An Experiment in Game-Based Classifier Selection

Keywords: game theory, multiple classifier systems, optical character recognition, MNIST

Abstract

We present a two-player game against nature for the study of classifier combination. The game is an extension of the prediction with expert advice framework developed by Cesa Bianchi et al.. In the game, the player selects from a set of base classifiers and their combinations, playing a closed-world competitive prediction with expert advice game, with the aim of selecting one of the classifiers that will achieve the minimum error. To demonstrate our approach we present a simple game for a binary classification task using the MNIST data set. From our exhaustive evaluation of this scenario we develop two simple strategies for selecting the best forecaster. In the future, the game presented may be used to study various classification contexts, structural pattern recognition problems, and the use of learning algorithms to infer strategies for the game.

1. Introduction

In this paper we extend previous work on game-based approaches to online prediction (Cesa-Bianchi & Lugosi, 2006; Freund & Schapire, 1996) for the task of predicting the best classifier in a set of base classifiers and their combinations. It has been repeatedly demonstrated that often a set of classifiers may be combined so that the combination outperforms the individual classifiers (Kittler et al., 1998; Kuncheva, 2004; Fumera & Roli, 2005). A number of different architectures (e.g. ensembles and cascades, containing some combination of classifiers with matching and different output spaces) and combination rules (e.g. static rules such as majority vote, linear combinations learned during training) have been studied. For ensembles, we have a reasonable understanding of regression, but not classification (Brown et al., 2005a).

The motivation for this work is to improve our understanding of the factors determining the relative performance of classifiers and their combinations. A related long-term goal is to be able to reliably predict which classifier(s) in a set of base classifiers and their combinations will produce the minimum error for a given classification problem.

Towards these ends, we present a game-based formalism for a class of classifier combination experiments. This is a two-player game against nature in which our player tries to correctly select one of the classifiers producing the minimum error (base or combined), and then loss is computed as the difference in losses between the best classifier(s) and the classifier chosen by the player. Our game provides a complete, concrete, and closed-world framework for comparing theories of classifier combination performance in an empirical setting.

To illustrate the approach taken in our game, we present some simple preliminary results demonstrating instances of this game: we consider binary classification using MNIST (LeCun et al., 1998) handwritten digit images, pixel intensity sums for features and 1-NN classification. Our example games consider the exhaustive combination of features with 1-NN classifiers, and combinations of these classifiers using the majority voting. Here we do not attempt to be prescriptive, but rather to demonstrate our game-based approach and point towards promising research directions.

In the remainder of the paper, we present the prediction with expert advice framework and our extensions of it to a closed-world competitive setting (Section 2), our game for predicting the best forecaster (Section 3) and then provide results and discussion for our preliminary instances of our game (Section 4 and 5).

2. Prediction with Expert Advice

In the model of prediction with expert advice (Cesa-Bianchi & Lugosi, 2006) a *forecaster* plays a turn-based two-player game against nature (*the environ-*

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ment). The forecaster's goal at each turn is to choose a prediction belonging to a *decision space* D that matches the outcome chosen by the environment from an *outcome space* Y. In addition to choosing outcomes, the environment determines the *advice* provided to the forecaster by a set of *experts*.

For each turn t ($t = 1, 2, \ldots$), the environment chooses the next outcome $y_t \in Y$ and the expert advice ($f_{E,t} \in D$, for each expert E). The forecaster then selects a prediction $\hat{p}_t \in D$. Outcome y_t is then revealed, and the loss for the forecaster and each expert is computed ($\ell(\hat{p}_t, y_t)$ and $\ell(f_{E,t}, y_t)$, respectively).

The forecaster's goal in this game is to minimize the *cumulative regret* with respect to each expert E ($R_{E,n}$), over all possible outcome sequences:

$$R_{E,n} = \sum_{t=1}^{n} (\ell(\hat{p}_t, y_t) - \ell(f_{E,t}, y_t))$$

Forecasters may use *side information* before making a prediction, in addition to the expert advice. In our case this will be features for the next digit to be classified.

