no cross-validation is mentioned.

Marginentau and Dietterich [42] use the idea that combining diverse classifiers leads to better ensembles: They first use boosting to form an ensemble of 50 decision trees from which they then prune (i.e., using backward search) non-diverse classifiers using five different diversity criteria. They use decision trees as base classifiers and they compare their results on ten data sets with 10-fold cross-validation.

Ting and Witten [62] propose the MLR (Multiresponse Linear Regression) algorithm which combines the outputs of base learners linearly. For each class, there is a linear model to separate it from all other classes, and they choose the maximum for decision (We use a single model with softmax at the output). They use three different base classifiers. They use the same classifiers also on the stacking level and compare them with their proposed linear combiner and deduce that their proposed MLR works the best. Note that there is no subset selection (at the level of classifiers or discriminants) here.

Tamon and Xiang [63] use a method called "weight shifting" to prune ensembles of decision trees constructed using AdaBoost. They evaluate this method by using 10-fold cross-validation on eight data sets, but no statistical testing is mentioned. They also show that subset selection problem is intractable and propose a solution using integer programming.

Sharkey et al. [64] propose the "test and select" approach to ensemble construction. The focus is on selecting a subset from a larger set. If the number of classifiers is small, they use exhaustive search; else they generate and test a number of candidates. One method is to generate random subsets, especially when there are too many base classifiers. They compare their results on two data sets and use neural networks for

the first data set and self organizing maps for the second data set. The data sets are divided into training, validation and testing sets but statistical comparison is not mentioned.

Ueda [65] combines multiple neural networks linearly to come up with an ensemble, similar to the work of Ting and Witten [62]. The approach is to first find best neural networks for each class and combine them linearly using optimal weights. Only neural network classifiers are used in the study and weight decay regularization is used. The method is compared with voting, bagging and majority voting. Leave-one-out cross-validation is used for evaluating the results on one artificial and two real data sets; no statistical testing is provided.

Prodromidis and Stolfo [66] form a meta decision tree for the classifiers in an ensemble and use cost-complexity based pruning to prune the tree. They first model the ensemble (constructed by any method) into a decision tree which mimics the behavior of the original ensemble. They then prune this constructed tree, which results in deletion of some of the classifiers in the original ensemble. The remaining classifiers are used to construct the final ensemble with the same method used to construct the original ensemble. The methods used for constructing the initial ensemble are weighted voting, majority voting, SCANN [22], and stacking, and five base classifiers were used in the study.

Roli et al. [67] define the "overproduce and choose" paradigm where they initially construct a set of candidate classifiers and then select a subset amongst them. Their forward and backward search is the same as ours, using accuracy and diversity criterion; they also implement a forward version which does not start with the best classifier but a random one. They use only one data set and compare their results with ALL and BEST.1, with three different sets of base classifiers. No statistical evaluation is presented in the study and they combine using majority voting.

Zhou et al. [68] show that choosing a subset of classifiers may be more accurate than combining all, and propose a genetic algorithm based method to find the subset

and evaluate their results on ten regression and ten classification data sets. Their base classifiers are neural networks with one hidden layer. They compare their results with bagging and AdaBoost and show that their method finds ensembles which are smaller and which have better generalization ability. They use ten-fold cross validation in their experiments and statistical tests are used to assess the performance.

Kim et al. [69] train multiple ensembles using genetic algorithms. They evaluate their algorithm on 17 data sets using cross-validation and using neural networks as base classifiers. The diversity of the ensembles is achieved using multiple feature subsets. The difference of their method from most of the methods in the literature is that, they build multiple ensembles at the same time. The genetic algorithm determines the membership of classifiers for the ensembles and the selected feature subsets.

Bakker and Heskes [70] use clustering to reduce the size of bootstrap ensembles. They use cluster centers as representatives of each cluster and use a measure of methodological diversity to make the cluster centers optimally diverse. Islam et al. [71] use incremental algorithms based on negative correlation learning to come up with neural network ensembles, taking into account both accuracy and diversity. They evaluate their proposed method on eight data sets. The diversity measure is added to the error function so that networks try to maximize diversity during training.

Caruana et al. [39] propose an incremental algorithm which adds classifiers to the ensemble one at a time, as in ICON. The stopping conditions are different: They stop after a predefined number of steps or when all of the classifiers are chosen. Their work uses thousands of models trained with different hyperparameters using six base algorithms, and they use seven data sets (which are binary or binarized) for evaluation purposes using ten performance metrics which are normalized to $[0, 1]$.

Goebel and Yan [72] use ρ-correlation based diversity measure with an incremental learning algorithm to find an ensemble. They claim that ρ-correlation is a better indicator of contribution to overall accuracy than that of the individual accuracy of the base classifier to be added. They use neural network classifiers as base classifiers

and they evaluate their results on one data set. No statistical testing is done.

Ruta and Gabrys [73] evaluate various methods for combining classifiers and compare error-based and diversity-based selection criteria. They make experiments on 27 data sets using 15 classifiers (different algorithms) using crisp (0/1) outputs. They evaluate various search methodologies including forward and backward search, single best and m-best, and evolutionary algorithms. They use 16 measures (based on accuracy, diversity or both) to assess the performance of constructed systems and their conclusion is the "inappropriateness of diversity measures used as selection criteria in favor of the direct combiner error based search". They also conclude that "greedy algorithms are the best resistant to bad selection criteria". After the selection process they use majority voting for the combination of the classifiers. They divide the data set into 100 different train/test splits (half of test set is used for validation and the other half is used for evaluation). Their empirical results support our findings: It is best to use accuracy as the search criteria; using diversity alone as the selection criteria gives very bad results because it chooses most diverse but inaccurate classifiers. They use crisp (0/1) outputs whereas using posterior probabilities as we do is more informative.

Perhaps the best way to form independent base classifiers is to have them use different representations of the same object or event. Different representations make different characteristics apparent and an object ambiguous in one representation may be clearly recognizable in another [4]. Demir and Alpaydın [14] generate multiple feature sets from different representations, feed them to separate base classifiers and incrementally find the best subset of classifier/representation pairs; different representations may have different costs associated with them and the chosen "cost conscious" ensemble yields the best trade-off between accuracy and cost. They perform their experiments on a handwritten digit data set with multiple representations using multilayer perceptrons as base classifiers, and compare the accuracy with voting over all and boosting.

Rokach et al. [74] propose the selective voting algorithm which orders base models in terms of goodness and combine the best m. The diversity of the base classifiers is

achieved through using multiple input feature sets. In their work, they give results on a two-class data set using 52 different representations, pointing out the advantage of using data from multiple representations. They compare their proposed selective voting algorithm with voting, stacking with decision trees, and weighted voting. They also see that performance degrades when many classifiers are included in the ensemble.

Zhang et al. [75] define the problem of pruning an ensemble as a quadratic integer programming problem and use semi-definite programming to get a better approximate solution.

Kuncheva and Rodriguez [76] propose a hybrid selection-fusion approach to classifier combination. For each classifier, a random linear oracle is created as a hyperplane and the data residing in each half are trained using the ensemble approach. They test their method on 35 data sets from UCI and 7 other medical data sets. 10-fold cross validation was carried out with decision trees used as base classifiers. They see that all of the ensemble methods benefit from this approach, with bagging and the random subspace methods having the highest benefits.

Martínez-Muñoz and Suárez [77] train L classifiers using bagging, and then use AdaBoost to prune the ensemble (i.e. change the random order of bagging and early stop). They use 18 data sets in their experiments and use decision trees as base classifiers; they also check for statistical improvement. Their conclusion is that, by selecting a subset of base classifiers, one can achieve better accuracy, with less complex ensembles. Their method outperforms bagging and is comparable to Adaboost, though when the noise level is high, their method outperforms Adaboost, which makes it a better alternative ensemble method when the noise level is not known.

Sohn and Shin [78] compare bagging and combination using linear regression and see that on large data sets, they work equally well and bagging is better on small data sets (where probably a trained combiner overfits). They find that a trained combiner is suitable when there is strong correlation between input variables.

Yang et al. [79] combine Bayesian network classifiers and compare subset selection (in forward/backward direction using various criteria) vs trained weighted combiner. They make their experiments on 58 benchmark data sets and use four different statistical testing methods for comparison. They conclude that there is no clear winner and that the choice between model selection and model weighing depends on the problem at hand.

3.7. Conclusions

We discuss two ensemble construction methods. In an ensemble of classifiers, we choose a subset from a larger set of base classifiers. In an ensemble of discriminants, we choose a subset of base discriminants, where a discriminant output of a base classifier by itself is assessed for inclusion in the ensemble.

A greedy, forward, incremental classifier ensemble finds ensembles that are small, as accurate as the optimal ensemble, and does this in polynomial time. It is best to maximize the final overall ensemble accuracy, rather than some intermediate diversity criterion; one may also envisage some combination of accuracy and diversity to be able to get the best of both worlds.

When the base classifiers are trained with enough data and are accurate, there is no need for a trained linear combiner, and the fixed sum rule works as well, in a cheaper manner. Our experience in this and other studies is that stacking works better than voting when there is disparity between the accuracies of the base classifiers. When all base classifiers are accurate and equally trustworthy, voting works fine; stacking is needed when we need to weight some, those that are more accurate, higher and some, the erroneous ones, less.

4. COMBINING REPRESENTATIONS

We will show two sets of experiments on two data sets from which we construct multiple representations. The first set uses neural networks (linear perceptron and multilayer perceptrons) as base classifiers, and the second set uses support vector machines as base classifiers. We will show that with base classifiers using different representations (even correlated representations) we have better accuracy than ones using a single representation. First, we use the pendigits data set defined in Section 2.2.4, then we use a processed version of the 3D RMA face data set [80][5] .

4.1. Pen-Based Digit Recognition

We use the data defined in Section 2.2.4, but this time using 10-fold cross-validation. For example, *sv3-s8* denotes the *svm* classifier with polynomial kernel of degree 3 using s8 (static 8×8) representation.

4.2. Face Recognition

For testing the strength of ICON for multiple representations, we also use a preprocessed version of the 3D RMA data set for face recognition, which consists of five or six images of each subject with 106 subjects (classes). One example of each representation is used for test and the remaining four (or five) for training. We apply 4-fold cross-validation where for each fold we have 318 examples for training and 193 examples for validation. The separate test set has 106 examples, one from each class. From the three dimensional data, four representations are extracted [81, 82] (Figure 4.1):

- *Profile Set (P):* The depth values of seven profiles of the face (the center profile, three right and three left profiles.
- *Point Cloud (C):* The (x, y, z) coordinates on a 3-dimensional face.
- *Surface Normals (N):* Surface normals of the face at each point.

[5]This processed version of the 3D RMA face data set is provided by Berk Gökberk.

- *Depth Map (D):* The depth values of the face image.

For these four representations, we also use principal components analysis to decrease dimensionality (to explain 90 per cent of variance) and get four new representations (with subscript R, standing for Reduced). For Point Cloud and Surface Normals, we also use downsampling (with subscript S, standing for subsampled). We therefore have a total of ten different representations.

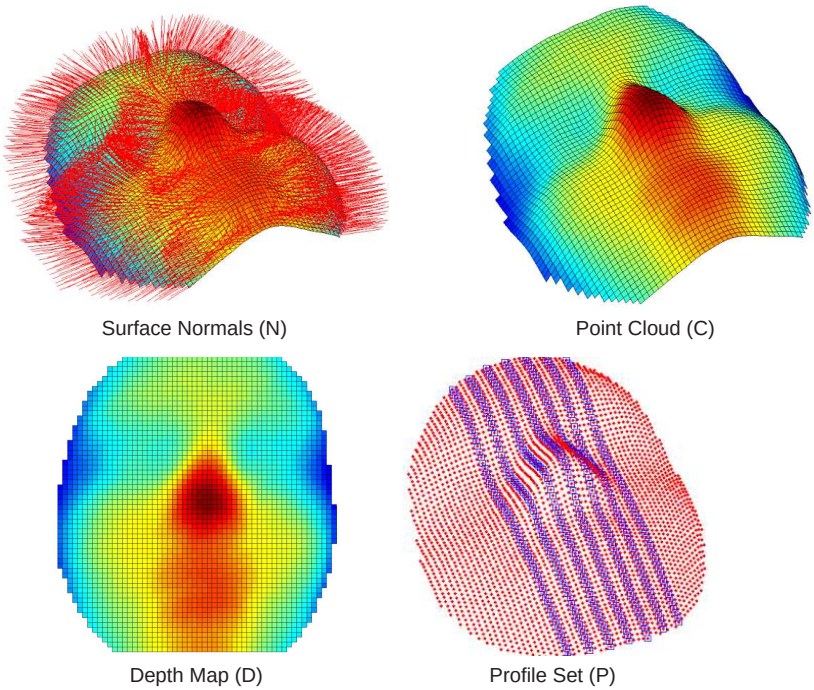


Figure 4.1. Four main face representations

4.3. Neural Network Classifiers

In this section, we present our results on pendigits and 3D RMA data sets using neural network base classifiers. We use linear and multi-layer perceptrons as base classifiers and in them, the number of weights define both the space and computational complexity. With D inputs and K classes, a linear model has $d \equiv (D+1)K$ weights (free parameters) and similarly an *mlp* with H hidden units has $d \equiv (D+1)H+(H+1)K$ weights. We used $H = 24, 30, 36, 42$ in our experiments, and report the one with the highest accuracy.

4.3.1. Pendigits Data Set

Note that when choosing the individual models, we do not need to optimize any hyperparameter such as the number of hidden units. In ensemble construction, it is not the accuracy of the models by themselves that should be optimized but how well they complement each other. As we see next, a model which is not accurate by itself may be included in the ensemble because it is accurate for the cases where another is not.

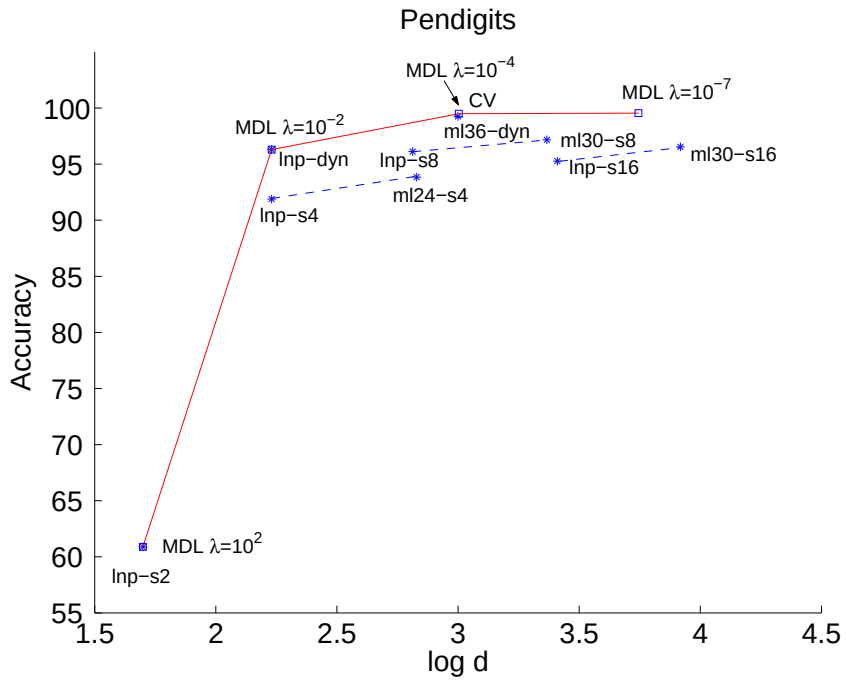


Figure 4.2. Test accuracy vs. log free parameters of single models and ICON results on pendigits. Dashed lines connect models with the same representation. Continuous line connects the results of MDL for various values of λ

Table 4.1 shows the results of individual models and variants of ICON. With CV, the chosen models are first *ml30-dyn*, and then *lnp-s4*. *lnp-s4* is not very accurate but it complements *ml30-dyn* and increases ensemble accuracy significantly. Note also that these two models use different representations and the fact that their combination is useful is automatically discovered by ICON. The accuracy of this combination is significantly better than that of the single best model *ml36-dyn* with $p = 0.98$ and also significantly better than ALL which combines all models with $p = 0.998$. There is no significant difference between CV and ACC. Although the single best classifier

is *ml36-dyn* since it is not significantly more accurate than *ml30-dyn*, CV starts with *ml30-dyn* which has less cost but statistically the same accuracy. This then yields an ensemble which is not much costlier than the single best, but is statistically more accurate. This is the advantage of ICON that, we improve accuracy while not paying too much in terms of cost.

