Mixing Independent Classifiers

Jan Drugowitsch and Alwyn M. Barry
Department of Computer Science
University of Bath
Bath, BA2 7AY, UK
J.Drugowitsch@bath.ac.uk, A.M.Barry@bath.ac.uk

ABSTRACT

In this study we deal with the mixing problem, which concerns combining the prediction of independently trained local models to form a global prediction. We deal with it from the perspective of Learning Classifier Systems where a set of classifiers provide the local models. Firstly, we formalise the mixing problem and provide both analytical and heuristic approaches to solving it. The analytical approaches are shown to not scale well with the number of local models, but are nevertheless compared to heuristic models in a set of function approximation tasks. These experiments show that we can design heuristics that exceed the performance of the current state-of-the-art Learning Classifier System XCS, and are competitive when compared to analytical solutions. Additionally, we provide an upper bound on the prediction errors for the heuristic mixing approaches.

Categories and Subject Descriptors: G.1.2 Numerical Analysis: Approximation — *Least squares approximation*

General Terms: Performance, Algorithms, Theory.

Keywords: Learning Classifier Systems, Information Fusion.

1. INTRODUCTION

In modern Michigan style Learning Classifier Systems (LCS), predictions of classifiers have been mixed to give a "system prediction", which is in contrast to using the prediction of single classifiers (for example, SCS [8]). In fact, the distinction is the one of mixed model prediction vs. local model predictions. Mixed model prediction requires the prediction of the local model to be combined in some way. We will call the problem of how to combine these local models the "mixing problem".

In this paper we will formalise the mixing problem and will introduce a set of analytical and heuristic solutions. We will concentrate on model architectures where the local models are trained independently of each other, and then combined to form the global model. To our knowledge there exists no systematic study with respect to LCS of how to form a global model from local models with such a model architecture[1].

The motivation behind this study is to improve the prediction quality of any system using such a model architecture, as seen in XCS and its derivatives. XCS was introduced by Wilson in [13] for use in classification and reinforcement learning and was later also extended to function approximation tasks [14], tasks that both classification and reinforcement learning [2] can be reduced to. Just as he did, we will interpret all local models as function approximators, and will define the global model prediction as a combination of the predictions of all relevant local models. Wilson defined his mixing model in [13] as follows:

> "There are several reasonable ways to determine [the global prediction] $P(a_i)$. We have experimented primarily with a fitness-weighted average of the prediction of classifier advocating a_i. Presumably, one wants a method that yields the system's "best guess" as to the payoff [...] to be received if a_i is chosen",

and he maintains this model for all XCS derivatives without further discussion. The fitness he is referring to is a complex measure of the quality of a classifier. In this study we do *not* aim at redefining the fitness of a classifier but rather we question whether this fitness is really the best measure to use to combine the prediction of different classifiers.

One might argue that the genetic algorithm in XCS only relies on the fitness of classifiers, which subsequently depends on the local but not the global prediction, and hence mixing is only important for the global prediction once the GA has found a suitable set of classifiers. This might be

[1]Naïvely, one could think that the model we are describing is the mixture-of-experts (MoE) architecture [10]. Even though it is closely related, it differs in some significant points. Foremost, there is a strong interaction between training the experts in MoE and combining them. If trained by the EM-algorithm [4], the optimal mixing is evaluated based on the current states of the experts in the expectation step, and both mixing and experts are modified in the maximisation step. In the model we are considering, the mixing does not influence the training of the experts. Hence, we do have a bilateral relation between experts and the mixing model in MoE, but only a unilateral relation in the our model. The redistribution of classifiers is performed by a genetic algorithm rather than by the maximisation step of the EM-algorithm.

[2]In the case of reinforcement learning tasks, XCS acts as an adaptive function approximator for the action-value function, while it uses Q-Learning to update this function.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

true in tasks such as function approximation and classification. However, when using XCS for reinforcement learning tasks, the reinforcement learning algorithm strongly interacts with the global prediction model, and thus relies on an accurate mixing model. Hence, the mixing model not only determines the final prediction quality for function approximation and classification tasks, but is particularly influential in reinforcement learning tasks.

Paper Structure: To study mixing we will first introduce a formal model that allows us to define the mixing models, which is our aim, and how we can evaluate the performance of a mixing model. We continue by describing linear mixing models and their complexity which makes them unsuitable for real-world applications. Next, mixing models based on heuristics are introduced, and are then compared to each other and to mixing based on linear models in the following experimental section. Finally, our results are discussed and brought into the wider perspective of different applications of LCS.

2. FORMALISING MIXING

Let f be a target function that maps some input space $\mathcal{X}$ into the reals $\mathcal{R}$. From this target function we have a finite number of N samples $\{(x_1, f(x_1)), \ldots, (x_N, f(x_N))\}$. We want to find a model $\hat{f} : \mathcal{X} \to \mathcal{R}$ of f that minimises the squared error over these samples, that is we want to minimise

$$\sum_{n=1}^{N} \Big(f(x_n) - \hat{f}(x_n)\Big)^2. \tag{1}$$

We will describe how LCS with independent classifiers constructs $\hat{f}$ by combining a set of local models, given by the classifiers.

2.1 Independent Local Models

We have a set of K classifiers, each of which describes a local model of the target function. For each classifier $k \in \{1, \ldots, K\}$ its locality is determined by the set of inputs $\mathcal{X}_k \subseteq \mathcal{X}$ that it *matches*. The local model of a classifier k is only built for the inputs that are in $\mathcal{X}_k$. Hence, a classifier provides the local model $\hat{f}_k : \mathcal{X}_k \to \mathcal{R}$. This model is usually parametric, but we do not need to specify the parameters here as they are of no relevance to this paper. Note that this study applies to any form of local models, such as simple averagers, linear regression or neural networks.