2.1. Competitive Prediction

To define the interaction of the environment, available features, and classifiers and their combinations more concretely, we now consider a variation of prediction with expert advice in which a set of forecasters (F) play simultaneously, with some forecasters doubling as 'experts' for other forecasters. In this variation, the environment selects the outcome, but not the expert advice: expert advice now comes from other forecasters in the game.

Each forecaster aims to minimize cumulative regret as defined above, with forecasters ranked in decreasing order of regret relative to the best forecaster(s) at the end of the game. This is a game of complete information, where a history for completed turns is recorded and made available to all forecasters in the game. For each turn, the game history h contains:

1. Feature values ($s_t \in S$)
2. Parameters for forecasters (θ_{it})
3. Predictions ($\hat{P}_t = \{\hat{p}_{t1}, \ldots, \hat{p}_{t|F|}\}$)
4. Outcome y_t

S is a fixed feature space, defined at the beginning of the game. Note that forecasters may not (likely will not) use all information in the game history; for example, forecasters will likely use different subspaces of S (i.e. different features).

The behavior of forecasters is defined by a decision function (d_i), and a learning function (l_i):

$$d_i : S \times \Theta_i \times D^{|A|} \rightarrow D$$
$$l_i : \Theta_i \times H \rightarrow \Theta_i$$

Θ_i is the parameter space for forecaster $f_i \in F$, $D^{|A|}$ is the space of possible expert advice (defined by some subset of forecasters $A \subseteq F$), and H is the space of possible game histories. d_i is used by forecaster f_i to make its prediction $\hat{p}_t$ each turn. Forecasters apply their learning function l_i to update their parameters ($\theta_i \in \Theta_i$) based on their current values and the game history.

The game as posed here assumes hierarchical dependencies between forecasters, where all expert advice (other forecaster predictions) must be computed *before* a forecaster makes its prediction. Without this restriction, cyclic dependencies between two or more forecasters may arise, making it impossible to compute their predictions.

3. Picking the Best Forecaster

Let us now define the game used in the experiments in this paper. In this game we ask a player to choose the *best* forecaster within a competitive prediction game before it has ended. Assume that we can see only the first m turns of the game, with $m < n$, the (finite) number of turns in the game. The best forecasters are those that achieve the minimum loss after the prediction game continues to the final turn n. As before, this is a two-player game against nature, with the environment choosing the outcomes within the prediction game.

In this new game the outcome space contains sets of forecasters with equivalent losses ($Y_L = \{L_1, \ldots, L_n\}$), while the decision space D_F is the set of forecasters ($D_F = F, D \neq Y$). The measure of regret for our player in this new game is the difference between the cumulative regret for one of the best experts (E_b) and for the selected expert (E_s):

$$\sum_{j=1}^{q} R_{E_s, turns(j)} - R_{E_b(k), turns(j)}$$

where $turns(j)$ is the number of turns played in prediction game j.

This 'new' game is really an instance of the original prediction with expert advice game (Section 2), with no experts and side-information (the provided game history). One might consider having the environment again provide expert advice to our player; however, we

Figure 1. Examples of digits from the MNIST database (LeCun et al., 1998)

will assume that our player must pick the best forecaster in the absence of expert advice in this paper.

4. Empirical Evaluation of Forecasters

The goal of the experiment in this paper is not to achieve a competitive error rate on MNIST, but rather to demonstrate our approach and observe interactions between a classification rule, available features, and the combination technique in a 'pick the best forecaster' game. Towards that end, we use a subset of the MNIST data to evaluate the game.

The MNIST (LeCun et al., 1998)[1] data set consists of pre-processed 28×28 grayscale examples of handwritten digits divided into training (60,000 examples) and test sets (10,000 examples). Some digits are shown in Figure 1. MNIST has been used extensively in machine learning research, making it an ideal benchmark for studying the properties of classifier combinations.

