Classifier Ensemble Recommendation

Pyry Matikainen[1] Rahul Sukthankar[2,1] Martial Hebert[1]
pmatikai@cs.cmu.edu rahuls@cs.cmu.edu hebert@ri.cmu.edu

[1]The Robotics Institute, Carnegie Mellon University
[2]Google Research

Abstract. The problem of training classifiers from limited data is one that particularly affects large-scale and social applications, and as a result, although carefully trained machine learning forms the backbone of many current techniques in research, it sees dramatically fewer applications for end-users. Recently we demonstrated a technique for selecting or *recommending* a single good classifier from a large library even with highly impoverished training data. We consider alternatives for extending our recommendation technique to *sets* of classifiers, including a modification to the AdaBoost algorithm that incorporates recommendation. Evaluating on an action recognition problem, we present two viable methods for extending model recommendation to sets.

1 Introduction

Classifiers continue to be impractical for web-scale uses, and when they are used, they tend to be in the form of *generic* classifiers (*e.g.*, face detectors) rather than the carefully tuned custom classifiers seen in research applications. The reasons for this are twofold. First, properly training a classifier is a surprisingly difficult affair — aside from the amount of data required (which might be substantial), methodological issues such as cross-validation for appropriate SVM parameters and the production of appropriately 'hard' negative examples complicate the neat theoretical story. Second, classifiers are computationally expensive to run and train when compared to web-scale technologies like approximate nearest neighbor methods and hashing techniques.

While this second concern will likely be alleviated in time by advances in the amount of available computing power, the first concern, that classifiers are in practice difficult to produce, must be given more care.

Our recent work [1] suggests a way out: rather than requiring end users to train classifiers, we recast the problem as a *selection* problem from a *pre-trained* library of classifiers. Instead of an end-user trying to directly learn a classifier from limited training data, the training data should be instead be used *select* a classifier from a pre-trained library according to which classifier had the best accuracy. But of course there is no "free lunch": with limited labeled training data, it is also hard to even *measure* the accuracy of a classifier. What we realized is that if many users are attempting to perform such a selection at the same time, then this problem of selecting the best classifier from a large library is *exactly analogous* to that addressed by collaborative filtering techniques for recommender systems, such as Netflix and Amazon.com! While any one user's measured classifier accuracies will be inaccurate, when considered jointly, there are correlations between classifier accuracies across different tasks that collaborative filtering methods can exploit to improve the classifier accuracy estimates for individual tasks. We were able to show that it is possible in practice to select a better classifier than by directly trying *every* option on the limited *training* set.

That technique, termed "model recommendation", can naturally be adapted to web-scale, since it is built around collaborative filtering techniques that were designed with web-scale data (*e.g.*, the Netflix ratings database) in mind. Furthermore, the formulation has a number of other benefits: it allows powerful classifier techniques to be used; in fact, any type of underlying classification machinery can be used, because the recommendation only uses the measured ratings or accuracies of those classifiers, and does not need to know anything about their underlying implementations. The work can be distributed: because the most computationally expensive step (evaluating the classifiers) can be performed by the users themselves in parallel. Additionally, since the users can evaluate classifiers locally and only return the ratings, some measure of privacy is preserved, since the raw samples (*e.g.*, videos) do not need to be sent to the recommendation system.

Although model recommendation might naturally be applied to banks of classifiers such as in exemplar SVMs [2] or Action Bank [3], the major limitation of the technique is that in the original formulation it is only designed to recommend a *single* model or classifier from the library. This naturally raises the question of how to extend the method to jointly recommend sets or ensembles of classifiers, and in this paper we evaluate several alternatives for doing so. Our key contribution in this paper is to demonstrate that the model recommendation framework can be extended to recommending sets of models, and that this ensemble recommendation outperforms both an AdaBoost baseline and a direct training baseline.

