Transferring Knowledge by Prior Feature Sampling

Victor Eruhimov, Vladimir Martyanov, Aleksey Polovinkin

Advanced Analytics, LTDA, Intel Corporation
30 Turgenev st., Nizhny Novgorod, 603950, Russia

Abstract. The paper presents a novel method for transfer learning through prior variable sampling. A set of problems defined in the same feature space with similar dependencies of target on features is considered. We suggest a method for learning a decision tree ensemble on each of the problems by prior estimation of variable importance on other problems in the set and using it for regularizing model learning for a small amount of training samples. The method is tested on several simulated and real datasets. In particular, we apply our method for a set of time series classification (TSC) problems. Our analysis demonstrates an intriguing result: a model trained on several TSC problems can learn a new problem with high accuracy from a low number of samples.

1 Introduction

The classical task of discriminative learning is to fit a model to a given data called a training set. The validity of the fitting method is checked on a test set that is assumed to be i.i.d. drawn from the same distribution as the training set. A critical parameter that influences the model predictive power is the size of the training set. Learning from too few samples can lead to poor generalization of the model coming from either under- or over-fitting. This is also related to the number of features used for prediction – the more predictors we have the more samples we need to learn the dependence. A rule of thumb says that the number of samples should be 5 times larger than the number of features. In many practical problems the predictive power of a model is limited by the training set size. Problems like face recognition or gene expression analysis generate datasets with the number of features much larger than the number of samples. A way to overcome the limitation of the small training set is to introduce prior knowledge transferred from similar problems. Some of pioneering works in the field of transfer learning are [1–3]. [1, 4] suggest a transfer learning method based on learning important features from different classification tasks and using feature importance information (w.r.t. prediction of the target) to fit a model for a new task. [5] shows how co-occurence of words calculated on several text classification problems can be used as a prior for solving a new problem.

This paper proposes a novel approach to transfer learning. We show a simple modification to the Gradient Boosting Trees learning algorithm [6, 7] that

significantly improves learning efficiency by using information about the importance of features with regard to target prediction. This method can be applied to learn a set of problems that are defined in the same feature space and where the importances of each feature (to be defined in the next section) are similar for all problems. We use several artificial and real datasets from different domains to show that the approach is universal and can be applied to a very wide range of problems. The next section contains an overview of Gradient Boosting Trees and feature importance, Section 3 describes Prior Feature Sampling algorithm and Section 4 discusses its application to transfer learning.

2 Feature Importance

We combine transfer learning with one of the best off-the-shelf classification methods – Gradient Boosting Trees (GBT). It has all the properties of a universal learner: fast, works with mixed-type data, elegantly handles missing data, invariant to monotone transformations of the input variables (and therefore resistant to outliers in input space), has been proven to be among the most accurate and versatile state-of-the-art learning machines. GBT is a serial ensemble of decision trees [8] where every new tree constructed relies on previously built trees. At every iteration c of GBT a new tree T_c is fitted to the generalized residuals with respect to a loss function, where the size of the ensemble is chosen to avoid overfitting (usually by monitoring validation errors.)

GBT provides (as a byproduct) a reliable estimate of the variable importance. The importance measure from a single tree can be defined as [8]:

$$VI(i,T) = \sum_{t \in T} \Delta I(x_i, t) \tag{1}$$

where $\Delta I(x_i,t) = I(t) - p_L I(t_L) - p_R I(t_R)$ is the decrease in impurity due to an actual (or potential) split on variable x_i at a node t. Here p_L and p_R are probabilities of a sample that appeared in the node t to fall into the left t_L and right t_R nodes correspondingly. We use Gini index [8] as the impurity function for categorical target and the square of the target standard deviation for numeric target.

