Machine Learning, 0, 1–25 (1997)

A Principal Components Approach to Combining Regression Estimates

CHRISTOPHER J. MERZ AND MICHAEL J. PAZZANI {cmerz,pazzani}@uci.edu
Department of Information and Computer Science, University of California, Irvine, CA 92697-3425

Received October 1, 1997; Revised July 17, 1998

Editor: Sal Stolfo

Abstract. The goal of combining the predictions of multiple learned models is to form an improved estimator. A combining strategy must be able to robustly handle the inherent correlation, or multicollinearity, of the learned models while identifying the unique contributions of each. A progression of existing approaches and their limitations with respect to these two issues are discussed. A new approach, PCR*, based on principal components regression is proposed to address these limitations. An evaluation of the new approach on a collection of domains reveals that 1) PCR* was the most robust combining method, 2) correlation could be handled without eliminating any of the learned models, and 3) the principal components of the learned models provided a continuum of "regularized" weights from which PCR* could choose.

Keywords: Regression, principal components, multiple models, combining estimates.

1. Introduction

Combining a set of learned models to improve classification and regression estimates has been an area of much research in machine learning and neural networks A learned model may be anything from a decision/regression tree to a neural network. The challenge of this problem is to decide which models to rely on for prediction and how much weight to give each.

Suppose a physician wishes to predict a person's percentage of body fat, `PBF`. S/he has a collection of patient records with simple measurements/attributes such as height, weight, chest circumference, leg circumference, etc., along with a measurement of `PBF` derived from a water displacement test. The task is to predict `PBF` for future patients using only the simple measurements without performing the expensive water displacement test. The physician has derived several models for predicting `PBF` using various linear regression methods, several neural network configurations, and some existing heuristic functions. The goal is to combine the learned models to obtain a more accurate prediction than can be obtained from any single model. The general problem of combining estimates robustly is the focus of this paper.

One major issue in combining a set of learned models is the amount of correlation in the set of predictors. A high degree of correlation is expected because the learned models are attempting to perform the same prediction task. Correlation reflects the amount of agreement or linear dependence between models when making a set of predictions. The more the models agree, the more correlation, or redundancy, is present. In some cases, one (or more) models may be expressed as a linear combination (with various numerical coefficients) of the other models. Such a high degree of correlation in the model set can cause some combining schemes to produce unreliable estimates. In statistical terms, this is referred to as the *multicollinearity problem*.

Another issue in combining the predictions of learned models is detecting each model's unique contribution to predicting the target outcome. Models generated using different learning algorithms are more likely to make such contributions. For example, a neural network may discover useful non-linear interactions amongst the initial attributes, whereas a standard linear regression method may employ an attribute deletion strategy which simplifies the prediction task. A good combining strategy must be able to weigh each model according to its unique contribution.

A tradeoff exists in solving the problems mentioned above. Solutions to the multicollinearity problem are likely to ignore the unique contributions of each model. On the other hand, methods that are good at finding the unique contributions of each model are more susceptible to the multicollinearity problem. A point between these two extremes where prediction error is minimized is sought.

The focus of this paper is to present and study an algorithm for solving the problems of multicollinearity and discovering unique contributions. The paper begins by defining the task of combining regression estimates (Section 2) and discussing the limitations of existing approaches with respect to the problems discussed above. More advanced approaches to solving the multicollinearity problem are described in Section 3. A novel approach, called PCR*, based on principal components regression is outlined in Section 4. Analytical and empirical analyses are given in Sections 5 and 6, respectively. Related work is discussed in Section 7. Directions for future work are given in Section 8, and concluding remarks are given in Section 9.

2. Motivation

The problem of combining a set of learned models is defined using the terminology of [25]. Suppose two sets of data are given: a training set $\mathcal{D}_{Train} = (x_m, y_m)$ and a test set $\mathcal{D}_{Test} = (x_l, y_l)$. Now suppose $\mathcal{D}_{Train}$ is used to build a set of functions, $\mathcal{F} = \hat{f}_i(x)$, each element of which approximates $f(x)$. The goal is to find the best approximation of $f(x)$ using $\mathcal{F}$.

Most approaches to this problem limit the space of approximations of $f(x)$ to linear combinations of the elements of $\mathcal{F}$, i.e.,

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \hat{f}_i(x)$$

where α_i is the coefficient or weight of $\hat{f}_i(x)$.

The focus of this paper is to develop a method for setting these coefficients that overcomes the limitations of earlier approaches. To do so, a brief summary of these approaches is now provided progressing from simpler to more complex methods pointing out their limitations along the way.

The simplest method for combining the members of $\mathcal{F}$ is by taking the unweighted average, (i.e., $\alpha_i = 1/N$). Perrone and Cooper refer to this as the Basic Ensemble Method (BEM), written as

$$\hat{f}_{BEM} = 1/N \sum_{i=1}^{N} \hat{f}_i(x)$$

This equation can also be written in terms of the *misfit function* for each $\hat{f}_i(x)$. These functions describe the deviations of the elements of $\mathcal{F}$ from the true solution and are written as

$$m_i(x) = f(x) - \hat{f}_i(x).$$

Thus,

$$\hat{f}_{BEM} = f(x) - 1/N \sum_{i=1}^{N} m_i(x).$$

Perrone and Cooper show that as long as the $m_i(x)$ are mutually independent with zero mean, the error in estimating $f(x)$ can be made arbitrarily small by increasing the size of $\mathcal{F}$. Since these assumptions break down in practice, they developed a more general approach which finds the "optimal"[1] weights while allowing the $m_i(x)$'s to be correlated and have non-zero means. This Generalized Ensemble Method (GEM) is written as

$$\begin{aligned} \hat{f}_{GEM} &= \sum_{i=1}^{N} \alpha_i \hat{f}_i(x) \\ &= f(x) - \sum_{i=1}^{N} \alpha_i m_i(x) \end{aligned}$$

where

$$\begin{aligned} \alpha_i &= \frac{\sum_{j=1}^{N} C_{ij}^{-1}}{\sum_{k=1}^{N} \sum_{j=1}^{N} C_{kj}^{-1}}, \\ C_{ij} &= E[m_i(x) m_j(x)], \end{aligned}$$

and $E[\cdot]$ is the expected value function.

C is the symmetric sample covariance matrix for the misfit function and the goal is to minimize $\sum_{i,j}^{N} \alpha_i \alpha_j C_{ij}$. Note that the misfit functions are calculated on the training data and $f(x)$ is not required. The main disadvantage to this approach is that it involves taking the inverse of C which can be unstable. That is, redundancy in the misfits leads to linear dependence in the rows and columns of C which in turn leads to unreliable estimates of C^{-1}.

To circumvent this sensitivity redundancy, Perrone and Cooper propose a method for discarding member(s) of $\mathcal{F}$ when the strength of its agreement with another member exceeds a certain threshold. Unfortunately, this approach only checks for linear dependence (or redundancy) between pairs of $\hat{f}_i(x)$. In fact, $\hat{f}_i(x)$ could be a linear combination of several other members of $\mathcal{F}$ and the instability problem would be manifest. Also, depending on how high the threshold is set, a member of $\mathcal{F}$ could be discarded while still having some degree of uniqueness and utility. An ideal method for weighting the members of $\mathcal{F}$ would neither discard any models nor suffer when there is redundancy in the model set.