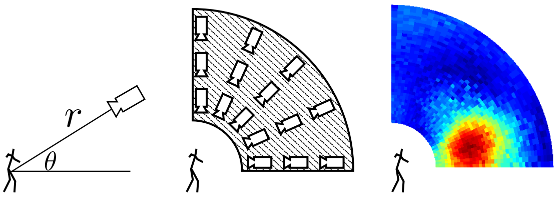


Fig. 1: We adopt the same illustrative example as [1] for visualization purposes. A library of classifiers produced by training SVMs to detect walking, where each SVM is trained only on samples from a narrow viewpoint, defined by its elevation θ and mean to subject r (left). The accuracies of all 1600 classifiers on a training set can be visualized as a heat map (right).

To help visualize model recommendation and our extensions to it, throughout the paper we use the illustrative example from [1], which is a scenario where the problem is to select a set of classifiers to detect the action “walking” from a fixed camera with an unknown vantage point. Since the camera is fixed, it can be described in terms of its mean distance r to the subject and its angle from the horizon θ (see Fig. 1). A library of 1600 “walking” classifiers is arranged in a 40×40 polar grid of possible viewing locations, and each is trained from synthetic data (rendered motion capture videos of people walking) generated for its assigned viewing location. Then, given a small training set of data from a new camera at an *unknown* viewing point, the goal is then to select a *set* of classifiers from the library which when combined gives the best classification accuracy.

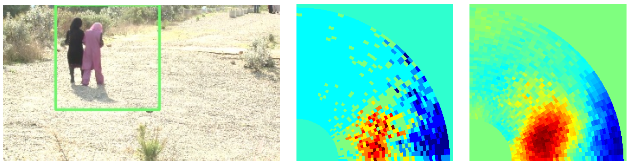
2 Related Work

This work combines our model recommendation technique [1] and boosting, and is loosely related to multi-task learning. In particular, common multi-task learning techniques attempt to enforce sparsity in the selection of features [4, 5], or support vectors [6], or kernels [7], across multiple tasks in order to enforce the sharing of some kind of information across tasks. Unlike these sparsity based approaches, there is no explicit forced sharing of features in model recommendation; indeed, it is possible to recommend a model for a target task that is not shared with any other task. This distinction is important because multi-task learning is known to fail when the tasks jointly learned are insufficiently related to one another, and hence an open area of research in multi-task learning is how to select which tasks to learn together [8].

Given the popularity of boosting, it has unsurprisingly been applied to multi-task learning as well. These methods tend to follow the standard multi-task approach of enforcing sparsity, such as in Chapelle *et al.* [9] where boosting selects a common set of weights for weak learners across all tasks, and then individual tasks are allowed to sparsely deviate from that common weighting. Wang *et al.* [10] take a slightly different approach, where the sparsity is enforced by learning a partitioning (clustering) of the tasks, where all the tasks in a cluster are forced to share the same weights for the weak learners. Faddoul *et al.* [11] take yet another approach, in which the weak learners are joint classifiers of two tasks, and so the boosting naturally selects a compromise between two tasks. However, the limitation of their approach is that it does not easily scale to more than two tasks, and these techniques generally need closely related tasks.

3 Method

Since model recommendation is a new and unconventional technique, we start with a brief functional overview of what it requires as input and what it accomplishes. Then, we briefly review the standard AdaBoost algorithm, and we describe how it can be modified to incorporate model recommendation.



(a) Camera view (b) Raw accuracies (c) Predicted

Fig. 2: An illustration of the model recommendation technique [1]. A set of 1600 viewpoint-tuned classifiers are evaluated (rated) on a dataset of 12 labeled samples from a particular camera viewpoint (Fig. 2(a)). Because there are only 12 training samples, there is quantization noise in measuring the accuracy of the classifiers (Fig. 2(b)), since the accuracies can only vary in increments of $\frac{1}{12}$. Model recommendation is able to take these noisy estimates of the accuracies and predict improved estimates (Fig. 2(c)). We adopt this display convention to aid comparison to [1].

3.1 Recommendation in brief

Model recommendation is a method for selecting an item from a library, where the goal is to return an item ("model") which is likely to be *rated* highly for a specific task. More formally, suppose there is some problem-specific rating function $r(c_j, d_k)$ which returns a numerical score rating item c_j on task d_k; if the items in the library are classifiers and the tasks are labeled sets of samples, then the rating function can be the accuracy of a given classifier on a dataset.