The aim of each classifier is to minimise the squared error over its matched inputs; that is, it wants to minimise

$$\sum_{n=1}^{N} \mathbf{1}_{\mathcal{X}_k}(x_n) \Big(f(x_n) - \hat{f}_k(x_n)\Big)^2,$$

where $\mathbf{1}_{\mathcal{X}_k}$ is the indicator function for set $\mathcal{X}_k$ that returns $\mathbf{1}_{\mathcal{X}_k}(x) = 1$ if $x \in \mathcal{X}_k$ and $\mathbf{1}_{\mathcal{X}_k}(x) = 0$ otherwise. Even though we are restricting ourselves to 0/1 matching, all the concepts introduced here are still valid when $\mathbf{1}_{\mathcal{X}_k}$ returns real values of the range $[0, 1]$. The local models of the classifiers are independent in the sense that they are trained independently of the models built by other classifiers, even if those classifiers match the same inputs.

2.2 Mixing Local Models

In order to get a model over the whole input space we need to combine the local models. We will do this by defining K parametric mixing functions $\{\psi_k : \mathcal{X} \times \mathcal{R}^P \to \mathcal{R}\}_{k=1}^{K}$ with shared P real-valued parameters, denoted θ_M, that determine for each input the influence of each of the local models to the global model $\hat{f}$.

Hence, the global model is given by

$$\hat{f}(x) = \sum_{k=1}^{K} \mathbf{1}_{\mathcal{X}_k}(x) \psi_k(x, \theta_M) \hat{f}_k(x), \qquad \forall x \in \mathcal{X}. \tag{2}$$

Thus, for each input x the global model is the sum of the matching local models weighted by the mixing functions. All K mixing functions in combination describe the mixing model.

2.3 Identifying Good Mixing Models

As we now have all the definitions that we require, we can state what constitutes a good mixing model. As previously described, we want to minimise the squared error (1) of the global model $\hat{f}$. Hence, given a set of local models $\{\hat{f}_k\}_{k=1}^{K}$ and our definition of how we mix these local models (2), our aim becomes to find a mixing model $\{\psi_k\}_{k=1}^{K}$ that minimises the global squared error (1) which, when substituting (2) for $\hat{f}$, is given by

$$\min_{\theta_M} \sum_{n=1}^{N} \left(f(x_n) - \sum_{k=1}^{K} \mathbf{1}_{\mathcal{X}_k}(x_n) \psi_k(x_n, \theta_M) \hat{f}_k(x_n)\right)^2. \tag{3}$$

We will continue by describing which mixing model structures allow us to find a closed-form solution to the above problem.

2.4 Analytical Solutions

We can see from (2) that the global prediction for a certain input is a linear combination of $\mathbf{1}_{\mathcal{X}_k}$, $\hat{f}_k$, and ψ_k. Additionally, both $\mathbf{1}_{\mathcal{X}_k}$ and $\hat{f}_k$ are independent of θ_M. Hence, given that each ψ_k are linear with respect to θ_M, the global model (2) is also linear with respect to these parameters. Therefore, solving (3) is a linear least-squares problem that has an analytical solution.

However, if the ψ_k's are non-linear with respect to θ_M we have a non-linear least-squares problem for which there exists no general analytical solution. Therefore, if we want to find an analytical solution we need to assume the mixing functions to be linear. In the next section we will discuss what such a model may look like, and how such a model allows us to find a close-form solution to (3).

3. LINEAR MIXING MODELS

The most general form of linear mixing model is to have a set of P parameters $\{\alpha_{kp}\}_{p=1}^{P}$ per mixing function, and an equal number of basis functions $\{\vartheta_p : \mathcal{X} \to \mathcal{R}\}_{p=1}^{P}$ that map the inputs into the reals. The linear mixing function ψ_k is defined as

$$\psi_k(x, \theta_M) = \sum_{p=1}^{P} \vartheta_p(x) \alpha_{kp}, \tag{4}$$

where $\theta_M = \{\alpha_{kp}\}$ is the set of all parameters for the mixing model. Hence, the problem (3) becomes

$$\min_{\theta_M} \sum_{n=1}^{N} \left(f(x_n) - \sum_{k=1}^{K} \sum_{p=1}^{P} \mathbf{1}_{\mathcal{X}_k}(x_n) \hat{f}_k(x_n) \vartheta_p(x_n) \alpha_{kp}\right)^2. \tag{5}$$

As this expression is a linear least squares problem, we can find its solution by some least squares method or an approximation to it, as we will elaborate in the next section.

3.1 Mixing by Least Squares

Given that we have all our input/output pairs available at the same time, and have access to all local models, we can minimise (5) directly by matrix inversion. However, such an approach is of computational and space complexity $O((KP)^3)$ and thus does not scale well with the number of local models. In addition, it only is applicable if all input/output pairs are available at once.

Given that we have an incremental learner, XCS, we need to resort to recursive least squares which updates the model with every additional input/output pair. Unfortunately, this method maintains and updates the covariance matrix for each input/output pair at the computational and space cost of $O((KP)^2)$ and therefore does not scale well either. Additionally, it does not consider that the local models $\hat{f}_k$ are also updated incrementally with every additional input/output pair and therefore influence the past. Consequently, we are not able to find the exact solution to (5) by recursive least squares.