The next approach reviewed is linear regression (LR). GEM and LR are closely related in that GEM is a form of linear regression with the added constraint that $\sum_{i=1}^{N} \alpha_i = 1$. The weights for LR are found as follows[2],

$$\hat{f}_{LR} = \sum_{i=1}^{N} \alpha_i \hat{f}_i(x)$$

where

$$\begin{aligned} \alpha &= (\hat{\mathbf{f}}^T \hat{\mathbf{f}})^{-1} \hat{\mathbf{f}}^T \mathbf{y} \\ \hat{f}_{ji} &= \hat{f}_i(x_j) \quad 1 \le j \le M, \quad and \\ y_j &= f(x_j). \end{aligned}$$

A more general form of linear regression is linear regression with a constant term (LRC). LRC is calculated the same way but with member $\hat{f}_0$, which always predicts 1. According to [17] having the extra constant term will not be necessary (i.e., it will equal zero) because in practice, $E[\hat{f}_i(x)] = E[f(x)]$.

Like GEM, LR and LRC are subject to the multicollinearity problem because finding the α_i's involves taking the inverse of a matrix. That is, if the f matrix is composed of $\hat{f}_i(x)$ which strongly agree with other members of $\mathcal{F}$, some linear dependence will be present.

Given the limitations of these methods, the goal of this research is to find a method which finds weights for the learned models with low prediction error without discarding any of the original models, and without being subject to the multicollinearity problem.

3. Methods for Handling Multicollinearity

In the abovementioned methods, multicollinearity leads to inflation of the variance of the estimated weights, α. Consequently, the weights obtained from fitting the model to a particular sample may be far from their optimal values. To circumvent this problem, several approaches have been developed:

1. One method for handling multicollinearity is to build models which make decorrelated errors by adjusting the bias of the learning algorithm [24] or the data

which it sees [19]. This approach ameliorates, but does not solve, the problem because redundancy is an inherent part of the task of combining estimators.

2. Gradient descent procedures (i.e., Widrow-Hoff learning, GD, EG and EG^{+}_{-} [12]) search for the coefficients by making iterative multiplicative or exponentiated updates to the α-coefficients as a function of their performance on the training data. This avoids the matrix inversion step which is susceptible to the multicollinearity problem. The potential problems with gradient descent approaches are the possibility of getting trapped in a local minima, choosing the appropriate initial weights, and deciding how large the weight updates should be.

3. Least squares regression methods which rely on matrix inversion for finding the weights (i.e., LR and LRC) can be made more reliable by constraining the types of weights they may produce. Ridge regression, RIDGE [23] has a parameter that may be used to restrict or regularize the α-coefficients. Breiman [2] has devised an approach based on constrained least squares regression [16] where the coefficients are required to be nonnegative.

The focus of this paper is on a flexible approach to weight regularization based on principal components regression (described in Section 4. Now the discussion turns to a more precise description of weight regularization and why it is effective at handling the multicollinearity problem.

Leblanc and Tibshirani [17] have proposed several ways of constraining or *regularizing* the weights to help produce estimators with lower prediction error:

1. Shrink $\hat{\alpha}$ towards $(1/K, 1/K, \ldots, 1/K)^T$ where K is the number of learned models.
2. $\sum_{i=1}^{N} \alpha_i = 1$
3. $\alpha_i \geq 0, i = 1, 2 \ldots K$

Breiman [2] provides an intuitive justification for these constraints by pointing out that the more strongly they are satisfied, the more interpolative the weighting scheme is. In the extreme case, a uniformly weighted set of learned models is likely to produce a prediction *between* the maximum and minimum predicted values of the learned models. Without these constraints, there is no guarantee that the resulting predictor will stay near that range and generalization may be poor. An effective weight regularization technique must decide the appropriate level of constraint to be placed on the weights. We demonstrate that selecting the number of principle components in principal components regression allows the appropriate amount of weight regularization to be selected for a given set of learned models.

4. The PCR* Algorithm

The PCR* algorithm may be broken down into four parts: representation, regression, search and evaluation. Section 4.1 discusses the first two parts by describing how the model set may be mapped into a new representation using principal components analysis, and how the resulting components may be used to build a regression model. Section 4.2 discusses the latter two parts of the algorithm: the asterisk in PCR* denotes the search for the number of principal components to retain which is tightly coupled with the evaluation metric for a given model.

4.1. Representation and Regression

"PCR*" is named partly for the modeling method at its core, "Principal Components Regression" (see [3] for a summary). This section discusses the central role PCR plays in representation and regression in PCR*.

In PCR*, the representation of the final regression estimate, $\hat{f}(\mathbf{x})$, is restricted to linear combinations of the learned models in $\mathcal{F}$, i.e.,

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \hat{f}_j(\mathbf{x}) \tag{1}$$

where α_j is the coefficient or *weight* of $\hat{f}_j(x)$.

PCR* uses an intermediate representation in order to derive the final regression estimate. The main idea is to map the original learned models to a new set of models using Principal Components Analysis (PCA). The new models are a decomposition of the original models' predictions into N independent components. The more useful initial components are retained to build an estimate of f, and the mapping is reversed to get the weights for the original learned models. The following discussion elaborates on this process.

The intermediate representation is derived using Principal Components Analysis (PCA). Define $\mathbf{A}^{\mathcal{F}}$ as the matrix of learned models' predictions where

$$\mathbf{A}^{\mathcal{F}}_{ij} = \hat{f}_j(\mathbf{x_i}). \tag{2}$$

PCA takes as its input the square, symmetric covariance matrix of $\mathbf{A}^{\mathcal{F}}$, denoted $\mathbf{C} = cov(\mathbf{A}^{\mathcal{F}})$, where

$$\mathbf{C}_{ij} = E[\mathbf{A}^{\mathcal{F}}_{*,i}\mathbf{A}^{\mathcal{F}}_{*,j}] - E[\mathbf{A}^{\mathcal{F}}_{*,i}]E[\mathbf{A}^{\mathcal{F}}_{*,j}]. \tag{3}$$

The output of PCA is a new representation called the "principal components," i.e., $\{\mathbf{PC}_1, \ldots, \mathbf{PC}_N\}$. Each principal component is a column vector in the matrix, $\mathbf{PC}$, where

$$\mathbf{PC}_{i,j} = \gamma_{j,1}\hat{f}_1(\mathbf{x}_i) + \ldots + \gamma_{j,N}\hat{f}_N(\mathbf{x}_i). \tag{4}$$

Associated with each principal component is an eigenvalue, λ_j, denoting the percentage of variance that component j captures from the original matrix, $\mathbf{A}^{\mathcal{F}}$.

One advantage of this representation is that the components are independent which means the correlation between $\mathbf{PC}_i$ and $\mathbf{PC}_j$ is zero for all $i \neq j$. Another advantage is that the components are ordered by their eigenvalues, i.e.,

$$\lambda_1 > \lambda_2 > \ldots > \lambda_N.$$

Given this new representation, the goal is to choose the number of principal components to include in the final regression by retaining the first K which meet a preselected stopping criterion. Choosing K is the search aspect of PCR* and is covered in section 4.2.

Once K has been selected, an estimate of f is derived via linear least squares regression using $\mathbf{PC}_1$ through $\mathbf{PC}_K$, i.e.,

$$\hat{f}^{\beta} = \beta_1 \mathbf{PC}_1 + \ldots + \beta_K \mathbf{PC}_K \tag{5}$$

where

$$\beta = (\mathbf{PC}_K^T \mathbf{PC}_K)^{-1} \mathbf{PC}_K^T \mathbf{y} \tag{6}$$

This is known as Principal Components Regression (PCR).