As a notational convenience we denote $r(c_j, d_k) = r_{jk}$. In order to make recommendations, a ratings matrix R (or "ratings store") is needed; the entries of this matrix are the ratings of items on different tasks, where $R_{j,k} = r_{jk}$, which is the rating of item j on task k.

Model recommendation takes a subset of ratings of items on a task, and uses that subset of rated items along with the ratings store to predict the ratings of other items. Functionally, supposing that A is a vector of item ratings for a target task, then model recommendation returns a prediction of a given classifier j's rating on that task $r'_j = \text{RecommendPredictAccuracy}(c_j, R, A)$. We perform the prediction using the factorization method presented in [1].

Note that this "subset" of probe ratings might very well be the entire set of classifiers. For example, in Fig. 2, all 1600 classifiers are used as probes on a small training set of 12 samples, producing a very noisy heat map of accuracies. When this full set of accuracies is fed into model recommendation, the returned predicted accuracies are a de-noised version of the input.

3.2 Ensemble recommendation methods

Here we present four options for selecting classifier ensembles. The simplest, top-k recommendation, just selects the top-k classifiers according to their predicted accuracies from model recommendation. AdaBoost is an unmodified, standard boosting algorithm. Recommendation boosting takes AdaBoost, but uses model recommendation to select the classifier at each iteration. Recommendation boosting+ uses the same underlying mechanism as recommendation boosting, but afterwards combines its selection with the top-k selection to add more variation to the selected set. Note that we use the boosting methods as feature selection mechanisms and discard the final weights of the selected classifiers in favor of simply training an SVM on the selection; this use of AdaBoost is common in vision [12, 13] and occasionally sees use in other domains [14, 15].

Top *k* recommendation Given the predicted ratings r' according to recommendation, rather than selecting only the top one, we select the top k, where k is the size of the desired set to be recommended. This is the obvious way of recommending multiple classifiers, but the downside is that it can potentially recommend a highly redundant set.

AdaBoost Although the AdaBoost has been extended to more than two classes, for simplicity in this paper we consider only a binary classification problem. The algorithm learns a classifier from a training set, where X_i is the ith data sample in the training set of a target task, and $y_i \in \{-1, 1\}$ is the associated binary label for that sample, and where n is the number of training samples.

AdaBoost is an iterative algorithm, where each iteration considers a different weighted version of the training set; we denote the weight of data sample i in iteration t by w_{it}. Then, given a classifier f_j, the weighted error of that classifier at an iteration is given by $\text{WeightedErr}(f_j, W, X, y) = \frac{\sum_i I(f_j(X_i) \neq y_i) \cdot w_{it}}{\sum_i w_{it}}$, where $I(.)$ is an indicator function. The weighted accuracy of the classifier is simply $\text{WeightedAccuracy}(f_j, W, X, y) = 1 - \text{WeightedErr}(.)$.

At each iteration the classifier with the lowest weighted error is selected, and the weights modified to increase the weights of misclassified samples and decrease the weights of correctly classified ones. The algorithm is given in Alg. 1.

Recommendation Boosting We modify AdaBoost to incorporate recommendation using the the key insight that boosting algorithms can be seen as a series of tasks, and therefore model recommendation can be used to pick the weak learner at each iteration of the algorithm

Recommendation boosting simply replaces the selection of the classifier with the lowest weighted error with a model recommendation step. That is, instead of using the weighted errors of the classifiers to directly select the classifier for an iteration, the measured accuracies (along with a ratings matrix R of the accuracies of classifiers in the library evaluated on other action recognition tasks) are fed into model recommendation to predict the accuracies for the classifiers, and the classifier with the highest predicted accuracy selected. This modified algorithm is given in Alg. 2.

```
Input : Classifiers $F = \{f_1, f_2, \ldots, f_j, \ldots\}$
Input : Training-samples and labels X, y
Output: A selected set $S \subseteq F$
$\forall w_i \in W, w_i \leftarrow \frac{1}{n}$ ;
$S \leftarrow \{\}$ ;
for $t = 1, \ldots, k$ do
    $s_t \leftarrow \text{argmin}_{f_j}$ WeightedErr($f_j, W, X, y$) ;
    $e_t \leftarrow$ WeightedErr($s_t, W, X, y$) ;
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-e_t}{\max(e_t, \epsilon)}$ ;
    $\forall w_i \in W, w_i \leftarrow w_i \cdot e^{-\alpha_t \cdot \text{sign}(y_i \cdot s_t(X_i))}$ ;
    $W \leftarrow$ Normalize($W$) ;
    $S \leftarrow S \cup \{s_t\}$ ;
end
```

