A Novel Domain Adaptation Solution to the Transductive Transfer Learning Problem

Jordan Thomas Ash

A Thesis

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Master of Science in Engineering

Recommended for Acceptance

by the Department of

Computer Science

Adviser: Robert Schapire

June 2015

© Copyright by Jordan Thomas Ash, 2015.

All rights reserved.

Abstract

Most classification algorithms rely on the assumption that training and testing data come from the same distribution. When this assumption is violated, classification performance can be seriously affected. Transfer learning, a new and increasingly popular branch of machine learning, seeks to remedy this potential problem. In this thesis, we introduce a new transductive transfer learning technique that functions by leveraging two separate classification hypotheses to geometrically align the source and target datasets before classification. We show several examples of this technique, and compare it to other methods commonly used in transfer learning.

Introduction

From the perspective of human intelligence, transfer learning can be thought of as responsible for why it's easier to learn to speak French after having learned Spanish, or to learn to program in C++ after having learned Java. Transfer learning becomes important any time we want to apply a model obtained from one dataset to a similar dataset that comes from a different distribution.

In the transfer learning paradigm, rather than having training data and testing data, we are given source data, from which we want to transfer knowldedge, and target data, to which we want to transfor knowledge. Transfer learning comes in two flavors, inductive and transductive, in which we are either given some labels or no labels on our target data respectively. As one might expect, transductive transfer learning is generally regarded as significantly more difficult than inductive transfer learning.

This research was largely motivated by a particular transductive transfer learning problem that will be discussed in detail later. This problem is an extreme case of transfer learning because, as we'll show, none of the sources have any significant geometric overlap. Because of its difficulty, the problem was outsourced as a competition on a popular data science dataset repository in 2014. We will show that our algorithm gets results that are superior to what's been published on this problem [1], and that even surpass the winners of the online competition.

This thesis is organized in the following way. We first describe transfer learning in detail, in addition to the assumptions we'll be making. Next, we will discuss other transfer learning algorithms that are frequently used. Then, we'll introduce our technique at a high level. Finally, we will demonstrate the algorithm in two very different settings, one of which being the outsourced problem mentioned above. We'll compare these results to those produced by other transfer learning algorithms.

Transfer Learning and Assumptions

Let X_s^j be a particular source dataset (of which there may be several), meaning that we would like to transfer some model trained on X_s^j to our unlabeled target data, X_t. Further, let $\mathcal{D}(X_s^j)$ and $\mathcal{D}(X_t)$ be defined as the underlying distributions generating the j^{th} source and target datasets respectively. The typical machine learning paradigm assumes that $\mathcal{D}(X_s^j) = \mathcal{D}(X_t)$. However, in a transfer learning framework, we instead assume that $\mathcal{D}(X_s^j) \neq \mathcal{D}(X_t)$, and further that $\mathcal{D}(X_s^j) \neq \mathcal{D}(X_s^i)\ \forall j \neq i$. In some cases, we are given some labels on data points from $\mathcal{D}(X_t)$. In that case, the problem is called inductive. Otherwise, the problem is called transductive. In either case, we are given X_t before classification. Here, we will assume that sample labels are either zero or one, $y(x) \in \{0, 1\}$.

In this thesis, we will be assuming that there exists some transformation, A_j, such that $\mathcal{D}(X_s^j) = \mathcal{D}(A_j X_t)$ for any particular source j. Our goal will be to uncover this transformation, allowing us to use the classification model trained on X_s^j. Even stronger, we will be assuming that this transformation, A_j, will capture a translation, dilation, and rotation. In other words, we are assuming that $\mathcal{D}(X_t)$ is translated, dilated, and rotated with respect to $\mathcal{D}(X_s^j)$.