Many techniques have been used to recognize MNIST with an error rate of less than 1%, including shape context features paired with k-NN classification (Belongie et al., 2002), image deformation models (Keysers et al., 2007), synthetic training data generation using elastic distortions (Simard et al., 2003), convolutional neural networks using boosted ensembles (LeCun et al., 1998) and Support Vector Machines for classification in the final layer (Lauer et al., 2007), and using an energy based model (Ranzato et al., 2007) (with error rate 0.39%). These techniques produce few enough errors that confused digits are often listed in the papers.

4.1. Experimental Design

Using the MNIST data we run the game on the pairs of digits ("0", "1"), ("1", "7") and ("4", "9") to provide a range of levels of difficulty for our chosen features. We use only one function for feature extraction: the sum of a rectangular array (represented as $\sum_I$) of pixel values. This can be used to provide histograms of pixel values within individual rows and columns in an image, or within subregions of an image (for example, after splitting an image into 4 regions). For this evaluation, we select the regions as three columns of widths (left to right) 9, 10 and 9, which we label as $FS1$, $FS2$ and $FS3$, respectfully.

[1] http://yann.lecun.com/exdb/mnist/

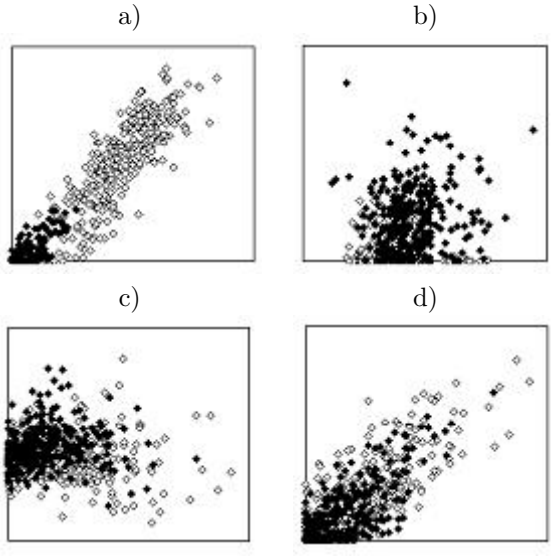

Figure 2. Examples of training feature values plotted as pairs using ground truth classifications to distinguish classes for a) ("0" , "1") (white, black) for $FS1$ (x-axis) versus $FS3$ (y-axis); b) ("1", "7") for $FS2$ versus $FS3$; c) ("4", "9") for $FS1$ versus $FS2$; and d) ("4", "9") for $FS1$ versus $FS3$.

The features are chosen for both simplicity and their potential to discriminate with varying levels of performance between digits. For example, all three features provide good discrimination for ("0", "1"), as can be seen in Figure 2a for $FS1$ versus $FS3$, with a clear discrimination boundary. In contrast, for ("1", "7"), there is no clear discrimination boundary between FS2 and FS3 (Figure 2b), which perhaps matches the similarity of the written digits, whereas for ("4", "9"), the distinction is a little clearer if we take into account neighborhoods of features (Figure 2c, d).

To help quantify the amount of similarity between features for a given task, we calculated the correlation coefficients for pairs of features to show the overall correlation across and within classes as specified by ground truth (Table 1). While the correlation coefficient is not the best measure of feature dependence, it does allow us to see the potential redundancy our features provide. Here we can see that, in general, $FS1$ and $FS3$ are highly correlated, especially for "0", "1" and "4", with "7" and "9" showing a slightly lesser correlation. This would suggest that having just one of $FS1$ or $FS3$ is sufficient for classification of "0", "1" or "4". In contrast, $FS1$ and $FS2$, and $FS2$ and $FS3$ show little or no correlation, except for "7" (0.44) and across both classes for ("0", "1") and ("1", "7"), where

Table 1. Correlation coefficients for the training data sets for the three tasks. Coefficients are shown between pairs of features for the whole data set, and then for each data set split by ground truth (GT) classification.