Variable importance for GBT is defined as the variable importance (1) averaged over all trees in the ensemble [6]:

$$VI(i) = \frac{1}{C} \sum_{c=0}^{C} VI(i, T_c) \tag{2}$$

GBT builds shallow trees using all variables (on a subsample of the training data), and hence, it can handle large datasets with a moderate number of inputs. Very high dimensional data is extremely challenging for GBT. Apart from computational complexity problems (time complexity of the algorithm for GBT learning is $O(MNlog(N))$ where N is the number of samples and M –

the number of features) the model tends to overfit on data with few samples and many irrelevant variables. The primary reason for that is the greedy algorithm for learning a decision tree that can choose an irrelevant variable for a split because it provides larger impurity reduction by chance. [9] addresses both issues by introducing a "Dynamic Feature Selection" (DFS) method for learning a GBT ensemble. The cornerstone idea of the method is to sample a subset of variables for each split and choose the best one among this subset instead of the whole set. Sampling weights for each variable are different and are updated iteratively after each new tree is added to the ensemble. So a variable that provided large impurity reduction for the already learned trees has a higher probability to be selected for splits in the next tree. However this method still requires a considerable amount of samples as it is based on reliable estimation of feature importance. The next section presents a very simple and effective modification to a GBT learner that can generalize from very few samples given prior feature importance.

3 Learning GBT with Prior Feature Sampling

Let us assume that we know feature importance $VI(i)$ for each variable x_i, i.e. we know that some variables participate in splits more often with higher impurity reduction values. However impurity reduction that is used as a primary measure for a split has high variance for a small training set size. As a result even a variable that is not relevant to the target can be selected for a split. In order to regularize the search for a split we introduce variable pre-sampling with probabilities proportional to variable importance. Let $F = \{x_i\}|_{i=1..M}$ be the set of all variables, $\Delta I(i,t)$ be the maximal impurity reduction obtained by a split on variable x_i in the node t, $VI(i)$ – variable importance. We sample a subset F_L of L variables from F with probabilities $p_i \propto VI(i)$. The classical GBT algorithm splits on variable $i^* = \arg\max_{x_i \in F} \Delta I(i,t)$. We suggest searching for the optimal split in a much smaller feature subspace F_L: $i^* = \arg\max_{x_i \in F_L} \Delta I(i,t)$. The number of features L should be much smaller than the total number M, in our experiments we use $L = \lfloor \sqrt{M} \rfloor$ unless stated otherwise. We call this algorithm Prior Feature Sampling (PFS). Note that PFS is a method for training a base learner so it is compatible with other ensemble learning methods such as Adaboost [10] and Random Forests [11]. Various theoretical results obtained for ensembles of generic base learners (e.g. [12]) or tree-based learners ([11]) about generalization error bounds hold for ensembles of trees learned with PFS.

We will illustrate the method on a simple artificial dataset called *Signum*. Let the feature space $F = F_1 \otimes F_2 \otimes ... \otimes F_G$ be the Cartesian product of G subspaces of the same dimensionality S so that the total number of features $M = SG$. Each variable x_i, $i = 1..M$ is sampled from a uniform distribution in the region $[-1, 1]$. Let $c_i = \beta^{g(i)}$ be the importance of each group where $g(i)$ is

the index of the group x_i belongs to. Target variable

$$y = \gamma \sum_{i=1..G} c_i sign \left(\sum_{\{j|g(j)=i\}} sign(x_j) \right), \tag{3}$$

where $sign(x)$ is the sign function

$$sign(x) = \begin{cases} 1, x > 0 \\ -1, x \leq 0 \end{cases}. \tag{4}$$

Note that importances are decreasing exponentially so that for small β most of target variation is provided by the first few groups.

We generate a dataset of $M = 30$ features consisting of $G = 10$ groups, $\beta = 0.5$. We learn a GBT model using PFS algorithm with variable importance $VI(i) \propto c_i^2$ (since impurity is defined as the square of standard deviation of target). The number of trees in GBT $C = 200$, and the regularization parameter $\nu = 0.1$ are constant throughout the paper. Figure 1 shows the dependence of test error of classical GBT, GBT-DFS and GBS-PFS on the training set size. The number of samples in the test set is fixed and equal to 10000. GBT-PFS is superior to other methods for any number of training samples but the difference is more significant for smaller number of samples when the variance of impurity reduction is high and prior information is of more value to the learning engine. This is also demonstrated by Figure 2 that compares test error for PFS with $L = 1$ and $L = \lfloor \sqrt{M} \rfloor$. The model with $L = 1$ that we will call PFS_1 samples only one variable for each split so that the choice of the variable to split on is entirely based on prior knowledge and does not depend on the training data. PFS_1 shows lower test error than PFS on a smaller number of training samples when high variance of impurity reduction prevents PFS splitting on right variables. It is no surprise that PFS wins for larger training set size when estimation of impurity reduction is closer to the expected value.