Algorithm 1: AdaBoost

Recommendation Boosting+ As explained in [16], eventually AdaBoost will converge to a 'limit cycle' in which the same weak learners are cyclically selected. If the number of training samples is small, this convergence can happen very quickly. As a result, in the quantitative experiments performed later, AdaBoost only selects a mean of 11 unique classifiers over 20 iterations, while recommendation boosting only selects 10 unique classifiers over those same 20 iterations.

In recommendation boosting+ we add variety to the set of classifiers selected by recommendation boosting– if recommendation boosting only selects b unique classifiers, but a set of k is desired, then the remaining $k-b$ classifiers are selected as the $k-b$ classifiers with the highest predicted accuracies according to model recommendation.

4 Evaluation

We evaluate our alternatives on action recognition in two ways. First, we consider a simplified qualitative example (Fig. 1) using synthetic data (rendered videos of motion capture data using the same viewing angle setup as in [1], with histogram of optical flow descriptors).

For quantitative results we consider an action recognition problem on the UCF50 dataset [17] using limited training data (10.2 training samples on average, compared to the approximately 100 per action that are available when UCF is evaluated as a single task). We use STIP [18] plus HOG3D [19] bag-of-words histograms as our low-level representation; this is commonly used as the foundation for action recognition systems and often performs similarly to more complex approaches [20]. The UCF50 dataset is a difficult dataset of videos of various actions harvested from YouTube; for this setup, we limit the amount of training data by splitting the dataset into a number of tasks. UCF50 contains approximately 5500 videos, divided into 50 actions, with each action further subdivided into groups of videos, where each group comprises a set of related videos. We

```
Input  : Classifiers $F = \{f_1, f_2, \ldots, f_j, \ldots\}$
Input  : Training-samples and labels X, y
Input  : Ratings matrix $R$ of classifier accuracies on other tasks
Output: A selected set $S \subseteq F$
$\forall w_i \in W, w_i \leftarrow \frac{1}{n}$ ;
$S \leftarrow \{\}$ ;
for $t = 1, \ldots, k$ do
    $A = [\texttt{WeightedAccuracy(}f_j, W, X, y\texttt{)}$ for $f_j \in F]$ ;
    $s_t \leftarrow \text{argmax}_{f_j}$ $\texttt{RecommendPredictAccuracy(}f_j, R, A\texttt{)}$ ;
    $e_t \leftarrow \texttt{WeightedErr(}s_t, W, X, y\texttt{)}$ ;
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-e_t}{\max(e_t, \epsilon)}$ ;
    $\forall w_i \in W, w_i \leftarrow w_i \cdot e^{-\alpha_t \cdot \text{sign}(y_i \cdot s_t(X_i))}$ ;
    $W \leftarrow \texttt{Normalize(}W\texttt{)}$ ;
    $S \leftarrow S \cup \{s_t\}$ ;
end
```

Algorithm 2: Recommendation Boosting

produce each task by merging 1-3 groups of the same action, dedicating 2/3 of the video groups to training data, and the remaining 1/3 to testing data. We augment each group with an equal number of negative samples drawn at random from the other actions, so that each task is then a one vs. all binary classification problem with an equal number of positive and negative samples (so that chance is 50% accuracy). The mean number of training samples per task is 10.2, and the mean number of testing samples 24.3. Each group may be used in multiple training or test tasks (but there is no overlap between test and training).