	$FS1, FS2$	$FS1, FS3$	$FS2, FS3$
("0", "1")	0.46	0.95	0.38
"0" (GT)	0.12	0.84	-0.03
"1" (GT)	0.04	0.80	-0.04
("1", "7")	0.51	0.77	0.44
"1" (GT)	-0.05	0.82	-0.17
"7" (GT)	0.44	0.63	0.17
("4", "9")	-0.23	0.79	-0.16
"4" (GT)	-0.21	0.83	-0.16
"9" (GT)	-0.01	0.61	0.10

some degree (at least 0.38) of correlation is shown.

For a given game, we select separate training and testing sets from the respective MNIST data in the order supplied. For both training and testing we select 1/20 of the available digits, sufficient to allow our classifiers to learn, but small enough to facilitate evaluation without being too computationally intensive. For example, for the ("0", "1") game, we select the first 296 "0"s and the first 337 "1"s for training. The complete set of sample sizes is shown in Table 2.

A training phase of the game is executed first in which the forecasters are permitted to adjust their parameters θ_{it}, and which uses the training set to construct the feature values s_t. The number of turns n in the game is set to the total number of digits in the training set (in our example 633) so that each example is used only once. The testing phase then uses the testing set to construct features for samples, but with no forecaster learning, the last set of trained forecaster parameters θ_{nt} are used. During both training and testing the environment uses a uniform random sampling method to select examples per turn, assigning y_t to be the ground truth classification for the sample from the associated MNIST label.

To explore different types of forecaster in a hierarchy, we use both base and multiple classifiers. The base classifiers are binary 1-nearest neighbor (1-NN) classifiers, using the Euclidean distance metric, taking values provided by $\sum_I$ and the current classifier's training sample set within the parameters θ_{it} as input, and returning the class with the closest instance. Ties are resolved by taking the class of just the first closest sample. The learning function for the 1-NN classifiers

Table 2. MNIST digit training and testing sample sizes.

Digit	Training	Testing
"0"	296	49
"1"	337	57
"4"	292	49
"7"	313	51
"9"	297	50

Table 3. Base classifier feature spaces.

Base classifier	$FS1$ (left 9)	$FS2$ (center 10)	$FS3$ (right 9)
f_1			•
f_2		•	
f_3		•	•
f_4	•		
f_5	•		•
f_6	•	•	
f_7	•	•	•

is trivial (l_{NN}), as it simply adds observed instances in the feature space to the classifiers sample set (θ_{it}), until all the training turns have completed. No changes to the parameters are permitted during testing. The idea of combining 1-NN classifiers through voting is not new: there is at least one published method for applying k-NN classifiers to individual features, and then combining their results using a plurality voting scheme (Akkus & Guvenir, 1996).

To provide us with a distinctive set of base classifiers, we construct 1-NN classifiers that take as input the exhaustive combination of the features $FS1$, $FS2$ and $FS3$, so that f_1 has feature space $FS3$, and so forth, as shown in Table 3. The multiple classifiers evaluate the exhaustive combination of each of the 7 base classifiers with 120 ensembles, combining from 2 to 7 of the base classifiers. Each ensemble determines its output as a majority vote on the outputs of its associated base classifiers, with ties broken randomly (uniform distribution).

To evaluate the best performing forecasters in the game we use a zero-one loss function ℓ (1 if the classifier output is wrong, 0 if correct). For ease of comparison in the results, we rank the losses for each forecaster in ascending order (lowest number of losses first). For an equal number of losses, the rank function preserves the order of the forecasters. We also report the overall performance of a forecaster as a cumulative percentage binary classification accuracy at each turn.