3.2 Gain Adaptation Approximations

Methods of gain adaptation approximate the least squares solution by only maintaining the diagonal of the covariance matrix rather than keeping the whole matrix, and hence scale linearly with the number of local models. Probably the best known variants of gain adaptation are K1, K2 and IDBD, developed by Sutton [12]. As they do not keep information on how the different components of the inputs interact, they cannot be expected to perform well when this interaction is crucial, such as when applied to multivariate linear regression. This observation was, for example, confirmed by Lanzi et al. [11], where gain adaptation did not perform much better than gradient descent when applied to training the local models of classifiers.

When searching good parameters for the mixing model we are particularly interested in how the different local models interact. Hence, the correlation between the different components of the input[3] are certainly important. Therefore, we do not expect gain adaptation methods to perform well when applied to our problem.

In summary, the least squares method does not scale well, and its approximations are expected to feature bad performance. Hence, the use of analytical solutions to the mixing problem might not be applicable to real world problems. Therefore, we will introduce a set of heuristics that we will show to come close to the quality of analytical solutions and perform significantly better than currently used heuristics.

4. HEURISTIC MIXING MODELS

The current (implicit) approach in LCS with independently trained classifiers is based on heuristic mixing. In this section we will introduce some new heuristics and will compare them to the approach used in XCS and its derivatives. They are all based on a weighted average of the predictions of the local models and differ only in how they measure the prediction quality of the local models.

Throughout the section we will assume the local model of a classifier to be linear. Let $\{\phi_l : \mathcal{X} \to \mathcal{R}\}_{l=1}^{L}$ be a set of L basis functions that map the input space into the reals and in combination form the feature column vector $\boldsymbol{\phi}(x) = (\phi_1(x), \ldots, \phi_L(x))'$. We denote the transpose of a vector $\boldsymbol{v}$ by $\boldsymbol{v}'$. Let $\boldsymbol{w}_k \in \mathcal{R}^L$ be the weight vector that defines the parameters of the local model of classifier k. The prediction of classifier k is given by

$$\hat{f}_k(x) = \boldsymbol{\phi}(x)'\boldsymbol{w}_k, \qquad \forall x \in \mathcal{X}, \quad k = 1, \ldots, K,$$

which is the inner product of the features of state x and the parameters of classifier k. The parameters are implicit in $\hat{f}_k$. Each classifier minimises the squared error over the inputs that it matches; that is, it aims to find $\boldsymbol{w}_k$ such that

$$\min_{\boldsymbol{w}_k} \sum_{k=1}^{K} \mathbf{1}_{\mathcal{X}_k}(x) \left(f(x) - \boldsymbol{\phi}(x)'\boldsymbol{w}_k\right)^2.$$

This problem is a linear weighted least squares problem and we have given an extended LCS-related discussion on how to solve it in [7].

The heuristics introduced below do not directly depend on the assumption of linearity of the classifier model. However, if other local model types are to be used, the model quality measures introduced below need to be reformulated adequately.

4.1 Mixing by Weighted Average

For each state $x \in \mathcal{X}$ each matching classifier provides a prediction for the value of $f(x)$, according to its best knowledge. Hence, it is sensible to assume that the true value of $f(x)$ is somewhere in between the lowest and the highest of those predictions. Therefore, all the heuristics that we will use ensure that the mixed global prediction is bounded from above and below by the highest and lowest local prediction respectively.

Let $\{\gamma_k : \mathcal{X} \times \mathcal{R}^P \to \mathcal{R}_0^+\}_{k=1}^{K}$ be a set of functions, one for each classifier, that map the input space into the non-negative reals, based on a set of P shared scalar parameters. We define the mixing functions to be

$$\psi_k(x, \theta_M) = \frac{\gamma_k(x, \theta_M)}{\sum_{\bar{k}=1}^{K} \mathbf{1}_{\mathcal{X}_{\bar{k}}}(x)\gamma_{\bar{k}}(x, \theta_M)}. \tag{6}$$

Combining the above with (2), we get the global prediction

$$\hat{f}(x) = \frac{\sum_{k=1}^{K} \mathbf{1}_{\mathcal{X}_k}(x)\gamma_k(x, \theta_M)\hat{f}_k(x)}{\sum_{\bar{k}=1}^{K} \mathbf{1}_{\mathcal{X}_{\bar{k}}}(x)\gamma_{\bar{k}}(x, \theta_M)}, \qquad \forall x \in \mathcal{X}, \tag{7}$$

which is the weighted average of the predictions of all matching classifiers. The magnitude of $\gamma_k(x, \theta_M)$ determines the influence of classifier k to the prediction of $f(x)$. Thus, $\gamma_k(x, \theta_M)$ needs to reflect our estimate of the quality of the prediction $\hat{f}_k(x)$.

Note that the mixing functions ψ_k are non-linear with respect to the mixing parameters θ_M. Consequently, there is no analytical solution to the least squares problem (3). There might not even be a unique minimum. Hence, we do not attempt to solve (3) but give some heuristics for possible γ_k's below.

[3] In case of the mixing model, the components of the input to the gain adaptation methods are the combination of indicator functions, values of the local models, and the basis functions ϑ.

