Regularized Linear Models in Stacked Generalization

Sam Reid and Greg Grudic

University of Colorado at Boulder, Boulder CO 80309-0430, USA

Abstract. Stacked generalization is a flexible method for multiple classifier combination; however, it tends to overfit unless the combiner function is sufficiently smooth. Previous studies attempt to avoid overfitting by using a linear function at the combiner level. This paper demonstrates experimentally that even with a linear combination function, regularization is necessary to reduce overfitting and increase predictive accuracy. The standard linear least squares regression can be regularized with an L2 penalty (Ridge regression), an L1 penalty (lasso regression) or a combination of the two (elastic net regression). In multi-class classification, sparse linear models select and combine individual predicted probabilities instead of using complete probability distributions, allowing base classifiers to specialize in subproblems corresponding to different classes.

1 Introduction

Multiple classifier systems combine the predictions of many classifiers to produce the ensemble prediction [1, 2, 3]. Simple techniques such as voting or averaging can improve predictive accuracy by combining diverse classifiers [4]. More sophisticated ensemble techniques, such as ensemble selection, train a combination function in order to account for the strengths and weaknesses of the base classifiers and to produce a more accurate ensemble model [5].

Stacked generalization is a flexible method for multiple classifier systems in which the outputs of the base-level classifiers are viewed as data points in a new feature space, and are used to train a combiner function [6][1]. Ting and Witten [7] applied stacked generalization to classification problems, and found that a multiple response linear combiner outperformed several nonlinear combiners on their problem domains and selection of base classifiers. They also showed that in classification problems, it is more effective to combine predicted posterior probabilities for class membership than class predictions.

Caruana et al. [5] evaluated stacked generalization with logistic regression with thousands of classifiers on binary classification problems, and reported that stacked generalization tended to overfit, resulting in poor overall performance. In this paper, we remedy this overfitting and improve overall generalization accuracy through regularization.

[1] Wolpert introduced the ideas of internal cross-validation and trainable combiner functions together in his article; we use the term 'stacked generalization' to refer to the latter.

J.A. Benediktsson, J. Kittler, and F. Roli (Eds.): MCS 2009, LNCS 5519, pp. 112–121, 2009.

Regularization attempts to improve predictive accuracy by reducing variance error at the cost of slightly increased bias error–this is known as the bias-variance tradeoff [8]. In this paper, regularization is applied to linear stacked generalization for multi-class classification in order to improve predictive accuracy. In particular, Ridge regression [8], lasso regression [8], and elastic net regression [9] are used to regularize the regression model by shrinking the model parameters. Lasso regression and some settings of elastic net regression generate sparse models, selecting many of the weights to be zero. This means each class prediction may be produced by a different subset of base classifiers.

In our experiments, many classification algorithms and many parameter settings are used to build a library of base models as in Caruana et al. [10]. We also perform resampling at the ensemble level in order to obtain more statistically reliable estimates of performance without the expense of retraining base classifiers. We look at the correspondence between performance on subproblems and overall classifier performance, and interpret the behavior of sparse linear models in stacked generalization.

This paper is organized as follows: Section 2.1 formally describes stacked generalization, including usage of indicator functions to transform the multi-class problem into several regression problems and the class-conscious extension, StackingC. Section 2.2 describes linear regression, Ridge regression, lasso regression and elastic net regression, which are used to solve the indicator subproblems in stacked generalization. Section 3 describes empirical studies that indicate the advantage of regularization. Section 4 discusses the results and Section 5 concludes with a summary and future work.