Finally, the weights, α, can be derived for the original learned models by expanding (5) according to

$$\mathbf{PC}_K = \gamma_{K,0} \hat{f}_0 + \ldots + \gamma_{K,N} \hat{f}_N,$$

where $\gamma_{K,j}$ is the j-th coefficient of the K-th principal component. The α-coefficients can be calculated as follows,

$$\alpha_i = \sum_{k=1}^{K} \beta_k \gamma_{k,i} \tag{7}$$

Equations 2 through 7 make up the core of the PCR* algorithm and are summarized in Table 1. The third step, i.e., choosing K, constitutes the search aspect of PCR*. The next section elaborates on this process.

Table 1. The PCR* algorithm.

Input: $\mathbf{A}^{\mathcal{F}}$ is the matrix of predictions of the models in $\mathcal{F}$.

1. $\mathbf{C} = cov(\mathbf{A}^{\mathcal{F}})$
2. $\mathbf{PC} = PCA(\mathbf{C})$.
3. K = Choose_Cutoff($\mathbf{PC}$).
4. $\hat{f}^{\beta} = \beta_1 \mathbf{PC}_1 + \ldots + \beta_K \mathbf{PC}_K$ where $\beta = (\mathbf{PC}_K^T \mathbf{PC}_K)^{-1} \mathbf{PC}_K^T \mathbf{y}$.
5. $\alpha_i = \sum_{k=1}^{K} \beta_k \gamma_{k,i}$.
6. Return α.

4.2. Search Procedure and Evaluation

The main search component of PCR* is step 3 which involves choosing K (see Table 1). The basic idea is to include successive principal components in the regression estimate of $f(x)$ (see eqn.(5)) until all N components are used.[3] The reason for searching for K in this manner is that the principal components are ordered by the amount of variance they capture in the original learned models. The first principal component explains the most variance in the data which is where the models agree the most. The subsequent (orthogonal) components capture more and more of the variations in the models' predictions. Therefore, the number of components retained directly affects how much attention is paid to the variations in the predictions of the learned models. The value of k (where $1 \leq k \leq N$) with the lowest estimated error is chosen. This step is very important because choosing too few or too many principal components may result in underfitting or overfitting, respectively.

The evaluation criterion for selecting K is the measure of error for each possible value, k. Table 2 shows how v-fold cross-validation is used to estimate the error for each k. For a given k, as partition v is held out it is evaluated on the regression equation derived from a modified set of principal components, $\mathbf{PC}^{(-v)}$, where $\mathbf{PC}_i^{(-v)}$ is the same as $\mathbf{PC}_i$ with the examples from partition v removed. The k with the smallest cross-validation error is chosen as K. Other approaches to choosing K have been explored in [20].

Table 2. The Choose_Cutoff() algorithm.

Input:

$\mathbf{A}^{\mathcal{F}}$ is the matrix of predictions of the models in $\mathcal{F}$.

$\mathbf{\Gamma}$, the eigenvectors derived by PCR*.

$\mathbf{y}$, the target output.

Output: K, the number of components retained.

1. Form V random partitions of $\mathbf{A}^{\mathcal{F}}$
2. For partition v
   - Create new principal components:
     For $i = 1, N$
     $$\widehat{\mathbf{PC}}_i^{(-v)} = \gamma_{i,0}\hat{f}_0^{(-v)} + ... + \gamma_{i,N}\hat{f}_N^{(-v)},$$
     where $\hat{f}^{(-v)}$ is $\hat{f}$ with the examples/rows of partition v removed.
   - For $k = 1, N$
     - $\hat{f}^{\beta} = \beta_1\mathbf{PC}_1 + ... + \beta_k\mathbf{PC}_k$ where $\beta = (\mathbf{PC}_k^T\mathbf{PC}_k)^{-1}\mathbf{PC}_k^T\mathbf{y}$.
     - Error[k] $= Error$[k] $+ \sum_{i \in v}(\hat{f}^{\beta}(\mathbf{x})_i - f(\mathbf{x}_i))^2$.
3. Return $\underset{1 \leq k \leq N}{arg\ min}\ Error[k]$

5. Understanding PCR* Analytically

This section provides an analysis which illuminates how PCR* addresses some of the open problems discussed in Section 1. Artificial data sets will be used to show that PCR* provides a continuum of regularized weights for the original learned models. Section 5.1 shows how PCR* produces a highly regularized set of weights to avoid the multicollinearity problem. Section 5.2 demonstrates how PCR* handles the problem of detecting areas of specialization of each learned model by producing a less regularized set of combining weights. Section 6 will then evaluate PCR* on real problems.

5.1. The Multicollinearity Problem

The multicollinearity problem, as described in Section 3 leads to an increase in the variance of the estimated weights, α. The resulting prediction error can be quite high because the weights are very sensitive to minor changes in the data. To avoid this the weights must be regularized.

Weight regularization in PCR* is controlled via the number of principal components retained. Let PCR_k denote the instantiation of PCR* where the first k principal components are retained. Now consider deriving the α-weights using PCR_1. The first principal component is defined as the linear combination of the members of $\mathcal{F}$ with the highest average correlation with the members of $\mathcal{F}$. In this case, the weights, $\gamma_{1,*}$, will tend to be quite similar because the learned models are all fairly accurate, i.e., $E[\hat{f}_i - f] = 0$. Equation (7) shows that the γ-weights are in turn multiplied by a constant, β_1, as derived in equation (6). Thus, the resulting α_i's will be nearly uniform. The later principal components serve as refinements to those already included producing less constrained weight sets until finally the N-th principal component is included resulting in an unconstrained estimator theoretically equivalent to standard linear regression, LR.

Now an experiment will be conducted using an artificial data set to demonstrate that the weight sets derived by PCR* become less regularized as the number of principal components retained grows from 1 to N, where $N = 20$. Let f be a Gaussian function with mean zero and standard deviation one, i.e., $f \sim N(0,1)$. Model $\hat{f}_i$ ($1 \le i \le 10$) was derived as follows:

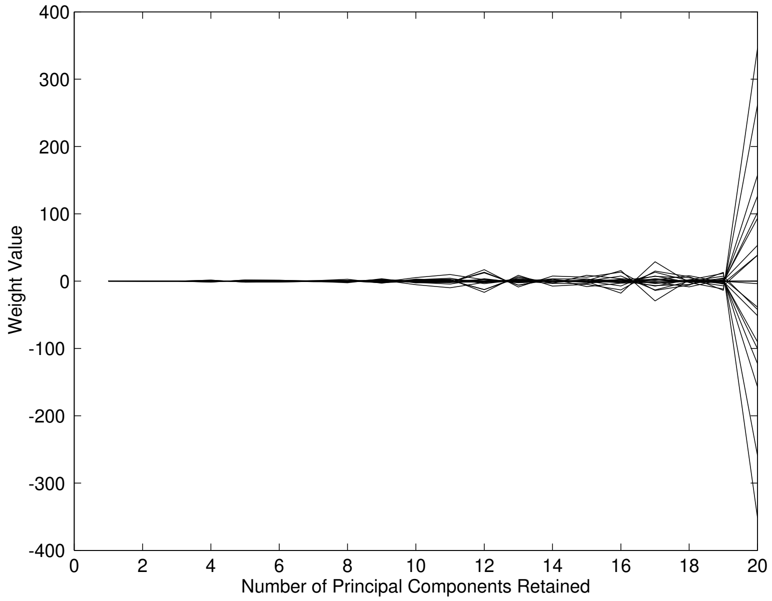
$$\begin{aligned}\hat{f}_i &= f + c_i \\ \hat{f}_{i+10} &= \hat{f}_i\end{aligned}$$

where $c_i \sim N(0, 0.1)$. This will produce ten unique models and a total of twenty models for $\mathcal{F}$. The first ten models are duplicated in the second set of ten, creating multicollinearity. Each model will produce a slight variation of f because c has a standard deviation of 0.1. One would expect a high degree of regularization to be in order for this data because of the extreme multicollinearity and the fact that the models, $\hat{f}_i$, are uniformly distributed about f.