With MDL, the selected models are given in Table 4.2 as a function of $\log \lambda$. As we can see, if we increase the importance of cost, the algorithm tends to choose the simplest model. If we increase the importance of accuracy, the algorithm behaves similar to CV. We see that ACC, additional to the two models chosen by CV, also uses two more models, one of which uses a different representation. Figure 4.2 shows the test accuracies for single models and ICON variants as a function of complexity, where we see that ICON variants attain higher accuracy with less cost.

4.3.2. Face Data Set

Again with this data set, we do not optimize any of the models, we just train a number of candidate models and let ICON do the picking. Note that with this data set, because both the number of inputs and the number of classes is high, *mlp* models have less free parameters than LP models. Again, we see that CV improves accuracy (Table 4.3): It chooses *lnp*-C_S and *lnp*-N_R. CV is significantly better than the single best with $p = 0.96$ and ALL with $p = 0.99$. There is no significant difference between CV and ACC. MDL results are given in Table 4.4. Figure 4.3 shows accuracies for ICON variants and single models as a function of $\log d$.

CV selects an ensemble which is costlier but is statistically significantly more accurate. We again see that CV and ACC add models using different representations. Both CV and ACC are statistically significantly more accurate than the single best classifier and ALL.

Table 4.1. Results of individual models and ICON variants on pendigits

Model	d	Val	Test
lnp-s4	170	91.8±0.8	88.6±0.1
lnp-dyn	170	96.3±0.8	91.7±0.2
lnp-s8	650	96.1±0.3	93.5±0.1
ml24-s4	675	93.8±0.8	89.8±0.6
ml30-dyn	837	**99.2±0.5**	**96.5±0.2**
ml36-dyn	999	**99.2±0.4**	**96.8±0.2**
ml36-s4	999	94.2±0.4	90.6±0.6
ml42-s4	1,161	94.7±0.5	90.6±0.4
ml42-dyn	1,161	**99.2±0.3**	**96.7±0.2**
ml30-s8	2,325	97.1±0.5	95.0±0.1
lnp-s16	2,570	95.2±0.6	92.4±0.2
ml36-s8	2,775	97.2±0.4	95.0±0.3
ml42-s8	3,225	97.3±0.5	95.3±0.2
ml30-s16	8,277	96.5±0.6	94.6±0.2
ml36-s16	9,879	96.8±0.5	94.8±0.3
ml42-s16	11,481	97.0±0.4	94.8±0.2
ALL	49,069	98.9±0.3	97.4±0.1
ACC	5,555	**99.5±0.2**	**98.0±0.1**
ml36-dyn	*lnp-s4*	*ml42-dyn*	*ml42-s8*
CV	1,007	**99.5±0.3**	**97.5±0.3**
ml30-dyn	*lnp-s4*		

Table 4.2. MDL results on pendigits as a function of λ

$\log \lambda$	Combination				d	Val	Test
-6	*ml36-dyn*	*lnp-s4*	*ml42-dyn*	*ml42-s8*	5,555	99.5±0.2	98.0±0.1
-5	*ml36-dyn*	*lnp-s4*	*ml42-dyn*	*ml42-s8*	5,555	99.5±0.2	98.0±0.1
-4	*ml30-dyn*	*lnp-s4*			1,007	99.4±0.3	97.5±0.3
-3	*ml30-dyn*	*lnp-s4*			1,007	99.4±0.3	97.5±0.3
-2	*lnp-dyn*				170	96.2±0.8	91.7±0.1
-1	*lnp-dyn*				170	96.2±0.8	91.7±0.1
0	*lnp-dyn*				170	96.2±0.8	91.7±0.1
1	*lnp-dyn*				170	96.2±0.8	91.7±0.1

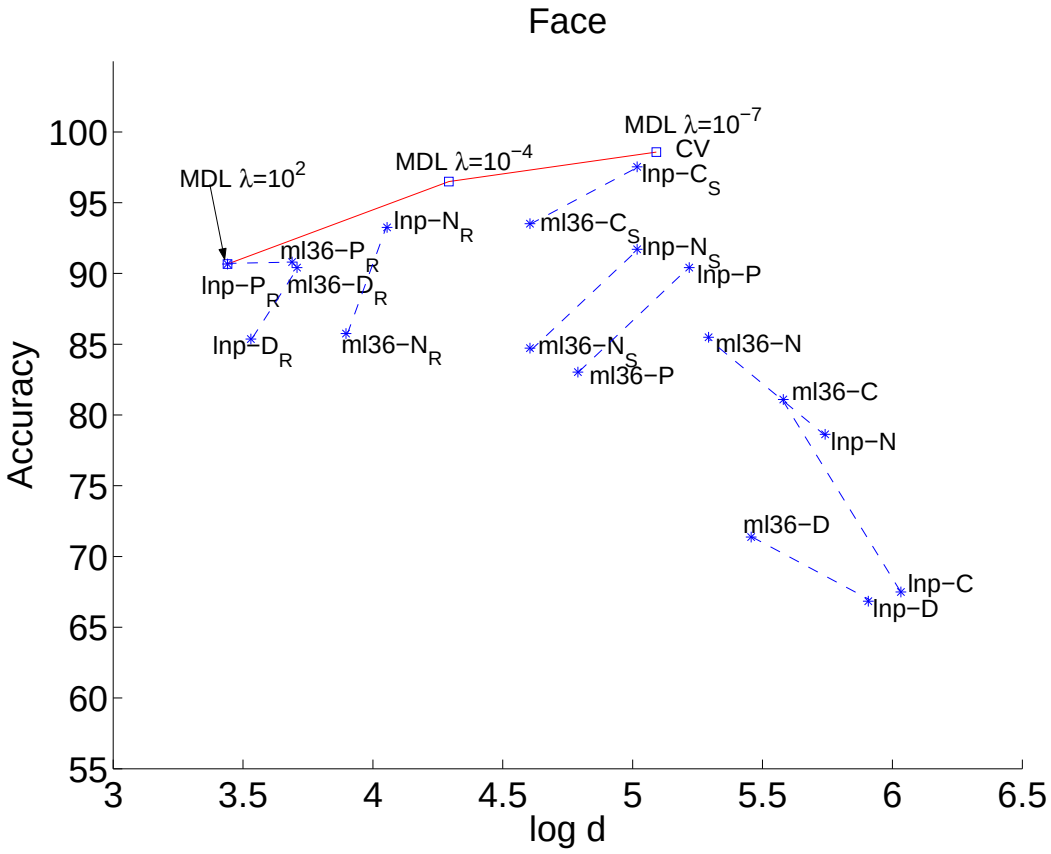


Figure 4.3. Test accuracy vs. log free parameters of single models and ICON results on face. Dashed lines connect models with the same representation. Continuous line connects the results of MDL for various values of λ

Table 4.3. Results of individual models and ICON variants on face

Model	d	Val	Test
lnp-P_R	2,756	90.7±2.0	89.4±4.0
lnp-D_R	3,392	85.3±4.4	85.1±3.8
ml36-P_R	4,884	90.8±1.0	91.7±3.2
ml36-D_R	5,106	90.4±3.7	87.2±3.4
ml36-N_R	7,881	85.7±3.6	87.9±4.7
lnp-N_R	11,342	93.2±1.4	95.9±1.4
ml36-C_R	16,206	27.7±5.2	25.4±3.2
lnp-C_R	35,192	53.2±5.7	57.7±3.8
ml36-C_S	40,256	93.5±2.1	93.6±3.6
ml36-N_S	40,256	84.7±4.2	81.8±6.2
ml36-P	61,568	83.0±5.2	84.6±4.7
lnp-N_S	104,092	91.7±2.7	92.4±1.5
lnp-C_S	104,092	**97.5±1.0**	**97.4±1.4**
lnp-P	165,148	90.4±3.2	92.9±1.9
ml36-N	196,100	85.4±4.0	83.9±4.4
ml36-D	286,010	71.3±4.6	70.5±4.7
ml36-C	380,138	81.0±6.4	78.5±4.1
lnp-N	550,564	78.6±5.4	78.3±3.5
lnp-D	808,144	66.8±5.9	67.4±3.4
lnp-C	1×10^6	67.4±5.6	65.5±7.4
ALL	3.9×10^6	96.5±2.0	97.4±0.9
ACC	123,315	**98.5±0.6**	**97.8±0.4**
lnp-C_S *lnp*-N_R *ml36*-N_R			
CV	115,434	**98.2±1.3**	**97.6±0.9**
lnp-C_S *lnp*-N_R			

Table 4.4. MDL results on face as a function of λ

$\log\lambda$	Combination			d	Val	Test
-6	*lnp*-C_S	*lnp*-N_R	*ml36*-N_R	123,315	98.5±0.6	97.8±0.4
-5	*lnp*-C_S	*lnp*-N_R	*ml36*-N_R	123,315	98.5±0.6	97.8±0.4
-4	*lnp*-N_R	*ml36*-P_R	*lnp*-D_R	19,618	96.5±1.4	96.2±2.0
-3	*lnp*-P_R			2,756	90.6±1.9	89.3±4.0
-2	*lnp*-P_R			2,756	90.6±1.9	89.3±4.0
-1	*lnp*-P_R			2,756	90.6±1.9	89.3±4.0

4.3.3. Conclusions

We see that using a subset of possible models as chosen by ICON works better than taking a vote over all models. With voting, we would use all of the candidate models, which both increase complexity and is not guaranteed to improve accuracy; but with ICON, we construct ensembles which are simple and accurate, because the increase in complexity should be justified by a parallel increase in accuracy.

The strength of combining multiple representations is justified with ICON. The models added to the ensemble use different representations. A model which is very simple and which may never be used as a single model, may contribute to combined accuracy and is added to the ensemble if it complements the models already in the ensemble. For example on pendigits, the model *lnp-s4* is a very simple model but is selected by CV; this creates an ensemble which is statistically more accurate than the single best model, without an increase in cost. The algorithm operates in the "Low-cost High-accurate" part of the search space to come up with accurate ensembles (Top left in Figures 4.3 and 4.2).

On pendigits, CV does not have a significant penalty for cost and selects a significantly more accurate model. On face, CV selects a more accurate but costlier model. CV chooses a simple and accurate ensemble if one is obtainable; if not, CV selects accurate models at the risk of increasing cost.

The main advantage of ICON is that the models need not be optimized for best performance. The user just prepares a palette of models (representation/classifier) and ICON automatically chooses a subset from these for optimized performance. The subset found is also automatically tailored for best accuracy-cost trade-off, depending on the application.

In this section we use simple voting in our experiments. Any other algorithm to combine the decisions of the models e.g., weighted voting, stacking [9], mixture of experts [7] can be used with ICON (we will see in the following sections). Note that voting does not use any parameters but a combiner will have an extra cost.

4.4. Support Vector Classifiers

In this section, we present our results using *svm* base classifiers. We also inspect the effect of fixed rules, and forward/backward search on ensemble accuracy.

4.4.1. Pendigits Data Set

We do not need to optimize kernel type of the *svm* classifier or the type or dimensionality of the input representation, we try all and generate a palette of candidates; it is ICON that chooses the best subset from these, keeping what are most accurate and complementary, ignoring the costly and the redundant.

Table 4.5 shows the results of the individual models where we see that the dynamic representation gives the simplest models and that 4×4 images contain quite a bit of information; we see overfitting as models get more complex.

Table 4.6 shows SUM.F and PRO.F results for ACC, and CV results. In terms of complexity, we have the following ordering: CV $<$ ACC. ACC.SUM is significantly more accurate than CV.SUM ($p = 0.99$), whereas ACC.PRO and CV.PRO find the same ensemble. This shows us that ACC finds more accurate (though more complex) ensembles. Note that with CV, at each step, we check for statistical significance,

Table 4.5. Results of individual models on pendigits

Model	d	Val	Test
svl-dyn	7,600	98.4±0.5	95.3±0.3
sv2-dyn	7,696	99.5±0.3	97.6±0.1
sv3-dyn	**8,400**	**99.6±0.3**	**97.6±0.1**
sv4-dyn	8,432	99.6±0.2	97.6±0.2
svr-dyn	14,240	99.7±0.2	98.3±0.1
sv2-s4	19,312	96.0±0.6	92.3±0.2
sv3-s4	19,952	95.3±0.6	91.3±0.3
sv4-s4	20,752	94.7±0.6	90.5±0.3
svl-s4	20,960	93.7±0.9	90.3±0.1
svr-s4	49,856	96.5±0.5	93.5±0.2
svl-s8	70,720	97.2±0.3	94.8±0.2
sv2-s8	139,328	98.4±0.4	96.6±0.1
sv3-s8	145,344	98.2±0.3	96.3±0.1
sv4-s8	155,328	98.1±0.4	95.8±0.1
svr-s8	173,952	98.4±0.4	96.9±0.1
svl-s16	596,224	96.9±0.5	94.4±0.2
svr-s16	1,137,920	96.1±0.6	94.2±0.1
sv2-s16	1,870,850	87.6±1.1	84.2±0.7
sv4-s16	1,878,020	83.8±3.5	80.4±3.2
sv3-s16	1,918,460	71.9±6.4	67.4±4.7

whereas for ACC, it is enough that average accuracy increases. It may be the case that a single model increases the accuracy but not significantly and a further second model increases it significantly; ACC catches such ensembles though CV stops early. So in terms of accuracy, we have the ordering as: ACC > CV. These results support the findings of [73]: "The results clearly show that searching directly according to the combiner error is optimal and returns well performing combinations".

Table 4.6. Results of SUM.F and PRO.F using different model selection criteria on pendigits

ICON			d	VAL	TEST
			ENSEMBLE FOUND		
FORWARD	SUM	ACC	618K	99.8±0.1	98.5±0.1
			svr-dyn svl-s16 sv4-dyn		
		CV	22K	99.7±0.2	98.1±0.1
			sv3-dyn svr-dyn		
	PRO	ACC	618K	99.8±0.1	98.9±0.1
			svr-dyn svl-s16 sv3-dyn		
		CV	618K	99.8±0.1	98.9±0.1
			sv3-dyn svr-dyn svl-s16		

Table 4.7 shows the effect of fixed rules on CV.F and ACC.F. With CV, we see that all fixed rule variants except PRO find the same ensemble *sv3-dyn svr-dyn*. CV.PRO adds another model (with a different representation) to this ensemble and has higher accuracy ($p \geq 0.97$) (Note that two fixed rule methods choosing the same ensemble does not necessarily indicate that they have the same accuracy; that is why CV.PRO can add the third model, whereas the others cannot). With ACC though, things change: All the fixed rule methods find different ensembles. All but MAX are significantly more accurate than MIN (though MIN finds a much simpler ensemble). There is no significant difference between PRO, MAX, SUM, MED, except that MED is significantly more accurate than MAX. For this data set, considering accuracy, we have the ordering: PRO, SUM, MED > MAX > MIN. PRO, in this case outperforms the others because the base classifiers do not create outlier posteriors [58].

Table 4.7. Results of Cv.F and Acc.F variants on pendigits

Icon			d	VAL	TEST
			ENSEMBLE FOUND		
FORWARD	Acc	Sum	618K	99.8±0.1	98.6±0.1
			svr-dyn svl-s16 sv4-dyn		
		Max	618K	99.8±0.1	98.6±0.1
			svr-dyn svl-s16 sv3-dyn		
		Min	21K	99.8±0.2	98.0±0.1
			svr-dyn sv2-dyn		
		Med	638K	99.8±0.1	98.7±0.1
			svr-dyn svl-s16 sv4-dyn sv2-s4		
		Pro	618K	99.8±0.1	98.9±0.1
			svr-dyn svl-s16 sv3-dyn		
	Cv	Sum	22K	99.7±0.2	98.1±0.1
			sv3-dyn svr-dyn		
		Max	22K	99.7±0.2	98.1±0.1
			sv3-dyn svr-dyn		
		Min	22K	99.8±0.2	98.0±0.1
			sv3-dyn svr-dyn		
		Med	22K	99.7±0.2	98.1±0.1
			sv3-dyn svr-dyn		
		Pro	618K	99.8±0.1	98.9±0.1
			sv3-dyn svr-dyn svl-s16		

Table 4.8 shows the effect of backward vs forward search with ACC using different rules. With SUM and MAX, backward finds ensembles that are more in number but less in terms of parameters than forward (With MIN the found ensemble is similar). We see that there is no significant difference between the accuracies of forward and backward results.