2 Model

2.1 Stacked Generalization

Given a set of L classifiers $\hat{y}_i(\boldsymbol{x}|\boldsymbol{\theta})$, $i = 1..L$, the predictions of each classifier on a validation dataset $\mathcal{D}_{val}$ are aggregated and combined with the known labels to create a meta-level training dataset $\mathcal{D}'_{val}$. The combiner function is then trained on this meta-level validation dataset. Given a test point, the predictions of all base-level classifiers are combined to produce a new data point $\boldsymbol{x}'_i$. The combiner function is evaluated at the new data point $\boldsymbol{x}'_i$, and its output is taken as the ensemble output. Formally, the ensemble prediction of stacked generalization is given by $sg(\boldsymbol{x}) = c(y_{11}(\boldsymbol{x}), ..., y_{1K}(\boldsymbol{x}), ..., y_{L1}(\boldsymbol{x}), ..., y_{LK}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the test point, c is the classifier combiner function and y_{lk} is the posterior prediction of the l^{th} classifier on the k^{th} class. Following Ting and Witten, a regression function can be used at the meta-level by constructing one regression subproblem per class with an indicator function [7]. At prediction time, the class corresponding to the subproblem model with the highest output is taken as the ensemble output. A more general discussion of reducing classification to linear regression problems is given in Hastie et al. [8].

The most general form of stacked generalization includes all outputs from all base classifiers. To simplify the problem, Seewald recommends using a

class-conscious approach in which each indicator model is trained using predictions on the indicated class only, called StackingC [11]. Formally, the StackingC class prediction is given by $sc(\boldsymbol{x}) = \text{argmax}_k\, r_k(y_{1k}(\boldsymbol{x}), ..., y_{Lk}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the test point, $k = 1..K$ is an index over classes, r_k is the regression model for the indicator problem corresponding to the k^{th} class and y_{lk} is the posterior prediction of the l^{th} classifier for the k^{th} class. Ting and Witten report that StackingC gives comparable predictive accuracy while running considerably faster than stacked generalization [7], and Seewald also reports increased predictive accuracy. Based on these arguments, the experiments in this paper use StackingC rather than complete stacked generalization.

2.2 Linear Models and Regularization

The least squares solution is given by $\hat{y} = \hat{\boldsymbol{\beta}}\boldsymbol{x}$, where $\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$. Here $\hat{\boldsymbol{\beta}}$ is the vector of model parameters determined by the regression, N is the number of training data points, y_i is the true output on data point i, x_{ij} is the jth feature of the ith data point, and p is the number of input dimensions for the problem. In StackingC, the features are predicted probabilities from base classifiers. When the linear regression problem is underdetermined, there are many possible solutions. This can occur when the dimensionality of the meta-feature space L is larger than the effective rank of the input matrix (at most N), where L is the number of classifiers and N is the number of training points. In this case, it is possible to choose a basic solution, which has at most m nonzero components, where m is the effective rank of the input matrix.

Ridge regression augments the linear least squares problem with an L2-norm constraint: $P_R = \sum_{j=1}^{p} \beta_j^2 \leq s$. This has the effect of conditioning the matrix inversion problem by adding a constant k to the diagonal: $\boldsymbol{\beta} = (X^T X + kI)^{-1} X^T \boldsymbol{y}$. There is a one-to-one correspondence between s and k [8].

Lasso regression augments the linear least squares problem with an L1-norm constraint: $P_l = \sum_{j=1}^{p} |\beta_j| \leq t$. The L1-norm constraint makes the optimization problem nonlinear in y_i, and quadratic programming is typically used to solve the problem. Unlike Ridge regression, lasso regression tends to force some model parameters to be identically zero if the constraint t is tight enough, thus resulting in sparse solutions.

Zou and Hastie describe a convex combination of the Ridge and lasso penalties called the elastic net [9]. The penalty term is given by $P_{en}(\boldsymbol{\beta}|\alpha) = (1 - \alpha)\frac{1}{2}||\boldsymbol{\beta}||_{l2}^2 + \alpha||\boldsymbol{\beta}||_{l1}$, where $0 \leq \alpha \leq 1$ controls the amount of sparsity. The elastic net is particularly effective when the number of predictors p (or classifiers in StackingC) is larger than the number of training points n. The elastic net performs groupwise selection when there are many correlated features (unlike the lasso, which instead tends to select a single feature under the same circumstances). When there are many excellent classifiers to combine, their outputs will be highly correlated, and the elastic-net will be able to perform groupwise selection.