Related Work

Covariate Shift

[1] discusses two approaches to solving this problem. One, termed covariate shift, seeks to show labeled source instances to a learner with a frequency proportional to the degree with which they look like they may have been emitted from the target distribution [2]. Accordingly, it weighs the likelihood of the classification algorithm seeing a labeled source sample during training by

$$\frac{P_{\sim\mathcal{D}(X_t)}(x_{s,i}, y(x_{s,i}))}{P_{\sim\mathcal{D}(X_s)}(x_{s,i}), y(x_{s,i})},$$

where $x_{s,i}$ is some labeled source datum. Covariate shift makes the assumption that $P_{\sim\mathcal{D}(X_t)}(Y|X_t) = P_{\sim\mathcal{D}(X_s)}(Y|X_s)$, so that

$$\frac{P_{\sim\mathcal{D}(X_t)}(x_{s,i}, y(x_{s,i}))}{P_{\sim\mathcal{D}(X_s)}(x_{s,i}, y(x_{s,i}))} = \frac{P_{\sim\mathcal{D}(X_t)}(x_{s,i})}{P_{\sim\mathcal{D}(X_s)}(x_{s,i})}.$$

So, to build a classifier that will hopefully generalize well to target data, one only needs to show a learner source data with a frequency proportional to $\frac{P_{\sim\mathcal{D}(X_t)}(x_{s,i})}{P_{\sim\mathcal{D}(X_s)}(x_{s,i})}$ during training.

To approximate these probabilities, the authors suggest using logistic regression, trained on labels corresponding to whether a sample comes from the source or target dataset.

Stacked Generalization

Second, [1] employs a technique called stacked generalization [3]. This is a general ensemble learning procedure in which several classifiers, C_i, $i = 1, 2, ..., k$, are trained on different portions of the training data. Then, the hypotheses generated by these classifiers are used as features for another learner trained with all of the available training data, $G : C_1(X_s), C_2(X_s), ..., C_S(X_s) \rightarrow y(X_s)$, where X_s is the set of all source subjects. The authors of [1] claim that the best way to adapt this method to fit the transfer learning framework is to make a first-level classifier for each of the source subjects.

We will compare our results to both of these techniques, as well as a combination of the two. In the latter case, each C_i sees samples from its corresponding X_s^i during training with a frequency computed by the covariate shift algorithm.

A Classification Procedure

For any held-out target data, we will leverage two feature spaces in which we could work. One is the higher-dimensional space that the data already lies in, X_t, while the other is simply some arbitrary lower-dimensional embedding of this data, $\tilde{X}_t$. The goal of this algorithm is to align the low-dimensional representation of the target data with that of a particular source, $\tilde{X}_s^j$. After this alignment, a classifier learned on the low-dimensional source should be able to perform well on the low-dimensional target. The classification procedure outlined in this section will leverage both spaces. We will use the high-dimensional space to align the low-dimensional $\tilde{X}_t$ for the actual classification.

For a fixed target subject, we first train a classification algorithm on all available source subjects X_s^j, $j = 1, 2, ..., S$ in high-dimensional space. The classifier outputs some hypothesis, $h(X_t)$, on the target between zero and one, reflecting the learner's confidence that the sample corresponds to the true or false class, respectively. We will use $h(X_t)$ to align the data in the low-dimensional space.

In the low-dimensional space, we fix a particular source subject to which we will align our target data. This alignment will involve rotating, translating, and dilating the target data to match it to the source data. Translation and dilation can be aligned by simply whitening both datasets, which makes the covariance matrices of both datasets equal to the identity matrix. In practice, this could be achieved by PCAing the data, then dividing each dimension by the square root of its corresponding eigenvalue. Across each dimension, whitened data has a mean of zero, making the source and target centered on the same point, and a standard deviation of one, making the datasets spread over the same geometric region. Another major advantage of whitened data, especially in this framework, is that if it is rotated, the resulting data is still white. From this point forward, $\tilde{X}_t$ and $\tilde{X}_s^j$ will refer to a whitened target and source dataset respectively, rather than their unwhitened counterparts.