Table 4.8. Comparison of forward and backward search on pendigits

ICON			d	VAL	TEST
			ENSEMBLE FOUND		
ACC	FORWARD	SUM	618K	99.8±0.1	98.5±0.1
			svr-dyn svl-s16 sv4-dyn		
		MAX	618K	99.8±0.1	98.5±0.1
			svr-dyn svl-s16 sv3-dyn		
		MIN	21K	99.8±0.2	98.0±0.1
			svr-dyn sv2-dyn		
		MED	638K	99.8±0.1	98.7±0.1
			svr-dyn svl-s16 sv4-dyn sv2-s4		
		PRO	618K	99.8±0.1	98.9±0.1
			svr-dyn svl-s16 sv3-dyn		
	BACKWARD	SUM	216K	99.8±0.1	98.5±0.1
			svl-dyn sv2-dyn svr-dyn sv3-s4 sv3-s8		
		MAX	161K	99.8±0.1	98.6±0.1
			sv3-dyn svr-dyn sv2-s8		
		MIN	22K	99.8±0.2	98.0±0.1
			sv3-dyn svr-dyn		
		MED	695K	99.8±0.1	98.3±0.1
			svl-dyn sv3-dyn sv4-dyn svr-dyn sv2-s4 sv3-s4 svl-s16		
		PRO	637K	99.8±0.1	98.9±0.1
			sv2-dyn svr-dyn sv2-s4 svl-s16		

Overall, we see that the fixed rule and search direction changes the results of CV and ACC. ACC.F and CV.F tend to find models with different representations.

We also present MDL.SUM.F results with different λ in Table 4.9. As we increase λ, complexity is penalized more and we get simpler, but less accurate ensembles. Figure 4.4 shows all individual models and the path MDL.F follows as λ varies.

Table 4.9. MDL.SUM.F results on pendigits as a function of λ

$\log\lambda$	Combination	d	Val	Test
-9	*svr-dyn svl-s16 sv4-dyn*	618K	99.8±0.1	98.5±0.1
-8	*svr-dyn svl-s16 sv4-dyn*	618K	99.8±0.1	98.5±0.1
-7	*svr-dyn sv2-dyn sv2-s4*	41K	99.8±0.2	98.5±0.1
-6	*svr-dyn sv2-dyn sv2-s4*	41K	99.8±0.2	98.5±0.1
-5	*svr-dyn*	14K	99.5±0.2	98.2±0.1
-4	*sv4-dyn*	8K	98.7±0.2	96.7±0.2
-3	*sv2-dyn*	7.7K	91.8±0.3	89.9±0.1
-2	*sv2-dyn*	7.7K	91.8±0.3	89.9±0.1
-1	*svl-dyn*	7.6K	98.4±0.5	95.3±0.3
0	*svl-dyn*	7.6K	98.4±0.5	95.3±0.3
1	*svl-dyn*	7.6K	98.4±0.5	95.3±0.3
2	*svl-dyn*	7.6K	98.4±0.5	95.3±0.3

Table 4.10 compares ACC.F results with three cases: BEST.1, ALL, and OPT (Instead of all 20, we did an exhaustive search over 12 of the 20, choosing the most accurate three models from each representation). Here we also see the effect of combination rule. We expect that MIN rule gives very bad results (as we will see in face) but since the individual base classifiers are strong even the MIN rule using all of the classifiers finds good ensembles. Still all of the ACC.F variants are statistically more accurate than using all models. There is no significant difference between the accuracies of OPT and ACC.F results. With MAX and PRO, ACC and OPT find the same ensemble. With the remaining three fixed rule combinations ACC has same accuracy with comparable complexity as in [73]. "Virtually all the search algorithms work well with this (combiner error) criterion and the combinations of classifiers they return are very similar or identical to the optimal combinations found by exhaustive search".

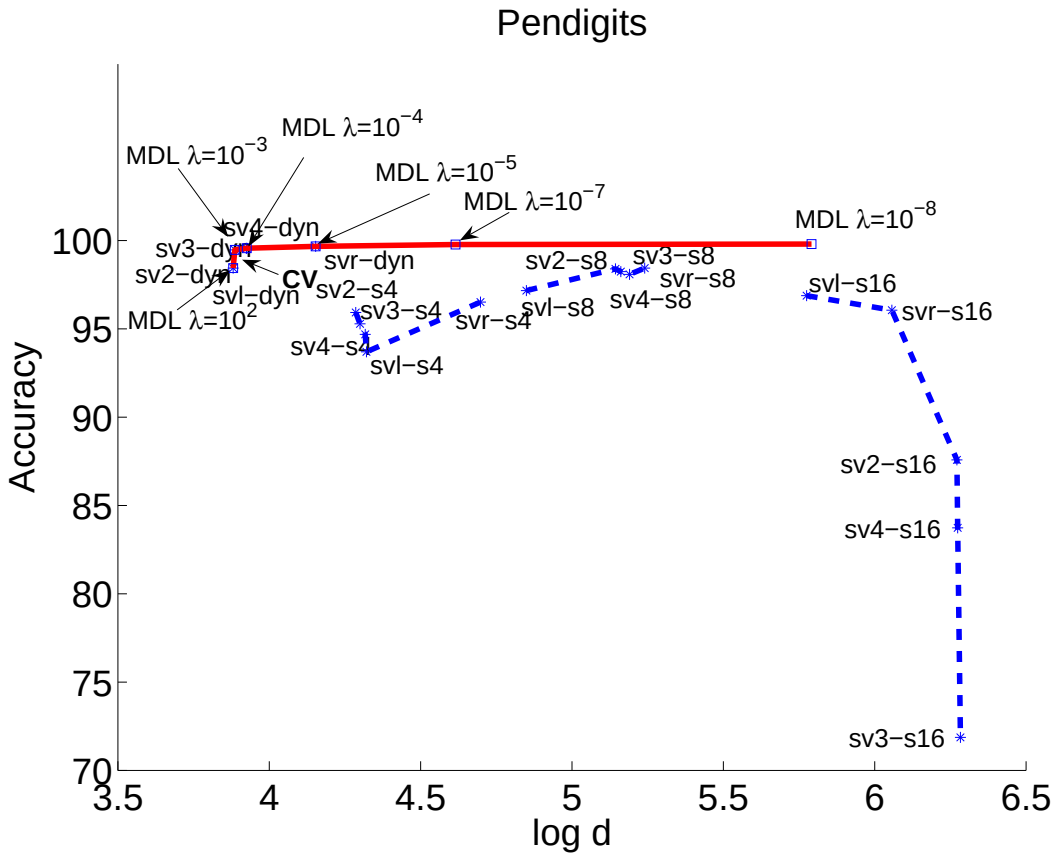


Figure 4.4. Test accuracy vs. log free parameters of single models and MDL.SUM.F results on pendigits. Dashed lines connect models with the same representation. Continuous line connects the results of MDL.SUM.F for various values of λ

Table 4.10. Results of comparing ACC with OPT and ALL on pendigits

			d	VAL	TEST
ICON			ENSEMBLE FOUND		
BEST.1			8K	99.55±0.3	97.60±0.1
			sv3-dyn		
FORWARD	ALL	SUM	8M	99.2±0.2	97.9±0.1
		MAX	8M	99.2±0.2	97.6±0.1
		MIN	8M	99.0±0.4	97.4±0.1
		MED	8M	99.0±0.2	97.5±0.1
		PRO	8M	99.3±0.2	98.2±0.1
	ACC	SUM	647K	99.8±0.1	98.5±0.1
			svl-s16 sv3-s4 svr-dyn sv4-dyn sv3-dyn		
		MAX	618K	99.8±0.1	98.5±0.1
			svl-s16 svr-dyn sv3-dyn		
		MIN	22K	99.8±0.2	98.0±0.1
			svr-dyn sv3-dyn		
		MED	1M	99.8±0.1	98.8±0.1
			sv2-s16 sv2-s4 svr-dyn sv3-dyn		
		PRO	618K	99.8±0.1	98.9±0.1
			svl-s16 svr-dyn sv3-dyn		
	OPT	SUM	618K	99.8±0.1	98.5±0.1
			svr-dyn svl-s16 sv4-dyn		
		MAX	618K	99.8±0.1	98.5±0.1
			svr-dyn svl-s16 sv3-dyn		
		MIN	21K	99.8±0.2	98.0±0.1
			svr-dyn sv2-dyn		
		MED	638K	99.8±0.1	98.7±0.1
			svr-dyn svl-s16 sv4-dyn sv2-s4		
		PRO	618K	99.8±0.1	98.9±0.1
			svr-dyn svl-s16 sv3-dyn		

4.4.2. Face Data Set

Table 4.11 shows the single model results, where we see that models have a wide range of complexities from 10^3 to 10^6 with high accuracies early on and clearly overfit with complex models (representations).

Table 4.12 shows SUM.F and PRO.F results for ACC and CV. We can again see that in terms of complexity, the ordering is: CV < ACC. CV.SUM finds only one model which is the single best, but with other fixed rules CV finds ensembles consisting of multiple models. ACC finds ensembles which are more complex, but have high accuracy. ACC.SUM is significantly more accurate than CV.SUM ($p = 0.95$) and ACC.PRO is significantly more accurate than CV.PRO ($p = 0.96$), but note that ACC ensembles are ten times more complex. We see that an ACC ensemble contains models from different representations to best complement each other.

Table 4.13 shows the effect of fixed rules on CV.F and ACC.F. With CV, we see that all fixed rule variants are significantly more accurate than SUM. The other fixed rules do not have any significant difference between them. With ACC though, things are different. All are significantly more accurate than MIN. PRO is significantly more accurate than all, except SUM. SUM is better than MAX; MED and MAX are comparable. For this data set, considering accuracy, we have: PRO, SUM > MED, MAX > MIN.

Table 4.14 shows the effect of backward vs forward search with ACC using different fixed rules. Except with MIN, forward finds simpler ensembles than backward, with the exception of MAX where the found ensemble is the same. We also see that SUM.F finds a significantly more accurate ensemble than SUM.B. We therefore conclude that it is better to use forward search. This is also true with respect to time when there are many models; if small ensembles are good, backward search will iterate more than forward search. Most of the time F and B find ensembles which have similar accuracies, but backward stops early and finds more complex ensembles.

Table 4.11. Results of individual models on face

Model	d	Val	Test
svr-P_R	10600	80.2±5.7	81.6±5.3
svl-P_R	10600	77.9±6.7	78.3±7.7
svl-N_R	44944	81.7±5.7	85.4±6.2
svr-N_R	44944	78.0±4.4	80.4±5.6
svl-C_R	140344	85.1±5.3	86.1±4.0
svr-C_R	140344	82.0±4.9	85.9±3.4
***sv4*-C_S**	**405153**	**90.9±7.1**	**93.4±5.7**
sv3-C_S	407115	92.8±5.2	93.2±4.3
sv2-C_S	413001	92.6±5.1	93.2±4.3
sv4-N_S	413982	81.5±9.5	85.9±8.9
sv3-N_S	414963	85.8±6.1	87.0±3.6
svl-C_S	415944	92.4±5.3	92.9±4.3
svr-N_S	415944	86.3±6.1	88.4±3.7
sv2-N_S	415944	85.8±6.1	87.0±3.6
svl-N_S	415944	85.5±6.0	86.8±4.1
svl-P	660168	74.6±5.3	80.2±7.1
svr-N	2.46E+06	79.4±4.8	84.2±2.7
sv3-N	2.46E+06	77.9±4.9	82.3±4.8
svl-N	2.46E+06	77.7±5.5	82.8±4.0
sv2-N	2.46E+06	77.7±4.5	82.8±4.0
sv2-C	4.31E+06	88.0±5.1	86.6±5.0
sv3-C	4.31E+06	88.0±5.1	86.3±4.8
svl-C	4.31E+06	87.8±5.4	86.8±4.6
sv4-C	4.31E+06	85.5±8.0	88.4±7.6
svr-C	4.31E+06	84.2±5.1	86.3±4.4

Table 4.12. Results of SUM.F and PRO.F using different model selection criteria on face

ICON			d	VAL	TEST
			ENSEMBLE FOUND		
FORWARD	SUM	ACC	4M	95.0±4.3	94.3±2.6
			sv3-C_S *sv4*-C_S *sv2*-N_R *svr*-C_R *svr*-D *svl*-P_R		
		CV	405K	90.9±7.1	93.4±5.7
			sv4-C_S		
	PRO	ACC	5M	95.5±4.4	94.1±2.5
			sv3-C_S *sv4*-C_S *svl*-N_R *svr*-C_R *sv3*-D *sv4*-N_R ..		
		CV	812K	93.5±5.4	93.4±4.4
			sv4-C_S *sv3*-C_S		

On the other hand, CV.F variants select ensembles which are costlier but are statistically significantly more accurate. We again see that CV.F and ACC.F add models using different representations. Only CV.SUM.F finds an ensemble with one classifier which is the same as the single best.

We also present ACC.SUM.F results with different λ in Table 4.15. As expected, as we increase λ, complexity is penalized more and we get simpler but inaccurate ensembles. Figure 4.5 shows all individual models and the path MDL.SUM.F follows as λ varies.

Table 4.16 compares ACC.F results against BEST.1, combining ALL models, and OPT (Instead of all 47 we used 20 of the individual models, choosing the most accurate two models from each representation). ALL.MIN gives very inaccurate results, because of the nature of the rule. But with ICON, since we are selecting the best to insert next, the inaccurate models are not included in the final ensemble unless they are complementing another strong model. That's why a subset selection strategy is better, and ICON finds a good subset of all the models in the given palette. All of the ACC.F variants are statistically better than using ALL. There is no significant difference between OPT and ACC.F results and ensembles found by ACC.F are simpler.

Table 4.13. Results of Cv.F and Acc.F variants on face

Icon			d	VAL	TEST
			ENSEMBLE FOUND		
FORWARD	Acc	Sum	4M	95.0±4.3	94.3±2.6
			sv3-C_S *sv4*-C_S *sv2*-N_R *svr*-C_R *svr*-D *svl*-P_R		
		Max	8M	93.8±4.6	93.2±3.1
			sv3-C_S *sv3*-N_R *svr*-C *svr*-D *sv4*-C_S		
		Min	4M	93.1±4.8	92.0±4.1
			sv3-C_S *svl*-C		
		Med	7M	94.0±4.7	93.0±4.1
			sv3-C_S *sv3*-N_R *svl*-C *sv4*-N		
		Pro	5M	95.5±4.4	94.1±2.5
			sv3-C_S *sv4*-C_S *svl*-N_R *svr*-C_R *sv3*-D *sv4*-N_R ..		
	Cv	Sum	405K	90.9±7.1	93.4±5.7
			sv4-C_S		
		Max	4M	92.6±6.2	93.4±4.4
			sv4-C_S *svr*-C		
		Min	5M	92.8±6.6	92.7±5.8
			sv4-C_S *sv2*-C_S *svl*-C		
		Med	4M	92.6±6.2	93.4±4.4
			sv4-C_S *svr*-C		
		Pro	812M	93.5±5.4	93.4±4.4
			sv4-C_S *sv3*-C_S		

Table 4.14. Comparison of forward and backward search on face

ICON			d	VAL	TEST
			ENSEMBLE FOUND		
ACC	FORWARD	SUM	4M	95.0±4.3	94.3±2.6
			sv3-C_S *sv4*-C_S *sv2*-N_R *svr*-C_R *svr*-D *svl*-P_R		
		MAX	8M	93.8±4.6	93.2±3.1
			sv3-C_S *sv3*-N_R *svr*-C *svr*-D *sv4*-C_S		
		MIN	4M	93.1±4.8	91.0±4.1
			sv3-C_S *svl*-C		
		MED	7M	94.0±4.7	92.9±4.1
			sv3-C_S *sv3*-N_R *svl*-C *sv4*-N		
		PRO	5M	95.5±4.4	94.1±2.5
			sv3-C_S *sv4*-C_S *svl*-N_R *svr*-C_R *sv3*-D *sv4*-N_R ..		
	BACKWARD	SUM	26M	94.3±4.5	93.6±2.1
			sv4-P_R *sv3*-P_R *sv2*-P_R *svl*-P_R *sv2*-D_R *svl*-D_R ..		
		MAX	8M	93.8±4.6	93.1±3.1
			sv3-N_R *sv4*-C_S *sv3*-C_S *svr*-D *svr*-C		
		MIN	877K	91.8±3.7	92.5±2.0
			svl-P_R *svl*-N_R *sv3*-C_S *sv3*-N_S		
		MED	34M	94.2±3.4	93.9±2.2
			sv4-P_R *sv2*-P_R *svl*-P_R *svr*-P_R *svl*-D_R *svl*-N_R ..		
		PRO	24M	95.0±4.7	93.2±2.4
			svl-P_R *sv3*-D_R *svl*-N_R *sv2*-N_R *svl*-C_R *sv2*-C_R ..		

