Critical Analysis of Transfer Learning Algorithms

Colin P. McLean, #1139518, cmmclea@student.unimelb.edu.au

1 Introduction

The field of transfer learning (domain adaptation) has made numerous gains over the last two decades. Some of the most widely used and effective algorithms to solve these domain adaptation problems are straight forward aprroaches from the Frustratingly Easy Domain Adaptation (FEDA) paper (Daume, 2007). This paper will use the SRCONLY, TGTONLY, ALL, WEIGHTED, PRED, and LININT methods as baseline scores, as well as the AUGMENT method for further exploration of the data.

Since the original publication of the FEDA paper in 2007, there have been many new algorithms which outperform these baselines. This paper explores Extreme Learning Machines (ELM) and how they can be exploited in order to best address transfer learning problems.

The data set used in this paper is regarding school exam scores in the United Kingdom over a period of three years. The exam scores are for three different types (domains) of schools: single sex male, single sex female, and co-educational. The goal is to develop machine learning models which predict exam scores with Mean Squared Error (MSE) as the measure of success.

2 Baseline Approaches

2.1 Exposition of Method

The baseline tests and the augment approach use ridge regression and neural network learning algorithms to predict the test scores. Ridge regression analyzes multiple regression data and includes a regularization parameter to penalize the use of large weights in the model. It produces strong MSE values and generalized well. Neural networks are in a different class of machine learning models than ridge regression, so they can exploit the data in ways regression models cannot. Critical analysis of the models will be discussed in section 2.2 and can be observed in Table 1.

Table 1: FEDA Approaches MSE Scores

Model	Dom	SrcOnly	TgtOnly	All	Weight	Pred	LinInt	Augment
Ridge	M/F	114.929	116.197	114.745	113.007	109.270	110.125	110.173
NN	M/F	130.668	250.896	115.021	283.876	134.756	150.525	117.132
Ridge	F	164.345	173.607	164.370	167.141	160.247	161.635	162.853
NN	F	163.317	212.455	179.370	199.361	130.613	197.079	159.884
Ridge	M	157.651	150.927	157.341	150.441	148.938	159.378	148.651
NN	M	132.914	274.015	159.583	276.218	150.421	181.543	146.425

The SRCONLY, TGTONLY, and ALL approaches train on the source data, target data, and all data respectively. The WEIGHTED approach trains on all data and up-samples the target data. The PRED approach trains a SRCONLY model and then predicts the target values using that model. These predictions are then appended to the target data and a new TGTONLY model is made. The LININT model trains SRCONLY and TGTONLY models and then finds a combination of the two.

The AUGMENT approach is as follows: for each of the domains, make a general version of it, a source version, and a target version (Daume, 2007). Given k domains, the augmented feature vector will result in K+1 copies of it, in our case 4. Consider a feature vector x and a feature transformation ϕ . There will be four entries in the feature transformation, the first being the general case, the second being the male specific case, the third being the female specific case, and the fourth being the co-ed case. For example, $\phi^{\text{male}} = \langle x, x, 0, 0\rangle$, $\phi^{\text{female}} = \langle x, 0, x, 0\rangle$, and $\phi^{\text{co-ed}} = \langle x, 0, 0, x\rangle$.

2.2 Results

Ridge regression is a better predictor than the neural network in most cases. In particular, the TGTONLY and WEIGHT approaches had weak performance with the neural networks. Due to the low amount of training data, the neural networks were not able to see enough data to generalize well. Similarly in the WEIGHTED approach, this small subset of target data was given more weight in the final model by up-sampling the target data. The target data might not truly be representative of the target domain as a whole, therefore giving more weight to the non-representative data and explaining the variability in the neural network performance. Due to their unpredictability while training, neural networks also give varying results based on the data partitions, which will be discussed further in section 2.3.

The feature augmentation approach using ridge regression produced MSE values approximately the same as the lowest of the six baseline. However the neural networks are very strong predictors with this approach. Across all domains, the neural network achieved the second lowest MSE, and a MSE that was very close to the regression score, if not lower. This could be explained by the increase in feature space and the allowance for individuality among domains, as discussed in FEDA (Daume, 2007). The different features can take on different meanings across domains, so the feature augmentation approach allows this to be reflected in the data and results in low MSE.

The paper conducted a secondary experiment to see how test set size affected MSE values. When the test sets were increased in size from 100 to 200, the observed results achieved 20 points lower MSE values on average. However when the test sets were decreased in size from 100 to 50, the observed results were more inconsistent, with an average increase of MSE of 12. These results can be explained by the fact that larger testing data better represent the data.

2.3 Hyperparameter Tuning

The process of hyperparameter tuning was completed via grid search. Given a list of possible parameters values, the grid search algorithm will tune a model for every combination of given hyper parameter and return the combination providing lowest error rate. This paper uses the hypopt package's implementation of grid search, namely the GridSearch class ("hypopt", 2019).

The hyperparameters were tuned on all of the data from the three domains compiled into one set of x and y data. When tuning the ridge regression model, the grid search algorithm was given the possible list of alpha values = [.001, .01, .1, 1, 10]. These given values then generated a best alpha of 10. When tuning the neural network, the grid search algorithm was given the following list of hyper parameters: learning rate = [.0001, .001, .01, .1], maximum iterations = [350, 400, 450], and hidden layer size = [(64,64), (16,16,16,16), (8,8,8,8,8,8,8,8)]. The optimal values returned were learning rate = .01, iterations = 350, and hidden layer size = (16,16,16,16). These parameters resulted in an improved MSE of nearly 30 for each of the models on every approach implemented.