3 Experimental Studies

For empirical evaluations, we selected publicly available datasets with numerical attributes and $k \geq 3$ classes. Table 1 indicates the datasets and relevant properties. For the 26-class letter dataset, we randomly subsampled a stratified selection of 4000 points.

Table 1. Datasets and their properties

Dataset	*Attributes*	*Instances*	*Classes*
balance-scale	4	625	3
glass	9	214	6
letter	16	4000	26
mfeat-morphological	6	2000	10
optdigits	64	5620	10
sat-image	36	6435	6
segment	19	2310	7
vehicle	18	846	4
waveform-5000	40	5000	3
yeast	8	1484	10

Approximately half the data points (with stratified samples) in each problem are used for training the base classifiers. The remaining data is split into approximately equal disjoint segments for model selection at the ensemble level (e.g. stacking training data or select-best data) and test data, again in stratified samples. In a real-world application, the base classifiers would be re-trained using the combination of base-level data and validation data once ensemble-level hyperparameters are determined, but this is not done in our studies due to the expense of model library construction.

Previous studies with $L \geq 1000$ classifiers obtain one sample per problem, with no resampling due to the expense of model library creation, such as in Caruana et al. [5]; we partially overcome this problem by resampling at the ensemble training stages. In particular, we use Dietterich's 5x2 cross-validation resampling [12] over the ensemble training data and test data. We use the Wilcoxon signed-rank test for identifying statistical significance of the results, since the accuracies are unlikely to be normally distributed [13].

We generate around 1000 classifiers, including neural networks, support vector machines, k-nearest neighbors, decision stumps, decision trees, random forests, and adaboost.m1 and bagging models. See the Appendix for full details on construction of base classifiers. Source code and data sets are available at `http://spot.colorado.edu/~reids/reg-09/`

3.1 Ensemble Techniques

As a baseline for comparison, we select the best classifier as identified by accuracy on the held-out ensemble training set (*select-best*). We also compare our

linear models to voting (*vote*) and averaging (*average*) techniques. The StackingC approaches are denoted *sg-linear*, *sg-ridge* and *sg-lasso*.

For the majority of our datasets, there are more linear regression attributes p (same as the number of classifiers L) than data points n (equal to the number of stacking training points, roughly $\frac{N}{4}$)[2]. To solve this underdetermined system without resorting to the typical Ridge solution, we choose a basic solution as implemented in the Matlab *mldivide* function.

In order to select the Ridge regression penalty, we search over a coarse grid of $\lambda = \{0.0001, 0.01, 0.1, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$ using cross-validation, then use all validation data to train the ridge regression model with the selected penalty parameter. We use the Matlab implementation of Ridge regression from the Matlab Statistics Toolbox. Parameters are selected by cross-validation for each subproblem rather than choosing a single λ for all subproblems. For example, the regularization hyperparameter for the first indicator problem λ_1 may differ from λ_2. For lasso regression, we use the LARS software by Efron and Hastie [14], and search over a grid of $fraction = 0$ to 1 in increments of 0.01 to select the regularization penalty term by cross-validation for each subproblem. We search over a finer grid in *sg-lasso* than in *sg-ridge* since model selection is much more efficient in LARS. For the elastic net, we use the glmnet package written by Friedman, Hastie and Tibshirani and described in the corresponding technical report [15].

4 Results

The test set accuracies of all ensemble methods are shown in Table 2. Each entry in this table is an average over 10 folds of Dietterich's 5x2 cross-validation [12] over ensemble training/validation data. According to the pairwise Wilcoxon

Table 2. Accuracy of each model for each data set. Entries are averages over the 10 samples from Dietterich's 5x2 cross-validation at the ensemble level. Variances are omitted based on arguments in Demšar [13]. See Section 3 for a description of the methods and Section 4 for discussion.