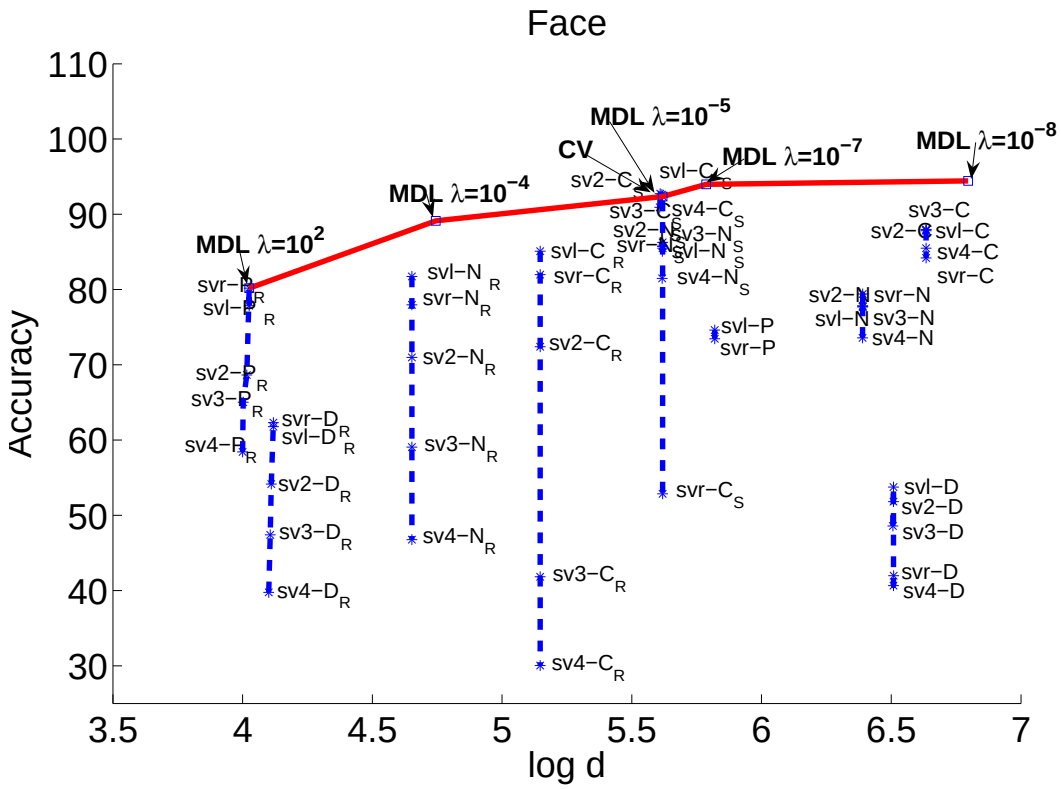


Figure 4.5. Test accuracy vs. log free parameters of single models and ICON results on face. Dashed lines connect models with the same representation. Continuous line connects the results of ACC.SUM.F for various values of λ

Table 4.15. ACC.SUM.F results on face as a function of λ

$\log\lambda$	Combination	d	Val	Test
-10	*svl*-C_S *svr*-C_R *svr*-N *svr*-D *svl*-P_R	6M	94.4±4.0	94.1±2.4
-9	*svl*-C_S *svr*-C_R *svr*-N *svr*-D *svl*-P_R	6M	94.4±4.0	94.1±2.4
-8	*svl*-C_S *svr*-C_R *svr*-N *svr*-D *svl*-P_R	6M	94.4±4.0	94.1±2.4
-7	*svl*-C_S *svr*-C_R *svl*-N_R *svr*-P_R	611K	94.0±4.3	93.1±3.1
-6	*svl*-C_S *svr*-C_R *svl*-N_R *svr*-P_R	611K	94.0±4.3	93.1±3.1
-5	*svl*-C_S	415K	88.2±5.3	88.8±4.3
-4	*svr*-P_R *svl*-N_R	55K	83.6±4.4	83.6±2.9
-3	*svr*-P_R	10K	69.6±5.7	71.0±5.3
-2	*svr*-P_R	10K	69.6±5.7	71.0±5.3
-1	*svr*-P_R	10K	69.6±5.7	71.0±5.3
0	*svr*-P_R	10K	69.6±5.7	71.0±5.3
1	*svr*-P_R	10K	69.6±5.7	71.0±5.3
2	*svr*-P_R	10K	69.6±5.7	71.0±5.3

4.4.3. Conclusions

We use two data sets with multiple representations to test our algorithm ICON, with backward and forward search, with different model selection techniques such as ACC, CV, and using different fixed rule combination techniques such as SUM, MAX, MIN, MED, PRO. We see that combining models using different representations can affect ensemble accuracies significantly.

We again see that the advantage of combining multiple representations is justified with ICON. The models added to the ensemble use different representations. A model which is very simple may contribute to combined accuracy and is added to the ensemble if it complements the models already in the ensemble. For example on pendigits, the model *sv2-s4* is a very simple model (uses a 4×4 image) but is selected by ACC.MED.F; this creates an ensemble which is statistically more accurate than the single best model, without a significant increase in cost.

Table 4.16. Results of comparing ACC with OPT and ALL on face

			d	VAL	TEST
ICON			ENSEMBLE FOUND		
BEST.1			8K	99.55±0.3	97.60±0.1
			sv4-C_S		
FORWARD	ALL	SUM	56M	92.6±3.8	92.9±2.2
		MAX	56M	84.6±4.5	89.2±3.7
		MIN	56M	41.8±2.8	46.9±3.6
		MED	56M	92.4±4.0	92.7±2.6
		PRO	56M	91.3±4.4	92.7±2.7
	ACC	SUM	3M	95.0±3.3	92.9±2.9
			svr-N *sv3*-N_S *sv2*-C_S *sv3*-C_S *svr*-C_R		
		MAX	407K	92.8±5.2	93.2±4.3
			sv3-C_S		
		MIN	4M	92.9±4.6	91.8±4.6
			sv3-C *sv3*-C_S		
		MED	13M	94.6±3.7	93.6±2.1
			svl-C *svr*-D *svl*-D *svr*-N *svl*-C_S *svr*-C_R		
		PRO	4M	94.7±3.5	92.7±2.7
			svr-N *svr*-N_S *sv3*-N_S *sv2*-C_S *sv3*-C_S *svr*-C_R *svl*-N_R		
	OPT	SUM	4M	95.0±4.3	94.3±2.6
			sv3-C_S *sv4*-C_S *sv2*-N_R *svr*-C_R *svr*-D *svl*-P_R		
		MAX	8M	93.8±4.6	93.2±3.1
			sv3-C_S *sv3*-N_R *svr*-C *svr*-D *sv4*-C_S		
		MIN	4M	93.1±4.8	92.0±4.1
			sv3-C_S *svl*-C		
		MED	16M	92.0±3.3	92.0±4.2
			sv2-C *sv3*-C *sv3*-N *svr*-N *svl*-P *svr*-P ...		
		PRO	5M	95.5±4.4	94.1±2.5
			sv3-C_S *sv4*-C_S *svl*-N_R *svr*-C_R *sv3*-D *sv4*-N_R ..		

In general PRO, SUM and MED rules find more accurate ensembles, whereas MAX and MIN rules are most of the time significantly worse than the other three.

ICON finds ensembles which are statistically significantly not worse than OPT, and most of the time ICON finds simpler ensembles.

Forward and backward ICON tend to find ensembles with similar accuracies. But if the number of base models is high, it is better to use forward (backward may stop early). If the number of models is not so high, backward ICON may find ensembles which have same accuracy but are simpler.

We can also see from our experiments that with ICON, we do not need to define an explicit diversity measure. The diversity between the classifiers is detected automatically using ICON according to the model selection methodology used.

In this section we use fixed combination rules in our experiments. Any other algorithm to combine the decisions of the models e.g., weighted voting, stacking [9] can be used with ICON.

Because of the high dimensionality of the pendigits and face applications which increase the number of free parameters, CV.F and ACC.F are able to find ensembles that are more accurate than the single best model without significantly increasing cost. We believe that this has the potential to improve performance in data sets like face where training set is small with respect to the number of weights. It would also be interesting to apply ICON in other sensor fusion applications, for example speech or other biometric applications.

5. EXTRACTING METACLASSIFIERS FOR AGGREGATE DECISIONS

5.1. Comparison of Combination Rules on Real Data Sets

We compare the accuracies of all fusion methods, namely, the fixed rules, SUM, MIN, MAX, MED, and the trained linear combiner LIN, on the *test* sets of all 38 data sets. The pairwise comparison results are given in Table 5.1 where entry (i, j) shows the number of wins, that is, the number of data sets on which method i is significantly more accurate than j using the 5×2 cv F test: $38 - (i, j) - (j, i)$ gives the ties. The entries in boldface show statistically significant difference between the methods using sign test on 38 trials with 0.95 confidence.

Table 5.1. Pairwise comparisons of fusion rules

	SUM	MIN	MAX	MED	LIN
SUM	0	**22**	**20**	2	0
MIN	0	0	1	1	0
MAX	0	6	0	1	1
MED	0	**20**	**18**	0	0
LIN	5	**24**	**23**	4	0

We see that LIN seems to be better than the fixed rules, though it is not significantly more accurate than SUM and MED. As we have seen in Section 2.1.2, when there are groups of classifiers with different intergroup correlations, a trained linear combiner is more accurate than a fixed rule. Then, next, there is no statistically significant difference between SUM and MED methods, and these methods are significantly more accurate than MIN and MAX methods. We do further statistical analysis with nonparametric tests. Friedman's test rejects the hypothesis that the five have equal ranks. Doing Nemenyi's post-hoc test, we get the results in Figure 5.1 [51]. We see that MED, LIN and SUM construct a group and they are statistically significantly more accurate than the other group which consists of MIN and MAX.

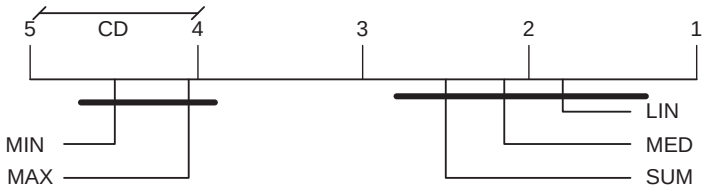


Figure 5.1. Graphical representation of post-hoc Nemenyi test results for fusion rules. The numbers on the line represent the average ranks, CD is the critical difference for statistical significance, and bold lines connect the methods which have no significant difference

5.2. Extracting Metaclassifiers for Aggregate Decisions

Given a set of positively correlated base classifiers, the usual approach is to keep a subset, pruning those that are correlated with those in the subset. However, unless there is perfect correlation (of 1.0), this causes a loss of information as those base classifiers which can potentially help in new cases are removed, decreasing fault tolerance. The approach we propose is to keep all the base classifiers even if there is correlation between them but combine their predictions taking into account the fact that they are not independent.

We consider the outputs of base classifiers as dimensions of a new feature space in which a new classification problem is defined and we view the problem of combining base classifier outputs as the problem of choosing input features. With this view in mind, when we are interested in extracting the best features (that is, choose the most informative base classifier values), one way is to do *feature selection*, where we keep some of the features and discard the rest. Actually, this is the idea that underlies methods which choose a subset of base classifiers from a large ensemble of candidates [42, 67, 39, 73, 14, 40].

In this section, we advocate the other approach of combining the base classifier outputs to define new *metaclassifiers*. This is similar to *feature extraction* in pattern recognition where we define new features that are combinations of the original features. In particular, we use Principal Component Analysis (PCA) which defines new aggregate

dimensions that are linear combinations of the original features, which in our case correspond to outputs of the base classifiers.

We use PCA as follows: Given the 14 classifiers trained on the training fold tra_i, we calculate their outputs for the correct class as a 14-dimensional vector on *val-*A_i and their correlation matrix, which is a 14×14 matrix. The leading M eigenvectors of this matrix are the new metaclassifier directions. We map the original input to this space by first calculating classifier outputs and then multiplying all the K outputs of a classifier by the corresponding value of these eigenvector. The output is then used to train a linear classifier in this new space, using again *val-*A_i. The input to the linear combiner (which is a linear perceptron in our case) has $M \cdot K$ dimensions where we decide on M, the number of eigenvectors (components, metaclassifiers), based on the average accuracy on *val-*B_i, the other half of the validation fold (unused during training of the base classifiers or the linear combiner). We report and check for significant difference on the accuracies of the ten folds on the *test*.

In using LDA, we use the full $14 \cdot K$ dimensional output of the base classifiers, where K is the number of classes, and not the 14-dimensional outputs for the correct class as we do with PCA. This is because LDA uses the class information and for it to work, instances of different classes need to look different. When we use the full vector, instances belonging to different classes have different representations and LDA can work. The disadvantage is that now the eigenvectors are $14 \cdot K$ dimensional and we need more calculation and they are not as easily interpretable[6] . One problem with LDA is that with K classes, one can have up to $K-1$ new aggregate dimensions which makes LDA not accurate on data sets with few classes.

One advantage of a linear combiner is that there is no need for scaling or any other normalization [56, 57]. As we have shown in Section 2.1.2, when there is a number of groups with different intragroup and intergroup correlations, a trained linear combiner works better than any fixed rule in decreasing error. The superiority of the linear

[6]Doing the same, that is using the full $14 \cdot K$ vector with PCA, does not increase or decrease accuracy.

perceptron over other (linear) combiners has been shown by Raudys [83].

The eigenvectors of the correlation matrix can also be analyzed for information extraction. In Figure 5.2, we show the first five eigenvectors of the correlation matrix averaged over all data sets (Table 2.2) (These are not the eigenvectors used; actually, for each data set, in each fold, there is a different correlation matrix and a different set of eigenvectors). The numbers in the top right corner of the figure is the proportion of variance explained by the components up until then.

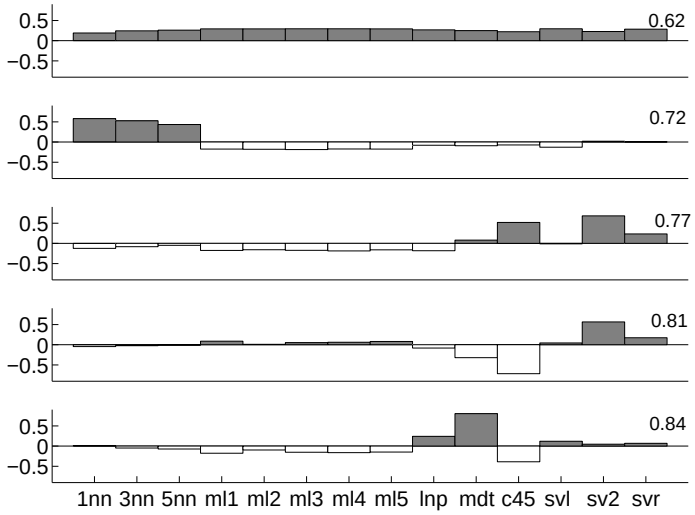


Figure 5.2. The first five eigenvectors of the correlation matrix of all fourteen classifiers averaged over all data sets, shown in Table 2.2. The gray boxes show positive dimensions, and the white boxes show negative dimensions. The numbers on the top-right of each subfigure represents the proportion of variance explained upto and including that eigenvector

In all data sets, we always see that the first eigenvector is a vector of roughly equal positive values; doing a dot product with it is almost like a simple averaging. It is known [84] that if we have a covariance matrix where all variances are equal and all correlations are also of similar magnitude, the first principal component is proportional to the mean of the input variables. In our case, the proportion of variance explained by the first eigenvector is 0.62 on the average and this value is higher as the base classifiers become more correlated. We interpret this as follows: Even when the classifiers are

correlated, the best aggregate decision is to have a simple average and this gives us more than half of the information provided by the classifiers, as measured by the proportion of variance explained. A similar conclusion has also been reached at by Fumera and Roli [85], where they say that "simple averaging is the optimal linear combining rule only if the individual classifiers exhibit identical error rates and identical correlations between estimation errors." This also implies that in cases when this is not true, taking only an average corresponds to discarding the variance carrying dimensions that the other components represent.