The more difficult task is to find the rotation for the target data that best matches it to the source data. To accomplish this, we first train another classifier on a fixed low-dimensional source dataset we want to transfer knowledge from, $\tilde{X}_s^j$. This will be yet another classifier, which like the high-dimensional learner, can output classifications on data between zero and one. We will refer to this low-dimensional classifier as f_j. If we assume that the high-dimensional hypothesis $h(X_t)$ performs better than guessing, and that f_j is the correct shape for a low-dimensional dichotomizer, then it seems reasonable to rotate the whitened target data until $f_j(\tilde{X}_t)$ best matches

$h(X_t)$. So, to find the optimal rotation in low dimensions, we want to minimize the disagreement between the low and high dimensional classifications.

$$disagreement_j(R) = \sum_{i=1}^{N} (h(x_i) - f_j(R\tilde{x}_i))^2$$

where x_i is the i^{th} target sample in high-dimensional space, $\tilde{x}_i$ is the i^{th} whitened target sample in low-dimensional space, and R is a rotation matrix in the same number of dimensions as $\tilde{X}$. Once we have found the minimizing rotation matrix, $R_j^* = \arg\min_R \ (disagreement_j)$, we compute a final hypothesis for X_t as

$$\left\lfloor \frac{1}{S} \sum_{j=1}^{S} f_j(R_j^* \tilde{X}_t) \right\rceil,$$

averaging the classifications produced by leveraging all S sources.

Experiments and Results

MEG Data I

This problem, which motivated the research in this thesis, was originally presented by Kaggle as a data science competition. The Kaggle website connects various organizations with a network of people interested in data science, effectively allowing large problems to be outsourced. This particular challenge was sponsored by several companies, including Elekta Oy, MEG International Services, Fondazione, and Besa.

The data consists of 16 training subjects. For each subject, hundreds of samples were collected where he/she was shown either an image of a face or of a "scrambled face," which was essentially just a noisy image with no discernible face-like structures. Each individual sample was collected by monitoring each subject's brain activity with a neuro-imaging technique called Magnetoencephalography (MEG). The goal is to correctly label, on subjects unseen during training, which samples correspond to faces and which correspond to noise.

MEG data is created from many sensors, each recording some time-varying signal at a different part of the skull. For this reason, we chose to represent the MEG data using the Common Spatial Pattern (CSP) algorithm, which is most frequently used on electroencephalogram (EEG) because EEG also has this time-varying multivariate signal property. The CSP algorithm finds a set of components such that, when the data is spread across them, the variance between the true-labeled data and false-labeled data is maximized [4].

After constructing this representation, we employed t-distributed Stochastic Neighbor Embedding (t-SNE) [5] to reduce its dimensionality. The t-SNE algorithm operates by minimizing the KL divergence between a distribution representing a high-dimensional datum and a distribution representing its corresponding low-dimensional embedding. Specifically, t-SNE begins by, for some general fixed data point X_i, computing the probability that all X_j, $j \neq i$, would be selected as X_i's nearest neighbor according to a distribution. This is done by centering a Gaussian at X_i and computing the conditional probability of all X_j according to this Gaussian. The resulting list of probabilities is then normalized to become a distribution over all neighbors of X_i. An analogous set of normalized probabilities is computed for the low-dimensional data by using a student's t-distribution, rather than a Gaussian. An embedding is found my minimizing the KL divergence between these high and low-dimensional distributions, summed over all data points. We used t-SNE to embed our data into two dimensions to create figure 1.

