ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

Domain Expert Approximation Through Oracle Learning

Joshua E. Menke[1] and Tony R. Martinez[2]

Brigham Young University - Computer Science Department
3361 TMCB - Provo, UT, USA

Abstract. In theory, improved generalization accuracy can be obtained by training separate learning models as "experts" over subparts of a given application's domain. For example, given an application with both clean and noisy data, one solution is to train a single classifier on a set of both clean and noisy data. More accurate results can be obtained by training separate expert classifiers, one for clean data and one for noisy data, and then using the appropriate classifier depedning on the environment. Unfortunately, it is usually difficult to distinguish between clean and noisy data outside of training. We present a novel approach using *oracle learning* to approximate the clean and noisy domain experts with one learning model. On a set of both noisy and clean optical character recognition data, using oracle learning to approximate domain experts resulted in a statistically significant improvement ($p < 0.0001$) over using a single classifier trained on mixed data.

1 Introduction

The main idea in *oracle learning* [1, 2] is that instead of training directly on a set of data, a learning model is trained to approximate a given *oracle's* behavior on a set of data. The oracle can be another learning model that has already been trained on the data, or it can be any given functional mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ where n is the number of inputs to both the mapping and the *oracle-trained model* (OTM), and m is the number of outputs from both. The main difference with oracle learning is that the OTM trains on a set of data whose targets have been labeled by the oracle instead of training on given targets. Having an oracle to label data also means that previously unlabeled data can be used to train the OTM since the OTM's goal is only to approximate its oracle, not to improve in accuracy over the it. The key to oracle learning's success is that it attempts to use data that fits the observed distribution of the given problem to accurately approximate the oracle on those sections of the input space that are most relevant in real-world situations. In [1] small *artificial neural networks* (ANNs) are trained to approximate larger ANNs instead of being trained directly on the data. In addition, the smaller ANNs are trained on previously unlabeled data since the larger ANNs can serve as *oracle ANNs* to label data that did not originally have labels. In the following, instead of approximating a single ANN, we use a set of *domain expert* ANNs as an oracle to train a single ANN on real data.

For a given application, higher generalization accuracy can be obtained by training separate learning models as "experts" over specific domains. For example, given an application where it is common to observe at least two varying levels of noise, clean and noisy, one solution is to train a single classifier on both clean and noisy data. It is possible to achieve better accuracy by training one classifier on only noisy data and one classifier on only clean data, and then choosing between them during classification depending on the environment. The clean and noisy domain experts will have higher accuracy on their respective domains than a classifier trained on a mix of both clean and noisy data. Unfortunately, it is difficult to know beforehand whether a given data point belongs to the clean or noisy section of the data, and therefore it is difficult to know whether to use the clean or noisy domain expert. Here, we present the *bestnets* method which uses *oracle learning* to approximate the behavior of both the clean and noisy domain experts with a single learning model.

In [1], oracle learning is used to approximate a single, larger ANN. With the bestnets method, oracle learning is used to approximate the behavior of multiple ANNs, each expert on parts of a given application's domain. When given a problem that has both noisy and clean data, one domain expert ANN is trained only on clean data, and another expert is trained only on noisy data. Then, each domain expert relabels the original training data on that expert's part of the domain. Furthermore, because the domain experts can be used to label a given data point, the original training data can be augmented by previously unlabeled data, creating an even larger training set. Note that this unlabeled data is only used to better approximate the domain experts, not for inferring additional concepts above what the domain experts learned from the original training set. Finally, a single ANN, the *bestnets-ANN*, is trained on the domain expert-labeled training set instead of training directly on the data. On a set of both noisy and clean optical character recognition data, using oracle learning to approximate the domain experts resulted in a statistically significant improvement ($p < 0.0001$) over standard training on the mixed data.

2 Background

The idea of approximating a model is not new. [3] used Quinlan's C4.5 decision tree approach [4] to approximate a bagging ensemble. [5] and [6] used an ANN to approximate a similar ensemble [5]. Craven and Shavlik used a similar approximating method to extract rules [7] and trees [8] from ANNs. Domingos [3] and Craven and Shavlik [7, 8] used their ensembles to generate training data where the targets were represented as either being the correct class or not. Zeng and Martinez [6] used a target vector containing the exact probabilities output by the ensemble for each class. The following research also uses vectored targets similar to Zeng and Martinez since Zeng's results support the hypothesis that vectored targets "capture richer information about the decision making process ..." [6]. Menke et. al used *oracle learning* in [1] to reduce the size of ANNs by approximating larger ANNs with smaller ANNs, using unlabeled data. While

previous research has focused on either extracting information from ANNs [7, 8], using statistically generated data for training [3, 6], or reducing the size of ANNs [1], the novel approach presented here is that a single ANN can be trained using oracle learning to approximate multiple domain experts.