In order to illustrate how PFS learns a better model than GBT we plot train and test errors versus the number of trees in the ensemble in Figure 3. To make the results more interpretable we learn trees of depth equal to 1 – so-called *stumps*. One can see that while GBT is able to optimize its training error better than PFS, its generalization accuracy characterized by test error is weaker. GBT selects variables to split on corresponding to maximum impurity reduction on the *training set* but since there are few training samples these splits do not provide low test error. Figure 4 shows the number of variables outside the first two groups (i.e. two groups of the most important varaibles that explain 75% of target variation) selected for splitting normalized on the total number of splits (that is equal to the ensemble size). One can see that GBT selects many more unimportant variables than PFS.

In this paper we leave open the question of how good is our sampling strategy compared to others (such as, for example, taking into account the number of splits on each variable rather than impurity reduction). However we show below

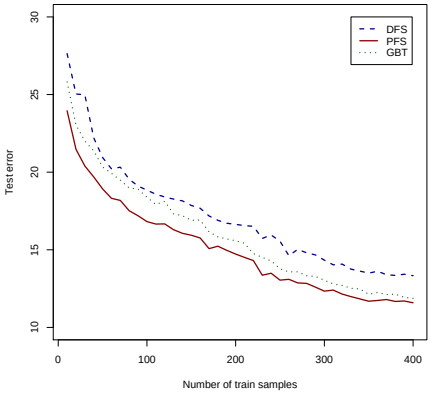


Fig. 1: Test error vs. training set size.

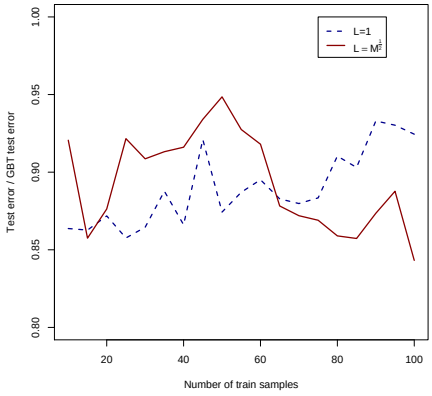


Fig. 2: PFS test error normalized on GBT test error vs. training set size.

that importance-based sampling provides a significant improvement over GBT on a wide range of machine learning problems.

Practical machine learning problems rarely assume known variable importance. The next section considers transferring variable importance from other problems.

4 Prior Feature Sampling for Transfer Learning

Let us assume that we have a set of problems $D = \{D_k\}|_{k=1..K}$ defined on the same feature space F. Furthermore we will assume that problems in D have similar variable importance. However all of the problems have small training sets so that an estimation of variable importance on one dataset has high variance. Suppose that we want to learn a model of $D_{\hat{k}} \in D$. We start by learning a GBT model for each of the datasets $D_k \in D \backslash D_{\hat{k}}$ and calculating variable importance $VI^{(k)}(i)$. Then we average importance over datasets $VI_p^{(\hat{k})}(i) = \frac{1}{K-1} \sum_{k \neq \hat{k}} VI^{(k)}(i)$ and use $VI_p^{(\hat{k})}(i)$ as feature importance for the PFS method. Note that because of our leave-one-out strategy sampling weights do not depend on $D_{\hat{k}}$. If variable importances for D_k are different and in order to benefit from more samples we might want to take into account variable importance from $D_{\hat{k}}$.

Below we demonstrate the results on several artificial and real datasets. In each experiment we learn DFS and PFS models 10 times on each dataset to account for model variance coming from random variable selection for each split. We also learn 10 GBT models for each experiment. We compare test errors

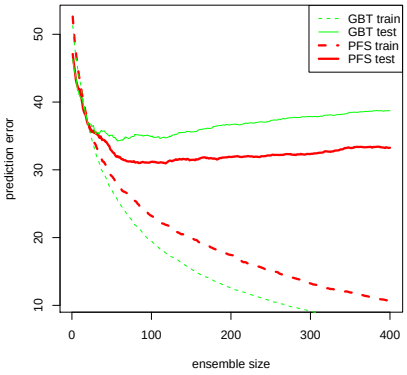


Fig. 3: GBT and PFS stumps training and test error versus ensemble size.