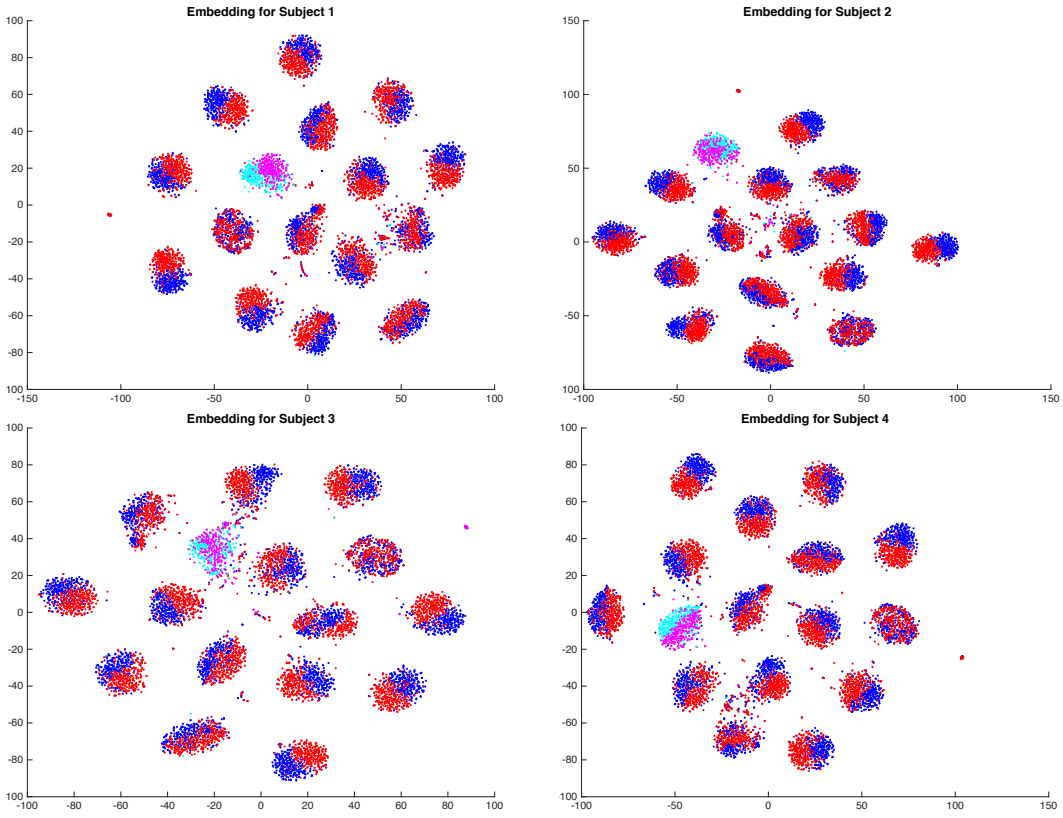

Figure 1: Four example embeddings from t-SNE. Red and blue respectively correspond to positive and negative examples for training (source) subjects, while Cyan and Magenta respectively correspond to positive and negative examples for test (target) subjects.

Figure 1 instantly gives some validation to our assumptions. Indeed, each subject appears to be rotated, translated, and dilated with respect to every other subject. Because the data is now embedded into two dimensions, and two-dimensional rotation matrices have the explicit form

$$R = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix},$$

we can find R^* by just searching over all $\theta \in [0, 2\pi)$. Figure 2 shows $disagreement_j$ and target error as a function of θ, demonstrating the strong correlation between the value of $disagreement_j$ and target error. Of course, these plots also show what we're really interested in, which is that when the objective function is minimized, target classification error is minimized as well.

As can be seen in table 1, we are sometimes unable to find a high-quality alignment for two datasets. This obliterates classification performance, as in the case of subjects 11, 3, 16, and 6. Some of these cases are plotted in figure 3.

The plots in figure 3 are so poor that the high-dimensional classifier actually outperforms the low-dimensional classifier. In other words, we see no performance