4.2 Bounding the Global Model Error

Before discussing several measures γ_k for use in (6) for the prediction quality of a local model in the next sections, let us emphasise their importance through the following observation about the local squared errors:

THEOREM 4.1. *When using weighted averaging mixing, for all $x \in \mathcal{X}$, the squared prediction error is bounded from above by the weighted sum of squared prediction errors of the local models, where the weights are those used to mix these models. That is*

$$\left(f(x) - \hat{f}(x)\right)^2 \leq \sum_{k=1}^{K} \mathbf{1}_{\mathcal{X}_k}(x)\psi_k(x,\theta_M)\left(f(x) - \hat{f}_k(x)\right)^2.$$

PROOF SKETCH. (full proof in [6]). Observing that $\mathbf{1}_{\mathcal{X}_k}\psi_k$ describes a K-dimensional simplex for each $x \in \mathcal{X}$, the result follows from the convexity of $\cdot^2$ and Jensen's Inequality applied to $(f(x) - \hat{f}(x))$ with $\hat{f}$ expanded. □

This theorem is independent of the form of the local models, that is, it also applies to non-linear local models. Additionally, it naturally extends to all data pairs:

COROLLARY 4.2. *When using weighted averaging mixing, we can bound the sum of squared errors for a set of inputs from above by*

$$\sum_{n=1}^{N}\left(f(x_n) - \hat{f}(x_n)\right)^2 \leq \sum_{k=1}^{K}\sum_{n=1}^{N}\mathbf{1}_{\mathcal{X}_k}(x_n)\psi_k(x_n,\theta_M)\left(f(x_n) - \hat{f}_k(x_n)\right)^2.$$

Hence, the global model error is never worse than the weighted sum of errors of all local models, and is kept minimal by assigning a low weight to classifiers whose local model can be expected to have a high prediction error. This again emphasises the importance of having a good prediction quality measure for the local models.

4.3 Inverse Variance Mixing

The unbiased variance estimate of the prediction of the local linear model of classifier k is given by

$$\hat{\sigma}_k^2 = \left(-L + \sum_{n=1}^{N}\mathbf{1}_{\mathcal{X}_k}(x_n)\right)^{-1}\sum_{n=1}^{N}\mathbf{1}_{\mathcal{X}_k}(x_n)\left(f(x) - \hat{f}_k(x)\right)^2,$$

and is therefore proportional to the sum of squared prediction errors, where L refers to the size of the feature vector. As we can expect classifiers with a lower variance estimate to given better predictions on average, we can use the inverse of the variance as a measure for the quality of the prediction of a classifier's model.

We define the inverse variance mixing model to be a weighted averaging mixing model with the classifier quality measures being defined input-independently by $\gamma_k(x,\theta_M) = (\hat{\sigma}_k^2)^{-1}$. Hence, the parameter vector of this mixing model is formed by the unbiased variance estimates $\theta_M = \{\hat{\sigma}_1^2,\ldots,\hat{\sigma}_K^2\}$. In [7] we show how the variance can be estimated in a batch and incrementally, and how this form of mixing relates to the principle of maximum likelihood.

4.4 Confidence Mixing

Under the standard assumption of a normally distributed constant-variance zero-mean model error of the linear classifier model its prediction is also normally distributed [7]. As the confidence interval of a distribution is the distance from the mean at which the probability density accumulates a certain mass, this interval is in the case of the normal distribution proportional to the standard deviation[4]. In our case the standard deviation of the prediction is given by

$$\operatorname{var}\left(\hat{f}_k(x)\right) = \left(\hat{\sigma}_k^2\boldsymbol{\phi}(x)'\left(\sum_{n=1}^{N}\mathbf{1}_{\mathcal{X}_k}(x_n)\boldsymbol{\phi}(x_n)\boldsymbol{\phi}(x_n)'\right)^{-1}\boldsymbol{\phi}(x)\right),$$

where $\hat{\sigma}_k^2$ is the unbiased variance estimate of classifier k, as introduced in the last section. The inverted sum inside the brackets on the right-hand side is the covariance matrix of the feature vectors. In [7] we show how this term can be updated incrementally.

The confidence of the prediction gives input-dependent information of how certain the local model is about its prediction. As it is dependent on the input, we can expect it to give a more fine-grained information than the variance of the classifier. A low confidence interval indicates a high confidence in the given prediction. Therefore, we define the confidence mixing model to be an averaging mixing model with the quality measure of a classifier being given by $\gamma_k(x,\theta_M) = \left(\operatorname{var}\left(\hat{f}_k(x)\right)\right)^{-1/2}$. The parameters of this mixing model are on one hand the variance estimates of the classifier and on the other hand the covariance matrices of the feature vectors. As the latter can also be used in the recursive least squares algorithm to update the local model of the classifier, the parameters are shared between the mixing model and the local models.

Note that in contrast to the inverse variance mixture model the confidence measure relies on the assumption of the local model error being normally distributed and of constant variance. Therefore its performance relative to inverse variance mixing depends upon this assumption holding.

4.5 Maximum Confidence Mixing

Given that the predictions of the local models are Gaussian, mixing them by non-negative weights that sum up to 1 results in a Gaussian mixture. Our aim in maximum confidence mixing is to minimise the confidence interval of this mixture and subsequently maximise the confidence of the global model prediction.

Let k be the classifier with the highest confidence for predicting $f(x)$. In other words, this classifier has the least spread probability density function (pdf), which is given by assigning full weights to the local model with the highest prediction confidence.

As noted in the previous section, the confidence interval is proportional to the standard deviation of the classifier's normal prediction. Hence, the maximum confidence mixing model is defined as

$$\gamma_k(x,\theta_M) = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_k \operatorname{var}\left(\hat{f}_k(x)\right) \\ 0 & \text{otherwise} \end{cases}.$$

[4]We can use the knowledge of the confidence interval of a local model to give the confidence of the prediction of the global model. This, however, is not as straightforward as it initially seems. Therefore, we postpone its presentation to a later paper that is currently in preparation.

The parameters of maximum confidence mixing are the same as for confidence mixing.