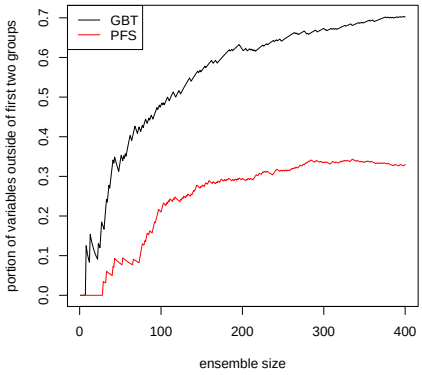


Fig. 4: The number of variables outside the first two groups selected for splits normalized on ensemble size.

distribution of different learning algorithms for each dataset using two one-sided t-tests with p-values 0.05.

4.1 Signum

The first example is a class of datasets obtained by Signum generator described in the previous section. We generate 10 datasets with the same β equal to 0.5 and different γ sampled from a uniform distribution in $[0, 1]$. The number of samples in each training dataset is equal to $N = 100$. As responses in different datasets have different variation we calculate the ratio of PFS test error to GBT test error. The value of this ratio averaged over all 10 datasets is equal to 0.87. t-tests show that PFS is superior to GBT for 8 datasets and the difference between methods on the remaining two datasets is not statistically significant.

4.2 Linear regression

This section considers experimental results on the class of datasets with linear dependence of response on predictors. Let $x = (x_1, \ldots, x_m, x_{m+1}, \ldots, x_M)$ be a numeric vector of variables, uniformly distributed in $[0, 1]$, where $x_1, \ldots, x_m$ are features that influence the target, $x_{m+1}, \ldots, x_M$ are "noise" features. The target variable y is a linear function of $x_1, \ldots, x_m$: $y = \sum_{i=1}^{m} c_i x_i$. Coefficients $c_1, \ldots, c_m$ are drawn from the uniform distribution in $[0, 1]$ for each dataset independently. We generate 10 datasets with the number of training samples $N = 100$, number of test samples $N_{test} = 100$, $m = 4$, $M = 104$ (so the number of "noise" variables is equal to 100). As a result we get datasets where target depends on the same

4 variables but the dependences are different for different datasets. Also each dataset has 100 variables irrelevant to the target. As in the previous section we run PFS using leave-one-out scheme, calculating variable importance on 9 datasets and using it for learning a model on the remaining one. A success of a learning engine on this dataset depends strongly on the ability to filter out irrelevant variables. This is not the strongest side of GBT so we use a classical filter based on Pearson test (p-value 0.05) for each predictor variable versus target. Baseline GBT model is learned on variables selected by the test. The results are shown in Figure 5. When the training set size is sufficiently small PFS does a better job of filtering irrelevant variables than Pearson test as it takes advantage of more samples.

Let us note that there are methods such as [13] that successfully solve exactly the same problem. Our goal here is to show that PFS, being a general method that does not make any strong assumptions about the dependence of response on predictors, can filter out a large set of irrelevant variables and learn an accurate model.

We have repeated the experiment 3 times and for only one dataset and only one model instance (once in 300 cases – we have 10 datasets and we learn 10 models for each one) we got a PFS model with higher test error than DFS model (the rest of the data shows superiority of PFS).

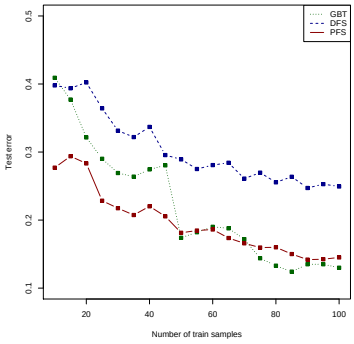


Fig. 5: Linear Regression problem: the dependence of test error on the number of samples.