3 Bestnets

There are three major steps in the bestnets learning process and the following sections describes each one in detail. First, the domain experts need to be trained correctly. Then, the domain experts are used to relabel the original training data changing the targets to the exact outputs of the correct domain expert on each data point. Finally, the bestnets-trained ANN is trained using the relabeled dataset.

3.1 Obtaining the Domain Experts

Since the accuracy of the domain experts directly influences the performance of the bestnets ANN, the domain experts must be the most accurate classifiers available for their domains, regardless of complexity (number of hidden nodes). In the case of ANNs, the most accurate classifier is usually the largest ANN that improves over the next smallest ANN on a validation set. The domain experts should be chosen using a validation (hold-out) set (or similar method) in order to prevent over-fitting the data. For our domain experts, we choose the most accurate ANN improving over the next smallest ANN on a validation or hold-out set in order to prevent the larger ANN from over-fitting.

3.2 Labeling the Data

The key to the bestnets method is being able to use knowledge of the domain at train-time to augment later generalization. Since the training set contains information indicating which data is clean and which noisy, that knowledge can be incorporated into a single classifier using the bestnets method allowing the bestnets-trained ANN to implicitly distinguish between clean and noisy data and "mimic" the behavior of an expert over that domain. This is accomplished by training an ANN to give the same outputs as the "clean oracle" when given clean data, and likewise give the same outputs a "noisy oracle" would give when presented with noisy data.

To train an ANN to behave in this manner, the clean data used to originally train the clean domain expert is relabeled by the clean domain expert with the exact outputs of the clean domain expert on each training point. The same is done with the noisy domain expert. In other words, this step creates a target vector $\mathbf{t^j} = \mathbf{t_1} \ldots \mathbf{t_n}$ for each input vector $\boldsymbol{x}^j$ from a given domain where each t_i is equal to the domain expert's activation of output i given the j^{th} pattern in the data set, $\boldsymbol{x}^j$. Then, the final bestnets data point contains both $\boldsymbol{x}^j$ and $\boldsymbol{t}^j$. In order to create the labeled training points, each available pattern $\boldsymbol{x}^j$ is presented as a pattern to its respective domain expert which then returns the output vector

$\boldsymbol{t}^j$. The final bestnets training set then consists of the pairs $\boldsymbol{x}^1\boldsymbol{t}^1 \ldots \boldsymbol{x}^m\boldsymbol{t}^m$ for all m data points—both clean and noisy.

3.3 Training the Bestnets ANN

For the final step, the bestnets-trained ANN is trained using the data generated in section 3.2, utilizing the targets exactly as presented in the target vector. The bestnets-trained ANN interprets each real-valued element of the target vector $\boldsymbol{t}^j$ as the correct output activation for the output node it represents given $\boldsymbol{x}^j$. The back-propagated error is therefore $t_i - o_i$ where t_i is the i^{th} element of the target vector $\boldsymbol{t}^j$ (and also the i^{th} output of the domain expert) and o_i is the output of node i. This error signal causes the outputs of the bestnets-trained ANN to approach the target vectors of the domain expert corresponding to each data point as training continues.

As an example, the following vector represents the output vector $\boldsymbol{o}$ of the noisy domain expert given the input vector $\boldsymbol{x}$ from a set of noisy data. $\hat{\boldsymbol{o}}$ represents the output of the bestnets-trained ANN. Notice the 4^{th} output is the highest and therefore the correct one as far as the domain expert is concerned.

$$\boldsymbol{o} = \langle 0.27, 0.34, 0.45, 0.89, 0.29 \rangle \tag{1}$$

Now suppose the bestnets-trained ANN outputs the following vector:

$$\hat{\boldsymbol{o}} = \langle 0.19, 0.43, 0.3, 0.77, 0.04 \rangle \tag{2}$$

The error is the difference between the target vector in 1 and the output in 2:

$$\boldsymbol{o} - \hat{\boldsymbol{o}} = \langle 0.08, -0.09, 0.15, 0.12, 0.25 \rangle > \tag{3}$$

In effect, using the domain expert's outputs as targets for the bestnets-trained ANN makes the bestnets-trained ANN a real-valued function approximator learning to behave like the appropriate domain expert on each domain.

4 Experiment and Results

In order to test the effectiveness of bestnets training, an experiment was conducted using *optical character recognition* (OCR) data containing both noisy and clean samples. The clean OCR data set consists of 500,000 alphanumeric character samples randomly partitioned into a 400,000 character training set, a 50,000 character validation set, and a 50,000 character test set. Each data point consists of 64 features from an $8x8$ grid of the gray-scale values of the character. The noisy OCR data set was created from the clean set by adding a random amount of noise to each of the 64 pixels in the $8x8$ grid. A different amount of noise was chosen for each pixel by randomly generating a gaussian distributed number, cubing it, and then adding it to the pixel's original value. This is repeated for each pixel on each pattern creating the "salt and pepper" effect seen in figure 1.