We use the training groups to generate a library of 1000 classifiers trained on different groups, and also to generate a ratings store of those 1000 classifiers rated on 1000 tasks. Thus, the ratings store has size 1000×1000. Ideally, different data would be used to train the classifier library and build the ratings store, as rating a classifier against the same group from which it was trained results in an overly optimistic accuracy (rating) and possibly distorts the computed factorization. For testing, we use a set of 250 tasks with no videos or video groups in common with the set used to generate the classifiers and store. This is the same general setup as in [1], but randomized train/test partitions mean that results cannot be exactly compared. However, model recommendation [1] can be compared to our proposed ensemble recommendation by noting that ensemble recommendation degenerates to model recommendation if a set of size one is to be recommended.

5 Results

Qualitative results for the viewing angle situation can be seen in Fig. 3. Note how top-k recommendation chooses a very redundant set of classifiers, where all five classifiers are tightly clustered around the predicted maximum. AdaBoost, on the other hand, is confused by the spurious classifiers that appear to have 100%

accuracy on the training set. Recommendation boosting picks a few classifiers near the predicted maximum, but then spreads the remainder out for better coverage of the region near the maximum. Thus the recommendation boosted set is more likely to be robust to variations in the action.

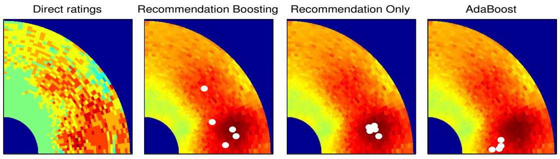


Fig. 3: A comparison of the selected classifiers according to top-k recommendation, AdaBoost, and recommendation boosting. Top-k recommendation selects a highly redundant set of classifiers, while AdaBoost is led astray by a few erroneously good classifiers. Recommendation boosting selects a nice spread of classifiers.

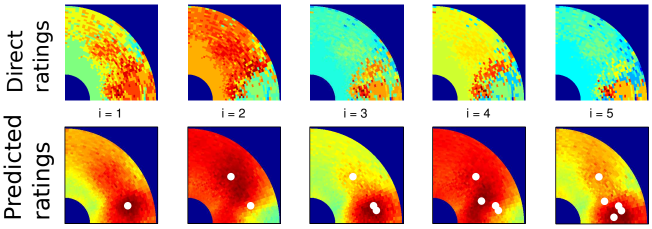


Fig. 4: Progression of the classifiers selected by recommendation boosting; note how each iteration's reweighting of the training samples shifts the distribution of predicted accuracies so that the selected classifiers do not all clump near one location, as in top-k recommendation (see Fig. 3).

A visualization of how recommendation boosting selects its classifier each iteration can be seen in Fig. 4. At the first iteration, the selected classifier is the same as the top recommended classifier, but then in subsequent iterations, as misclassified samples are more strongly weighted, the distribution of classifier ratings changes to promote the selection of classifiers other than those at the original maximum. At each iteration the method is able to smooth over the extremely noisy measured accuracies for the iteration to produce a better estimate of where the maximum accuracy is obtained for that iteration.

Quantitative results on the UCF50 dataset are shown in Fig. 5, where it can be seen that recommendation boosting outperforms both AdaBoost and direct training. Interestingly, the straightforward top-k recommendation method does better than basic recommendation boosting, suggesting that for this evaluation

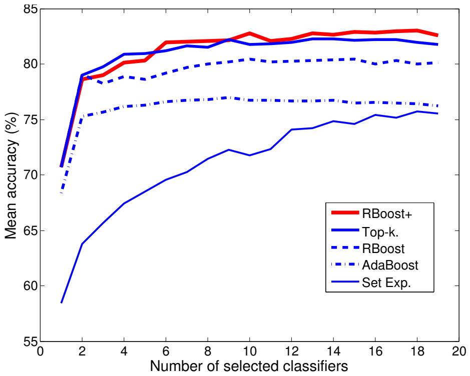


Fig. 5: Mean accuracy of the selected set of classifiers vs. the size of the selected set. Top-k recommendation and recommendation boosting+ have the best performance, with recommendation boosting+ having a slight edge at larger set sizes. The accuracy obtained by directly training each task on the low-level STIP+HOG inputs is 77%.

domain, redundancy in the classifier library is not as large a concern as expected. It is likely that the 1000 classifier library samples the space of possible classifiers so sparsely that there is too little redundancy in the classifiers to be detrimental.