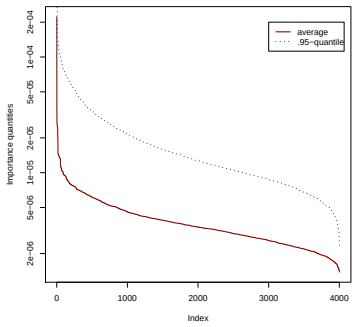
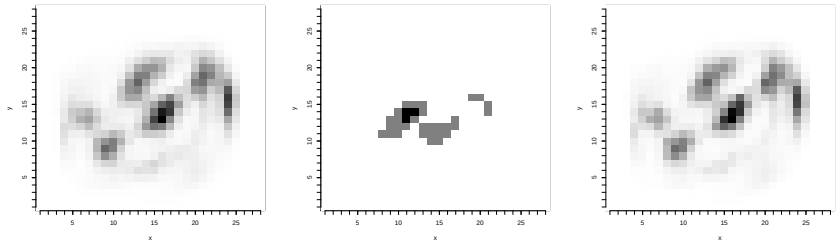


Fig. 6: The distribution of variable importance for UCR datasets.

4.3 Handwritten digit recognition

A large class of machine learning problems comes from computer vision. We illustrate PFS approach on the task of handwritten digit recognition with MNIST dataset [14]. We learned GBT models on even digits (classifying each versus each) represented by intensity values in each pixel and calculated feature importance

for each feature (pixel). Images of feature importance (resolution 28x28 pixels) are presented in Figure 7. Then GBT and PFS models were learned on odd digits, each versus each, with 10 samples per task (5 samples per class) chosen randomly from the training dataset. Each model was tested on 10000 samples sampled from the test dataset. Table 1 summarizes experimental results for GBT and PFS methods. Each model was trained 10 times independently. The table shows average test errors, bold indicates statistically significant difference in test error distributions (detected with one-sided t-tests, p-value equal to 0.05). One can see that PFS is able to learn more accurate models compared to GBT for all tasks except for one. The most of the PFS models have low test errors (less than 0.2) in spite of low number of training samples.



(a) (b) (c)

Fig. 7: Variable importance for MNIST dataset: (a) average, (b) 5% quantile, (c) 95% quantile. Black corresponds to the maximum importance.

Dataset	GBT	PFS
1 vs 3	0.23	**0.15**
1 vs 5	0.26	**0.11**
1 vs 7	0.13	**0.08**
1 vs 9	**0.20**	0.29
3 vs 5	0.39	**0.34**
3 vs 7	0.28	**0.09**
3 vs 9	0.37	**0.26**
5 vs 7	0.19	**0.14**
5 vs 9	0.23	**0.16**
7 vs 9	0.36	**0.28**

Table 1: Average test errors on MNIST dataset.

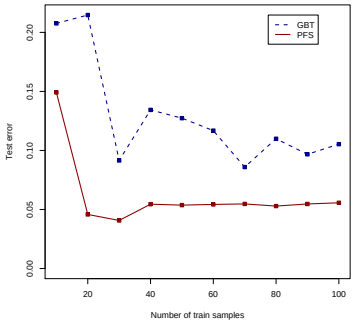


Fig. 8: The dependence of test error on the number of training samples for MNIST problem 1 vs 3.

4.4 Time series classification

Time Series Classification (TSC) is a field recently bursting with new results and applications. Variables used for prediction in a TSC problem are ordered and represent a process in time. An extensive overview of time series classification methods is given in [15]. One of the frequently used approaches to a TSC problem is to extract salient features from each signal and transform TSC task into a regular machine learning problem. Many feature sets have been evaluated including SVD (Singular Value Decomposition) features, DFT (Discrete Fourier Transform), coefficients of the decomposition into Chebyshev Polynomials, DWT (Discrete Wavelet Transform), PLA (Piecewise Linear Approximation), ARMA (AutoRegression Moving Average) coefficients, various symbolic representations [16, 17]. However none of these feature sets is universal enough to give competitive results on a wide variety of problems. A study done in [18] showed that a learning engine can benefit from multiple feature sets, but at the same time learning on too many features causes overfitting. Following this work we construct our feature set by combining several different types of features: Wavelets, Chebyshev polynomials, Raw features, first 5 statistical moments.

Surprising as it may seem, one of the best state-of-the-art methods for TSC is one-nearest-neighbor (1NN) with measure between two series called Dynamic Time Warping (DTW) [19]. One of the important reasons for strong performance of 1NN method on TSC problems is invariance to warping of time axis that is provided by DTW measure. In order to have our method generalize well on datasets with significant time warping we add warp-invariant features to our feature set. A detailed description of our feature set is given in [18].