As discussed before, the confidence measure relies on the normal distribution of the local model error. While confidence mixing provides some smoothing due to a weighted average of all local predictions according to their prediction confidence, maximum confidence mixing only considers the most confident prediction and therefore relies more heavily on the confidence measure. Consequently, if the local model error is not normally distributed we can expect maximum confidence mixing to perform worse than confidence mixing.

4.6 XCS Mixing

XCS and its derivatives also use a weighted averaging mixing model. Their γ_k's are given by the fitness of a classifier, which makes the mixing model closely linked to the definition of the fitness of a classifier in XCS. As such, it is more complex than any of the mixing models presented so far.

In XCS most classifier performance and fitness measures are approximated by gradient descent, which implies an incremental update. Rather than using this incremental update, we will present the convergence points of the gradient descent update equations. That allows us to calculate them directly rather than by gradient descent for the purpose of experimental comparison.

The *error* of classifier k in XCS is the mean absolute prediction error of its local model, and is given by

$$\epsilon_k = \left(\sum_{n=1}^{N} \mathbf{1}_{\mathcal{X}_k}(x_n)\right)^{-1} \sum_{n=1}^{N} \mathbf{1}_{\mathcal{X}_k}(x_n) \left| f(x_n) - \hat{f}_k(x_n) \right| .$$

The classifier's *accuracy* is some inverse function $\kappa(\epsilon_k)$ of the classifier error. This function was initially given by an exponential, but was later redefined to

$$\kappa(\epsilon) = \begin{cases} 1 & \text{if } \epsilon < \epsilon_0 \\ \alpha \left(\frac{\epsilon}{\epsilon_0}\right)^{-\nu} & \text{otherwise} \end{cases} ,$$

where the constant scalar ϵ_0 is the minimum error, the constant α is a scaling factor, and the constant ν is a mixing power factor [2]. The accuracy is constantly 1 up to the error ϵ_0 and then drops off steeply, with the shape of the drop determined by α and ν. The *relative accuracy* is a classifier's accuracy for a single input normalised by the sum of the accuracies of all classifiers matching that input. The *fitness* is the relative accuracy of a classifier averaged over all inputs that it matches, that is

$$F_k = \left(\sum_{n=1}^{N} \mathbf{1}_{\mathcal{X}_k}(x_n)\right)^{-1} \left(\sum_{n=1}^{N} \frac{\mathbf{1}_{\mathcal{X}_k}(x_n)\kappa(\epsilon_k)}{\sum_{\bar{k}=1}^{K} \mathbf{1}_{\mathcal{X}_{\bar{k}}}(x_n)\kappa(\epsilon_{\bar{k}})}\right) .$$

This fitness measure is used as the quality measure of a classifier's prediction, and hence γ_k is input-independently given by $\gamma_k(x, \theta_M) = F_k$. The parameters of this mixing model are all performance measures that need to be evaluated to get the classifier's fitness.

Note that the magnitude of a relative accuracy depends on both the error of a classifier, and on the error of the classifiers that match the same inputs. This makes the fitness of classifier k dependent on inputs that are matched by classifiers that share inputs with classifier k, but are not necessarily matched by this classifier. This might be a good measure for the fitness of a classifier (where prediction quality is not all that counts), but we do not expect it to perform well as a measure of the prediction quality of a classifier. This expectation is confirmed in the following experiments.

5. EXPERIMENTS

With the following experiments we show that i) XCS mixing performs worse than least square mixing and all of the other weighted averaging mixing models in nearly all of the experiments, ii) gain adaptation is not a viable alternative to least squares mixing, and iii) least squares mixing only performs better than weighted averaging mixing if the number of classifiers is small.

All experiments are performed on a fixed set of randomly distributed classifiers, such that each input is matched by several classifiers at once. This experimental setup was chosen to investigate the quality of mixing when there is significant overlap between the different classifiers that makes mixing their local predictions necessary. As the GA in XCS tends to distribute classifiers such that their overlap is minimised, we have not applied a GA in any of our experiments.

5.1 Experiment Design

To show performance of the different heuristic mixing models, we measure their performance in approximating a set of four commonly used test functions $f : \mathcal{R} \to \mathcal{R}$ as given in [1, 5, 6]. As a baseline, we compare these approximations to the performance of least squares and gain adaptation in combination with linear mixing models.

For each experiment 50 classifiers are generated once and not changed thereafter. The classifiers are distributed to match a subinterval of the function domain $[0, 1]$ such that on average 3 classifiers match any input. Each of the classifier's linear models is trained independently on the matched subset of 1000 samples from the target function taken in regular intervals over the range $[0, 1]$. For training, the QR-decomposition inverse RLS implementation [9] is used, with the covariance matrix initialised to $10^{-20}\boldsymbol{I}$. Each experiment is conducted using either features $\boldsymbol{\phi}(x) = (1)$, resulting in averaging classifiers, or features $\boldsymbol{\phi}(x) = (1, x)'$, resulting in classifiers that model a straight line.

The variance and confidence mixture models rely on the incremental variance estimate procedure we introduced in [7]. The parameters of XCS and linear mixing are learned post-hoc, using the fully trained classifier models and the same sample set as for training the classifiers. The XCS mixture model constants are set to $\epsilon_0 = 0.01$, $\alpha = 0.1$ and $\nu = 5$, as recommended in [2], but experiments with different settings yielded qualitatively similar results. For gain adaptation we apply the K1 algorithm due to its best performance in [12], initialised to $\mu = 0.004$, $\hat{R} = 1$ and $\hat{\boldsymbol{P}}(0) = \boldsymbol{I}$. Both K1 and least squares mixing are tested on model (4) with $P = 1$ and $\vartheta_1(x) = 1$. Least squares mixing is additionally tested on a linear mixing model with the same basis functions as the classifiers, that is $P = L$ and $\vartheta_p(x) = \phi_p(x)$. The least squares approach is implemented by a QR-decomposition inverse RLS algorithm[5] [9] with an initial covariance matrix of $10^{-40}\boldsymbol{I}$. For the rest of this paper we will refer to inverse variance mixing by *InvVar*, to confidence mixing by *Conf*, to maximum confidence mixing by *MaxConf*, to XCS mixing by *XCS*, to gain adaptation

[5]Both the direct least squares approach and the standard RLS approach proved numerically too unstable to be used to train the linear mixing model.

