# A Greedy Algorithm for Selecting Models in Ensembles

Andrei L. Turinsky
*University of Calgary, Canada*
*aturinsk@ucalgary.ca*

Robert L. Grossman
*University of Illinois at Chicago, USA*
*and Open Data Partners, USA*
*grossman@uic.edu*

## Abstract

*We are interested in ensembles of models built over k data sets. Common approaches are either to combine models by vote averaging, or to build a meta-model on the outputs of the local models. In this paper, we consider the model assignment approach, in which a meta-model selects one of the local statistical models for scoring. We introduce an algorithm called Greedy Data Labeling (GDL) that improves the initial data partition by reallocating some data, so that when each model is built on its local data subset, the resulting hierarchical system has minimal error. We present evidence that model assignment may in certain situations be more natural than traditional ensemble learning, and if enhanced by GDL, it often outperforms traditional ensembles.*

## 1. Introduction

In a standard approach to data mining, a learning algorithm F is applied to a single data set D, where D consists of instances of the form (x,y), with x a vector of data attributes and y a class label. The choice of the algorithm defines a parametric family of predictive models. Given the algorithm F and the dataset D, a predictive model $f:x \rightarrow y$ is built to predict the class value of unlabeled data instances.

Ensembles of models arise in several different ways. First, they can be built by taking a single data set and using sampling with replacement to produce k separate data sets [1]. Second, ensembles arise naturally in distributed data mining, where the data may be naturally distributed over k geographical sites. While it may be possible to move all data to a central site and build a single model there, it is often too costly or otherwise impractical. The alternative is to mine all data in-place and thus build k predictive models (base-models) locally. Third, ensembles of models arise naturally in hierarchical modeling. Here the outputs of one or more models are used as the inputs to another model [2]. Hierarchical modeling arises naturally when data comes from different underlying distributions, in which case a collection of specialized models is desirable. Consider an (idealieze) application where a separate model predicts the risk of a heart disease for each age group. When a new patient arrives, the *meta-model* will invoke the predictive model that corresponds to the patient's age.

There are several ways to produce a single score from an ensemble of models. The simplest is a voting/averaging ensemble [3], in which case the meta-model is just an averaging function. A more complex one is meta-learning, where individual base-models make predictions, after which the meta-model reads their scores and makes the overall prediction [4].

A third technique is *model assignment*, or *model selection*, where the meta-model delegates the scoring of an unlabeled data instance to one of the base-models, as in the example with the heart disease prediction above.

Consider k data sets $D_1,\ldots, D_k$, each represented by a different color. Denote by D their union as a bag (i.e. points may occur with multiplicity). A model assignment system is created as follows. A base-model $f_i$ is built on each data subset $D_i$ using a learning algorithm F. Then a sample of data from each color is used to train a meta-model that can predict colors.

In this paper, we assume that we are given the k data sets $D_1,\ldots, D_k$, but have the flexibility to move data between them. This assumption is natural for the cases of distributed data mining and hierarchical modeling mentioned above. We introduce a novel algorithm, called Greedy Data Labeling (GDL), for learning base-models on $D_1,\ldots, D_k$ and a meta-learner for model selection. The algorithm relies on moving small amounts of data between the various data sets $D_i$.

We give some preliminary experimental studies showing that in interesting cases involving heterogeneous data it outperforms traditional ensemble learning based upon voting. Since one of our interests is in exploring distributed data partitions, in this paper, we consider the special case in which data instances belong to only one color at a time. Naturally, the GDL algorithm depends on the initial distribution of data into the subsets $\{D_i\}$. This reflects the underlying structure of both distributed data mining and hierarchical modeling of heterogeneous data.

## 2. Related work

The GDL method is broadly related to unsupervised clustering, but uses a more general measure of tightness based on the choice of the learning algorithm F. It also employs an idea related to boosting [5], in which previously misclassified data is re-sampled more frequently to give the predictive model more chances to learn it. Bagging [1] is one of the main methods for building a collection of classifiers, where each classifier is trained on a subset of data drawn from the same distribution. In practice, the data subsets are usually created by re-sampling the original dataset with replacement and hence may share some data.