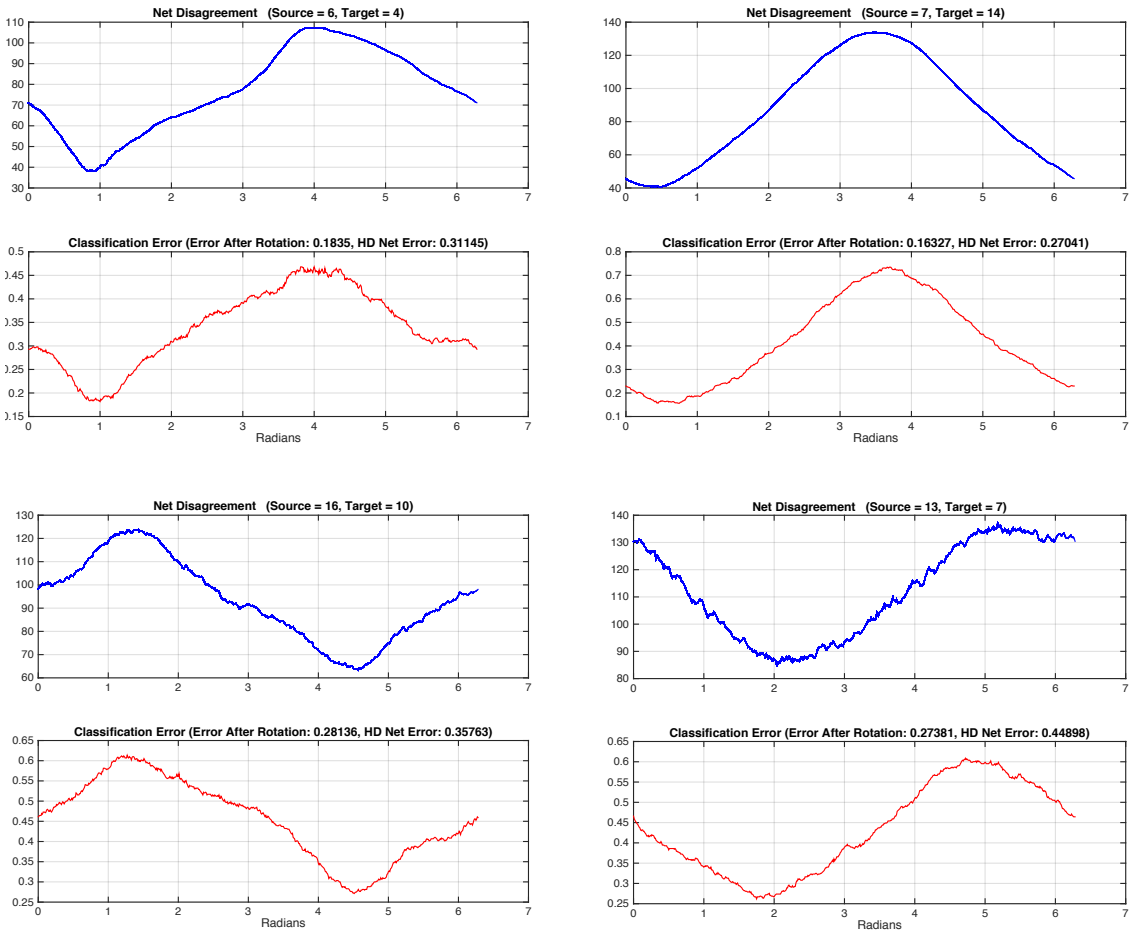

Figure 2: Four source-target pair examples, showing the value of $disagreement_j$ (blue) and the classification error on the target data (red) as a function of $\theta \in [0, 2\pi)$. Each plot also identifies the error after alignment and the error of the naive high-dimensional classifier h (HD Error).

Classification Error Summary			
S01: .22	S02: .32	S03: .37	S04: .19
S05: .25	S06: .33	S07: .31	S08: .20
S09: .16	S10: .29	S11: .41	S12: .25
S13: .28	S14: .17	S15: .22	S16: .37
Overall Error: .27			

Table 1: A per-subject breakdown of classification error rate. In this example, h and f_j are both feedforward neural networks with a single hidden layer.

improvement by doing the alignment at all. Still, it's worth noting that the $disagreement_j$ values remain correlated with the error plots shown below them. As one might expect, it turns out that alignments that produce high errors tend to