The partitions of data had a great effect on the MSE values. The three domains each had development sets of 100. When using target data to train models, only 100 samples from the target domain were included. However the size of the test data was variable. The final partitions use 94% of the mixed data, 93% of the female data, and 92% of the male data for training. This results in tests sets of around 200 for each of the three domains, and consistently lower MSE values upwards of 50 points.

Larger values for the alpha value in ridge regression resulted in smaller weights in the general case and larger values in the domain specific cases. The higher the penalty for large weights, the more important these weights were.

3 Domain Adaptation Extension

Extreme Learning Machines (ELM) were first introduced in 2006 as an extension to the existing uses of a feedforward neural network (Huang et al., 2006). Huang later went on to update the paper to demonstrate a kernelized version of the model (Huang et al., 2012). ELMs have gained popularity in the field of transfer learning due to their ability to process large amounts of data in an efficient manner (Salaken et al., 2017). In the paper, they discuss the most popular ELM approaches to transfer learning. Among them was the idea of adding the source data that was most similar to target data in order to train a model on only the target data with this added similar source data.

ELM is a type of single layer feedforward network whose input weights are chosen randomly and output weights are determined analytically by a Moore-Penrose generalized inverse (Serre, 2002). Huang et al. show that ELM models provide lower computational complexity and greater generalization performance—ideal for transfer learning (Huang et al., 2012).

They also extend further on the original proposed ELM to kernelize it. This means that through using a kernel function, such as radial basis function (RBF), the original feature matrix can derive a kernel. The benefits of using Kernelized ELMs include faster run times and strong performance on sparse data, the latter making it ideal for transfer learning problems.

For the sake of brevity, the full algorithm for ELM is not described in this paper, but the reader is encouraged to refer back to the Huang et al. (2012) description for further details on both algorithms.

Pan et. al (2014) discuss an approach to Q learning that utilizes the ELM model and transfer learning techniques. Among the transfer learning techniques used, they calculate likelihood ratios between source and target data and then generate probabilities from these likelihood ratios. This increases the amount of data in the data impoverished target domain, while still training on relevant data.

This addresses two problems encountered during the baseline implementations. Firstly, the target only baseline approach trains on only 100 data points. The suggested addition of similar source data to the target data would solve the issue of having too few training data points. Secondly, the training data would still be relevant to the target data. The benefit of the target only model is that it is training solely on the target data. As this data is within the target domain, one would think that this data will be the best predictor. Incorporating the most relevant source data ensures that the training data is still similar to the target domain data.

In order to implement this approach, the elm package was used, specifically the ELMKernel class (Almeida, 2015). The search_param() method was used for hyperparameter tuning, which performs time-series cross validation (CV) to determine the best parameters. The hyperparameters are the kernel function, the coefficient of regularization, and the number of hidden neurons. For each of the kernel functions (rbf, linear, and polynomial) the optimal hyperparameters are set and the models are ready to train data. The kernel function with the lowest CV score was polynomial. The final ELM used in this paper had the following hyperparameters: ["poly", 1.0722, [3.0674, 2.3780]].

To calculate source and target data similarity, a similarity score function was implemented. The function took in source data X_s, target data X_t , and a threshold of similarity α. For each point in X_s, the function looped through the 100 points in X_t and calculated the sum of squared differences between the two data points. For this paper, a threshold of 400 was used which resulted in 1500 extra data points being added from X_s to the training data.

The combination of kernelized ELMs along with the training on a selected portion of the source and target data led to a low MSE across all domains. The results can be observed in Table 2. The TgtOnly models did not produce strong results with the ELM models. However, the All model outperformed all of the other baseline models. Furthermore, the addition of less than 30% of source data allowed for a reduction of 50 points in most cases. It would be expected to see a lower MSE for the implemented model rather than the All model on data sets with more drastic differences between domains.

The use of kernelized ELM allows for lower MSE values on the school data set by a 20, 40, and 30 for mixed, female, and male data respectively. These are non-trivial decreases that show the efficacy of ELM models applied to domain adaptation.

Table 2: Kernelized ELM MSE Scores

Kernel Function	Dom	TgtOnly	Extension	All
Polynomial	M/F	169.395	113.387	100.269
Polynomial	F	175.597	124.936	120.454
Polynomial	M	172.915	135.168	119.065

References

Almeida, A. (2015). *Elm*. https://elm.readthedocs.io/en/latest/elm.html

Daume, H. (2007). Frustratingly Easy Domain Adaptation, In *45th annual meeting of the association of computational linguistics*, Association for Computational Linguistics. Prague, Czech Republic.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *"IEEE Transactions on Systems, Man, and Cybernetics—PArt B: Cybernetics"*, *42*(2). https://doi.org/10.1109/TSMCB.2011.2168604

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *"Neurocomputing"*, *70*, 489–501.

Hypopt. (2019). https://pypi.org/project/hypopt/

Salaken, S. M., Khosravi, A., Nguyen, T., & Nahavandi, S. (2017). Extreme learning machine based transfer learning algorithms: A survey. *"Neurocomputing"*, *267*, 516–524.

Serre, D. (2002). *Matrices: Theory and applications* (Vol. 216). New York, Springer.