Function	Mean Squared Error of Mixing Model						
	Inv Var	Conf	Max Conf	XCS	LS	LS-f	K1
Blocks	*1.0765*	*1.0956*	*1.1232*	1.4393	**0.8980**		5.0773
Bumps	*0.8305*	*0.8365*	*0.9482*	1.2135	**0.5796**		1.9729
Doppler	*0.0183*	*0.0188*	0.0213	0.0253	**0.0123**		0.0358
HeaviSine	**0.1690**	**0.1677**	0.3494	0.2664	*0.2224*		4.3772
Blocks + N(0,1)	*1.1286*	*1.1819*	*1.1563*	1.4531	**0.9387**		5.2202
Bumps + N(0,0.5)	*0.8347*	*0.8590*	0.9697	1.2015	**0.5928**		2.0256
Doppler + N(0,0.1)	*0.0184*	*0.0190*	*0.0212*	0.0247	**0.0129**		0.0376
HeaviSine + N(0,1)	**0.2035**	**0.2135**	0.3920	*0.2719*	*0.2752*		4.5886
Blocks lin	*0.5935*	*0.6341*	*0.6397*	0.8745	*0.6756*	**0.4186**	5.7835
Bumps lin	*0.4408*	*0.4653*	0.5110	0.5913	0.3759	**0.2382**	1.3873
Doppler lin	*0.0089*	0.0093	0.0104	0.0112	*0.0080*	**0.0048**	0.0263
HeaviSine lin	**0.0231**	**0.0236**	**0.0264**	*0.0349*	0.1485	0.0736	3.9757
Blocks + N(0,1) lin	*0.6558*	*0.7116*	0.8088	0.8747	*0.7302*	**0.5379**	6.6167
Bumps + N(0,0.5) lin	*0.0093*	*0.0097*	0.0111	0.0110	*0.0087*	**0.0060**	0.0282
Doppler + N(0,0.1) lin	*0.0183*	0.0189	0.0205	0.0202	*0.0172*	**0.0137**	0.0363
HeaviSine + N(0,1) lin	**0.0562**	**0.0561**	*0.0658*	**0.0579**	0.2130	0.1916	4.6260

Table 1: Average MSE for the different mixing models over 20 experiments with different classifier distribution of 50 classifiers and on average 3 classifiers per input. The vertical bar separates the heuristic models from the linear models. N(0,x) refers to added Gaussian noise with standard deviation x. The features are $\phi(x) = (1)$, except for *lin* functions where the features are $\phi(x) = (1, x)'$. The group of lowest MSE mixing models without any significant difference (0.1% level) to the lowest MSE mixing model within that group are written in bold. The group of second-lowest MSE mixing models is written in italics.

mixing by *K1*, to least squares mixing with a single basis function by *LS*, and to least squares mixing with the same basis functions as the classifiers by *LS-f*.

The mixing models are evaluated by the mean squared error of the global prediction with respect to the target function over the same samples that were used to train the classifiers and mixing models. If noise was added, the error was computed using noise-free samples. We have not performed n-fold cross validation as we do actually want to minimise the MSE for the given samples rather than for the function that was sampled.

The experiments were implemented in Java and Jython, using the matrix libraries of the Colt Project [3]. The source code is available on the primary author's webpage. For a more detailed description of the experimental setup see [6].

5.2 Prediction Quality

We have evaluated all mixing models on all four target functions with features $\boldsymbol{\phi}(x) = (1)$ and $\boldsymbol{\phi}(x) = (1, x)'$, with and without adding Gaussian noise. Noise was added at different strength, depending on the range of the target function. The mean squared errors are averaged over 20 experiments with a randomly generated set of 50 classifiers and on average 3 classifiers per input. The number of classifiers is deliberately kept low to emphasise the effect of mixing on the global predictions.

Table 1 summarises the results of these experiments and highlights the group of mixing models with the lowest and the second-lowest MSE's for each target function. An example prediction for the noise-free Blocks function is shown in Figure 1. The full set of prediction plots can be found in [6]. All significant differences reported are at the 0.1% level, evaluated by the two sample t-test.

Let us first concentrate on linear mixing models. Given averaging classifiers, both LS and LS-f produce the same results because they use the same basis functions $P = 1$ and $\vartheta_1(x) = 1$ for their linear model. Except when tested with the HeaviSine function, their MSE is significantly lower than that of any other mixing model. If we use the classifier features $\phi(x) = (1, x)'$, LS-f maintains that lead significantly (again with exception of the HeaviSine function), but LS is now only as good or worse than the heuristic mixing models. Using K1 to approximate the LS solution results in significantly worse MSE's than all other mixing models in all cases.

With respect to heuristic mixing models, InvVar is always among the group of best or at least second-best models if compared to all other mixing models. If only compared to heuristic mixing models, it is always in the best group, with the lowest MSE. Conf is generally worse than InvVar, but the difference is very often not significant. MaxConf, however, performs frequently significantly worse than both InvVar and Conf. Mixing as performed by XCS is usually significantly worse than both InvVar and Conf, with one exception where it only performs worse than InvVar, and not significantly.