We can extract more information by looking at the eigenvectors that follow (see Figure 5.2): In the second one, we have all *knn* variants with positive weights, perceptrons with negative weights, and all others are close to zero; we interpret this as the nearest neighbor dimension. In the third, *svm*s and the two trees (*c45* and *mdt*) have positive weights, perceptrons have negative weights and the others have small negative weights; this is the *svm*-tree dimension. The fourth separates trees from *svm*s. Once we make a distinction between the major groups (and having explained 81 percent of the variance), the fifth separates *lnp* from other perceptrons, and univariate and multivariate trees.

We compare our results using PCA with three other methods:

- BEST.1 is the accuracy of the most accurate single base classifier.
- ALL is the accuracy when all fourteen base classifiers are combined.
- OPT is the accuracy of the optimum subset, that is, the one found by exhaustive search over all $2^{14} - 1$ possible subsets.

Note that as with PCA and LDA, there are linear perceptrons trained to combine the outputs of the base classifiers also with BEST.1, OPT and ALL, and they are also trained on *val-A* folds.

5.2.1. Case Studies

Before proceeding to our complete results on all 38 data sets, we start by presenting our results on two data sets, *pageblock*, and *spambase*, in more detail, to get an initial feel.

5.2.1.1. Pageblock Data Set. The correlation matrix is given in Table 5.2 and the first five eigenvectors of the correlation matrix is given in Figure 5.3. This is a data set where the base classifiers are strongly correlated and the first component explains 79 per cent of the variance. The second separates *knn* variants from all others and the third separates *c45* from *mlp* variants. The fourth one makes a distinction between single-layer and multilayer perceptrons and *svm* with polynomial and Gaussian kernels. The fifth one separates univariate and multivariate trees.

Table 5.2. Average correlations on the *pageblock* data set

	kn1	*kn3*	*kn5*	*ml1*	*ml2*	*ml3*	*ml4*	*ml5*	*lnp*	*mdt*	*c45*	*svl*	*sv2*	*svr*
kn1	1.00	**0.80**	**0.74**	0.59	0.57	0.57	0.57	0.59	0.55	0.53	0.50	0.56	0.52	0.59
kn3	**0.80**	1.00	**0.95**	**0.76**	**0.74**	**0.75**	**0.74**	**0.77**	**0.71**	**0.69**	**0.61**	**0.73**	**0.68**	**0.77**
kn5	**0.74**	**0.95**	1.00	**0.81**	**0.79**	**0.79**	**0.78**	**0.81**	**0.74**	**0.74**	**0.65**	**0.77**	**0.73**	**0.83**
ml1	0.59	**0.76**	**0.81**	1.00	**0.93**	**0.94**	**0.92**	**0.92**	**0.82**	**0.79**	**0.68**	**0.89**	**0.78**	**0.86**
ml2	0.57	**0.74**	**0.79**	**0.93**	1.00	**0.95**	**0.93**	**0.91**	**0.79**	**0.81**	**0.70**	**0.89**	**0.78**	**0.85**
ml3	0.57	**0.75**	**0.79**	**0.94**	**0.95**	1.00	**0.94**	**0.93**	**0.81**	**0.79**	**0.69**	**0.88**	**0.76**	**0.86**
ml4	0.57	**0.74**	**0.78**	**0.92**	**0.93**	**0.94**	1.00	**0.92**	**0.82**	**0.79**	**0.67**	**0.88**	**0.77**	**0.85**
ml5	0.59	**0.77**	**0.81**	**0.92**	**0.91**	**0.93**	**0.92**	1.00	**0.84**	**0.80**	**0.67**	**0.88**	**0.77**	**0.87**
lnp	0.55	**0.71**	**0.74**	**0.82**	**0.79**	**0.81**	**0.82**	**0.84**	1.00	**0.73**	0.59	**0.85**	**0.76**	**0.80**
mdt	0.53	**0.69**	**0.74**	**0.79**	**0.81**	**0.79**	**0.79**	**0.80**	**0.73**	1.00	**0.60**	**0.81**	**0.78**	**0.83**
c45	0.50	**0.61**	**0.65**	**0.68**	**0.70**	**0.69**	**0.67**	**0.67**	0.59	**0.60**	1.00	**0.68**	**0.63**	**0.66**
svl	0.56	**0.73**	**0.77**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.85**	**0.81**	**0.68**	1.00	**0.90**	**0.88**
sv2	0.52	**0.68**	**0.73**	**0.78**	**0.78**	**0.76**	**0.77**	**0.77**	**0.76**	**0.78**	**0.63**	**0.90**	1.00	**0.82**
svr	0.59	**0.77**	**0.83**	**0.86**	**0.85**	**0.86**	**0.85**	**0.87**	**0.80**	**0.83**	**0.66**	**0.88**	**0.82**	1.00

We see the error of the PCA method compared to those of BEST.1, OPT and ALL in Figure 5.4, where we see that accuracy increases by including more components. On this data set, PCA, LDA, OPT and ALL are significantly more accurate than BEST.1. PCA with four metaclassifiers is as accurate as ALL.

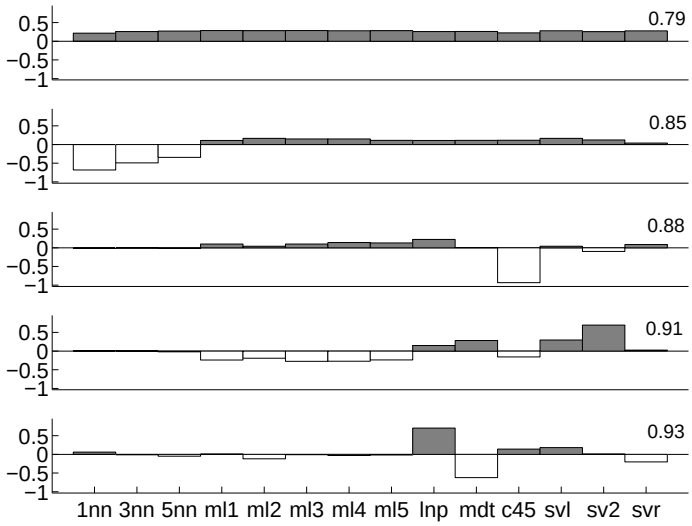


Figure 5.3. First five eigenvectors of the correlation matrix on the *pageblock* data set

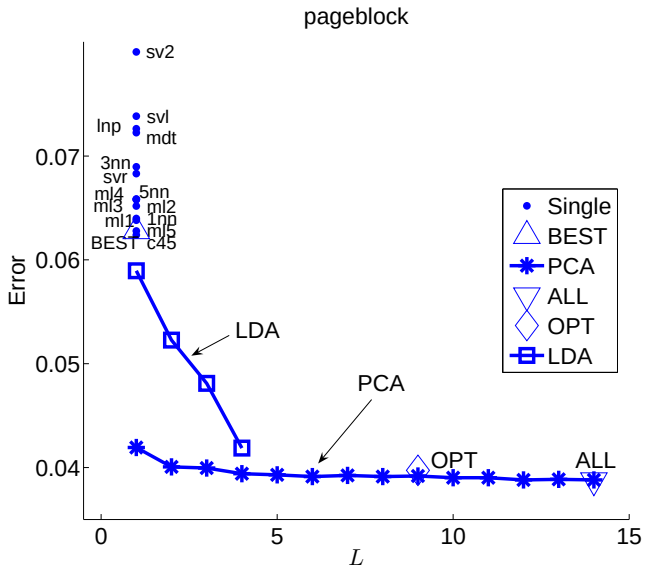


Figure 5.4. Classification errors of base classifiers, PCA, LDA, OPT and ALL on *pageblock*

5.2.1.2. Spambase Data Set. We can see the correlation matrix in Table 5.3 and the first five eigenvectors of the correlation matrix in Figure 5.5. The first eigenvector is again approximately the simple average. The second one separates *knn* variants from the others. The third one separates *knn*s, *mlp*s and the univariate tree from multivariate tree and *svm*s with polynomial kernel. The eigenvectors that follow can similarly be interpreted.

Table 5.3. Average correlations on the *spambase* data set

	kn1	*kn3*	*kn5*	*ml1*	*ml2*	*ml3*	*ml4*	*ml5*	*lnp*	*mdt*	*c45*	*svl*	*sv2*	*svr*
kn1	1.00	**0.74**	**0.65**	0.45	0.43	0.46	0.45	0.44	0.35	0.38	0.23	0.45	0.31	0.44
kn3	**0.74**	1.00	**0.90**	**0.61**	**0.60**	**0.61**	**0.60**	**0.60**	0.49	0.54	0.31	**0.63**	0.46	**0.61**
kn5	**0.65**	**0.90**	1.00	**0.66**	**0.65**	**0.65**	**0.64**	**0.64**	0.54	0.59	0.33	**0.70**	0.52	**0.65**
ml1	0.45	**0.61**	**0.66**	1.00	**0.84**	**0.88**	**0.87**	**0.87**	**0.76**	**0.67**	0.43	**0.79**	0.50	**0.74**
ml2	0.43	**0.60**	**0.65**	**0.84**	1.00	**0.84**	**0.84**	**0.83**	**0.75**	**0.68**	0.42	**0.80**	0.49	**0.75**
ml3	0.46	**0.61**	**0.65**	**0.88**	**0.84**	1.00	**0.91**	**0.87**	**0.76**	**0.66**	0.44	**0.79**	0.49	**0.75**
ml4	0.45	**0.60**	**0.64**	**0.87**	**0.84**	**0.91**	1.00	**0.87**	**0.76**	**0.67**	0.43	**0.78**	0.49	**0.73**
ml5	0.44	**0.60**	**0.64**	**0.87**	**0.83**	**0.87**	**0.87**	1.00	**0.75**	**0.65**	0.42	**0.79**	0.48	**0.74**
lnp	0.35	0.49	0.54	**0.76**	**0.75**	**0.76**	**0.76**	**0.75**	1.00	**0.61**	0.36	**0.72**	0.44	**0.65**
mdt	0.38	0.54	0.59	**0.67**	**0.68**	**0.66**	**0.67**	**0.65**	**0.61**	1.00	0.32	**0.77**	0.52	**0.68**
c45	0.23	0.31	0.33	0.43	0.42	0.44	0.43	0.42	0.36	0.32	1.00	0.40	0.23	0.43
svl	0.45	**0.63**	**0.70**	**0.79**	**0.80**	**0.79**	**0.78**	**0.79**	**0.72**	**0.77**	0.40	1.00	**0.68**	**0.78**
sv2	0.31	0.46	0.52	0.50	0.49	0.49	0.49	0.48	0.44	0.52	0.23	**0.68**	1.00	0.37
svr	0.44	**0.61**	**0.65**	**0.74**	**0.75**	**0.75**	**0.73**	**0.74**	**0.65**	**0.68**	0.43	**0.78**	0.37	1.00

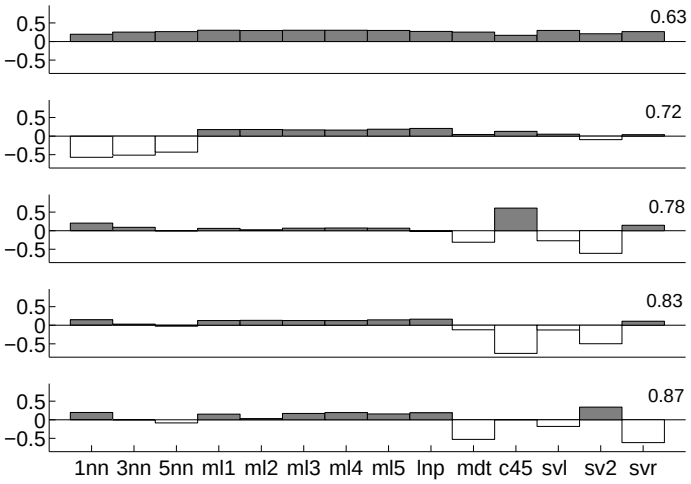


Figure 5.5. First five eigenvectors of the correlation matrix on the *spambase* data set

The classification errors of the proposed Pca, Best.1, Opt and All are given in Figure 5.6. On this data set, there is no statistically significant difference between the

accuracy of the combination algorithms. Note that PCA even with a single eigenvector is quite accurate; this is an example of a data set where the fixed fusion rules, average or median, would be expected to work well.

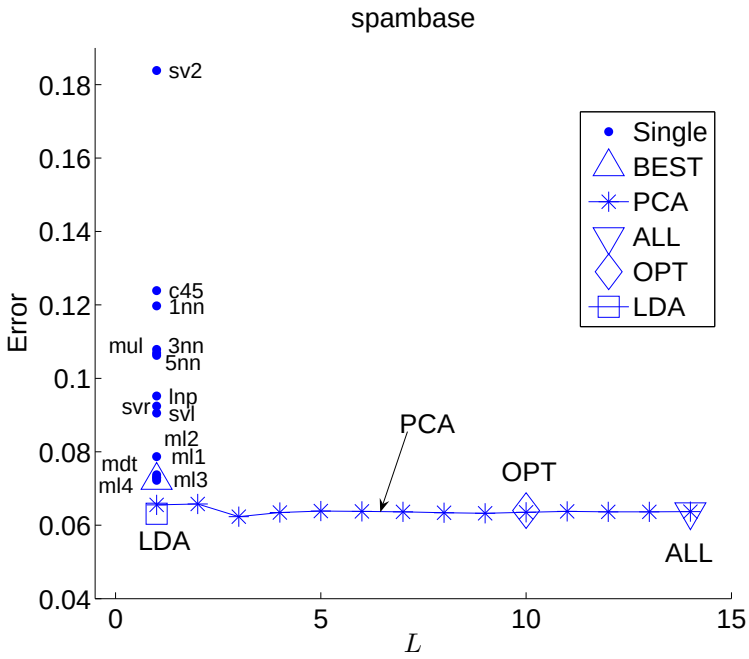


Figure 5.6. Classification errors of base classifiers, PCA, LDA, BEST.1, OPT and ALL on *spambase*

5.2.2. Overall Results

Comparing our PCA method with BEST.1, ALL and OPT on all 38 data sets, we get the pairwise comparison results in Table 5.4. We see that there is no significant difference between PCA, ALL and OPT but all three are significantly more accurate than BEST.1. ALL is not more accurate than PCA, or the other way around. Friedman's test rejects the hypothesis that the four methods have equal ranks. Doing Nemenyi's post-hoc test for pairwise comparison, we get the results in Figure 5.7. The test finds that PCA, OPT and ALL belong to one group, and this group is significantly more accurate than BEST.1. We see that PCA and ALL have the same accuracy, where generally a few components is enough with PCA; see Figure 5.8 for the histogram of number of components used by PCA on 38 data sets.

Table 5.4. Pairwise comparisons of BEST.1, PCA, OPT and ALL

	BEST.1	PCA	ALL	OPT	LDA
BEST.1	0	1	1	1	1
PCA	**8**	0	0	0	4
ALL	**8**	0	0	0	5
OPT	**13**	4	2	0	5
LDA	5	0	0	0	0

It can be said that the major advantage of a subset selection method over PCA is that once a subset is chosen during training, afterwards during test, not all, but only those in the subset need be evaluated, whereas PCA needs to calculate all base classifier outputs before doing the dot product and calculating the metaclassifier outputs. Note that if the aim is feature *selection*, there are methods which use PCA for this purpose [86]: We can decrease the ensemble size before doing PCA; for example, given that *c45* and *mdt* are both in the tree dimension, or, members of the tree "family," we can get rid of one, increasing the weight of the other.