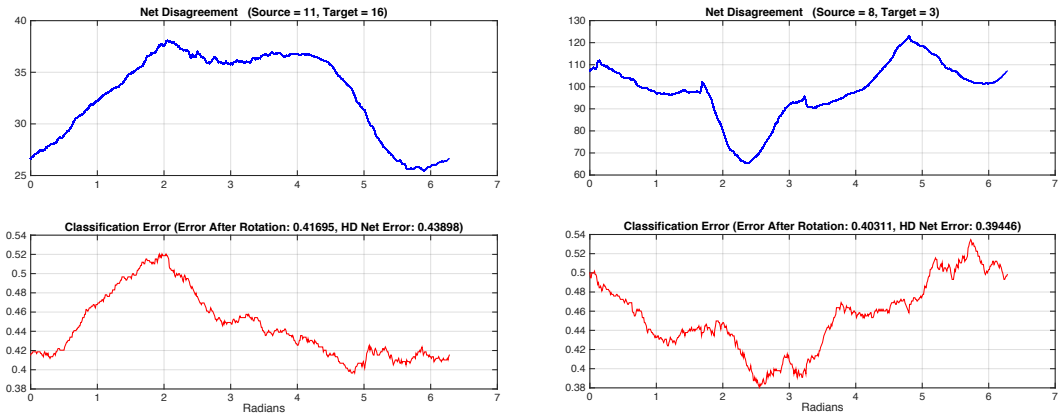

Figure 3: Two target-pair examples as in figure 2, but with particularly poor classification error values.

correspond to transferring knowledge from or to low-dimensional clusters that seem disorderly. Such clusters can be observed in figure 1 as inseparable.

This behavior is very predictable. Figure 4 plots classification error as a function of the disorder of each target subject in the low-dimensional space. Here we are defining disorder as the probability that a target point and its nearest neighbor have the same ground-truth class. We can easily compute this probability as

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{y(\tilde{x}_i) \neq y(NN(\tilde{x}_i))\}$$

where n is the number of points in the target subject's data, $NN(\tilde{x}_i)$ is the nearest neighbor to some $\tilde{x}_i$, and $y(\tilde{x}_i)$ is the ground-truth label of $\tilde{x}_i$. if a dataset is pure labeled randomly, then disorder would be .5. If instead it is perfectly separable, and there's some space between the true and false samples, disorder would be 0.

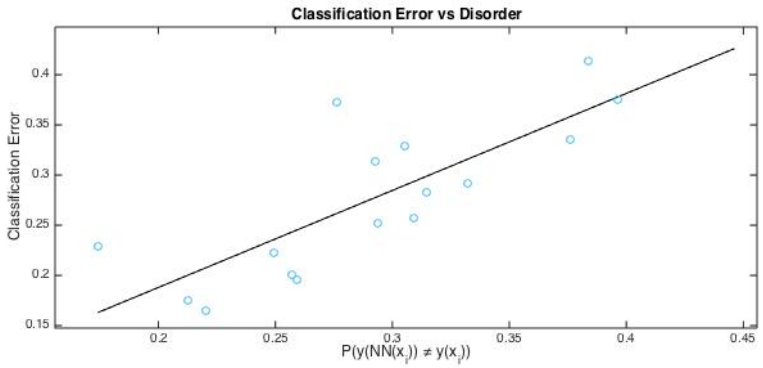

Figure 4: A plot showing disorder as a function of classification error for all 16 subjects.

It's highly possible that the high degree of disorder shown in some subjects, creating a similarly high classification error, is the result of an imperfect low-dimensional

embedding rather than inherently unclassifiable target data. One solution might be to embed the data into a space higher than two dimensions, but the current $dissimilarity_j$ optimization routine is just an exhaustive search, which becomes quickly intractable. Even worse, because high-dimensional rotation matrices don't necessarily have simple parametrizations like in the two-dimensional case, an exhaustive search over variables can't even be done most of the time. In the next instantiation of this technique, we will give an example of how to find higher-dimensional rotation matrices.