We have tested PFS on the UCR corpus [20] of TSC problems. Each of the datasets has a different number of features due to different time series length so for each dataset with time series length Q that we learn we scale time series from all other datasets to Q by piecewise-linear interpolation in order to match feature spaces.

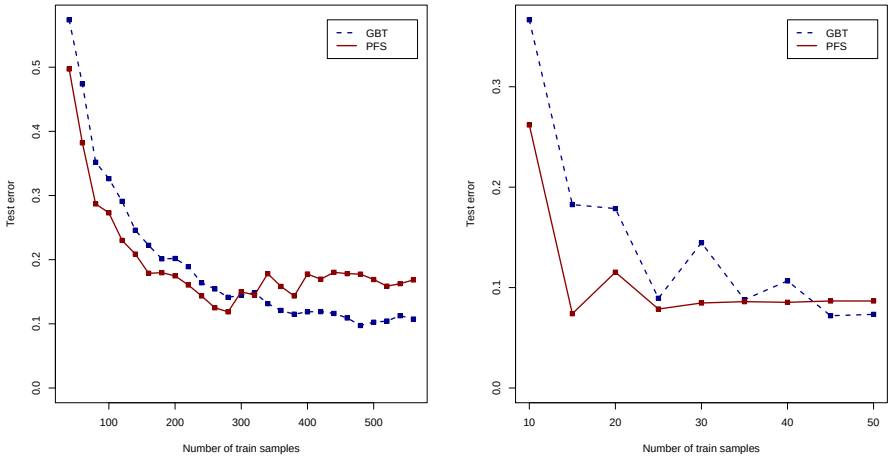
Although there is no explicit indication that different TSC problems have similar feature importance there are several clues pointing to this conclusion. To start with, most of the signals in UCR TSC problems are smooth and classes are defined by smooth features. This means that most of the time a class could be rather recognized by large time-scale features than by small time-scale. However this is not true for all datasets so we cannot filter out small-scale features. Another argument was pointed out in [18] where, although all feature types were important and removal of one of them caused increase of test errors on at least one dataset, the removal of wavelet features caused larger increase of test errors on more datasets compared to other feature types. Figure 6 shows variable importance averaged over all UCR datasets and 95% quantile versus the feature index, the features being sorted by the decreasing average variable importance. These results are taken for time series length $Q = 5051$ of Lighting2 dataset. Note that both curves decrease very fast so there are around 100 variables with average importance greater than 10^{-5}. However after this point both curves decrease slowly and 95% quantile is always much larger than the average

value. This means that there is a group of few variables that are consistently important for all datasets and another group of variables (much larger than the first one) that are important on few datasets and thus cannot be filtered for all TSC problems. Also many datasets in UCR corpus have less than 50 samples per class with time series length more than 100. This is another argument for applying PFS to UCR corpus, recalling that PFS strong sides show themselves on datasets with a small number of samples and a large number of features.

Dataset	Number of train samples	Number of features	GBT	PFS	DTW+1NN (uses whole train set)
Beef	30	6023	0.1233	0.1233	0.467
CBF	30	1113	0.0287	**0.0012**	0.004
Coffee	28	2462	**0**	0.1679	0.179
ECG200	50	740	**0**	0.151	0.12
FaceAll	50	2453	0.5303	**0.5028**	0.192
FaceFour	24	4420	0.0761	**0.0455**	0.114
Fish	50	5981	0.3411	**0.256**	0.160
GunPoint	50	1286	0.087	0.087	0.087
Lighting2	50	5051	0.1934	**0.1508**	0.131
Lighting7	50	5117	0.34	**0.28**	0.288
OliveOil	30	8080	0.1967	0.1867	0.167
OSULeaf	50	5765	0.5221	**0.4505**	0.384
SwedishLeaf	50	1667	0.4755	**0.4118**	0.157
SyntheticControl	50	875	0.061	**0.0407**	0.017
Trace	50	4045	0	0	0.01
TwoPatterns	50	1390	0	0	0.0015
Wafer	50	1292	0.0554	**0.03902**	0.005
Yoga	50	2882	0.4354	**0.333**	0.155

Table 2: Average test errors on UCR datasets.