For this experiment, the direct training baseline (in each task, directly train an SVM on the input STIP+HOG bag-of-words histograms, rather than appealing to the library) obtains an accuracy of 77%, which only AdaBoost fails to exceed. Recommendation boosting+ exhibits a slight improvement over top-k selection (this difference is statistically significant to $p < 0.05$ for set sizes ≥ 18).

All of the selection strategies show large gains at first and then quickly plateau, indicating that they front-load their selections with the strongest classifiers.

6 Conclusions

In this paper we have evaluated several alternatives for extending model recommendation to recommending sets. Out of these alternatives, recommendation boosting+ and top-k recommendation methods are the clear victors. Between these two alternatives, however, the distinction is less clear. In terms of quantitative performance, the two are very similar, with recommendation boosting+ having a narrow edge at larger recommended set sizes. Furthermore, qualitative results suggest that the top-k selection strategy should be prone to selecting redundant classifiers if the classifier library contains them.

Acknowledgments

This work was partially funded by the Army Research Laboratory under Cooperative Agreement #W911NF-10-2-0061. The views and conclusions are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation herein.

References

1. Matikainen, P., Sukthankar, R., Hebert, M.: Model recommendation for action recognition. In: CVPR. (2012)
2. Malisiewicz, T., Shrivastava, A., Gupta, A., Efros, A.: Exemplar-svms for visual object detection, label transfer and image retrieval. In: ICML. (2012)
3. Sadanand, S., Corso, J.: Action bank: A high-level representation of activity in video. In: CVPR. (2012)
4. Obozinski, G., Taskar, B.: Multi-task feature selection. In: ICML Workshop on Structural Knowledge Transfer for Machine Learning. (2006)
5. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: NIPS. (2007)
6. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. JMLR **6** (2005) 615–637
7. Rückert, U., Kramer, S.: Kernel-based inductive transfer. In: ECML. (2008)
8. Kang, Z., Grauman, K., Sha, F.: Learning with whom to share in multi-task feature learning. In: ICML. (2011)
9. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Multi-task learning for boosting with application to web search ranking. In: ACM SIGKDD. (2010)
10. Wang, X., Zhang, C., Zhang, Z.: Boosted multi-task learning for face verification with applications to web image and video search. In: CVPR. (2009)
11. Faddoul, J., Chidlovskii, B., Torre, F., Gilleron, R.: Boosting multi-task weak learners with applications to textual and social data. In: Machine Learning and Applications. (2010)
12. Shen, L., Bai, L.: Adaboost gabor feature selection for classification. In: Proceedings of the Image and Vision Computing Conference, New Zealand. (2004)
13. Zhou, M., Wei, H.: Face verification using gabor wavelets and adaboost. In: ICPR. (2006)
14. Zhou, X., Zhuang, X., Liu, M., Tang, H., Hasegawa-Johnson, M., Huang, T.: Hmm-based acoustic event detection with adaboost feature selection. Multimodal Technologies for Perception of Humans **4625** (2008) 345–353
15. Morra, J., Zhuowen, T., Apostolova, L., Green, A., Toga, A., Thompson, P.: Comparison of adaboost and support vector machines for detecting alzheimer's disease through automated hippocampal segmentation. Medical Imaging, IEEE Transactions on **29** (2010) 30 –43
16. C. Rudin, I. Daubechies, R.S.: The dynamics of adaboost: Cyclic behavior and convergence of margins. JMLR **5** (2004)
17. University of Central Florida: UCF50 action recognition dataset. `http://server.cs.ucf.edu/~vision/data.html\#UCF50` (2011)
18. Laptev, I., Lindeberg, T.: Space-time interest points. In: ICCV. (2003)
19. Kläser, A., Marszałek, M., Schmid, C.: A spatio-temporal descriptor based on 3d-gradients. In: British Machine Vision Conference. (2008) 995–1004
20. Kovashka, A., Grauman, K.: Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In: CVPR. (2010)