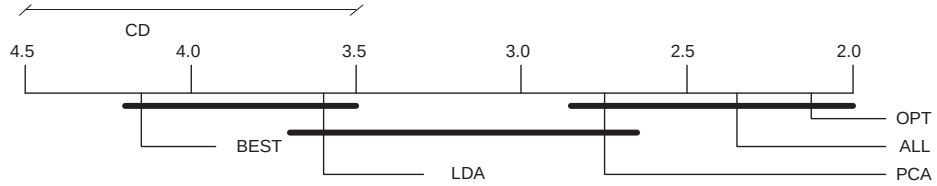


Figure 5.7. Graphical representation of post-hoc Nemenyi test for PCA, LDA, BEST.1, and OPT

5.2.3. Multiple Representations

On the *pendigits* data set with multiple representations, which we have discussed in Section 2.3.1.5, the same approach can also be used, this time to combine classifiers using different representations, and the eigenvectors of the correlation matrix can be analyzed for information extraction. In Table 5.5, we show the eigenvectors of the average correlation matrix given in Table 2.3; see also Figure 5.9.

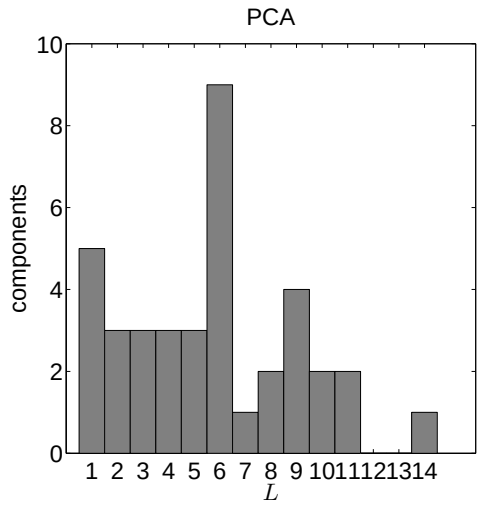


Figure 5.8. Histogram of the number of components used PCA on all 38 data sets

Table 5.5. The proportion of variance explained and the eigenvectors of the average correlation matrix given in Table 2.3

Comp	Var	*dyn*	*s4*	*s8*	*s16*
1	0.65	0.37	0.49	0.57	0.54
2	0.84	0.92	-0.28	-0.18	-0.19
3	0.96	-0.09	-0.81	0.24	0.54
4	1.00	-0.03	-0.18	0.76	-0.62

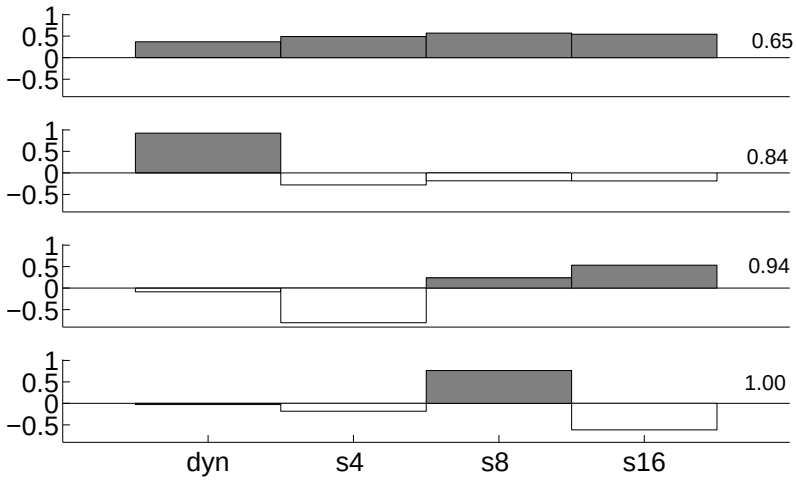


Figure 5.9. The eigenvectors of Table 5.5 visualized. The gray boxes show positive dimensions, and the white boxes show negative dimensions. The numbers on the right of each subfigure represents the proportion of variance explained upto and including that eigenvector

We again see that the first component is nearly the average of all classifiers explaining 65 percent of the variance. The second has a positive *dyn* dimension and all *static* dimensions are negative. If we want to go further than 84 percent variance, the third component has a negative *s4* dimension and positive *s16* and *s8* dimensions. The last distinguishes between *s16* and *s8*.

Figure 5.10 shows the errors of the methods on this data set. We see that Opt choosing *dyn* and *s8* is significantly more accurate than other methods. Next, there are All and Pca which are comparable and significantly more accurate than Best.1 and Lda. Best.1 and Lda are not significantly different. So according to accuracy, we have the following ordering: Opt > Pca, All > Lda, Best.1. Because the two, temporal and image-based, representations, namely *dyn* and *s8*, have low correlation between them, fusing them (as Pca does) is not better than taking both (as Opt does); this would be even more true for the case of different modalities.

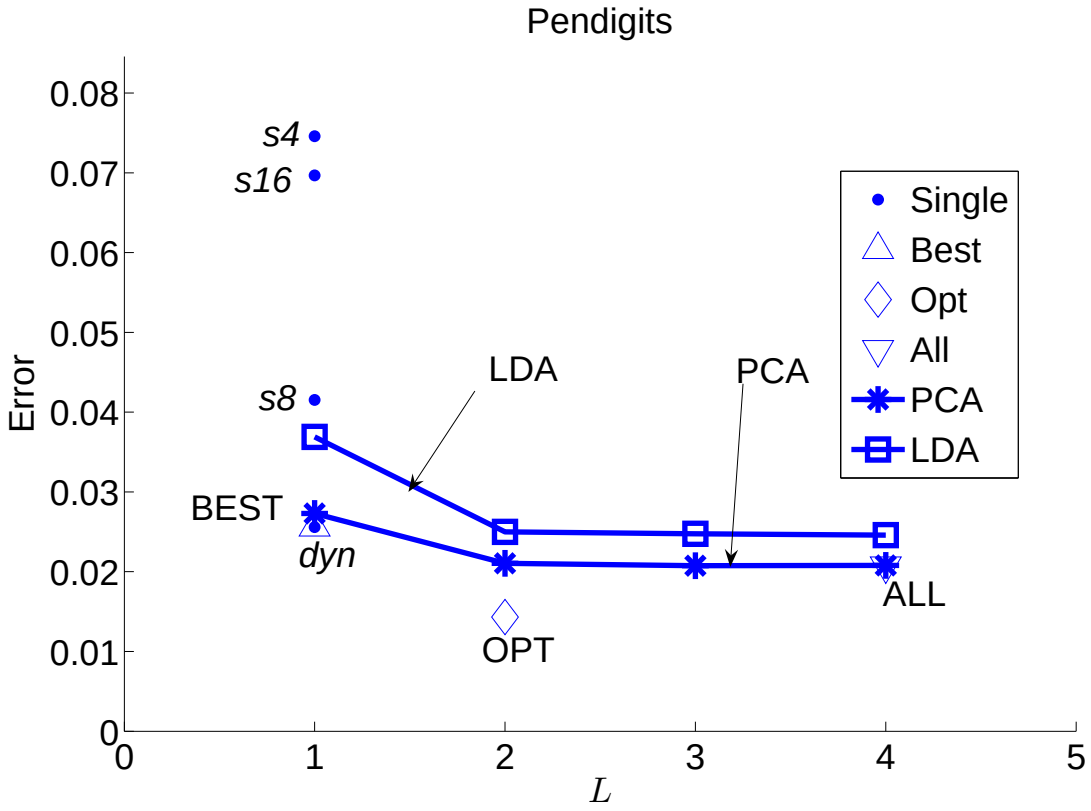


Figure 5.10. Classification errors of base classifiers and ensemble methods on *pendigits*

5.2.4. Related Work

All methods which consider the outputs of previously trained base learners as inputs to a new learner are variants of stacking [9, 62] and so are the methods we discuss above. Merz [22], [87] discuss work that are most similar to ours. Merz [22] proposes the SCANN algorithm which uses correspondence analysis on the crisp outputs of base classifiers and combines them using the nearest mean classifier. This corresponds to doing PCA on the 0/1 outputs of the base classifiers and one can show that the nearest mean classifier is also a linear classifier. In his work, Merz uses neural networks, rule lists, decision trees and nearest neighbor classifiers as base classifiers. The combination results are compared with voting, and using naive Bayes and multilayer perceptron as stackers. In theory, numeric outputs give more information and are preferred [62], but in our experiments, we have found no difference between using 0/1 or continuous outputs; this probably is because our outputs are posterior probabilities (normalized using softmax) and are close to 0/1 anyway. Note however that because we use a

trained linear combiner and not a fixed rule (e.g., sum, max, etc.), there is no need that the base classifier outputs be normalized. Note also that Merz works on the full $14 \cdot K$ dimensions instead of 14.

Merz and Pazzani [87] propose the PCR* algorithm which uses PCA on the outputs of base regressors. They show that the algorithm is able to cope with the inherent correlations amongst the base regressors, which is what we try to achieve. They compare their proposed method with other regression combination algorithms in the literature. After the reduction of the output dimensionality using PCA, they also use a linear combiner for the final decision. Since theirs is a regression problem, there is only one output in their study. As future work, they write that their algorithm can be used in classification problems using a separate model for each class. The method we propose pools all class information and learns a single set of metaclassifiers; we believe that this is better than learning a separate set of metaclassifiers for each class because, (i) it is a simpler solution, has less parameters, pools data, and therefore has less risk of overfitting, and (ii) we expect correlations between two classifiers to be similar for different classes: Let us say A and B are two algorithms each with its own inductive bias (assumption about the data); the way the two are correlated on examples from class 1 will be the same, or very similar, to the way they will be correlated on examples from class 2.

5.3. Conclusions

The fact that experts trained on the same (or slightly different versions of the same) task are positively correlated is well-known but has largely been swept under the rug. In this work, we (i) study the effect of this correlation on fusion rules, (ii) investigate how such correlation manifests itself on real-world data using well-known training algorithms on frequently used data sets, and (iii) how such correlated experts can be combined to define new dimensions that are uncorrelated so as to allow representing the same information succinctly for high accuracy and knowledge extraction.

As our first contribution, we analyze in detail the error using different fusion rules under the condition of correlation between experts. We investigate how the ensemble performs as dependency between classifiers, posterior value, and ensemble size change. We show that, as expected, the accuracy goes down as (positive) correlation between the classifiers increases. This is because as correlation increases, the classifiers begin to resemble each other, diversity decreases, and the correcting effect of the ensemble degrades. Instead of using similar classifiers, it is more reasonable to use diverse classifiers to improve accuracy, which implies that negative correlation is desirable.

As our second contribution, we test the effect of five factors on the correlation between classifiers, which are similarity in the algorithms, hyperparameters, overlapping folds, shared input features, and input representations. We see that no matter how we may vary algorithms, hyperparameters, folds, or input features, we still get positively correlated classifiers, and post-processing is needed to make them uncorrelated.

This correlation analysis allows us to see how ensembles should be formed so that combination is useful. For example, we see that bagging trees is a good idea but bagging support vector machines is not good; with the latter, it is better to play with the kernel or inputs. With knn, neither resampling nor varying k suffices to get uncorrelated versions, one should vary some other factor, for example, input features, or one should combine knn with some other algorithm, a linear perceptron for example. In general, it seems to be better to combine different algorithms or different inputs, rather than different training subsets or hyperparamaters. In case we have many classifiers from different families, we should prefer to use classifiers from different families instead of same family because then the correlation between groups is less than the correlation between variants of the same family; this is a fact also supported by our results in the first part.

Our third contribution is in the investigation of methods where we construct new uncorrelated metaclassifiers from a set of correlated classifiers. Our experimental results show that our PCA-based method is as accurate as using the whole ensemble or the optimum subset, and also allows knowledge extraction. We define metaclassifiers,

that is, linear combinations of existing base classifiers which are uncorrelated. Note that using a feature extraction method such as PCA instead of a feature selection method such as subset selection may be costlier to use but a subset keeps some and discards the rest and has the potential to remove information that may be useful on some instances; keeping and combining all allows redundancy and promises to be more fault tolerant.

The PCA-based method as we implement it, also allows knowledge extraction: We see that the first principal component corresponds to simple averaging which shows us the strength of simple voting. Looking at the other components, we can find new aggregate dimensions in terms of families of algorithms that best complement simple voting. Such families then allow us to decide how best to design an ensemble: I.e., it is better to have one from each family, rather than multiple algorithms from the same family.

We have also tried using linear discriminant analysis (LDA) instead of PCA; LDA is sometimes advantageous in that it uses class information but in practice, it does not work as well as PCA because the number of components should be less than K, the number of classes, which makes the approach based on LDA unsuccessful in cases where K is small. One can also use nonlinear dimensionality reduction methods, though in such a case we might lose from interpretability of the new metaclassifiers.

It is also possible to combine multiple representations of the same input by training experts on, for example, different representations of the same handwritten digit [4]. Here, we would like to make a distinction between a *representation* and a *modality*: Representations are differently preprocessed versions of the same signal and therefore we expect to have some correlation between them; we would expect less or no correlation if inputs come from different modalities; for example, we expect no correlation between a person's face image and his/her signature. The analysis we do in this work can also be carried out for the case of multiple representations, that is, checking how much correlation there is and how experts using different representations can be fused using PCA.

6. CONCLUSIONS

In this chapter, we will first begin with the contribution of this thesis, we will continue with an overall comparison of the proposed methods, and finish with future work.

6.1. Contributions of This Thesis

We present a greedy algorithm ICON for selecting a subset of classifiers, given a previously trained base classifier set. The algorithm has three dimensions: (i) The search direction, (ii) Ensemble evaluation method, and (iii) Combination method. We see that using backward search creates larger ensembles and has no advantage over forward search, and floating search is no different than forward search but visits more steps. As ensemble evaluation criteria, we used two diversity measures (CORR and QSTAT), two accuracy based measures (CV and ACC), and a measure which combines cost and accuracy (MDL). We see that the criterion based on diversity alone is not good for best combination performance; we conclude that it is best to use the accuracy as the model selection criteria. We also see that when the cost of the combination is important, MDL with a suitable λ trades-off complexity and accuracy, to come up with simple and accurate classifiers. Of the combination methods, amongst fixed rules, we see that SUM usually works the best and is robust to noisy base classifiers. PRO also is good, but when one of the base classifiers is inaccurate, the accuracy of the combined ensemble diminishes rapidly. MED rule works well and is robust, but MIN and MAX rules have low accuracy. The linear combiner is useful when we try to combine all classifiers, because it weeds out the inaccurate classifiers by assigning small weights, but a subset selection algorithm also has the same power with less cost. Also there is the need to have separate data to train the linear combiner. If the number of classes and number of base classifiers is high, the linear combiner overfits. So if one has a small number of classes and classifiers, or if one has inaccurate base classifiers, it may be useful to use a linear combiner, otherwise a fixed rule run over a selected subset is simple, and sometimes is even more accurate.

We see that using a subset of given base classifiers as chosen by ICON is better than the single best classifier and taking a vote over all the classifiers, and is not worse than using the optimum subset. Using all the candidate models increases complexity and is not guaranteed to improve accuracy; but when we use ICON, we have to justify an increase in complexity with an increase in accuracy.

We see that combining base classifiers using different representations increases accuracy significantly. We see on two multi-representation data sets that, the ensembles created by ICON contain base classifiers from different representations and are significantly more accurate than the single best classifier and using all classifiers. We also see that a base classifier which is inaccurate and would not be chosen on its own may be complementary to other base classifiers and increases accuracy significantly. For example on pendigits, the model *lnp-s4* is a very simple model but is selected by CV; this creates an ensemble which is statistically more accurate than the single best model, without an increase in cost.

The main advantage of ICON is that, we do not need to optimize the base classifiers, we just let ICON do the picking. The subset is selected according to the accuracy-cost trade-off depending on the application.