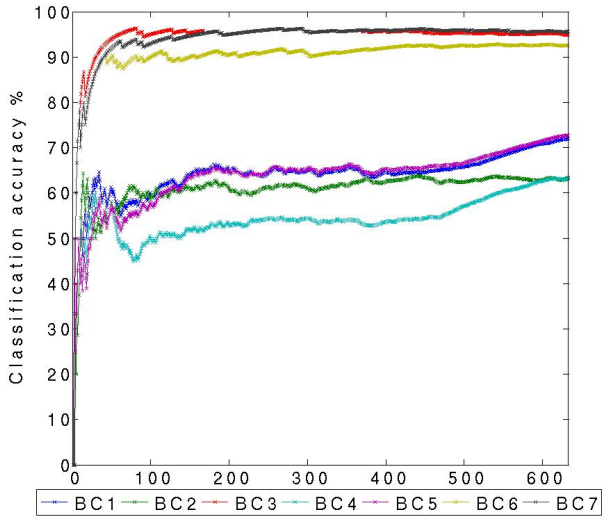

Figure 3. Base classifier cumulative training classification accuracy for ("0", "1").

4.2. Base Classifier Results

First we examine the performance of the base classifiers to determine which achieves the best ranking versus their corresponding features. Having selected only 1/20 of the available digits for training, we note that each base classifiers performance appears to stabilize after approximately 100 examples have been presented for training (for example, Figure 3). Table 4 shows the ranking for each base classifier and its corresponding classification accuracy for the three tasks.

During training, the highest ranking base classifier is f_7 for all three tasks, which uses $(FS1, FS2, FS3)$. This is then followed by either f_6 and f_3, which are always in second or third rank. Indeed, f_7, f_6 and f_3 become dominant as soon as the classifiers achieve stability in training (for example, Figure 3) for ("0", "1") and ("1", "7"), although this is less clear for ("4", "9") where the classifiers are more evenly matched. The worst ranked base classifier is f_4, which uses $(FS1)$, for ("0", "1") and ("4", "9"), and second worst for ("1", "7"), with f_1 worst using $(FS3)$. Note also that f_5 is always ranked in the middle using $(FS1, FS3)$. This implies that during training, classification accuracy is achieved most by a combination of at least $FS2$ and one other feature, with all three features achieving nearly the highest accuracy, even when compared to the multiple classifiers. These results match the intuition that the correlation coefficients between pairs of features give an indication of their relative performance, where, for example $FS1$ and $FS3$ are highly correlated and hence one conveys similar information to the other. Just using these two correlated features provides a median ranking, whereas just one of these combined with $FS2$ achieves at least a top three ranking. However, while this might follow intuition for training, these results are not carried through to testing.

With testing, the ranking of the classifiers changes moderately with f_7 only appearing in the top three base classifiers for ("0", "1"), however with the same number of losses (essentially equal ranking) as f_1 and f_5. For ("0", "1") and ("4", "9"), f_1 is ranked top, despite featuring in the bottom three during training. This shows that $FS3$ provides sufficient information for good generalization on its own for these two tasks over the combination of, say, $FS2$ and $FS3$ for f_3; even for ("1", "7"), f_1 achieves a significantly improved accuracy (77%), if not a much higher ranking.

Looking at the correlation based predictions for testing, f_6 with $FS1$ and $FS2$ achieves rank 2 in testing for ("1", "7"), but otherwise achieves no better than 118, while f_3, using $FS2$ and $FS3$, decreases in accuracy for ("0", "1") and ("1", "7"), but improves significantly for ("4", "9") to rank 33. This shows that, while our simple 1-NN classifiers have good generalization capability, the feature predictions from correlation coefficients do not provide a good indication of performance during testing, despite contrary observations during training. This is also in contrast to the predictions based upon the class discrimination provided by the visual inspection of the features (Figure 2), which suggests that the ("0", "1") task is the easiest, ("4", "9") moderate, and ("1", "7") the hardest, yet the test performance shows that ("4", "9") is the hardest (59% best test accuracy).