A system where classifiers may specialize and/or abstain from voting is considered in [6]. A method of selecting classifiers based on local accuracy is presented in [7]. A Behavior-Knowledge Space method considers each possible output combination in the space of combined outputs of multiple classifiers [8]. In [9], conceptual clustering creates a classification hierarchy based on how well objects fit descriptive concepts. For related results, see also [10]. There are various techniques to recover individual data distributions from a mixture of distributions, such as the popular EM algorithm [11]. Other related methods are described in [12].

The novelty of the GDL approach is in the fact that, unlike the above methods, it is geared towards applications and situations, such as distributed data mining with high bandwidth networks, in which moving some of the data between local data sets is practical. In particular, it uses data reallocation to improve the structure of local data distributions before the distributed learning process begins. The resulting system deploys local classifiers in a way that reflects the realities of the current data partition.

Finally, the GDL algorithm uses simulated annealing to escape local minima, which allows suboptimal exploration steps in greedy descend [13].

## 3. The Greedy Data Labeling algorithm

The GDL algorithm is concerned with the scenario in which we move some amount of data between the various $D_i$ to improve the data partition. We are not interested in the two extreme cases in which we either combine all the data to produce a single data set D or leave all the data in separate partitions. We also assume that the attributes are discrete (or discretized by binning), an assumption that is required for several types of classifiers commonly used in practice, such as the naïve Bayesian classifiers.

Once the data partition $\{D_i\}$ is found by the GDL algorithm, we take a $\mu$ percent sample of the data from all subsets (a *meta-sample*) and train a meta-model to learn the k colors, as defined by the current data partition.

Given a data partition $\{D_i\}$, we define the color $\eta$ as *native* for a data instance (x,y) if

$$\eta(x,y) = \arg\min_{1 \le i \le k} \| y-f_i(x)\|$$

i.e. if the base-model $f_\eta(x)$ built on the subset $D_\eta$ predicts the true class of that instance better than base-models of other colors. Here, $\|y-f_i(x)\|$ is the appropriately defined prediction error. In case of a tie, the native color is chosen randomly among the candidates.

The GDL algorithm performs the following greedy iteration until all data points are in their native subsets:

- Initialize a batch size parameter T.
- For all data points (x,y) in D:
  - Find the prediction error $\varepsilon(i) = \|y-f_i(x)\|$ on this data point for each base-model.
  - Identify the native color $\eta$ of the point.
  - Compute the greedy step $\delta = \varepsilon(c) - \varepsilon(\eta)$, where c is the current color.
  - Using simulated annealing, reallocate the data point with probability $p=\exp(-1/(\delta t))$.
  - Stop when T data points are reallocated.
- Update all base-models.
- Reduce batch size T in half for the next iteration.

Here, t is a control parameter tied to the batch size T. When $\delta=0$, $p=0$ and the data instance is never chosen. Alternatively, $p=\exp(-1/((\delta+1)t))$ would allow to select instances with $\delta=0$ with non-zero probability and escape local minima better at the expense of a larger number of iterations. GDL moves data in batch to minimize base-model rebuilding after each reallocation, and thus requires only a few base-model updates before convergence. We have not noticed any problems related to overfitting.

There are noticeable similarities between the GDL method and clustering techniques. For example, the k-means clustering is initially seeded with k centroids. New data is placed into the cluster with the nearest centroid, after which the centroid is recomputed. In the GDL, the algorithm is seeded by the initial data partition. New data is placed into the subset whose model gives the least prediction error, after which the model is recomputed. In both cases, instances are moved between data subsets to improve a certain measure of tightness within subsets.

## 4. Experimental datasets and partitions

The goal of our experiments was two-fold: (a) to show that a model assignment approach may in certain situations outperform traditional voting methods which use all base-models for prediction; (b) to explore the quality of data partitions produced by the GLD algorithm.

We selected several datasets from the UCI Machine Learning Repository [14] that satisfy three criteria: (a) all attributes are discrete, (b) the number of data instances is at least 500, sufficiently large to understand the behavior of the GDL algorithm on well studied data, (c) random data partitions results in at least 1% classification error, i.e. the domain is sufficiently complex. The UCI datasets we used in our experiments are:

- Balance Scale data: 625 instances, 4 attributes
- Tic-Tac-Toe data: 958 instances, 9 attributes
- Car Evaluation data: 1728 instances, 6 attributes
- Chess data: 3196 instances, 36 attributes
- Nursery data: 12960 instances, 8 attributes

To examine the quality of data partitions produced by the GDL method, we compared three types of partitions.