The artificial data set, *A1*, derived using these equations consists of 200 training examples and 100 test examples. Figure 1 displays the collection of possible weight sets derived by PCR* on *A1*. The y-axis is the range of coefficient values, and the x-axis is the number of principal components used to derive the α-weights. Each line traces a single model's weight, α_i, as it is derived using the first k principal components. The weights start out as small positive values. For PCR_1, $\alpha_i \approx 1/20$. As more principal components are included the weights become less regularized, e.g., when $k = 4$ some of the weights become negative. This continues as k approaches N at which point the weights take on a very broad range of values. PCR* chose to stop at $k = 1$.

The corresponding error curve for this experiment is shown in Figure 2. In this graph, the y-axis is the mean absolute error, and the x-axis is the same as in Figure 1. As k increases and approaches N, the error rate also increases. The lowest error rate occurred at $k = 1$, the same value PCR* chose. This experiment was repeated 20 times with PCR* consistently choosing highly regularized weights.

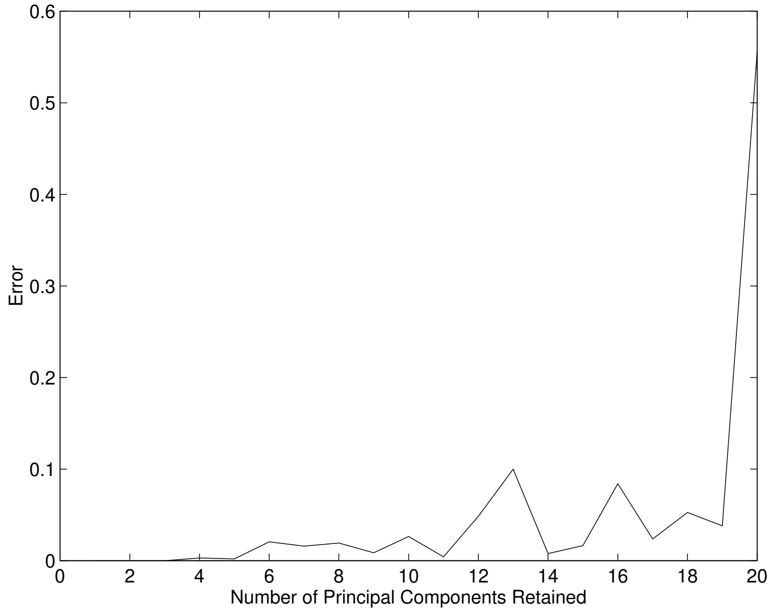
Figure 1. The α-weights for a single run with the artificial data set, *A1*. Each line corresponds to α_i as it is derived using the first k principal components.



5.2. Discovering Niches

The purpose of this section is to demonstrate that PCR* chooses less regularized weights in order to capture the unique abilities of each learned model in predicting

Figure 2. The error curve for one run with the artificial data set, *A1*. Each point corresponds to the error rate associated with the α_i-weights derived using the first k principal components.



f. Less regularized weights are needed when the errors committed by the learned models have patterns of error which cannot be canceled out by simple uniform weighting.

To demonstrate how PCR* handles this situation, another artificial data set was created where each model performs particularly well for a certain range of target values. The data set, *A2*, was generated in a similar fashion as *A1*; $f \sim N(0,1)$, and $\hat{f}_i$ $(1 \leq i \leq 10)$ was derived as follows:

$$\hat{f}_i(\mathbf{x}_j) = \begin{cases} f(\mathbf{x}_j) + c_i & : \; iff(\mathbf{x}_j) \in [c_{i'} - 0.25, c_{i'} + 0.25] \\ (0.7 + c_{i,j})f(\mathbf{x}_j) + c_i & : \; otherwise. \end{cases}$$

$$\hat{f}_{i+10} = \hat{f}_i$$

where $c_i \sim N(0, 0.1), c_{i'} \sim UNIF(-3,3), c_{i,j} \sim UNIF(0, 0.2)$. This function produces a set of 20 models where model $\hat{f}_i$ performs particularly well (i.e., with a minor offset) in the

interval $[c_{i'} - 0.25, c_{i'} + 0.25]$. Otherwise, the model randomly guesses uniformly between 70 to 90 percent of the true value for a particular point, $\mathbf{x}_j$, plus a minor offset. A data set, *A2*, of 200 training examples and 100 test examples was generated using this function.

Figure 3 displays the weights as a function of the number of principal components retained. As with data set *A1*, the weights become less regularized as k increases, but the range of values is narrower, even for $k = 20$ (see Figure 1). The corresponding error curve for the test data is plotted in Figure 4. The error rate starts out high and decreases as k approaches nine, and increases as k exceeds ten. In this case, PCR* chooses $k = 9$, the lowest point in the error curve. The initial decrease in error stems from PCR* including the unique contributions of each model (captured in the principal components) in the derivation of the α-weights. The increase in error as k exceeds ten is due to the multicollinearity contained in the model set. This experiment was repeated 20 times with PCR* consistently choosing the appropriate amount of regularization. Note that Figure 4 plots the error as measured on unseen test data while PCR* uses an estimate of error derived from only the training data.

Figure 3. The α-weights for a single run with the artificial data set, *A2*. Each line corresponds to α_i as it is derived using the first k principal components.

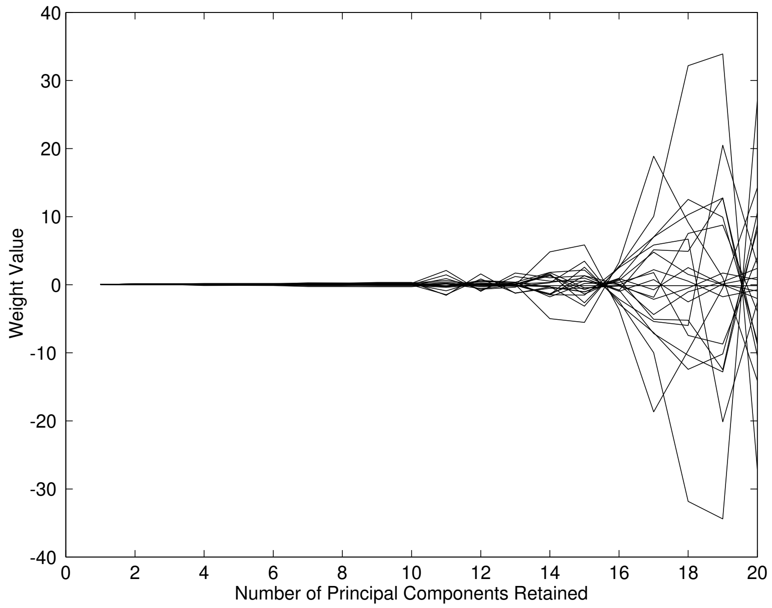
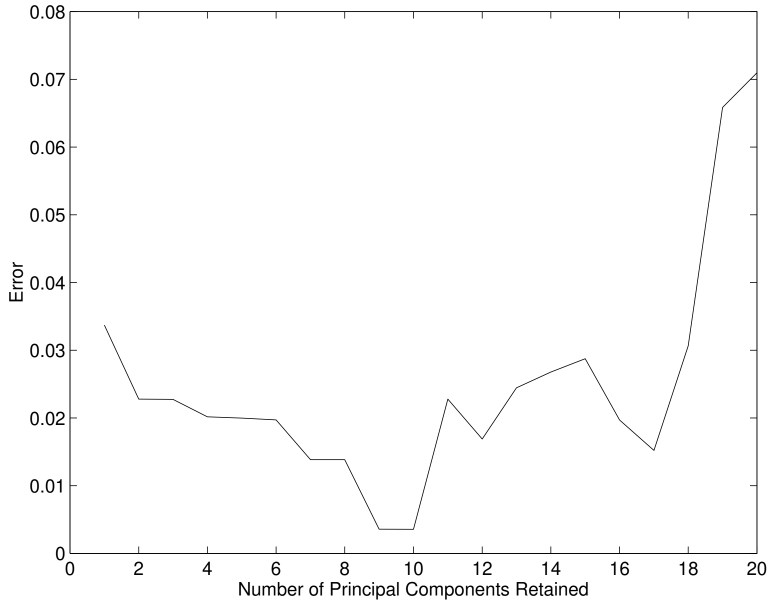