4.3. Multiple Classifier Results

Within the game we are bringing together a hierarchy of forecasters (base and multiple classifiers) to explore how we might predict the forecaster that will minimize the regret between that selected and the best forecaster. Our multiple classifiers do not adjust their parameters during training, but use instead the base classifiers, which do. Therefore, stability of the multiple classifiers is achieved after approximately 100 examples when the base classifiers themselves have stabilized. Table 5 shows the ranking for selected base and multiple classifiers and their corresponding classification accuracy for the three tasks. For example, we can see that for ("0", "1"), f_{58} is ranked highest, combining base classifiers f_3, f_5 and f_7, but during testing this only achieves a rank of 30.

Table 4. Base classifier training and testing rankings (with classification accuracy) for games run on digit pairings ("0", "1"), ("1", "7") and ("4", "9"). Note that rankings are in ascending order with 1 being the best and 127 the worst. The highest ranking classifier for each task is highlighted in italic.

Base classifier	("0", "1") Train		("0", "1") Test		("1", "7") Train		("1", "7") Test		("4", "9") Train		("4", "9") Test	
f_1 $(FS3)$	117	(72%)	***19***	***(95%)***	127	(49%)	112	(77%)	126	(49%)	***4***	***(59%)***
f_2 $(FS2)$	125	(63%)	127	(58%)	119	(56%)	127	(65%)	120	(51%)	79	(51%)
f_3 $(FS2, FS3)$	19	(95%)	43	(94%)	56	(70%)	126	(66%)	83	(55%)	33	(55%)
f_4 $(FS1)$	127	(63%)	119	(88%)	124	(54%)	65	(85%)	127	(48%)	125	(45%)
f_5 $(FS1, FS3)$	112	(73%)	20	(95%)	101	(60%)	55	(86%)	94	(55%)	44	(54%)
f_6 $(FS1, FS2)$	30	(92%)	118	(89%)	23	(76%)	***2***	***(91%)***	37	(58%)	120	(46%)
f_7 $(FS1, FS2, FS3)$	***2***	***(96%)***	21	(95%)	***5***	***(79%)***	66	(85%)	***7***	***(60%)***	6	(58%)

Table 5. Summary of training and testing rankings (with classification accuracy) for selected forecasters. For each task this shows 1) the highest ranked base classifier during training; 2) the highest ranked multiple classifier during training; 3) the highest ranked multiple classifier during testing, which for ("1", "7") is the same as 2); 4) the lowest ranked multiple classifier during training; and 5) the multiple classifier that combines all base classifiers. Testing loss and regret for each is also shown (see Section5).

	Train		Test		$\sum \ell$	R
			("0", "1")			
f_7	2	(96%)	21	(95%)	5	3
$f_{58}(3,5,7)$	1	(96%)	30	(95%)	5	3
$f_{65}(1,2,3,5)$	83	(80%)	1	(98%)	2	0
$f_{15}(2,4)$	126	(63%)	126	(70%)	32	30
$f_{127}(1-7)$	43	(89%)	42	(95%)	5	3
			("1", "7")			
f_7	5	(79%)	66	(85%)	16	8
$f_{63}(5,6,7)$	1	(80%)	1	(93%)	8	0
$f_8(1,2)$	126	(52%)	124	(71%)	31	23
$f_{127}(1-7)$	36	(75%)	40	(88%)	13	5
			("4", "9")			
f_7	7	(60%)	6	(58%)	42	4
$f_{97}(3,5,6,7)$	1	(62%)	30	(56%)	44	6
$f_{35}(1,3,5)$	111	(53%)	1	(62%)	38	0
$f_8(1,2)$	125	(49%)	34	(55%)	45	7
$f_{127}(1-7)$	5	(61%)	78	(52%)	48	10

In general, the multiple classifier rankings match well with their component base classifiers. During training, the highest ranking multiple classifiers all use at least one of the top 2 ranked base classifiers. For example, for ("4", "9"), the top 78 ranked multiple classifiers use either f_6 or f_7. If we calculate a mean rank for each group of multiple classifiers that uses each base classifier (take the mean of all that use base classifier f_1, then f_2 and so forth), then the order of the mean ranks match the base classifier rank orders with only a few minor exceptions.