5.3 Linear Models vs. Heuristics

In our previous analysis we have used 50 classifiers. We will now investigate how linear mixing models compare with heuristic mixing models when the number of classifiers is modified. We vary the number of classifiers from 20 to 420 in 100 steps. For each of these steps we compare the MSE's of InvVar, XCS, LS and LS-f, averaged over 20 experiments with different classifier arrangements. The results for Conf and MaxConf are not presented, as MaxConf frequently gave significantly worse results than InvVar, and Conf was mostly worse (but not significantly) than InvVar in all experiments.

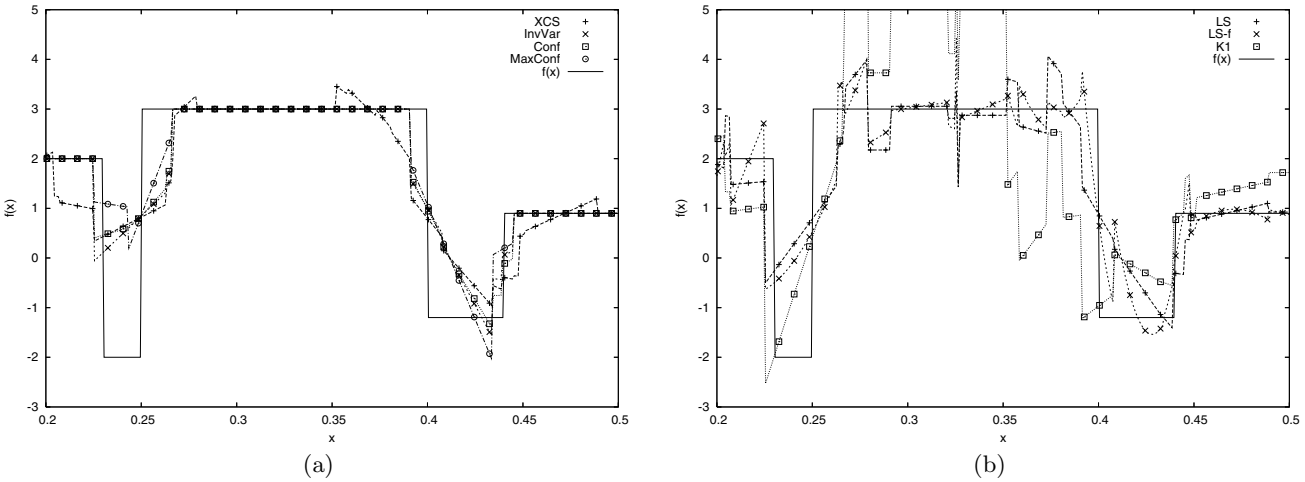

Figure 1: Example prediction for the Blocks function with 50 classifiers using the features $\phi(x) = (1, x)'$, with an average of 3 classifiers per input, plotted over the range $[0.2, 0.5]$. (a) shows the heuristic models, and (b) shows the linear models.

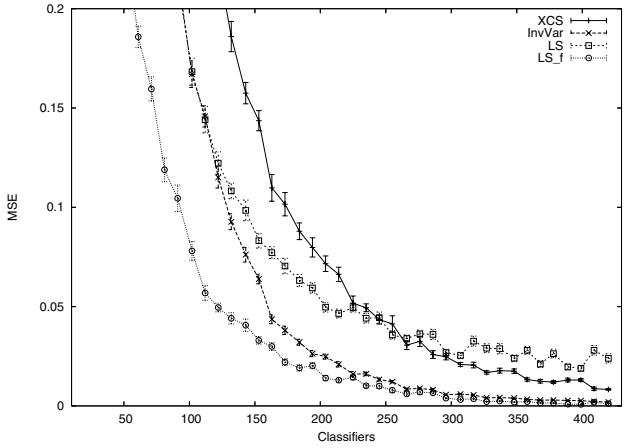

Figure 2: Change of the MSE for approximating the Bumps function with a change in the number of classifiers and the features $\phi(x) = (1, x)'$. The error bars show the standard error over 20 experiments.

Representative for all test function the results for the Bumps function are shown in Figure 2. The result for the other test functions can be found in [6]. The common pattern that we observed in all experiments is that the linear mixing models perform better when the number of classifiers are low. For a higher number of classifiers, InvVar is as good as or better than LS and LS-f. XCS is always worse than InvVar but the difference decreases with a higher number of classifiers. When using the more complex features $\phi(x) = (1, x)'$, the performance of LS in comparison to InvVar is worse even for smaller numbers of classifiers, but LS-f keeps up the lead of linear models to a higher number of classifiers.

5.4 Discussion

Let us first consider the difference in performance of the different heuristic mixing models. As we have already mentioned when introducing mixing models that rely on prediction confidence, this confidence measure is based on the assumption of having a normally distributed model error. While Conf still averages over all matching local models, MaxConf only picks the model with the highest prediction confidence. Hence, if the assumption of a normally distributed noise is violated, we can expect MaxConf to perform worse than Conf. This is confirmed by almost all experiments. In cases where MaxConf outperforms Conf their difference in MSE is not significant. However, InvVar is better than Conf and MaxConf in all cases, and it is computationally simpler. Therefore, it seems to be a better choice than the heuristics based on prediction confidence.

We have already described when introducing XCS mixing, that its classifier prediction quality metric also depends on inputs that the classifier does not match. Therefore, we did not expect it to perform as well as other heuristics that do not rely on the prediction of other classifiers or inputs outside of the set that it matches. This was confirmed in our experiment where XCS performed significantly worse than any other weighted averaging mixing model in all cases except for one.