18 datasets with the number of classes less than 30 were selected. For each dataset we have learned 10 GBT and PFS models and compared the results as described in the beginning of Section 4. The sizes of training and test datasets are taken as in [20] unless stated otherwise. The results are summarized in Table 2. We have limited the training set size to 50 samples if it was larger. Bold values show cases where the difference between two methods is statistically significant. The last column shows the performance of 1NN with DTW measure, cited from [20]. It is important to note that 1NN model uses the whole training set. PFS has lower test error compared to GBT for 11 datasets and higher test error for 2 datasets. These two, **Coffee** and **ECG200** have a very different distribution of variable importance from the rest of TSC problems. GBT is able to find features that help building extremely accurate model for these particular tasks. However



(a) (b)

Fig. 9: The dependence of test error on the training set size for UCR datasets: (a) FaceAll, (b) GunPoint

for the most of the datasets PFS achieves dramatic improvement over GBT model. Figures 9 (a-b) show the dependence of test error on the training set size N for two datasets. Note that PFS is always superor to GBT when N is small and regularization in the form of prior variable importance plays a significant role. Each curve tends to a constant in the area where N is sufficiently large and a small change in the number of training samples does not affect test error much. However PFS test error converges to this constant faster than GBT, this is more clear in Figure 9 (b).

5 Conclusion

We presented a transfer learning method that is based on sampling a subset of variables for each tree split with probabilities proportional to the importance of variables with regard to target prediction. Importance distribution was calculated on other datasets with similar feature spaces and targets. The method demonstrated a significant improvement in generalization error on a wide variety of datasets including Time Series Classification UCR corpus. We came to a surprising conclusion: different TSC problems are like each other and the model, trained on several TSC problems, can learn a new one with high accuracy from extremely low number of samples (5-10 per class).

This investigation opens several directions for future work. The problem of the optimal sampling weights even for a simple class of problems is still open.

Information such as the number of splits and, maybe, distribution of impurity reduction over splits has to be taken into account. Prior knowledge about a distribution function for each variable can be used to improve generalization error. Finally, the algorithm can be generalized to handle variable interactions so that sampling weights depend on the split one level up the tree.

References

1. Thrun, S.: Is learning the n-th thing any easier than learning the first? In Touretzky, D.S., Mozer, M.C., Hasselmo, M.E., eds.: Advances in Neural Information Processing Systems. Volume 8., The MIT Press (1996) 640–646
2. Caruana, R.: Multitask learning. Machine Learning **28**(1) (1997) 41–75
3. Baxter, J.: A bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning **28** (2007) 739
4. Thrun, S., Mitchell, T.M.: Learning one more thing. In: IJCAI. (1995) 1217–1225
5. Raina, R., Ng, A.Y., Koller, D.: Constructing informative priors using transfer learning. (2006) 713 – 720
6. Friedman, J.: Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University (1999)
7. Friedman, J.: Stochastic gradient boosting. Technical report, Dept. of Statistics, Stanford University (1999)
8. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont, MA (1984)
9. Borisov, A., Eruhimov, V., Tuv, E.: Dynamic soft feature selection for tree-based ensembles. In Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., eds.: Feature Extraction, Foundations and Applications. Springer, New York (2006)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. (1995) 23–37
11. Breiman, L.: Random forests. Machine Learning **45**(1) (2001) 5–32
12. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: a new explanation for the effectiveness of voting methods. In: Proc. 14th International Conference on Machine Learning, Morgan Kaufmann (1997) 322–330
13. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Advances in Neural Information Processing Systems 19 (NIPS 06), Mit Press (2006)
14. LeCun, Y., Cortes, C.: The mnist dataset of handwritten digits
15. Keogh, E.: Data mining and machine learning in time series databases (2004)
16. Huang, Y., Yu, P.S.: Adaptive query processing for time-series data. In: In proceedings of the 5th Int'l Conference on Knowledge Discovery and Data Mining, San Diego, CA (Aug 15-18 1999) 282–286
17. Geurts, P.: Pattern extraction for time series classification. In: In proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, Freiburg, Germany (Sep 3-7 2001) 115–127
18. to be added after review. (2007)
19. Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: International Conference on Machine Learning. (2006)
20. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: The ucr time series classification/clustering homepage (2006)