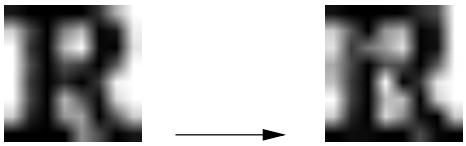

Fig. 1: The character R before and after adding random noise

The bestnets learning method as described in section 3 was applied to the OCR set. Both clean and noisy domain experts were created by training ANNs on clean-only and noisy-only data respectively. The domain experts were then used to relabel the original data, and then the domain expert-labeled data was used to train the bestnets-trained ANN. Results were obtained and compared to the domain experts and to training on mixed data without domain expert labels.

Results from McNemar tests [9] are shown in table 1. The ANN1 and ANN2 columns give the type of data used to train the models being compared. The data set column shows the type of data used to compare the two ANNs. The difference column gives the accuracy of ANN1 minus the accuracy of ANN2. The p-value column gives the p-value (lower is better) resulting from a McNemar test [9] for statistical difference between ANN1 and ANN2. A p-value of less than 0.0001 means that a difference as extreme or more than the difference between the two ANNs compared would be seen randomly less than 1 out of 10000 repeats of the experiment. In other words, the test is more than 99% confident that ANN1 is better than ANN2. Notice in the first two rows that the ANN trained on mixed data is significantly worse than the "expert" ANNs trained on their specific domains. This suggests there is room for improvement by using the bestnets method. The third row shows that the bestnets model was still significantly worse than the clean domain expert, and therefore there is still room for improvement on at least one of the domains. The fourth row in the table shows that the bestnets ANN is not significantly different than the ANN trained only on noisy data, yielding an improvement over directly training on the mixed data set and showing the bestnets method retaining the performance of one of the domain experts. Finally, the last row shows that the bestnets ANN is significantly better than the ANN trained on mixed data when compared using both the noisy and clean data. It is expected that the bestnets ANN will miss one less in every 250 characters than the mixed data trained-ANN. This final row is the most interesting because it compares two solutions to the problem that are realistic since the domain experts can not be used directly without a way of distinguishing explictly whether a given data point is clean or noisy.

5 Conclusions

The bestnets method improves over training on the mixed data and retains the performance of the noisy domain expert. Future work will investigate why

ANN1	ANN2	Data set	Difference	p-value
Clean Data Oracle	Mixed Data	Clean	0.0307	< 0.0001
Noisy Data Oracle	Mixed Data	Noisy	0.0092	< 0.0001
Clean Data Oracle	Bestnets	Clean	0.0298	< 0.0001
Noisy Data Oracle	Bestnets	Noisy	-0.0011	0.1607
Bestnets	**Mixed Data**	**Mixed**	**0.0056**	**<0.0001**

Table 1: Results comparing the bestnets method to training directly on mixed data.

there was not as significant of an improvement on the clean data. In order to determine where the bestnets method's ability to retain domain expert accuracy diminishes, experiments with several varying levels of noise will be conducted and then the bestnets method will be modified to preserve the experts' accuracy where there is currently room for improvement.

The bestnets method can be used for more than just approximating experts on varying levels of noise. One area of future work will develop methods to automatically identify subsections in a given application's domain. Then, the bestnets method can be applied over the subsections just as applied here on varying levels of noise. In this manner, bestnets can be applied to a wide range of problems, not just those for which divisions are known beforehand.

References

[1] Joshua Menke, Adam Peterson, Michael E. Rimer, and Tony R. Martinez. Neural network simplification through oracle learning. In *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'02*, pages 2482–2497. IEEE Press, 2002.

[2] Joshua Menke and Tony R. Martinez. Simplifying ocr neural network through oracle learning. In *Proceedings of the 2003 International Workshop on Soft Computing Techniques in Instrumentation, Measurement, and Related Applications*. IEEE Press, 2003.

[3] Pedro Domingos. Knowledge acquisition from examples via multiple models. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 98–106, San Francisco, 1997. Morgan Kaufmann.

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

[5] L. Breiman. Bagging predictors. *Machine Learning.*, 24(2):123–140, 1996.

[6] Xinchuan Zeng and Tony Martinez. Using a neural networks to approximate an ensemble of classifiers. *Neural Processing Letters.*, 12(3):225–237, 2000.

[7] Mark Craven and Jude W. Shavlik. Learning symbolic rules using artificial neural networks. In Paul E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 73–80, San Mateo, CA, 1993. Morgan Kaufmann.

[8] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 24–30, Cambridge, MA, 1996. The MIT Press.

[9] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.