During testing, this relationship between the multiple and base classifiers is maintained, with the same order of mean rankings for testing to that of the base classifiers, following the different rankings observed for testing as described above. We note however for both training and testing that combiner f_{127} (Table 5), which uses all base classifiers, performs no better than the base classifiers, except during for training for ("4", "9"). This acts as a useful benchmark if we consider an approach that subscribes to the principle that an ensemble of more components provides improved performance. We can see from this that improvement can only be obtained in this scenario by considering different strategies combining a lesser number of components, which we will discuss shortly.

4.4. Discussion

Our hierarchy of forecasters appears to provide sufficient complexity within a constrained scenario to evaluate successful strategies for picking the best forecaster, albeit with a limited amount of data to aid evaluation and reduce computational complexity. In particular, we have gone from the level of features to base classifiers to multiple classifiers so that the game of expert advice is applied to forecasters that them-

selves have their own set of experts. However, in selecting classifiers we have deliberately chosen simple algorithms to constitute both the base and multiple classifiers. In order to achieve higher levels of performance, we could have selected to use an increased number of neighbors, for example, or more complex combining algorithms that attempt to adjust the component classifiers based upon the overall performance of the ensemble. Using such algorithms, as discussed in section 4, can give error rates of less than 1%, so our achieved performances are not competitive, but at least for ("0", "1") is reasonable.

The relationship between the base and multiple classifiers is perhaps intuitive when we consider that the multiple classifiers are only as good as their components, and hence high performing components leads to high performing ensembles. However, the notion of *diversity* (Kuncheva & Whitaker, 2003) in an ensemble, implies that ensemble performance can be improved if the components complement each other, so that, for example, if one makes an error, the other components compensate for this. This principle, for example, may explain why f_{127} does not perform optimally.

While metrics to quantify diversity and use it to actively construct ensembles are still being developed (see for example (Brown et al., 2005b) for regression), a simple measure of the component similarities can be obtained by calculating the correlation coefficient between pairs of base classifier outputs. For example, the outputs of f_3 and f_7 during training for ("0", "1") are highly correlated with a coefficient of 0.97, similarly between f_6 and f_7, and f_3 and f_6, yet the ensemble f_{59} that combines just these base classifiers is ranked 6 during training, and 53 during testing. While the correlation coefficient is a crude measure of diversity, this does suggest that further investigation of the influence dissimilarity has is required in such an exhaustive environment to help inform theory. This follows the work on boosting that links game theory to a sampling strategy for components successfully (Freund & Schapire, 1996), but which it may be possible to extend to the more general scenario we have presented in which the forecasters themselves are hierarchical.

5. Strategies for Picking the Best Forecaster

Now that we have provided a empirical environment that compares a hierarchy of forecasters, the question remains as to how we can select an appropriate strategy for selecting a forecaster that results in the *practical* reduction in regret. Recall that the cumulative regret is the difference between a forecaster and an expert, so that our task is to minimize the regret between a selected forecaster and the best performing forecaster.

In Table 5, we list the cumulative loss during testing for each of the selected classifiers. We have a number of choices for strategy that these losses show us, based purely on the training results. To provide a benchmark for each of these strategies, we consider two cases. First, the worst performing forecaster for each task, which gives a regret of 30, 23 and 7 for ("0", "1"), ("1", "7") and ("4", "9") respectfully (recall that our results show that these tasks are in order of difficulty, hence the corresponding reduction in regret for each task). Second, the naïve multiple classifier that combines all base classifiers (f_{127}), which gives regrets of 3, 5 and 10, which are relatively small themselves, given the overall testing set sizes of 106, 108 and 99.

