Locally Weighted Ensembling

Kennan LeJeune

December 7, 2019

1 Introduction

Locally Weighted Ensembling is a transductive parameter knowledge transfer framework [1] intended to improve the learning of a target task τ on a testing domain T, transferring models $M_1, M_2, \ldots, M_k$ trained on labeled domains of interest $D_1, D_2, \ldots, D_k$. For any example x, we weight the model predictions according to their performance in the neighborhood of other examples clustered near x, making an overall prediction with a weighted average of the model outputs at x.

1.1 Example

To best illustrate the effectiveness of such a framework, consider a toy problem with datasets D_1, D_2 and respective models M_1, M_2 as shown by Figure 2. Although each training set has a linear decision boundary, the optimal boundary on the test set follows a v-shape. In this context, we have M_1 which can effectively classify the region R_1, and M_2 which can classify R_2. Rather than merging the test domains, or globally weighting the models, we can produce a weight based on the quality of M_i at any example x, so that the test set can weight M_1 highly to classify R_1, but also weight M_2 more highly to classify R_2 when appropriate [2]. Optimally, we would determine this with Bayesian model averaging to compute the posterior distribution of y as $P(y|x)$ [2]. However, in the case of an unlabeled test domain, we are unable to compute $P(M_i|D)$ in order to properly compute

$$P(y|x) = \sum_{i=1}^{k} P(y|x, D, M_i)P(M_i|D) \tag{1}$$

1.2 Estimating Model Output on an Unlabeled Domain

We can estimate $P(y|M_i, x)$ on the test domain by examining the similarity to M_i on examples in the space around x, and examples which are clustered in the test domain around x [2]. Consider graphs $G_M = (V, E_M)$ and $G_T = (V, E_T)$ where $V = \{x \in T\}$. We construct E_T by clustering the T and adding an edge between all pairs of test examples which are members of the same cluster.

Similarly, E_M is constructed by connecting any two examples $u, v \in T$ if a model M predicts the same class for both u and v. We can construct G_{M_i} for every model M_i of a training domain D_i, and G_T based on the clustered test domain [2].

For any x, we can compute the model weight at x, which is proportional to the similarity of its local structures. If we denote the sets of neighbors of $x \in G_M, G_T$ to be V_M, V_T respectively, then the weight calculation [2] is

$$w_{M,x} \propto s(G_M, G_T; x) = \frac{\sum_{v_1 \in V_M} \sum_{v_2 \in V_T} 1\{v_1 = v_2\}}{|V_M| + |V_T|} \tag{2}$$

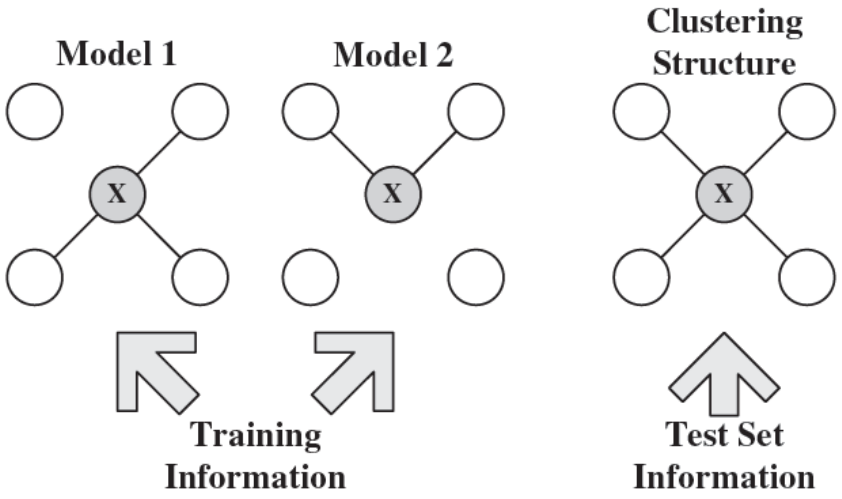

Figure 1: Example Similarity Graph Construction

Figure 1 depicts an example of three such graphs, where $w_{M_1,x} \propto \frac{3}{4}$, and $w_{M_2,x} \propto \frac{1}{2}$ since models 1 and 2 have 3 of 4 and 2 of 4 neighbors in common with x on the test set respectively. With this, we can effectively weight our models to maximize their utility on a per-example basis, providing insight into the power of this framework.