In general PRO, SUM and MED rules find more accurate ensembles, whereas MAX and MIN rules are most of the time significantly worse than the other three. ICON finds ensembles which are not statistically significantly worse than OPT, and most of the time ICON finds simpler ensembles. Forward and backward ICON tend to find ensembles with similar accuracies. But if the number of base models is high, it is better to use forward (backward may stop early). If the number of models is not so high, backward ICON may find ensembles which have the same accuracy and are simpler.

A greedy, forward, incremental method finds ensembles that are small, as accurate as the optimal ensemble, and does this in polynomial time. It is best to maximize the final overall ensemble accuracy, rather than some intermediate diversity criterion, or some combination of accuracy and diversity to be able to get the best of both worlds.

When the base classifiers are trained with enough data and are accurate, there is no need for a trained linear combiner, and the fixed sum rule works as well, in a cheaper manner. Our experience in this study is that stacking works better than voting when there is disparity between the accuracies of the base classifiers. When all base classifiers are accurate and equally trustworthy, voting works fine; stacking is needed when we need to weight some, those that are more accurate, higher and some, the erroneous ones, less.

By analyzing correlations between experts in an ensemble, we see that, whatever factor we play with (algorithms, hyperparameters, training set samples, features subsets, or representations), we still have positive correlation. We have also analyzed theoretically that, if we have positive correlation, ensemble accuracy decreases. Our proposed method PCA first gets rid of this correlation and constructs uncorrelated metaclassifiers, and is as accurate as the optimum classifier, with less cost.

By using a correlation analysis as we propose in this thesis, we see how to form ensembles for combination to be useful. For example, bagging trees is a good idea, but bagging support vector machines is not. With support vector machines, it is better to change inputs or kernels. For algorithms like *knn*, resampling and changing k does no good, so we need a different methodology, for example combining it with other algorithms. Combining different inputs and different algorithms seems to be better than combining sampled data and different hyperparameters. On the other hand, using different algorithms has the obvious disadvantage of coding and optimizing. Using algorithms with the same data set, but with half of the features is good for local algorithms such as *knn* and *svr*.

We also construct new, uncorrelated metaclassifiers from a set of correlated classifiers. We see in our experiments that PCA method is as accurate as using the whole ensemble or the optimal subset. This means that this method is as accurate as any subset selection algorithm. A subset selection algorithm (like ACC) has the potential to remove classifiers, which may contribute to accuracy, but our post-processing approach considers all the classifiers. PCA algorithm also allows us to extract knowledge

from the ensemble. We see that, most of the time, the first component corresponds to simple voting, which shows us the strength of simple voting. The other components allow us to find new aggregate dimensions in terms of families of algorithms.

6.2. Overall Comparison

We have seen that Acc obtains ensembles which are as accurate as All and Opt, but simpler. Also we have seen that Pca constructs ensembles which have the same accuracy as All with a few components. As further comparison, we compare the two proposed methods with Opt, All and Best.1. For the sake of fairness of comparison, we use the .Lin versions of Opt, All, Acc and Best.1. We compare accuracies of the five ensemble methods in a pairwise manner in Table 6.1. These are the number of significant wins and losses of method in the row over the method in the column over the test sets. The sum of wins and losses subtracted from 38 gives the number of ties. If the entry is bold, this means that the number of wins/losses over 38 is statistically significant using the Sign test. We do further statistical analysis with nonparametric tests using the average ranks of the six ensemble methods on 38 data sets (Table 6.2). Friedman's test rejects the hypothesis that the five methods have equal ranks. Doing Nemenyi's post-hoc test for pairwise comparison, we get the results in Figure 6.1. We see that there is no difference between Acc, Opt, All and Pca, which are significantly more accurate than Best.1.

Table 6.1. Pairwise comparison of accuracies (wins/losses over 38) of all methods using 5×2 cv F-test

	Best.1	Pca	All	Opt	Acc
Best.1	0	1	1	1	1
Pca	**8**	0	0	0	3
All	**8**	0	0	0	2
Opt	**13**	4	2	0	2
Acc	**16**	4	2	1	0

Table 6.2. Average ranks of compared methods

Best.1	Pca	All	Opt	Acc
4.32	3.07	2.70	2.45	2.47

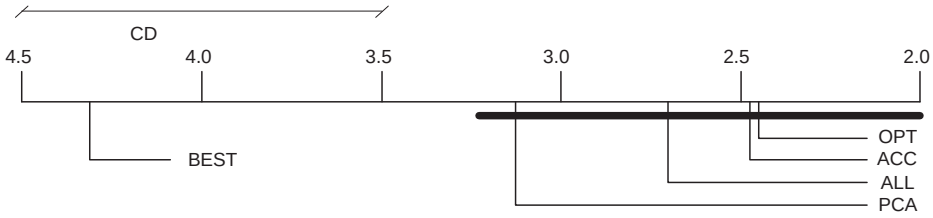


Figure 6.1. Graphical representation of post-hoc Nemenyi test results of compared methods with ranks given in Table 6.2

We propose another method to compare multiple algorithms over multiple data sets which is based on MultiTest algorithm [88]. Friedman's test takes into account only the average accuracy, but in our proposed method, we first rank the algorithms using MultiTest which checks for statistically significant difference in accuracy and also uses the complexity of base classifiers; we then use Friedman's test and its post-hoc test Nemenyi's test (if needed) on the results of MultiTest to get the final result. We see the ranks for each data set in Tables 6.3 and 6.4, average ranks in Table 6.5, and results of the post-hoc Nemenyi test in Figure 6.2. Our prior information about the complexity of the methods affect the outcome of this proposed test, we see that Pca and All, both being complex algorithms, are worse than the other three. In this experiment, we used a prior ordering of methods based on test complexity (Best.1<Acc<Opt<All<Pca) but it is possible to use other measures of complexity (training complexity) based on the application. If we want to create a final ordering of algorithms, we can apply the MultiTest algorithm on the results of Nemenyi's test. The graph can be seen in Figure 6.3; we see that since no complex algorithm is significantly more accurate than a simpler algorithm, the graph has no edges, and the ordering is Best.1>Acc>Opt>All>Pca. If we apply another prior ordering for these ranks (train complexity: Best.1<All<Acc<Pca<Opt), the graph in Figure 6.4 is formed. Using this graph, we find the following ordering: Best.1>Acc>Opt >All>Pca.

Table 6.3. Ranks of compared algorithms on each data set

	Best	Acc	Opt	All	Pca
segment	3	1	2	4	5
spambase	2	1	3	4	5
pageblock	5	1	2	3	4
iris	1	2	3	4	5
pendigits	1	2	3	4	5
optdigits	3	1	2	4	5
wine	1	2	3	4	5
monks	1	2	3	4	5
zoo	5	1	2	3	4
tae	1	2	3	4	5
hepatitis	2	1	3	4	5
flags	1	4	2	3	5
glass	5	1	2	3	4
heart	1	2	3	4	5
haberman	1	5	2	3	4
flare	3	1	2	4	5
ecoli	5	1	2	3	4
bupa	3	1	2	4	5
ionosphere	1	2	3	4	5

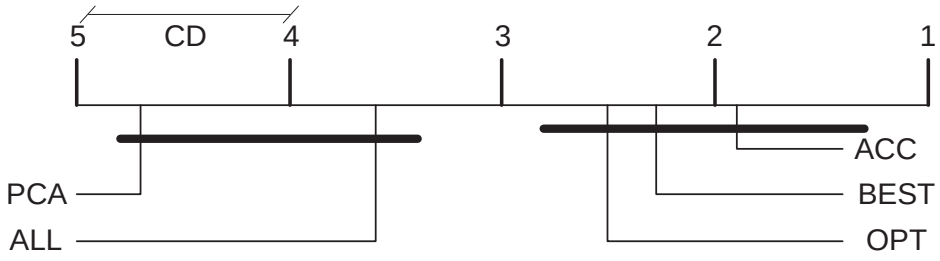


Figure 6.2. Graphical representation of post-hoc Nemenyi test results of compared methods with ranks given in Table 6.5

Table 6.4. Ranks of compared algorithms on each data set continued

	Best	Acc	Opt	All	Pca
dermatology	1	2	3	4	5
horse	5	1	2	3	4
vote	3	1	2	4	5
cylinder	1	2	3	4	5
balance	1	2	3	4	5
australian	1	2	3	4	5
credit	1	2	3	4	5
breast	1	5	2	3	4
pima	1	2	3	4	5
tictactoe	1	2	3	4	5
cmc	1	2	3	4	5
yeast	5	2	1	3	4
car	5	1	2	3	4
titanic	2	1	3	4	5
thyroid	5	4	1	2	3
ringnorm	1	2	3	4	5
twonorm	1	2	3	4	5
mushroom	1	2	3	4	5
nursery	3	1	2	4	5

Table 6.5. Average ranks of compared algorithms using MultiTest

Best.1	Pca	All	Opt	Acc
2.24	4.71	3.68	2.50	1.87

Figure 6.3. MultiTest graph using the average ranks given in Table 6.5 and test complexity

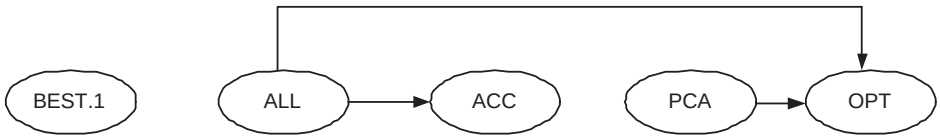


Figure 6.4. MultiTest graph using the average ranks given in Table 6.5 and train complexity

As an overall conclusion, we suggest using ACC with classifiers from different families of algorithms or using slightly correlated representations. Doing this, one has an ensemble, which is accurate (not worse than the ensemble combining the optimum subset), and cheap, i.e. the classifiers which were not selected need not be evaluated, even the features need not be extracted. However, if we have a large number of correlated classifiers, it is better to use PCA first to remove correlation and then combine the new, uncorrelated metaclassifiers.

6.3. Future Work

Knowing that diversity itself is not sufficient, one future research direction would be to construct a joint measure of diversity and accuracy, and use that measure as the ensemble evaluation criteria in ICON.

Since our aim is better accuracy with low cost, it would be beneficial to use our methods on data sets with huge amount of data. Two examples of these are text mining and bioinformatics. In these applications, real processing power is needed in the test phase, so a method which decreases the cost of testing would be beneficial. In bioinformatics applications, we believe that, our methods would construct cost-conscious ensembles using multiple representations, which would lead to better accuracy and less cost. Another application area where we believe that our methods would be interesting is in biometrics applications. These applications use multiple representations or modalities (e.g. face image, fingerprint, signature), and combining these modalities would lead to better ensemble accuracy. Also, keeping the cost of these modalities in mind, we would select those which are simple, discarding those which are redundant and complex, gaining both from accuracy and cost.

Our PCA and LDA algorithms use a linear transformation; a further research topic would be to use nonlinear transformations like ISOMAP [89], and see how we can improve in terms of accuracy and knowledge extraction.

The results we present here in using MultiTest on multiple data sets are new and can be extended. The methods which consider accuracy do not take into account the complexity of the algorithms, whereas methods using statistical significance are sometimes too conservative. We are working on a method to compare multiple algorithms over multiple data sets, which takes into account both accuracy and the complexity, using whichever complexity measure is more critical in the application.

APPENDIX A: STATISTICAL TESTS

In this appendix, we review the statistical tests used in evaluating the algorithms given α, the confidence value.

A.1. Tests for Comparing Individual Classifiers

We use the pairwise one-sided k-fold paired t-test [1] or 5×2 cv t-test to compare the expected error rates of the two ensembles to check whether the more costly ensemble is statistically significantly more accurate than the simpler one. We use 5×2 cv F test [50] to compare accuracies of different ensembles for statistically significant difference, and we used Sign test to check if the wins, ties, losses calculated by the 5×2 cv F test is significant over all the data sets. We also used non-parametric Friedman test for the same purpose.

A.1.1. k-fold paired t-test

Let $\boldsymbol{e}^1$ denote the error of the first classifier, and $\boldsymbol{e}^2$ denote the error of the second classifier on the k-folds. Then, μ_1 is the average error of classifier 1, and μ_2 is the average error of the second classifier. This test is used to test hypotheses: $H_0 : \mu_1 - \mu_2 = 0$ versus $H_0 : \mu_1 - \mu_2 \neq 0$. Let $p_i = \boldsymbol{e}_i^1 - \boldsymbol{e}_i^2$ be the difference of errors on folds. Under the null hypotheses, paired differences are t distributed with $k-1$ degrees of freedom. So, we calculate the estimates of the mean and the variance:

$$m = \frac{\sum_{i=1}^{k} p_i}{k} \ , \ S^2 = \frac{\sum_{i=1}^{k} (p_i - m)^2}{k-1}$$

We then calculate the t-statitics as $t' = \frac{m\sqrt{k}}{S}$. If $t' \in (-t_{\alpha/2,k-1}, t_{\alpha/2,k-1})$, then the test accepts, else the test rejects. This is the two-sided test. When we check for statistical improvement, we use the one-sided version. In this case, the test accepts if $t' \in (-\infty, t_{\alpha,k-1})$.

A.1.2. 5×2 cv t-test

Let $\boldsymbol{e}_i^1$ denote the error of the first classifier, and $\boldsymbol{e}_i^2$ denote the error of the second classifier on the ith replication of 2 folds. Let $p_i = \boldsymbol{e}_i^1 - \boldsymbol{e}_i^2$ be the difference of errors on each replication. Let $\bar{p}_i = (p_{i1} + p_{i2})/2$ be the estimated average on replication i and $s_i^2 = (p_{i1} - \bar{p}_i)^2 + (p_{21} - \bar{p}_i)^2$ be the estimated variance on replication i. So, we have:

$$M = \frac{\sum_{i=1}^{5} s_i^2}{\sigma^2} \text{ and } t' = \frac{p_{11}}{\sqrt{M/5}} = \frac{p_{11}}{\sqrt{\sum_{i=1}^{5} s_i^2/5}}$$

and t' is t-distributed with 5 degrees of freedom. If $t' \in (-t_{\alpha/2,5}, t_{\alpha/2,5})$, then the test accepts, else the test rejects. This is the two-sided test. When we check for statistical improvement, we use the one-sided version. In this case, the test accepts if $t' \in (-\infty, t_{\alpha,5})$.

A.1.3. 5×2 cv F-test

Since the numerator of the t', used for 5×2 cv t-test, is chosen randomly, it is possible to use all the ten values in the numerator. Alpaydın [50] introduced the robust F test which overcomes this issue. Let $p_i, \bar{p}_i, s_i^2, M$ be defined the same way as in t-test. Let

$$N = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} p_{ij}^2}{\sigma^2}.$$

It has been shown that

$$f' = \frac{N/10}{M/5} = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} p_{ij}^2}{2 \sum_{i=1}^{5} s_i^2}$$

is F distributed with ten and five degrees of freedom. The algorithms have the same error rate if $f' < F_{\alpha,10,5}$. This is a two-sided test, so if the test accepts, it means we have a tie, otherwise one of the algorithms is better. If the test rejects, we choose the algorithm which has a lower error rate as the winner.

A.2. Tests for Comparing Algorithms over Multiple Data Sets

We use these tests to compare the performance of algorithms over multiple data sets.

A.2.1. Sign Test

Given S data sets, let the number of wins of one classifier over the other be F, and let the number of losses be G (we ignore the ties), and number of wins + number of loses be S'. Sign test assumes that the wins/losses are binomially distributed, tests the null hypotheses that $F = G$. We calculate $p' = B(F, S')$ of the binomial distribution. If $p' < \alpha$, we accept the hypothesis. Otherwise we reject it (one of them is better). For large values of S', we can use an approximation for $p' = \frac{F-0.5S'}{\sqrt{0.25S'}}$. We accept the test if $p' \in (Z_{\alpha/2}, Z_{1-\alpha/2})$.

A.2.2. Friedman Test and Nemenyi Test

Friedman test is the non-parametric equivalent of ANOVA. First, all algorithms are ranked on each data set, giving the best algorithm rank 1. If there are ties, average values are given. If the algorithms have no difference, then their ranks should not be different, which is what Friedman test tests. Let r_{ij} be the rank of jth algorithm on ith data set. The Friedman test statistic is calculated as:

$$\chi_F^2 = \frac{12S}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]$$

where S is the number of data sets, k is the number of compared algorithms and $R_j = \frac{1}{N} \sum_i r_{ij}$ is the average rank of algorithms. This statistic is χ_F^2 distributed with $k-1$ degrees of freedom. If the test accepts, we say that the algorithms are not different, if the test rejects, we use the post-hoc Nemenyi test for pairwise comparison. Two classifiers are different if their average ranks differ by at least $CD = q_\alpha \sqrt{\frac{k(k+1)}{6S}}$, where values for q_α are based on the Studentized range statistic divided by $\sqrt{2}$.