Figure 4. The error curve for a single run with the artificial data set, *A2*. Each point corresponds to the error rate associated with the α_i-weights derived using the first k principal components.



5.3. Trading off Bias and Variance

The prediction error of a learned model can be attributed to two components: that which is due to the "bias" of the model, and that which is due to the "variance" of the model (for an elaborate decomposition of prediction error, see [8]). The *bias* of an algorithm measures how consistently the models it produces (for various data sets of the same size) differ from the true function, f. The *variance* measures how much the algorithm's predictions fluctuate for the possible data sets. To decrease the overall generalization error of an algorithm it is necessary to decrease the error due to bias and/or the error due to variance.

Now consider the PCR* algorithm when $k = 1$. The (nearly) uniform weights produced essentially ignore the patterns of predictions in $\mathcal{F}$. If the patterns in $\mathbf{A}^{\mathcal{F}}$ are useful in predicting f, then PCR* will be consistently off in its predictions producing a biased result. This corresponds to the points on the error curve in Figure 4 where small values of k result in higher error. On the other hand, if $k = N$ and multicollinearity is present in $\mathbf{A}^{\mathcal{F}}$, the weight estimates may be very sensitive to minor changes in the data causing the predictions to have high variance. This corresponds to the points in the error curve of Figure 2 where larger values of k produce more error. PCR* attempts to find the minimum in the error curve where the error is not being dominated by either bias or the variance.

5.4. *Computational Complexity*

The computational complexity of PCR* is analyzed independent of the model generation process. Given a set of N models built from M examples, the three largest procedures are:

1. The calculation of the covariance matrix. This is performed once and takes $\mathcal{O}(N^2M)$ time.

2. The inversion of a matrix. In general, computing the inverse of a matrix is cubed in the number of columns/rows. All matrix inversions are performed on $N \times N$ matrices taking $\mathcal{O}(N^3)$ time. The inversion procedure is performed a total of $N + 1$ times; once for determining the β coefficients for the final model, and once for each partition of $\mathcal{L}_1$ used in determining k. Note that the Choose_Cutoff() algorithm in Table 2 may be optimized by computing the β coefficients once using all N principal components. The β coefficients derived using any subset of the components will be the same because the principal components are uncorrelated. Therefore, matrix inversion takes $\mathcal{O}((V + 1)N^3)$ time, where V is typically ten.

3. The Singular Value Decomposition of a matrix. The SVD of an $N \times N$ matrix takes $\mathcal{O}(N^3)$ time (see [26]).

Therefore, the total time complexity of PCR* is $\mathcal{O}(N^2max(M, N))$.

6. Empirical Evaluation of PCR*

Two experiments were run to compare PCR* with other combining strategies. The first experiment aims to evaluate the combiners on a dozen models; half neural networks and half adaptive regression splines. The purpose of this experiment is twofold: to evaluate some of the combiners using stacking (described below), and to evaluate the combiners abilities to combine models generated using different learning algorithms. The second experiment tests the combiners' ability to handle a large number of correlated models. The combiners were evaluated for model sets of size 10 and 50. The parameter V in the Choose_Cutoff() algorithm was set to 10.

6.1. *Regression Data Sets*

Table 3 summarizes the eight data sets used. The "Source" column lists "UCI" for data sets taken from the UCI Machine Learning Repository [21], "CMU" for data sets taken from the Statistics Library at Carnegie Mellon University [22], "QSAR" for data sets taken from the QSAR Home Page [15], and UCI-MC for a proprietary

Table 3. Data set descriptions.

Data Set	Examples	Attributes	Numeric	Source
baseball	263	16	16	CMU
bodyfat	252	14	14	CMU
cpu	209	6	6	UCI
dementia	118	26	26	UCI-MC
hansch	111	13	0	QSAR
housing	506	12	12	UCI
imports	160	15	15	UCI
servo	167	4	0	UCI

data set from the UCI Medical Center. The *imports* data set had 41 examples with missing values which were not used due to limitations in one of the learning algorithms used.

6.2. Constituent Learners

The set of learned models, $\mathcal{F}$, were generated using Backpropagation networks (BP) [28] and Multivariate Adaptive Regression Splines (MARS) [7]. In both experiments, preliminary BP runs were conducted to find a network topology which gave good performance for each data set so that the combining methods would have to work well to improve upon a single model.

6.3. Other Combining Methods

The combining methods evaluated consist of all the methods discussed in Sections 2 and 3, as well as PCR_1 and PCR_N (to demonstrate PCR*'s most and least regularized weight sets, respectively). Now a more elaborate description is given of each of the methods briefly mentioned in Section 3.

The procedures based on Widrow-Hoff learning [12] are gradient descent (GD), and the exponentiated gradient procedures EG and EG^{+}_{-}. These are iterative approaches where the weights, α, are revised with multiplicative/exponentiated updates. Each revision attempts to move the weights in a direction of lower mean squared error on the training data.

In ridge regression, the equation for deriving the weights is similar to that of deriving the β-coefficients in PCR* using all N of the principal components:

$$\beta = (\mathbf{PC}^T\mathbf{PC} + \theta\mathbf{I}_M)^{-1}\mathbf{PC}^T\mathbf{y}$$

The major difference is that the $M \times M$ identity matrix, $\mathbf{I}_M$, multiplied by a constant, θ, is added to the matrix, $\mathbf{PC}^T\mathbf{PC}$. The effect is that as θ increases, the resulting regression coefficients generated by ordinary linear regression (LR) shrink towards zero proportionally. The α-coefficients are then derived as they are in PCR*. The end result is a more restricted set of coefficients. An iterative approach is used to searching for θ (as discussed in [23]).