It is crucial to note that this estimation is based upon a clustering assumption [2], asserting that an average clustered train set must be mostly representative of the actual distribution of class labels for examples in the set. If this assumption is not fulfilled, then we simply take an evenly weighted average of the model predictions for all $x \in T$ such that

$$P(y|x) = \sum_{i=1}^{k} w_{M_i,x} P(y|x, M_i) \tag{3}$$

where $P(y|x, M_i)$ denotes the output prediction of M_i for example x, and use $w_{M_i,x} = 1/k$ for all models.

Figure 2: Toy Problem

2 Algorithm Implementation

Figure 3 outlines the general structure which was followed for the algorithm implementation.

Algorithm: Locally Weighted Ensemble (LWE)
Input: (1)A training set D or k training sets $D_1, \ldots, D_k$
(2)k classification models $M_1, \ldots, M_k$ $(k > 1)$
(3)A test set T which comes from a different domain but the classification task is the same.
(4)A threshold δ and cluster number c'.
Output: The set of predicted labels Y for examples in T.
Algorithm:

1. Perform clustering on the training set(s), **IF** the average purity of clustering is less than 0.5, set $w_{M_i,\mathbf{x}} = \frac{1}{k}$ for all M_i and $\mathbf{x}$, and compute the posterior using Eq.(7) for each $\mathbf{x} \in T$. **RETURN**.
2. Group test examples into c' clusters and construct neighborhood graphs based on the clustering results and all the k models. Set $T' = \Phi$.
3. **FOR** each $\mathbf{x} \in T$,
   - **FOR** each model M_i, compute the model weight $w_{M_i,\mathbf{x}}$ according to Eq.(5).
   - **IF** $s_{\text{avg}}(\mathbf{x}) \geq \delta$, decide $\mathbf{x}$'s label based on the weighted ensemble's output $P(y|E,\mathbf{x})$ obtained using Eq.(7), **ELSE** put $\mathbf{x}$ into T'.
4. **FOR** each $\mathbf{x} \in T'$, predict $\mathbf{x}$'s label from the "unsupervised" classifier U, i.e., estimate $P(y|U,\mathbf{x})$ using Eq.(11) or Eq.(12). **RETURN**.

Figure 3: Locally Weighted Ensembling Framework

We begin by training models M_i corresponding to each training set D_i with Logistic Regression, and pass each trained model to the function. We perform clustering on the each training set D_i using `AgglomerativeClustering` from scikit-learn, and compute the confusion matrix according to the class labels for each $x \in D_i$. If the clustering assumption fails, then we cannot estimate the structure of the test domain using the structure of the training domains, so we return the average model output as demonstrated in Equation (3). Otherwise, we proceed with the LWE algorithm. We cluster the testing domain and then construct neighborhood graphs as proposed in Section 1.2.

Using these neighborhoods, we iterate over the examples in the test domain and compute a

normalized model weight [2],

$$w_{M_i,x} = \frac{s(G_{M_i}, G_T; x)}{\sum_{i=1}^{k} s(G_{M_i}, G_T; x)} \tag{4}$$

and if the the average $s_{avg}(x) = \frac{1}{k}\sum_{i=1}^{k} s(G_{M_i,G_T;x})$ satisfies a threshold value δ, compute the output for x using Equation (3). Otherwise, estimate the output of x by taking the average output of neighbors of x which are members of the same cluster, and also have an average structural similarity $s_{avg}(x) \geq \delta$.

This step allows us to approximate any example with reasonable accuracy regardless of a lack of confident model output, and is a significant portion of the manner in which the framework distinguishes itself from other ensembling methods, allowing us to leverage both structural similarity and model confidence for local regions of the test space.

We have additionally implemented a baseline classifier (SGA) [2] for comparison which simply uses the weighted output from Equation (3) with $w_{M_i} = 1/k$ without the clustering assumption, and a variant pLWE, which uses Equation (3) with the local model weight, regardless of structural similarity at x. We compare these approaches in Section 3.