The linear mixing model we have introduced is not restricted to weighted averaging mixing. Hence the values returned by the mixing functions ψ_k are not necessarily between 0 and 1, but can be larger than that or even negative. We have also observed such behaviour in our experiments, with mixing values up to 40. Hence, the prediction of a single local model was in some cases multiplied many-fold to produce the prediction of the global model. This makes us question the meaning of the prediction of such local models, that are supposed to represent the best fit of the matched data to the form of the model.

The aggressive mixing with linear models is also apparent when observing the shape of the predictions of the LS mixing model, for example in Figure 1. The steps in the prediction

appear at classifier matching boundaries where from a set of classifiers that match the input on one side of the step not all classifiers match the input on the other side of that step. This causes the global prediction on one side of the step to be offset by the weighted prediction of the classifier that does not match on the other side. Even though the overall prediction appears to be worse than the one produced by the heuristic mixing models, these spikes are narrow enough not to influence the MSE significantly, as our experiments have shown.

As the number of classifiers rises, so do the number of classifiers matching boundaries. Consequently, we can expect to have more spikes influencing the MSE. This explains why the performance of linear mixing models decreases with a higher number of classifiers. While this is more pronounced for LS, LS-f seems to cope better with a higher number of classifiers. This can be explained by the higher number of parameters that LS-f can adjust. Therefore, its model can react better to local changes in the target function and to classifier matching boundaries. Increasing the number of basis functions of the linear model even further will consequently also reduce the error even further. However, this increase reduces the generality of the model until we have one parameter per training sample and a prefect replication of these samples, which is certainly not useful.

The inverse variance mixing model comes close to mixing by least squares solutions of a linear model, and even exceeds its performance for higher numbers of classifiers. Additionally, it has a more appealing shape of the global prediction due to its weighted averaging architecture. An additional benefit is that it is easy to implement and that it scales linearly with the number of classifiers. The difference between inverse variance mixing and XCS mixing is significant but not substantial. Nonetheless, any improvement for small computational cost is desirable, and the concept of inverse variance mixing is more transparent and simpler to implement than XCS mixing. For all these reasons, it should be the preferred choice for a mixing model.

6. SUMMARY AND CONCLUSIONS

We have investigated how to best mix predictions by a set of local independently trained models that model the target function over a subset of its domain. This issue has, to our knowledge, never been investigated in any detail before.

We have formalised the task of mixing by specifying a parametric mixing model that forms the global prediction for a certain input by a weighted sum of the predictions of the matching local models. The aim of the mixing model is to minimise the MSE of the global predictions with respect to the target function.

Due to the nature of the problem, the only form of mixing model that admits an analytic solution to find the model parameters is the linear model, which we have described in more detail. However, computing its solution does not scale well with the number of local models. Using gain adaptation to approximate the solution is not an option because it ignores the interrelation between the predictions of the different local models.

To provide scalable solutions we have introduced a set of heuristics that are all based on a weighted average of the local predictions but differ in how they weight the different models. We have shown that using such heuristics allows us to bound the local and global prediction error by the weighted sum of prediction errors of the local models. We have introduced four heuristics: i) mixing by inverse variance of the local model, ii) mixing by prediction confidence, iii) mixing by maximum confidence, and iv) mixing by fitness, as currently performed in XCS.

We have shown in experiments that of the heuristic mixing models, mixing by inverse variance gives the lowest prediction error, and that the current XCS approach is not able to compete with any of the other heuristics. Even though linear models are able to outperform any of the heuristics when only few classifiers are used, their global prediction features many spikes, and their relative advantage to heuristic models drops when the number of classifiers increases. As they do not scale well either and cannot be applied to incremental learning, we recommend using mixing by inverse variance.

Its performance was so far only demonstrated in function approximation tasks. How well it will fare once it is applied to classification is the subject of further research. Furthermore, we expect the change of mixing model to have a significant impact on the performance of XCS in reinforcement learning tasks, a hypothesis that certainly requires further investigation.

7. REFERENCES

[1] L. B. Booker. Approximating value function in classifier systems. In L. Bull and T. Kovacs, editors, *Foundations of Learning Classifier Systems*, volume 183 of *Studies in Fuzziness and Soft Computing*. Springer Verlag, Berlin, 2005.

[2] M. V. Butz and S. W. Wilson. An Algorithmic Description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems*, volume 1996 of *LNAI*, pages 253–272. Springer-Verlag, Berlin, 2001.

[3] Colt project: Open source libraries for high performance scientific and technical computing in java, 2004.

[4] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM alrogithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[5] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[6] J. Drugowitsch and A. M. Barry. Mixing Independent Classifiers. Technical Report 2006–13, University of Bath, U.K., November 2006.

[7] J. Drugowitsch and A. M. Barry. A Formal Framework and Extensions for Function Approximation in Learning Classifier Systems. *Machine Learning*, in press.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley, MA, 1989.

[9] S. Haykin. *Adaptive Filter Theory*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ, 4th edition, 2002.

[10] R. Jacobs, M. Jordan, R. Jacobs, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[11] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Prediction update algorithms for XCSF: RLS, kalman filter, and gain adaptation. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1505–1512, New York, NY, USA, 2006. ACM Press.

[12] R. S. Sutton. Gain adaptation beats least squares? In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, pages 161–166, 1992.

[13] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995. http://prediction-dynamics.com/.

[14] S. W. Wilson. Function Approximation with a Classifier System. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *GECCO-2001: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 974–981, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.