The first strategy we consider is to use only the highest ranking base classifier as seen during training, which for these experiments is (f_7). This gives a regret of 3, 8 and 4, which improves upon the benchmarks provided, except for ("1", "7"). This strategy could be applied without computing any multiple classifiers. However, as we have seen, such multiple classifiers can give improved performance. As a consequence, our second strategy is to select the highest ranking multiple classifier. For each task this differs, with f_{58} giving a regret of 3, f_{63} 0 and f_{97} 6 for the three tasks, respectfully. For ("1", "7") this is optimal, but for ("4", "9") this is worse than the best base classifier, and hence using multiple classifiers for this strategy appears counterproductive, perhaps due to the difficulty of this task and the resulting poor accuracy (near chance) for all of the classifiers tested.

Both the strategies we consider here are straightforward and we have used these as a demonstration of our game-based approach only. Our approach provides sufficient generality, and indeed this was the aim, to develop more mature strategies. For example, the interaction between features and forecasters could be used by combining feature correlation coefficients with multiple classifier components to weight a forecaster's ranking, similar to how weighted majority voting works at the scale of the forecaster (Littlestone & Warmuth, 1994), rather than at the features.

6. Conclusion

In this paper we have presented a game-based approach to the task of predicting the best classifier in a set of base classifiers and the combinations. Our approach explicitly represents the interaction of fea-

tures, base classifiers and their combinations. Through a closed-world, well defined example, we have explored how this approach can be used to define strategies for the selection of the best forecaster. We show for this simple game that two basic strategies have moderate success in reducing the measure of regret, although these are obviously not optimal and are used for demonstration only.

In the future, we wish to apply the competitive prediction game to *structural* pattern recognition problems, using decision and output spaces containing sequences of operations that construct instances of a graph-based structural model. We will then use this as a basis to study how to learn effective combinations of structural pattern recognizers. We will also consider more sophisticated classifiers and theories (for example, (Cesa-Bianchi & Lugosi, 2006; Littlestone & Warmuth, 1994)) for making predictions, with the overall aim of developing automated methods for predicting the best classifier within a theoretical context that can be used in practice.

References

Akkus, A., & Guvenir, H. (1996). K nearest neighbour classification on feature projections. *Proc. Int'l Conf. Machine Learning* (pp. 12–19). San Francisco, USA.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, *24*, 509–522.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005a). Diversity creation methods: A survey and categorisation. *Information Fusion*, *6*, 5–20.

Brown, G., Wyatt, J. L., & Tiňo, P. (2005b). Managing diversity in regression ensembles. *Journal of Machine Learning Research*, *6*, 1621–1650.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. New York: Cambridge University Press.

Freund, Y., & Schapire, R. E. (1996). Game theory, on-line prediction and boosting. *Proceedings of the 9th Annual Conference on Computational Learing Theory* (pp. 325–332). New York: ACM Press.

Fumera, G., & Roli, F. (2005). A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Trans. PAMI*, *27*, 942–956.

Keysers, D., Deselaers, T., Gollan, C., & Ney, H. (2007). Deformation models for image recognition. *IEEE Trans. PAMI*, *29*, 1422–1435.

Kittler, J., Hatef, M., Duin, R., & Matas, J. (1998). On combining classifiers. *IEEE Trans. PAMI*, *20*, 226–239.

Kuncheva, L. (2004). *Combining pattern classifiers*. Hoboken, New Jersey: Wiley-Interscience.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles. *Machine Learning*, *51*, 181–207.

Lauer, F., Suen, C. Y., & Bloch, G. (2007). A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, *40*, 1816–1824.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, *86*, 2278–2324.

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, *108*, 212–261.

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. *Proc. NIPS 19* (pp. 1137–1144). Cambridge, MA: MIT Press.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Proc. Seventh Int'l Conf. Document Analysis and Recognition* (pp. 958–963).