Dataset	$select-best$	*vote*	*average*	$sg-linear$	$sg-lasso$	$sg-ridge$
balance-scale	**0.9872**	0.9234	0.9265	0.9399	0.9610	0.9796
glass	0.6689	0.5887	0.6167	0.5275	0.6429	**0.7271**
letter	0.8747	0.8400	0.8565	0.5787	0.6410	**0.9002**
mfeat-m	0.7426	0.7390	0.7320	0.4534	0.4712	**0.7670**
optdigits	0.9893	0.9847	0.9858	0.9851	0.9660	**0.9899**
sat-image	0.9140	0.8906	0.9024	0.8597	0.8940	**0.9257**
segment	0.9768	0.9567	0.9654	0.9176	0.6147	**0.9799**
vehicle	0.7905	0.7991	0.8133	0.6312	0.7716	**0.8142**
waveform	0.8534	0.8584	**0.8624**	0.7230	0.6263	0.8599
yeast	**0.6205**	0.6024	0.6105	0.2892	0.4218	0.5970

[2] The *waveform*, *letter*, *optdigits* and *sat-image* datasets are exceptions.

signed-ranks test [13], Ridge regression StackingC outperforms unregularized linear regression StackingC at $p \leq 0.002$. *Select-best* outperforms both unregularized and lasso regression StackingC at $p \leq 0.002$. Ridge regression StackingC outperforms *select-best* at $p \leq 0.084$, and has more wins than any other algorithm. On two problems, *select-best* outperforms all model combination methods. On all problems, *sg-linear* and *sg-lasso* perform less accurately than *sg-ridge*; this suggests that it may be more productive to assign nonzero weights to all posterior predictions when combining several base classifiers.

To study the effect of regularization on each subproblem, we plot the root mean squared error for a particular indicator subproblem as a function of the regularization penalty hyperparameter. Computation of a reasonable composite value over all data sets is difficult due to incommensurability of the problems, so we restrict our focus to a particular subproblem[3]. Figure 1(a) shows the root mean squared error in the first subproblem in the *sat-image* dataset[4]. As the ridge penalty λ increases from 10^{-8} to 10^{3}, the error decreases by more than 10%. With such a small penalty term, the error at 10^{-8} roughly corresponds to the error that would be obtained by unregularized linear regression. For individual subproblems, therefore, regularization dramatically improves performance.

Figure 1(b) shows the overall accuracy of the multi-response linear regression with Ridge regularization for the *sat-image* dataset. Regularization increases the accuracy of the overall model by about 6.5%, peaking around $\lambda = 10^{3}$. As the penalty is increased beyond 10^{3} (not pictured), the accuracy decreases, reaching 0.24, the proportion of the predominant class, around $\lambda = 10^{8}$. Please note that in this figure, λ is the same for all subproblems.

Figure 1(c) shows the correlation between the accuracy of the overall multi-response linear regression system and the root mean squared error on the first subproblem. The fit is approximately linear, with $a = -0.408e + 0.957$, where a is the accuracy of the multiclass classifier and e is the RMSE of the classifier on the first indicator subproblem.

Figure 2(a) shows the overall accuracy of the multi-response linear regression system as a function of the penalty term for lasso regression for the *sat-image* problem. Standard errors over the 10 folds are indicated. As in the Ridge regression case, λ is the same over all subproblems in this figure. The accuracy falls dramatically as the penalty increases beyond 0.2, stabilizing after $\lambda = 0.50$ at an accuracy of 0.24, the proportion of the predominant class.

In order to view the effect of the elastic net's mixing parameter α on the accuracy of the multi-response system, accuracy vs penalty curves are plotted in Figure 2(b) for $\alpha = \{0.05, 0.5, 0.95\}$. The $\alpha = 1.0$ curve indicated in Figure 2(a) is highly similar to the $\alpha = 0.95$ curve, and therefore omitted from Figure 2(b) for clarity. With a small penalty term $\lambda \leq 10^{-1}$, the curves are constant, and within one standard deviation of the *select-best* curve. As the penalty increases, the accuracy reaches a maximum that is dependent on α, with higher α values yielding

[3] Results are qualitatively similar for other subproblems.