A.2.3. MultiTest Algorithm

MultiTest algorithm [88] orders methods according to their accuracies and their complexities. Various time/space complexity measures can be used, and the algorithm is independent of these measures. The only assumption is that we have a prior ordering of algorithms in terms of the chosen complexity measure. Thus, given any two algorithms with the same expected error, the simple one is favored. The more complex algorithm is chosen only if it is significantly more accurate. Let us assume that we have a prior ordering of algorithms $1, 2 \ldots S$. A graph is formed as follows: $\forall i, j$, $i < j$, if M_j is significantly more accurate than M_i in terms of the statistical test used, a directed edge is placed from i to j. The in-degree of node i shows the number of algorithms that algorithm M_i is significantly more accurate and more complex. The out-degree of node i shows the number of algorithms which are more complex and significantly more accurate than M_i. Once this graph is constructed, the topological order is the resulting order of the selected algorithms.

REFERENCES

1. Alpaydın, E., *Introduction to machine learning*, The MIT Press, 2004.

2. Kuncheva, L. I., *Combining pattern classifiers: methods and algorithms*, Wiley-Interscience, 2004.

3. Breiman, L., "Bagging predictors", *Machine Learning*, Vol. 24, No. 2, pp. 123–140, 1996.

4. Alimoğlu, F. and E. Alpaydın, "Combining multiple representations and classifiers for pen-based handwritten digit recognition", *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR '97*, pp. 637–641, 1997.

5. Woods, K., J. W. Philip Kegelmeyer, and K. Bowyer, "Combination of Multiple Classifiers Using Local Accuracy Estimates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 4, pp. 405–410, 1997.

6. Kılıç, E., *Selecting from an Ensemble of Experts for Machine Learning*, Master's thesis, Boğaziçi University, 2006.

7. Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts", *Neural Computation*, Vol. 3, pp. 79–87, 1991.

8. Kittler, J., M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 226–239, 1998.

9. Wolpert, D. H., "Stacked generalization", *Neural Networks*, Vol. 5, pp. 241–259, 1992.

10. Freund, Y. and R. E. Schapire, "Experiments with a new boosting algorithm", *Proceedings of the International Conference on Machine Learning, ICML '96*, pp. 148–156, 1996.

11. Freund, Y. and R. E. Schapire, "A short introduction to boosting", *Journal of Japanese Society for Artificial Intelligence*, Vol. 14, No. 5, pp. 771–780, 1999.

12. Breiman, L., "Arcing classifiers", *The Annals of Statistics*, Vol. 26, No. 3, pp. 801–849, 1998.

13. Breiman, L., "Prediction games and arcing algorithms", *Neural Computation*, Vol. 11, No. 7, pp. 1493–1517, 1999.

14. Demir, C. and E. Alpaydın, "Cost-conscious classifier ensembles", *Pattern Recognition Letters*, Vol. 26(14), pp. 2206–2214, 2005.

15. Gökberk, B., H. Dutağacı, A. Ulaş, L. Akarun, and B. Sankur, "Representation Plurality and Fusion for 3D Face Recognition", *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, Vol. 38, No. 1, pp. 155–173, 2008.

16. Alpaydın, E., "Techniques for combining multiple learners", *Proceedings of the Engineering of Intelligent Systems, EIS '98*, Vol. 2, pp. 6–12, 1998.

17. Gama, J., "Combining classifiers by constructive induction", *Proceedings of the European Conference on Machine Learning, ECML '98*, pp. 178–189, 1998.

18. Gama, J., "Local cascade generalization", *Proceedings of the International Conference on Machine Learning, ICML '98*, pp. 206–214, 1998.

19. Alpaydın, E., "Voting over Multiple Condensed Nearest Neighbors", *Artificial Intelligence Review*, Vol. 11, No. 1-5, pp. 115–132, 1997.

20. Bay, S. D., "Nearest neighbor classification from multiple feature subsets", *Intelligent Data Analysis*, Vol. 3, No. 3, pp. 191–209, 1999.

21. Ho, T. K., "The Random Subspace Method for Constructing Decision Forests", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 8, pp. 832–844, 1998.

22. Merz, C. J., "Using correspondence analysis to combine classifiers", *Machine Learning*, Vol. 36, No. 1-2, pp. 33–58, 1999.

23. Bauer, E. and R. Kohavi, "An empirical comparison of voting classification algorithms: bagging, boosting, and variants", *Machine Learning*, Vol. 36, No. 1-2, pp. 105–139, 1999.

24. Alpaydın, E. and C. Kaynak, "Cascading classifiers", *Kybernetika*, Vol. 34, No. 4, pp. 369–374, 1998.

25. Kaynak, C. and E. Alpaydın, "MultiStage cascading of multiple classifiers: one man's noise is another man's data", *Proceedings of the International Conference on Machine Learning, ICML '00*, pp. 455–462, 2000.

26. Alpaydın, E., "REx: learning a rule and exceptions", Technical Report TR-97-040, International Computer Science Institute, Berkeley, CA, 1997.

27. Dietterich, T. G., "An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization", *Machine Learning*, Vol. 40, No. 2, pp. 139–157, 2000.

28. Dietterich, T. G., "Ensemble methods in machine learning", *Proceedings of the International Workshop on Multiple Classifier Systems, MCS '00*, Vol. 1857, pp. 1–15, 2000.

29. Seewald, A. K. and J. Fürnkranz, "An evaluation of grading classifiers", *Proceedings of the International Conference on Advances in Intelligent Data Analysis, IDA '01*, pp. 115–124, 2001.

30. Ženko, B., L. Todorovski, and S. Džeroski, "A comparison of stacking with meta

decision trees to other combining methods", *Proceedings of the International Multi-Conference Information Society, ICDM'01*, pp. 144–147, 2001.

31. Rahman, A. F. R. and M. C. Fairhurst, "Multiple classifier decision combination strategies for character recognition: a review", *International Journal on Document Analysis and Recognition*, Vol. 5, No. 4, pp. 166–194, 2003.

32. Hastie, T., R. Tibshirani, and J. Friedman, *The elements of statistical learning*, Springer-Verlag, 2001.

33. Kuncheva, L. I., "A theoretical study on six classifier fusion strategies", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 2, pp. 281–286, 2002.

34. Cabrera, J. B. D., "On the impact of fusion strategies on classification errors for large ensembles of classifiers", *Pattern Recognition*, Vol. 39, pp. 1963–1978, 2006.

35. Kuncheva, L. I., C. J. Whitaker, C. A. Ship, and R. P. Duin, "Is Independence Good For Combining Classifiers?", *Proceedings of the 15th International Conference on Pattern Recognition, ICPR '00*, pp. 168–171, 2000.

36. Jacobs, R. A., "Bias/variance analysis of mixtures-of-experts architectures", *Neural Computation*, Vol. 9, No. 2, pp. 369–383, 1997.

37. Kuncheva, L. I. and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy", *Machine Learning*, Vol. 51, No. 2, pp. 181–207, 2003.

38. Kuncheva, L. I., "Special issue on diversity in multiple classifier systems.", *Information Fusion*, Vol. 6, No. 1, pp. 1–115, 2005.

39. Caruana, R., A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models", *Proceedings of the International Conference on Machine Learning, ICML '04*, pp. 137–144, 2004.

40. Ulaş, A., M. Semerci, O. T. Yıldız, and E. Alpaydın, "Incremental Construction of Classifier and Discriminant Ensembles", 2008, accepted to Information Sciences.

41. Partridge, D. and W. B. Yates, "Engineering multiversion neural-net systems", *Neural Computation*, Vol. 8, No. 4, pp. 869–893, 1996.

42. Margineantu, D. D. and T. G. Dietterich, "Pruning adaptive boosting", *Proceedings of the International Conference on Machine Learning, ICML '97*, pp. 211–218, 1997.

43. Alkoot, F. M. and J. Kittler, "Experimental evaluation of expert fusion strategies", *Pattern Recognition Letters*, Vol. 20, No. 11-13, pp. 1361–1369, 1999.

44. Asuncion, A. and D. J. Newman, "UCI Machine Learning Repository", 2007, `http://www.ics.uci.edu/∼mlearn/MLRepository.html`.

45. Rasmussen, C. E., R. M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani, "Delve Data for Evaluating Learning in Valid Experiments", 1995-1996, `http://www.cs.toronto.edu/∼delve/`.

46. Yıldız, O. T. and E. Alpaydın, "Linear discriminant trees", *Proceedings of the International Conference on Machine Learning, ICML '00*, pp. 1175–1182, 2000.

47. Chang, C. C. and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001, `http://www.csie.ntu.edu.tw/∼cjlin/libsvm`.

48. Dietterich, T. G., "Approximate statistical tests for comparing supervised classification learning algorithms", *Neural Computation*, Vol. 10, No. 7, pp. 1895–1923, 1998.

49. Yıldız, O. T., A. Ulaş, M. Semerci, and E. Alpaydın, "AYSU: Machine Learning Data Sets for Model Combination", 2007, `http://www.cmpe.boun.edu.tr/∼ulas/aysu`.

50. Alpaydın, E., "Combined 5×2 cv F test for comparing supervised classification learning algorithms", *Neural Computation*, Vol. 11, No. 8, pp. 1885–1892, 1999.

51. Demsar, J., "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research*, Vol. 7, pp. 1–30, 2006.

52. Semerci, M., *Discriminant Ensembles and Error Analysis of Classifier Fusion Rules*, Master's thesis, Boğaziçi University, 2007.

53. Turney, P. D., "Types of cost in inductive concept learning", *Proceedings of the Workshop on Cost-Sensitive Learning, ICML '00*, pp. 15–21, 2000.

54. Akaike, H., "A new look at the statistical model identification", *IEEE Transactions on Automatic Control*, Vol. 19, pp. 716–723, 1974.

55. Schwarz, G., "Estimating the dimension of a model", *Annals of Statistics*, Vol. 6, pp. 461–464, 1979.

56. Liu, C.-L., "Classifier combination based on confidence transformation", *Pattern Recognition*, Vol. 38, pp. 11–28, 2005.

57. Jain, A., K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems", *Pattern Recognition*, Vol. 38, pp. 2270–2285, 2005.

58. Tax, D. M. J., M. van Breukelen, R. P. Duin, and J. Kittler, "Combining multiple classifiers by averaging or multiplying?", *Pattern Recognition*, Vol. 33, No. 9, pp. 1475–1485, September 2000.

59. Ulaş, A., M. Semerci, O. T. Yıldız, and E. Alpaydın, "The Effect of Correlation Between Experts in an Ensemble", 2007, submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

60. Duin, R. P., "The combining classifier: to train or not to train?", *Proceedings of the International Conference on Pattern Recognition, ICPR '02*, pp. 765–770, 2002.

61. Duin, R. P. W. and D. M. J. Tax, "Experiments with classifier combining rules", *Proceedings of the International Workshop on Multiple Classifier Systems, MCS '00*, pp. 16–29, 2000.

62. Ting, K. M. and I. H. Witten, "Issues in Stacked Generalization", *Journal of Artificial Intelligence Research*, Vol. 10, pp. 271–289, 1999.

63. Tamon, C. and J. Xiang, "On the boosting pruning problem", *Proceedings of the European Conference on Machine Learning, ECML '00*, pp. 404–412, 2000.

64. Sharkey, A. J. C., N. E. Sharkey, U. Gerecke, and G. O. Chandroth, "The "Test and Select" approach to ensemble combination", *Proceedings of the International Workshop on Multiple Classifier Systems, MCS '00*, Vol. 1857, pp. 30–44, 2000.

65. Ueda, N., "Optimal linear combination of neural networks for improving classification performance", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 2, pp. 207–215, 2000.

66. Prodromidis, A. L. and S. J. Stolfo, "Cost complexity-based pruning of ensemble classifiers", *Knowledge and Information Systems*, Vol. 3, No. 4, pp. 449–469, 2001.

67. Roli, F., G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems", *Proceedings of the International Workshop on Multiple Classifier Systems, MCS '01*, pp. 78–87, 2001.

68. Zhou, Z.-H., J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all", *Artificial Intelligence*, Vol. 137, pp. 239–263, 2002.

69. Kim, Y., W. N. Street, and F. Menczer, "Meta-evolutionary ensembles", *Proceedings of the International Joint Conference on Neural Networks, IJCNN '02*, Vol. 3, pp. 2791–2796, 2002.

70. Bakker, B. and T. Heskes, "Clustering ensembles of neural network models", *Neural Networks*, Vol. 16, No. 2, pp. 261–269, 2003.

71. Islam, M., X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles", *IEEE Transactions on Neural Networks*, Vol. 14, pp. 820–834, 2003.

72. Goebel, K. F. and W. Yan, "Choosing classifiers for decision fusion", Svensson, P. and J. Schubert (editors), *Proceedings of the International Conference on Information Fusion, Fusion '04*, Vol. I, pp. 563–568, Jun 2004.

73. Ruta, D. and B. Gabrys, "Classifier selection for majority voting", *Information Fusion*, Vol. 6, No. 1, pp. 63–81, 2005.

74. Rokach, L., O. Maimon, and R. Arbel, "Selective voting: getting more for less in sensor fusion", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 20, No. 3, pp. 329–350, 2006.

75. Zhang, Y., S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming", *Journal of Machine Learning Research*, Vol. 7, pp. 1315–1338, 2006.

76. Kuncheva, L. I. and J. J. Rodriguez, "Classifier Ensembles with a Random Linear Oracle", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 4, pp. 500–508, 2007.

77. Martínez-Munoz, G. and A. Suárez, "Using boosting to prune bagging ensembles", *Pattern Recognition Letters*, Vol. 28, No. 1, pp. 156–165, 2007.

78. Sohn, S. Y. and H. W. Shin, "Experimental study for the comparison of classifier combination methods", *Pattern Recognition*, Vol. 40, pp. 33–40, 2007.

79. Yang, Y., G. I. Webb, J. Cerquides, K. B. Korb, J. Boughton, and K. M. Ting, "To Select or To Weigh: A Comparative Study of Linear Combination Schemes for SuperParent-One-Dependence Estimators", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 12, pp. 1652–1665, 2007.

80. Beumier, C. and M. Acheroy, "Automatic 3d face authentication", *Image and Vision Computing*, Vol. 18, No. 4, pp. 315–321, 2000.

81. İrfanoğlu, O., B. Gökberk, and L. Akarun, "3D shape-based face recognition using automatically registered facialsurfaces", *Proceedings of the International Conference on Pattern Recognition, ICPR '04* , pp. 183–186, 2004.

82. Gökberk, B., M. O. İrfanoğlu, and L. Akarun, "3D shape-based face representation and feature extraction for face recognition", *Image and Vision Computing*, Vol. 24, No. 8, pp. 857–869, August 2006.

83. Raudys, S., "Trainable fusion rules. I. Large sample size case", *Neural Networks*, Vol. 19, pp. 1506–1516, 2006.

84. Rencher, A. C., "Interpretation of canonical discriminant functions, canonical variates, and principal components", *The American Statistician*, Vol. 46, No. 3, pp. 217–225, 1992.

85. Fumera, G. and F. Roli, "A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems", *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. 27, No. 6, pp. 942–956, 2005.

86. Jolliffe, I. T., "Discarding variables in a principal component analysis. II:Real data", *Applied Statistics*, Vol. 22, No. 1, pp. 21–31, 1973.

87. Merz, C. J. and M. J. Pazzani, "A principal components approach to combining regression estimates", *Machine Learning*, Vol. 36, No. 1-2, pp. 9–32, 1999.

88. Yıldız, O. T. and E. Alpaydın, "Ordering and Finding the Best of $K > 2$ Supervised Learning Algorithms", *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. 28, No. 3, pp. 392–402, 2006.

89. Tenenbaum, J. B., V. de Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Science*, Vol. 290, pp. 2319–2323, 2000.