3 Results and Analysis

We first run the algorithm on the Multi-Domain Sentiment Dataset, consisting of Amazon Reviews. We apply training on DVD reviews to produce sentiment classification for reviews of Kitchen products, training on 1000 reviews from a single dataset and test on a set of 300 reviews, allowing for a 5000 dimensional feature space.

DVD to Kitchen	**SGA**	**LWE**	**pLWE**
Avg. Accuracy	0.7506666667	0.774	0.7673333333
St. Dev	0.01404753834	0.01311487705	0.01535195

Figure 4: Amazon Review Sentiment Classification

While the classifier's accuracy is not particularly stellar, LWE manages to significantly outperform the baseline classifier, SGA, with the true mean of the difference between LWE and SGA's accuracy $\mu \in [0.0033, 0.0427]$ with 95% confidence. However, pLWE fails to be statistically significant, with $\mu \in [-0.00447, 0.03647]$ with 95% confidence, despite overall poor clustering quality on both the training and testing domains.

Next, we run the algorithm on ECMLL/PKDD Task A and Task B, spam datasets, as described in the group paper, with results shown in Figure 5.

Task A has 4000 source domain examples, resulting in a high initial baseline, which is only slightly improved upon by LWE and pLWE, although the differences are weakly statistically significant.

ECML/PKDD Task A	SGA	LWE	pLWE
Avg. Accuracy	0.9476666667	0.9553333333	0.9513333333
St. Dev	0.002516611478	0.001527525232	0.003785938897

Figure 5: Spam: ECMLL/PKDD Task A Results

ECML/PKDD Task B	SGA	LWE	pLWE
Avg. Accuracy	0.8733333333	0.9066666667	0.8966666667
St. Dev	0.01154700538	0.0132231652	0.0144

Figure 6: Spam: ECMLL/PKDD Task B Results

Task B has 15 sets containing 100 examples each, yielding a lower base accuracy, since not all models are equally effective when training across multiple domains, yielding an inferior approach when predicting with the averaged output in Equation (3). The results of LWE and pLWE are both statistically significant when compared against the base classifier as demonstrated in the group writeup.

Finally, we run on the 20 Newsgroups dataset [2], yielding the best overall performance of this algorithm as demonstrated by results in Figure 7.

20 Newsgroups	SGA	LWE	pLWE
Avg. Accuracy	0.8186666667	0.975	0.9276666667
St. Dev	0.003511884584	0.01311487705	0.03453018004

Figure 7: 20 Newsgroups Results

We find this dataset to be optimal for this approach since it is extremely purely clustered, but also has substantial depth to produce highly confident classifiers for nearly any local $x \in T$, although weaker points still exist near decision boundaries, as indicated by differing accuracy between results on LWE and pLWE. The accuracy of LWE outpaces the baseline by nearly 16%, and demonstrates one of the most effective applications of this framework in this problem space.

4 Comparison to Other Relevant Literature

We have explored related work detailing with an alternative approach to transform the model of training examples to produce a Bayesian prior which can be applied to the test domain [3]. Other approaches alternatively attempt to reduce covariate shift by reweighting training examples according to their ratio of test frequency to train frequency to minimize the reweighted probability [4]. In each case, the methods fail to consider multiple source domains, and lose out on the benefit of model localization on a test domain.

5 Conclusions

We have implemented the Locally Weighted Ensembling framework, in addition to a Partially Locally Weighted variant, and a baseline classifier on which to compare performance across three datasets. In each case, the LWE framework proved to provide statistically significant performance gains, approaching 16% accuracy improvement on some datasets.

With this framework, we effectively outperform a standard classifier in cases where source and target domains differ, or match in cases where we deal with a standard machine learining problem. Finally, we have demonstrated the effectiveness of this approach to problems where target domain data is limited, but plentiful in other related areas, and demonstrated its utility for future use.

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, 1345–1359, 2010. DOI: 10.1109/tkde.2009.191.

[2] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*, 2008. DOI: 10.1145/1401890.1401928.

[3] X. Li and J. Bilmes, *A bayesian divergence prior for classifier adaptation*, 2007.

[4] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, no. 2, 227–244, 2000. DOI: 10.1016/s0378-3758(00)00115-4.