[4] The root mean squared error is used instead of the accuracy because the subproblem in multi-response is a regression problem.

higher accuracy at smaller penalty values. In this case, fine-tuned regularization increases accuracy by about 1.5%.

In Ridge-regularized stacked generalization, all predictors are given some portion of the total weight. In lasso regression and some settings of elastic net regression, it is possible to obtain a sparse model in which many weights are identically zero. This reduces computational demand at prediction time and makes it

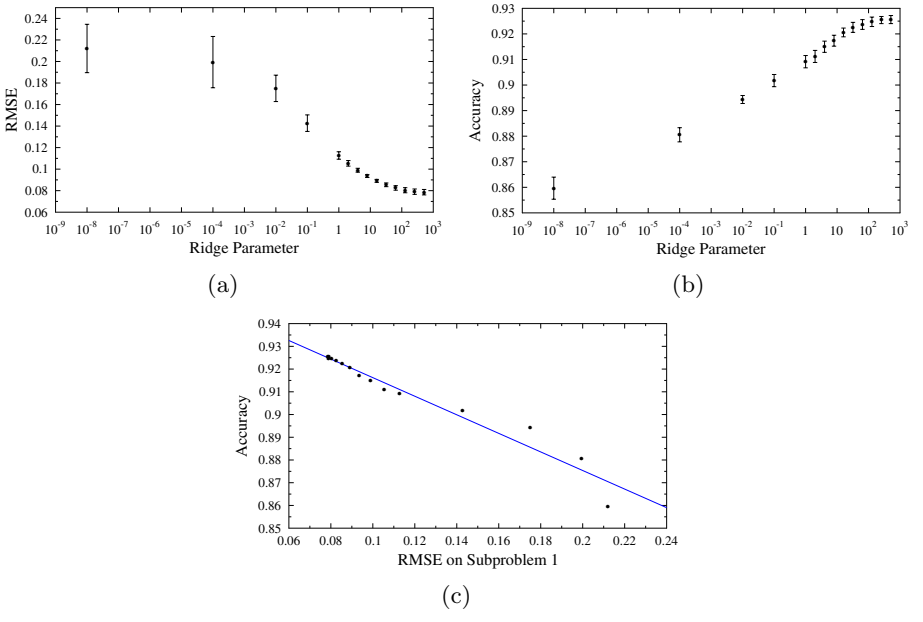
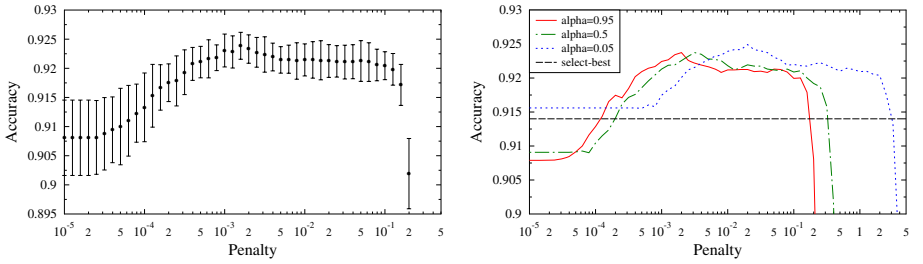


Fig. 1. Figure 1(a) shows root mean squared error for the indicator problem for the first class in the *sat-image* problem. Figure 1(b) shows overall accuracy as a function of the Ridge parameter for *sat-image*. The error bars indicate one standard deviation over the 10 samples of Dietterich's 5x2 cross validation. Figure 1(c) shows the accuracy of the multi-response linear regression system as a function of mean squared error on the first class indicator subproblem for Ridge regression.