A "stacked" constrained regression (SCR) procedure [2] has also been included in the evaluation. The two main components of this approach are stacking and constrained regression. Stacking [32] is simply a method of approximating the matrix of predictions, $\mathcal{A}^{\mathcal{F}}$. The idea is that rather than using the actual predictions of the learned models, it is better to use an estimate because the estimate will give more information as to how to correct for the errors in each learned model. The estimated predictions are generated using a 10-fold cross-validation technique. It should be noted that the stacking component can be computationally expensive because for each learned model in the final set, 10 approximations must be generated. The other major component of SCR is constrained regression. The α-weights are obtained using ordinary least square regression with the restriction that the weights be nonnegative. A simpler version of stacked constrained regression without the stacking component (referred to as CR) is also included to evaluate the utility of constrained regression alone.

6.4. Experiment 1

This experiment aims to evaluate the combining strategies on a smaller number of learned models generated by different learning algorithms. A smaller model set was used here to make the evaluation of SCR more tractable.

Twelve models were generated. Six were generated using MARS (version 3.5) [7]. In the first three models, the variables were entered in an unrestricted, restricted, and linear fashion, respectively. The other three models were generated by entering the variables in an unrestricted fashion with each model deleting one of the three most relevant variables as determined by diagnostic output from a preliminary run of MARS. Six BP models were generated using three different network topologies with random weight initialization.

Thirty runs were conducted for each data set. On each trial the data was randomly divided into 70% training data and 30% test data. Tables 4 and 5 report the means and standard deviations of absolute error rate. The rows of the tables are divided into two blocks. The former block consists of crude methods for obtaining highly constrained or unconstrained weights. The latter block consists of the more advanced methods capable of producing weight sets with varying degrees of regularization. Bold-faced entries indicate methods which were significantly different from PCR* via two-tailed paired t-tests with $p \leq 0.01$.

6.5. Discussion of Experiment 1

Observing the combining methods in the first block of rows reveals that more regularization appears necessary for the `baseball`, `cpu`, `dementia` and `hansch` data sets, and little or no regularization appears necessary for the `servo` data set. No method in the first block does particularly well for the `bodyfat` or `housing` data sets indicating that a moderate amount of regularization is required there.

Examining the more advanced methods for handling multicollinearity in the second block of rows reveals that PCR*, EG, and CR have the best overall perfor-

Table 4. Means and standard deviations of absolute error rate for combining strategies on first four data sets.

Method	baseball	bodyfat	cpu	dementia
GEM	6.5E+3(3.3E+4)	**19.1(27.6)**	37.0(10.65)	1.318(2.8)
BEM	199.95(23)	**0.645(0.14)**	34.2(7.21)	0.384(0.04)
LR	**1.3E+4(1.8E+4)**	1.0E+5(2.5E+5)	37.0(10.48)	3.7E+2(7.8E+2)
LRC	**1.6E+4(3.2E+4)**	4.1E+5(1.3E+6)	36.7(10.37)	5.0E+2(1.2E+3)
GD	**567.36(24)**	**0.644(0.14)**	**112.6(14.08)**	0.388(0.04)
EG	194.13(21)	**0.497(0.16)**	35.8(9.63)	0.394(0.04)
EG^{+}_{-}	**502.99(30)**	**0.664(0.13)**	**97.9(12.78)**	0.390(0.04)
RIDGE	**202.65(26)**	**0.545(0.14)**	36.7(10.36)	0.421(0.07)
CR	194.29(21)	**0.497(0.16)**	35.9(9.59)	0.397(0.04)
SCR	198.52(20)	**0.547(0.14)**	36.6(10.15)	0.379(0.04)
PCR*	196.29(25)	0.454(0.15)	35.6(8.86)	0.405(0.06)

Table 5. Means and standard deviations of absolute error rate for combining strategies on last four data sets.

Method	hansch	housing	imports	servo
GEM	6.229(12.8)	6.41(10.2)	**11,292(5.3E+3)**	0.364(0.05)
BEM	0.238(0.02)	2.56(0.12)	**5,847(444)**	**0.482(0.07)**
LR	1.9E+5(6.7E+5)	3.8E+3(1.0E+4)	**5,130(5.6E+3)**	0.363(0.05)
LRC	6.3E+5(1.5E+6)	2.2E+3(5.4E+3)	**20,930(3.4E+4)**	0.363(0.05)
GD	0.238(0.02)	2.56(0.12)	**11,583(714)**	**0.382(0.05)**
EG	**0.223(0.02)**	2.48(0.18)	1,903(299)	0.370(0.05)
EG^{+}_{-}	0.236(0.02)	2.48(0.15)	**11,572(712)**	0.373(0.05)
RIDGE	0.237(0.04)	2.50(0.19)	1,987(296)	0.362(0.05)
CR	0.224(0.03)	2.48(0.18)	1,904(300)	0.372(0.05)
SCR	0.231(0.03)	2.50(0.17)	1,896(314)	0.377(0.05)
PCR*	0.231(0.03)	2.48(0.20)	1,969(289)	0.362(0.05)

Table 6. Average rankings for CR, EG, and PCR* for each data set.

Data Set	CR	EG	PCR*
baseball	6.93	6.4	7.17
bodyfat	5.73	5.8	3.13
cpu	9.23	8.27	7.767
dementia	11.27	9.97	9.13
hansch	6.13	5.83	8.37
housing	8.48	7.92	7.52
imports	6.867	6.93	7.8
servo	8.267	8.03	5.73

mances. PCR* is statistically indistinguishable from the best method in all but the **hansch** data set. In this case EG and CR have a 3.5% relative reduction in error over PCR*. EG and CR are statistically indistinguishable from the leading method in all but the **bodyfat** data set where PCR* has a 9.6% relative reduction in error over EG and CR.

GD and EG^{+}_{-} do better than the methods in the first block, but have the most difficulty finding a good weight set. These methods occasionally converge to poor local minima in spite of setting the initial weights and the learning rate as Kivinen and Warmuth [12] recommend.

Another interesting result is that constrained regression (CR) tends to outperform constrained regression with stacking (SCR) with slight losses for only two data sets. This raises the issue of whether stacking is a beneficial component of the SCR algorithm for real data sets. The extra work does not appear to improve results.

Average rankings were also calculated for each of the methods. For a given run, each method was assigned a rank according to the number of methods with lower error rates. The ranks were then averaged over the 30 runs for each data set. Table 6 reports the results for the three best combining strategies, i.e., PCR*, CR and EG. PCR* consistently performed well with respect to ranking scores too. The closest competitors were CR and EG, each having a better average ranking than PCR* on three data sets.

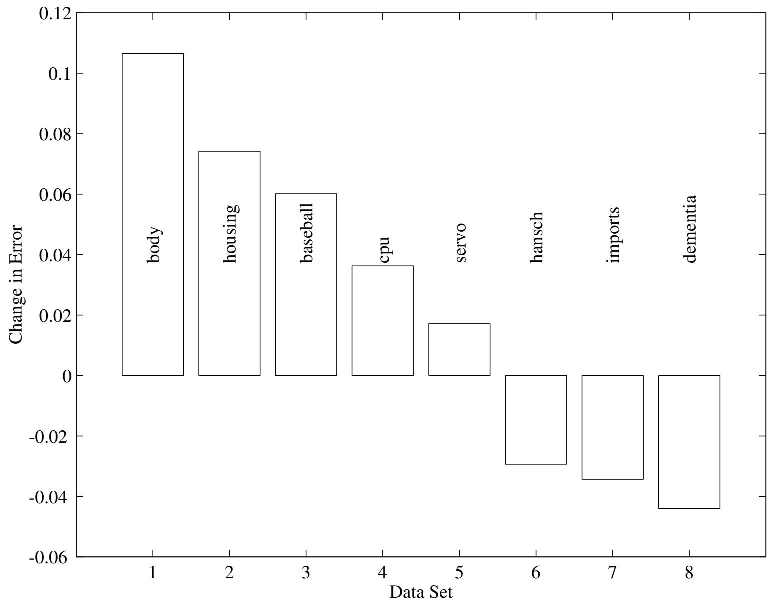
Figure 5 shows the relative error reduction made by PCR* as compared to the best individual model for each data set. PCR* improves performance by as much as 10.5%. The largest loss is a 4.3% increase in error. Overall, an improvement occurred in five data sets with an average reduction of 2.5%.