MEG Data II

Allowing the data to be embedded in a higher-dimensional space creates several immediate problems. First, high (greater than 3) order rotation matrices don't have a known simple parametrized form like in the two-dimensional case. We'd like to be able to use gradient descent on these parameters to minimize dissimilarity, but without having a parametrized rotation matrix form that could be generalized to higher dimensions, this isn't possible. We could instead perform gradient descent by just thinking of a rotation matrix R as a point in d^2 dimensions, approximating the gradient as

$$\nabla_R \, disagreement \simeq \frac{disagreement(R+\delta) - disareement(R-\delta)}{2\delta}$$

for some small δ, but then there would be nothing forcing R to actually be a rotation matrix. In this section, we will solve this optimization problem by leveraging a result from a similar problem.

Wahba's problem [6], which comes up in many computer-vision tasks, aims to find the rotation matrix that describes one set of points rotated with respect to another. The problem is to find the R that minimizes

$$\sum_{k=1}^{n} \|w_k - Rv_k\|^2,$$

where w_k and v_k are vectors rotated with respect to each other in arbitrary dimensions. Wahba's problem has a surprisingly simple solution [7], outlined below.

$$B = \sum_{k=1}^{n} w_i v_i^T$$

By singular value decomposition,

$$B = USV^T.$$

The optimal rotation matrix R^* is

$$R^* = UMV^T,$$

where $M = diag([1, 1, ..., det(U)det(V)])$. We will denote this solution to Wahba's problem as $\omega(X1, X2)$, which we will take to output some matrix R such that $RX2 \simeq X1$. This formulation will allow us to perform gradient descent to minimize $disagreement_j$. As stated before, we would like to be able to just think of the matrix we're solving for as a point in d^2, but performing gradient descent (or any optimization procedure for that matter) on $disagreement_j$ is unlikely to produce a rotation matrix (a member of the special orthogonal group $SO(d)$). To resolve this, we define

$$G(A) = \omega(A\tilde{X}_t, \tilde{X}_t)$$

Where A is an arbitrary dxd matrix. We could think of this as solving for a rotation matrix, $G(A)$, that minimizes the squared distance between $G(A)\tilde{X}_t$ and $A\tilde{X}_t$, taking an arbitrary matrix A and correcting it into a rotation matrix. Using this formulation, we minimize the function

$$OPT_j(A) = \sum_{i=1}^{n} (h(x_i) - f_j(G(A)\tilde{x}_i))^2$$

as an alternative to $disagreement_j$. Because A is an arbitrary dxd matrix, and doesn't have to embody any particular properties, we could perform simple gradient descent to find its optimal value A_j^*. Given this A_j^*, we are assuming that

$$R_j^* = \arg\min_R \{disagreement_j(R)\} = G(\arg\min_A \{OPT_j(A)\} = G(A^*).$$

Using this method, we can obtain updated results, shown in table 2. In this experiment f_j and h are, like in the previous example, both feedforward neural networks with a single hidden layer. However, unlike in the previous experiment, we are reducing the dimensionality to five with PCA rather than to two with t-SNE.

Classification Error Summary			
S01: .20	S02: .31	S03: .30	S04: .08
S05: .21	S06: .33	S07: .26	S08: .23
S09: .15	S10: .17	S11: .44	S12: .16
S13: .28	S14: .12	S15: .23	S16: .24
Overall Error: .23			

Table 2: An updated per-subject breakdown of classification error rate

Spam Classification

In this section, we will apply our algorithm in a more common space - spam classification. For this problem, we have inboxes from several different people, and want to use these sources to create a custom spam filter for a user unseen during training.

This dataset was obtained from the 2006 European Conference in Machine Learning meeting, and contains data from 15 different inboxes. Each email sample is represented according to a bag-of-words and each inbox contains 400 emails. This problem was selected because, as shown in figure 5, this is not obviously a transfer learning problem. Indeed, a classifier trained naively of all source subjects (which is exactly the same as h) in aggregate will perform well, acieving an error rate of .15.