Table 3. Selected posterior probabilities and corresponding weights for the *sat-image* problem for elastic net StackingC with $\alpha = 0.95$. Only the 6 models with highest total weights are shown here. *ann* indicates a single-hidden-layer neural network, and corresponding momentum, number of hidden units, and number of epochs in training.

$Classifier$	$class-1$	$class-2$	$class-3$	$class-4$	$class-5$	$class-6$	$total$
adaboost-500	0.063	0	0.014	0.000	0.0226	0	0.100
ann-0.5-32-1000	0	0	0.061	0.035	0	0.004	0.100
ann-0.5-16-500	0.039	0	0	0.018	0.009	0.034	0.101
ann-0.9-16-500	0.002	0.082	0	0	0.007	0.016	0.108
ann-0.5-32-500	0.000	0.075	0	0.100	0.027	0	0.111
knn-1	0	0	0.076	0.065	0.008	0.097	0.246



(a) (b)

Fig. 2. Overall accuracy of the multi-response linear regression system as a function of the penalty term for lasso regression for the *sat-image* problem, with standard errors indicated in Figure 2(a). Figure 2(b) shows accuracy of the multi-response linear regression system as a function of the penalty term for $\alpha = \{0.05, 0.5, 0.95\}$ for the elastic net for *sat-image*. The constant line indicates the accuracy of the classifier chosen by *select-best*. Error bars have been omitted for clarity, but do not differ qualitatively from those shown in Figure 2(a).

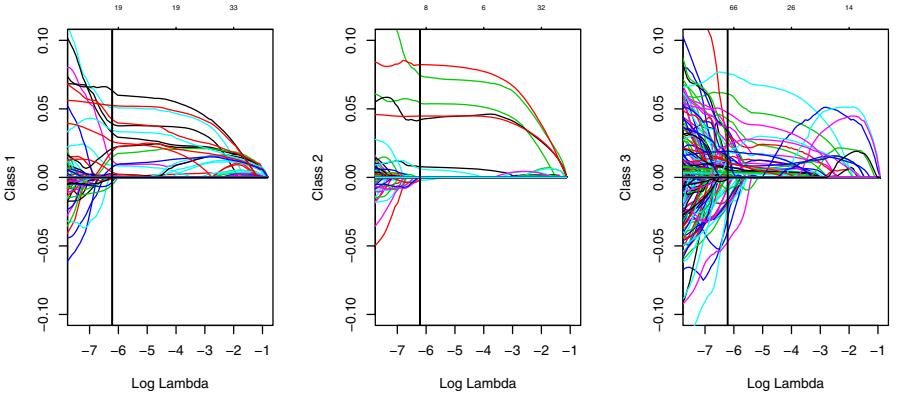


Fig. 3. Coefficient profiles for the first three subproblems in StackingC for the *sat-image* dataset with elastic net regression at $\alpha = 0.95$. This is over a single partition of ensemble training and testing data.

possible to identify a small subset of base classifiers and predictions that are responsible for making the overall prediction. Figure 3 shows the coefficient profiles for the *sat-image* dataset, for classes 1-3 with elastic net regularized StackingC with $\alpha = 0.95$. The optimal value of λ according to overall classification accuracy is shown as a vertical line. At $\lambda = \lambda_{opt}$, only 244 of the 999 classifiers are assigned weight for any of the subproblems. Table 3 shows the 6 classifiers with the highest total sum of weights for all classes. Sparse models obtained by L1-regularized linear regression can choose different classifiers for each class–that is, classwise posterior predictions are selected instead of complete classifiers. For instance, the classifier assigned the most total weight is $k = 1$-*nearest neighbor*,

which contributes to the response for classes 3-6, but doesn't appear in the predictions for classes 1 or 2. The base classifier that makes the largest contribution to the *class-1* prediction is boosted decision trees run for 500 iterations. Thus each classifier is able to specialize in different class-based subproblems rather than being required to predict accurate probabilities for all classes.