First, we were interested in modeling the situations for the initial partitions where data comes from either a single source or different sources. Because we used real data and could not control its source, we modeled these two situations as follows. For a *homogeneous* initial split, the training data was divided into k equal parts randomly. To model a *heterogeneous* partition, we sorted the training data by its most informative attribute (found by building a C4.5 tree) and split it sequentially into k subsets.

Second, random mixture partitions (RM) were produced by making each initial subset $D_i$ exchange a $\lambda$ percent of its randomly selected data with other subsets, divided equally among the receiving subsets.

Third, the GDL partitions were made by exchanging data instances selected by the GDL algorithm between initial subsets. We initialized the batch size parameter T to $(\lambda/2)|D|$ for each value of $\lambda$, using the same values as in random mixture (RM). Because GDL reduces T in half after each iteration, this choice ensures that the total data traffic does not exceed $\lambda/2+\lambda/4+\ldots < \lambda$ percent. Our goal was to demonstrate that with GDL, we get superior colorings moving less data than with the RM approach.

## 5. Test results and discussion

The models were built using the WEKA package [15], with C4.5 decision trees as the base-models [16] and three different types of meta-models: C4.5, naïve Bayesian, and one-nearest neighbor models. Three 5-fold cross-validations were used to test model assignment systems, as well as voting ensembles for non-GDL partitions. We set k=3 or k=10, the meta-sample percentage $\mu$ = 100, 50, 25 and 10%, and the data mixing parameter $\lambda$ = 10, 20, 40, 60, 80, and 90%. With zero-or-one classification error, annealing parameter $\delta(x,y)$ takes values 0 or 1 (native or non-native current color), which can be shown to reduce annealing to a random selection of T currently misclassified data points.

We observed that the nearest neighbor meta-model was the best choice. Table 1 shows classification errors for a voting ensemble of C4.5 base-models (ENS) vs. a model assignment system with the same base-models and a nearest neighbor meta-model (NN $\mu$) built for different values of $\mu$. The results are shown for each type of data partition for both homogeneous (same) and heterogeneous (different) data, averaged over the five UCI datasets.

**Table 1. Classification errors (%) of voting ensembles v. NN-based model assignment systems with different meta-sample sizes (%), on different type of data partitions, average for 5 UCI datasets**

| Data source | Data part. | ENS | NN 100 | NN 50 | NN 25 | NN 10 |
|---|---|---|---|---|---|---|
| Same k=3 | Init | 13.5 | 14.1 | 14.8 | 15.3 | 15.7 |
| | RM | 13.6 | 14.3 | 14.9 | 15.4 | 15.8 |
| | GDL | n/a | 11.3 | 12.5 | 13.7 | 14.7 |
| Same k=10 | Init | 16.0 | 17.3 | 18.2 | 19.0 | 19.8 |
| | RM | 16.1 | 17.0 | 18.1 | 18.9 | 19.8 |
| | GDL | n/a | 12.1 | 14.3 | 16.1 | 18.0 |
| Diff k=3 | Init | 28.4 | 10.7 | 11.6 | 13.9 | 16.6 |
| | RM | 15.7 | 12.3 | 13.3 | 14.5 | 15.7 |
| | GDL | n/a | 9.6 | 10.5 | 12.8 | 16.1 |
| Diff k=10 | Init | 30.5 | 12.3 | 14.5 | 18.3 | 22.0 |
| | RM | 17.9 | 14.1 | 16.1 | 18.1 | 19.7 |
| | GDL | n/a | 10.8 | 13.1 | 17.1 | 21.0 |

The tests confirm that combining data into fewer subsets results in better predictive systems of either type, and that the quality of the model assignment system drops as the size $\mu$ of the meta-sample decreases. Given the same collection of base models, care must be taken to build a sufficiently accurate meta-model.

For the homogeneous data (same-source), ensembles built on initial partitions are generally superior, and exchanging random samples of already homogeneous data has no effect. However, model assignment systems enhanced by GDL consistently outperform ensembles, sometimes even with meta-samples as small as 10%.

For the heterogeneous data (different-sources), ensembles perform poorly on the initial partitions. Their accuracy can be improved dramatically by exchanging random samples of data, which creates a more homogeneous data distribution across the k datasets. On the other hand, model assignment systems appear to be far superior to ensembles, and with GDL their accuracy improves even further. In effect, ensembles are "penalized" if their base-models are over-specialized, whereas model assignment systems are "rewarded"', and the GDL makes this effect even more pronounced.