6.6. *Experiment 2*

The second experiment tests the combiners to see how well they perform with a large number of correlated models. The combiners were evaluated for model sets of size 10 and 50. There were 20 trials run for each of the data sets. On each trial the data was randomly divided into 70% training data and 30% test data.

In this experiment, the collection of networks built differed only in their initial weights, and not their topology. There was no extreme effort to produce networks with more decorrelated errors. Even with such networks, the issue of extreme

Figure 5. Relative change in error for PCR* with respect to the best individual model for each data set.



multicollinearity would still exist because $E[\hat{f}_i(x)] = E[\hat{f}_j(x)]$ for all i and j. As more models are included the linear dependence amongst them goes up showing how well the multicollinearity problem is handled. Linear dependence is verified by observing the eigenvalues of the principal components and values in the covariance matrix of the models in $\mathcal{F}$.

Table 7 reports the results for the three most representative data sets (in terms of distinguishing the combiners), i.e., *bodyfat*, *cpu*, and *housing*. The means and standard deviations for absolute error are given for each of the methods on the data sets. Two new methods were included in Table 7, PCR_1 and PCR_N, representing PCR* stopping at the first and last component, respectively. They will serve to show PCR*'s performance relative to using highly constrained and unconstrained weight sets. Each row is a particular method and each column is the size of $\mathcal{F}$ for a given data set. Bold-faced entries indicate methods which were significantly different from PCR* via a two-tailed paired t-test with $p \leq 0.01$.

Table 7. Results with many learned models

Data	bodyfat		cpu		housing	
N	10	50	10	50	10	50
BEM	1.03(0.16)	**1.04(.16)**	38.6(7.9)	38.62(7.9)	**2.79(.19)**	**2.77(.18)**
GEM	1.02(0.17)	0.86(.26)	**46.6(15)**	**227(197)**	2.72(0.20)	2.57(0.28)
LR	1.02(0.16)	3.09(6.62)	44.9(13.8)	**238(189)**	2.72(0.20)	**6.44(5.6)**
LRC	1.02(0.17)	7.51(25.22)	44.8(13.7)	**255(177)**	2.72(0.21)	**8.88(17.5)**
RDG	1.02(0.16)	0.826(.27)	44.8(13.7)	**191(133)**	2.72(0.21)	2.55(0.26)
CR	0.99(0.16)	**1.03(.17)**	40.0(10.0)	38.36(7.9)	2.70(0.21)	2.66(0.23)
GD	1.03(0.16)	**1.04(.16)**	38.9(7.95)	38.8(7.99)	**2.79(.20)**	**2.77(.18)**
EG	1.04(0.17)	**1.03(.17)**	38.4(8.19)	38.0(7.75)	2.75(0.20)	2.64(0.22)
EG^{+}_{-}	1.03(0.17)	**1.07(.16)**	38.4(8.08)	38.0(7.86)	2.77(0.20)	**2.75(.17)**
PCR_1	1.04(0.15)	**1.05(.15)**	39.0(7.76)	39.0(7.80)	2.78(0.21)	**2.76(.19)**
PCR_N	1.02(0.17)	0.848(.27)	44.8(13.7)	**250(166)**	2.72(0.21)	2.57(0.29)
PCR*	0.99(0.16)	0.786(.21)	40.3(10.0)	40.8(10.1)	2.70(0.21)	2.56(0.26)

6.7. Discussion of Experiment 2

In experiment 1, PCR* performed most similarly to EG and CR. The results in Table 7 further distinguish PCR* from EG and CR. In the *bodyfat* data set, EG and CR converge on weight sets which are near uniform resulting in poor performance relative to PCR*.

PCR* is the only approach which is among the leaders for all three data sets. For the `bodyfat` and `housing` data sets the weights produced by BEM, PCR_1, GD, and EG^{+}_{-} tended to be too constrained, while the weights for LR tended to be too unconstrained for the larger collection of models. The less constrained weights of GEM, LR, RIDGE, and PCR_N severely harmed performance in the `cpu` domain where uniform weighting performed better.

The biggest demonstration of PCR*'s robustness is its ability to gravitate towards the more constrained weights produced by the earlier principal components when appropriate (i.e., in the `cpu` data set). Similarly, it uses the less constrained principal components closer to PCR_n when it is preferable as in the `bodyfat` and `housing` domains.

7. Related Work

Several other combining strategies exist in addition to the combining strategies described in Sections 2, 3, and 6.3. The next three sections discuss: two more general approaches, some data resampling techniques, and some methods for assigning weights as a function of the example being predicted.

7.1. Other General Approaches

Hashem and Schmeiser [9] have developed a combining scheme similar to GEM as well as a less constrained version which does not require the weights to sum to one. Like GEM, this method is susceptible to the multicollinearity problem.

Opitz and Shavlik [24] attempt to assign each model a weight according to an estimate of its accuracy, i.e.,

$$\alpha_i = \frac{(1 - E_i)}{\sum_{j=1}^{N}(1 - E_j)}$$

where E_i is the estimate of model i's accuracy based on performance on a validation set. Intuitively, model i gets more weight as its estimated performance increases relative to the estimated cumulative performance of the other models. The weights derived using this approach are less susceptible to the multicollinearity problem, but less robust because the intercorrelations of the models is not considered.

A technique for pruning weights in a neural network is given in [18]. This method is also applicable to the β coefficients produced in PCR*. A threshold, T, is set for pruning principal components as a function of training error. Any principal component with a small β weight and a small eigenvalue is pruned, i.e., $\beta_i^2 \lambda_i < T$. PCR* is similar in that it retains principal components as a function of training error, however, the pruning technique above focuses more on discarding components which have a negligible impact on the final equation. The criterion in PCR* prunes the later principal components which have small eigenvalues but have an unnecessarily large β weight.

7.2. Resampling Strategies

Resampling strategies are another approach to generating and combining learned models. In these approaches, the model generation phase is more tightly coupled with the model combination stage. The goal is to generate a set of models which are likely to make uncorrelated errors (or to have higher variance) thus increasing the potential payoffs in the combining stage. Each model is generated using the same algorithm, but different training data. The data for a particular model is obtained by sampling from the original training examples according to a probability distribution. The probability distribution is defined by the particular approach, Bagging or Boosting.

Bagging [1] is a method for exploiting the variance of a learning algorithm by applying it to various version of the data set, and averaging them (uniformly) for an overall reduction in variance, or prediction error. Variations on the training data are obtained by sampling from the original training data with replacement. The probability of an example being drawn is uniform, and the number of examples

drawn is the same as the size of the original training set. The underlying theory of this approach indicates that the models should be weighted uniformly. Unlike PCR*, bagging is limited to a single learning algorithm.