5 Conclusion

Stacked generalization has a tendency to overfit; overfitting is even more likely when using many highly correlated, well-tuned models. In order to avoid overfitting and to improve prediction accuracy, it is necessary to perform regularization at the combiner level, even when using a linear combiner. Regularization can be performed by penalization of the L2 norm of the weights (Ridge regression), L1 norm of the weights (lasso regression) or a combination of the two (elastic net regression). L1 penalties yield sparse linear models; in stacked generalization, this means selecting from a small number of classifier posterior predictions.

An interesting extension of this work would be to examine the full Bayesian solutions (under Gaussian and Laplacian priors for regularization), instead of the single-point maximum likelihood estimates implicit in the Ridge (Gaussian prior) and lasso (Laplacian prior) regularizers. Other work could study additional regularization by (a) selecting a single regularization hyperparameter for use in all subproblems or (b) constraining the weights to be non-negative for each subproblem.

Acknowledgments

The authors would like to thank PhET Interactive Simulations at the University of Colorado at Boulder for their support, the Turing Institute, Glascow, Scotland and the UCI Repository for providing datasets and the reviewers for their helpful remarks.

References

1. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
2. Roli, F., Giacinto, G., Vernazza, G.: Methods for designing multiple classifier systems. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 78–87. Springer, Heidelberg (2001)
3. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
4. Breiman, L.: Random forests. Mach. Learn. 45, 5–32 (2001)
5. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: ICML 2004: Proceedings of the twenty-first international conference on Machine learning, p. 18. ACM, New York (2004)
6. Wolpert, D.H.: Stacked generalization. Neural Netw. 5, 241–259 (1992)

7. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2003)
9. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B 67, 301–320 (2005)
10. Caruana, R., Munson, A., Niculescu-Mizil, A.: Getting the most out of ensemble selection. In: ICDM 2006: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, pp. 828–833. IEEE Computer Society, Los Alamitos (2006)
11. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: ICML 2002: Proceedings of the Nineteenth International Conference on Machine Learning, pp. 554–561. Morgan Kaufmann Publishers Inc., San Francisco (2002)
12. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1923 (1998)
13. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
14. Efron, B., Hastie, T., Johnstone, L., Tibshirani, R.: Least angle regression. Annals of Statistics 32, 407–499 (2004)
15. Friedman, J., Hastie, T., Tibshirani, R.: Regularized paths for generalized linear models via coordinate descent. Technical report, Stanford (2008)

Appendix: Base Classifiers

We generate about 1000 classifiers for each problem. For each classification algorithm, we generate a classifier for each combination of the parameters specified below. All implementations are in Weka except for the Random Forest (R), for which we used the R port of the Breiman-Cutler code by Andy Liaw, available through CRAN.

1. Neural Network decay={true, false} momentum={0.1, 0.5, 0.9} learningRate={0.5, 0.75, 0.9} trainingTime={100, 500, 1000} numHiddens={2, 4, 16, 32}
2. Support Vector Machine (C-SVM) kernelType={linear, polynomial, rbf, sigmoid} coef0={-1, 1} cost={0.1, 1.0, 10, 100, 1000} degree={1, 2, 3} eps={0.001, 0.01} gamma={0.1, 0.3, 0.8}
3. K-Nearest Neighbor k={1, 2, 4, 16, 32, 64}
4. Decision Stump
5. Decision Tree (J48) binarySplits={true, false} confidenceFactor={0.25, 0.5, 0.75} reducedErrorPruning={false, true} unpruned={true, false}
6. Random Forest (Weka) numTrees={1, 2, 30, 50, 100, 300, 500}
7. AdaBoost.M1 numIterations={10, 50, 100, 500} classifier={J48 binary Splits={true, false}, Decision Stump}
8. Bagging classifier={J48 binarySplits={true,false}} numBags={5, 10, 50}
9. Random Forest (R) numTrees={1, 2, 30, 50, 100, 300, 500}