In Table 2, we compare ensemble systems to model assignment under their respective optimal conditions: ensembles are built on random mixture partitions, and the model assignment systems are built on GDL partitions with meta-sample size $\mu$ =100%. We make several observations. The choice of a meta-model algorithm affects the performance significantly, with nearest neighbor being the best meta-models. GDL moves only a small percentage of data, especially when the initial partition already represents different data sources (5.0% vs. 10.3% for k=10). In contrast, the amount of traffic in random mixture approach is fixed, potentially large, and not reflecting the internal structure of data. In terms of efficiency, an average of about four greedy GDL iterations is required for convergence.

**Table 2. Classification errors, data traffic in GDL (% of total data size), and the number of greedy iterations, average for 5 UCI datasets**

| Data source | k | ENS % | NN % | C45 % | NB % | Traf % | Iter |
|---|---|---|---|---|---|---|---|
| Same | 3 | 13.6 | 11.3 | 13.5 | 15.2 | 6.7 | 3.8 |
| Same | 10 | 16.1 | 12.1 | 15.7 | 18.5 | 10.3 | 4.9 |
| Diff | 3 | 15.7 | 9.6 | 11.4 | 14.5 | 4.2 | 3.2 |
| Diff | 10 | 17.9 | 10.8 | 11.7 | 17.3 | 5.0 | 4.3 |

## 6. Conclusion

We compared several scenarios for selecting models out of an ensemble of k models. It appears that in the case of a homogeneous data distribution, traditional voting or averaging ensembles outperform model selection approaches. On the other hand, when the base models reflect a natural heterogeneity in the data, delegating the prediction to one model out of an ensemble outperforms voting/averaging. We also introduced an algorithm called Greedy Data Labeling (GDL) that enhances data partitions in model assignment problems by moving small portions of data between the different datasets before the learning algorithms are applied. The resulting model assignment system outperforms traditional ensembles for both homogeneous and heterogeneous datasets. Future work on GDL includes developing data weighting schemas and randomization mechanisms for distributed applications.

## 10. References

[1] L. Breiman, "Bagging predictors". *Machine Learning*, 1996, Vol.24, No. 2, pp.123-140.

[2] Draper, D., *Bayesian Hierarchical Modeling*, Springer-Verlag, New York, 2001.

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

[4] P.K. Chan and S.J. Stolfo, "Learning arbiter and combiner trees from partitioned data for scaling machine learning", *Proc. 1st Int. Conf. Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, 1995, pp. 39--44

[5] Y. Freund, "Boosting a weak learning algorithm by majority", *Information and Computation*, 1995, Vol. 121, No. 2, pp. 256--285.

[6] Y. Freund, R.E. Schapire, Y. Singer, and M.K. Warmuth, "Using and combining predictors that specialize", *Proc. of the 29th Annual ACM Symposium on the Theory of Computing*, 1997, pp.334--343.

[7] K. Woods, W.P. Kegelmeyer, and K. Bowyer, "Combination of Multiple Classifiers using Local Accuracy Estimates", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1997, Vol. 19, No. 4, pp. 405-410.

[8] Y.S. Huang and C.Y. Suen, "Combination of multiple experts for the recognition of unconstrained handwritten numerals", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1995, Vol. 17, pp. 90-94.

[9] R.S. Michalski and R.E. Stepp, "Automated construction of classifications: conceptual clustering versus numerical taxonomy", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1983, Vol. 5, pp. 396-410.

[10] U. Fayyad, C. Reina, and P.S. Bradley, "Initialization of iterative refinement clustering algorithms", *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, 1998, pp. 194-198

[11] M.I. Jordan and R.A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm", *Neural Computation*, 1994, Vol. 6, No. 2, pp. 181-214.

[12] Kuncheva, L.I., *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-InterScience, 2004.

[13] Van Laarhoven, P.J.M. and E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, D.Reidel, Norwell, 1987.

[14] C.L. Blake and C.J. Merz, *UCI Repository of machine learning databases*, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Avail. at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[15] Witten, I.H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, San Mateo, CA, 1999. WEKA software avail. at http://www.cs.waikato.ac.nz/ml/weka/.

[16] Quinlan, J.R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.