Another resampling method has its roots in what is known as Boosting, initially developed by Schapire [29]. Boosting is based on the idea that a set of moderately inaccurate rules-of-thumb (i.e., learned models) can be generated and combined to form a very accurate prediction rule. The initial development of this research was purely theoretical, but subsequent refinements [5, 4] have produced practical implementations of the boosting approach. This technique assigns a weight to each example in the training data and adjusts it after learning each model. Initially, the examples are weighted uniformly. For learning subsequent models, examples are reweighted as follows: "easy" examples which are predicted with low error by previously learned hypotheses (i.e., learned models) get lower weight, and "hard" examples that are frequently predicted with high error are given higher weight. The data sets for each learned model are resampled with replacement according to the weight distribution of the examples.[4]

A common combining strategy for boosting is described in Freund and Schapire's [4] AdaBoost.M1 algorithm. The i-th model's weight is a function of its error, ϵ_i, i.e.,

$$\alpha_i = log \frac{(1 - \epsilon_i)}{\epsilon_i}$$

In this scheme, learned models with less error (on the distribution of examples they see) tend to get higher weights. In boosting (and bagging), more emphasis has been placed on model generation than model combination. It's possible that a more elaborate combining scheme like that of PCR* may be a more effective method of combining the models generated.

Two recent experimental evaluations of Boosting and Bagging are given in [5, 27]. Both approaches have proven to be quite effective, but are currently limited to a single learning algorithm. Kong and Dieterrich [13] point out that combining heterogeneous learning algorithms can reduce bias as well as variance if the bias errors of the various algorithms are different.

Krogh and Vedelsby [14] have developed a method known as query by committee [30, 6, 6]. In this approach, as a collection of neural networks is trained simultaneously, patterns which have large ambiguity (i.e., the ensemble's predictions tend to vary considerably) are more likely to be included in the next round of training.

7.3. Non-constant Weighting Functions

Some combining approaches weigh each learned model as a function of the example being predicted. The most prevalent method in the literature for dynamically deciding how to weight a collection of regressors (or classifiers) is the "mixture of experts" approach [10] which consists of several different "expert" learned models (i.e., multilayer perceptrons) plus a gating network that decides which of the experts should be used for each case. Each expert reports a target attribute probability distribution for a given example. The gating network selects one or a few experts which appear to have the most appropriate target distribution for the example. During training, the weight changes are localized to the chosen experts (and the gating network). Experts which are more accurate for the example[5] are given more responsibility for that example and experts which are inaccurate for the example are given less responsibility. The weights of other experts which specialize in quite different cases are unmodified. The experts become localized because their weights are decoupled from the weights of other experts, and they will end up specializing on a small portion of the input space.

Jordan and Jacobs [11] expanded on this approach allowing the learned models/experts to be generalized linear models. The experts are leaves in a tree-structured architecture whose internal nodes are gating functions. These gating functions make soft splits allowing data to lie simultaneously in multiple regions. Currently, the weights generated by PCR* do not change as a function of the example being predicted. A comparison between the two approaches is needed.

Tresp and Taniguchi [31] derived a collection of non-constant weighting functions which can be used to combine regressors or classifiers. The proposed methods weigh a learned model according to its reliability in the region of the given example. Reliability is defined in terms of either the model's accuracy in the region of the given example, or the amount of variability of the model's predictions in that region. All of the approaches require that the weights be positive and sum to one. The methods proposed have not been evaluated empirically, but may prove useful in extending methods like PCR* to allow the weights of the learned models to change as a function of the example being classified.

8. Limitations and Future Work

PCR* is limited to just combining regression estimates with linear weights. One direction currently being explored is the extension of PCR* to the classification task. This can be accomplished by having one PCR*-like model for each possible class. Preliminary results indicate this is an effective method of combining classifiers.

Another direction of future work is to expand PCR*'s abilities allowing for non-constant weighting. It is not likely that each model performs consistently throughout the space of possible examples. Allowing a learned model's weight to change with respect to an example would further extend PCR*'s ability to find the strengths and weaknesses of each model.

9. Summary and Conclusion

This investigation suggests that the principal components of a set of learned models can be useful when combining the models to form an improved estimator. It was demonstrated that the principal components provide a continuum of weight sets ranging from highly regularized to unconstrained. An algorithm, PCR*, was developed which attempts to automatically select the subset of these components which provides the lowest prediction error. Experiments on a collection of domains demonstrated PCR*'s ability to identify the unique contributions of each learned model while robustly handling the inherent redundancy amongst the models.

Notes

1. Optimal here refers to weights which minimize mean square error for the training data.

2. Note that the constraint, $\sum_{i=1}^{N} \alpha_i = 1$, for GEM is a form of *regularization* [17]. The purpose of regularizing the weights is to provide an estimate which is less biased by the training sample. Thus, one would not expect GEM and LR to produce identical weights.
3. Least squares regression using all N principal components (denoted PCR_N) is equivalent to standard linear regression on the original members of $\mathcal{F}$.
4. Note that this resampling technique can be replaced by a reweighting technique when the learning algorithm is capable of directly accepting a *weighted* set of examples.
5. Here, "accurate" means to have less error than the weighted average of the errors of all the experts(using the outputs of the gating network to decide how to weight each expert's error). A less accurate prediction for an example will have more error than the weighted average.

References

1. L. Breiman. Heuristics of instability in model selection. Technical report, Department of Statistics, University of California at Berkeley, 1994.
2. L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.
3. N.R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, 1981.
4. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37. Springer-Verlag, 1995.
5. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*. Morgan Kaufmann, 1996.
6. Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Information, prediction, and query by committee. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 483–490. Morgan Kaufmann, San Mateo, CA, 1993.
7. J. H. Friedman. Multivariate adaptive regression splines. *Annal of Statistics*, 19:1–141, 1991.
8. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
9. Sherif Hashem and Bruce Schmeiser. Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Transactions on Neural Networks*, 6(3):792–794, May 1995.
10. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.
11. Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.
12. J. Kivinen and M. Warmuth. Exponentiated gradient descent versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, January 1997.
13. E. B. Kong and T. G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1995.
14. Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.
15. H. Kubinyi. The QSAR and modelling society home page, 1997.
16. J. Lawson and R. Hanson. *Solving Least Squares Problems*. Prentice-Hall, New Jersey, 1974.
17. M. Leblanc and R. Tibshirani. Combining estimates in regression and classification. Technical report, Department of Statistics, University of Toronto, 1993.
18. Asriel U. Levin, Todd K. Leen, and John E. Moody. Fast pruning using principal components. In Cowan, Tesauro, and Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, 1994.
19. Ronny Meir. Bias, variance and the combination of least squares estimators. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 295–302. The MIT Press, 1995.

20. C.J. Merz. *Classification and Regression by Combining Models.* PhD thesis, University of California, Irvine, 1998.
21. C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1996. http://www.ics.uci.edu/ mlearn/MLRepository.html.
22. M. Meyer. The CMU statlib home page, 1997.
23. D.C. Montgomery and D.J. Friedman. Prediction using regression models with multicollinear predictor variables. *IIE Transactions*, 25(3):73–85, 1993.
24. David W. Opitz and Jude W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 535–541. The MIT Press, 1996.
25. M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142, London, 1993. Chapman & Hall.
26. William H. Press. *Numerical Recipes in C: The Art of Scientific Computing*, pages 59–70. Cambridge University Press, 1992.
27. J. Ross Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1996.
28. David E. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP research group., editors, *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations.* MIT Press, 1986.
29. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990.
30. H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In David Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 287–294. ACM Press, July 1992.
31. Volker Tresp and Michiaki Taniguchi. Combining estimators using non-constant weighting functions. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 419–426. The MIT Press, 1995.
32. D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.