Still, as demonstrated in table 3, our algorithm is able to improve this result to .12. In this experiment, f_j and h are again neural networks, and we are reducing dimensionality to five using t-SNE.

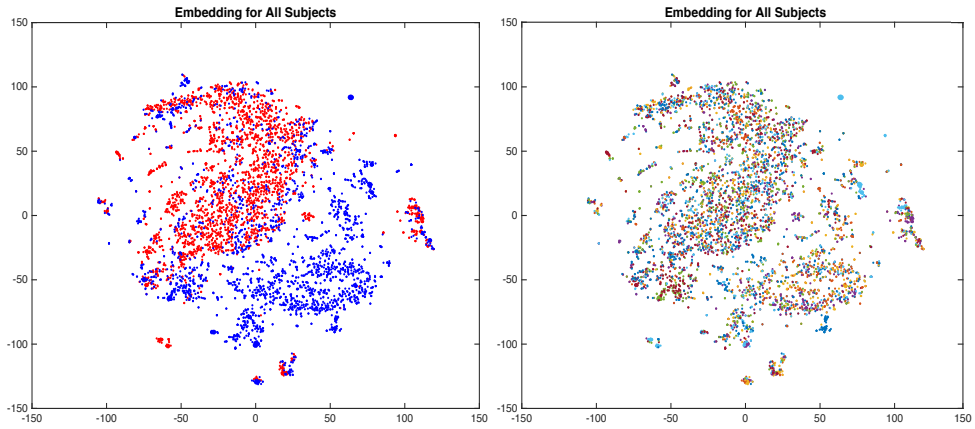

Figure 5: A two-dimensional embedding of the spam dataset as computed by t-SNE. On the left, samples are colored according to their label. On the right, they're colored according to the source from which they come.

Classification Error Summary				
I01: .06	S02: .07	I03: .14	I04: .09	I05: .18
I06: .19	S07: .14	I08: .10	I09: .08	I10: .04
I11: .15	S12: .15	I13: .07	I14: .18	I15: .13
Overall Error: .12				

Table 3: A per-inbox breakdown of classification error rate

Discussion

As mentioned earlier, we will be comparing our technique to covariate shift, stacked generalization, and a combination of the two, because that is what was used in [1]. For the MEG experiment, [1], rather than using a representation produced by the CSP algorithm, just used the high-pass filtered raw data to create a feature space. This space produces data that is more difficult to classify, but has more overlap between the subjects. They obtained hold-out classification errors of .38, .35, and .33 for covariate shift, stacked generalization, and the two combined respectively.

Because the CSP algorithm uses labels from each of the source subjects, it produces a space that is atypically extreme for transfer learning. Again, as seen in figure 1, there is essentially no geometric overlap of the subjects. Consequently, in this modified feature space, the covariate shift instance weighting ratio, $\frac{P_{\sim\mathcal{D}(X_t)}(x_{s,i})}{P_{\sim\mathcal{D}(X_s)}(x_{s,i})}$, collapses to near-zero for all $x_{s,i}$. So, the source instances are weighed almost uniformly, removing any gain we might get from using this method. Using a neural network as our learner, covariate shift does no better than guessing on this problem. Stacked generalization doesn't perform better than guessing in this representation either, probably also because the datasets don't overlap at all.

One major triumph of our algorithm is its ability to triumph in these particularly hostile settings. Figure 6 shows three source-target pairs before and after rotational alignment, demonstrating the algorithm's ability to reduce disorder even when data seems initially inseparable.

Both our first and second MEG classification results were superior to what was published in [1]. The online Kaggle competition first-place winner had an error rate of .25, which we were able to outperform in our second MEG data classification, but not our first.

On the spam classification problem, a covariate shift trained neural network can achieve .14 error, slightly lower than its non-covariate shift counterpart. Stacked generalization, also using neural networks, is able to produce an error rate of .12 (which is the same as our algorithm), but there is no improvement by combining stacked generalization with covariate shift.

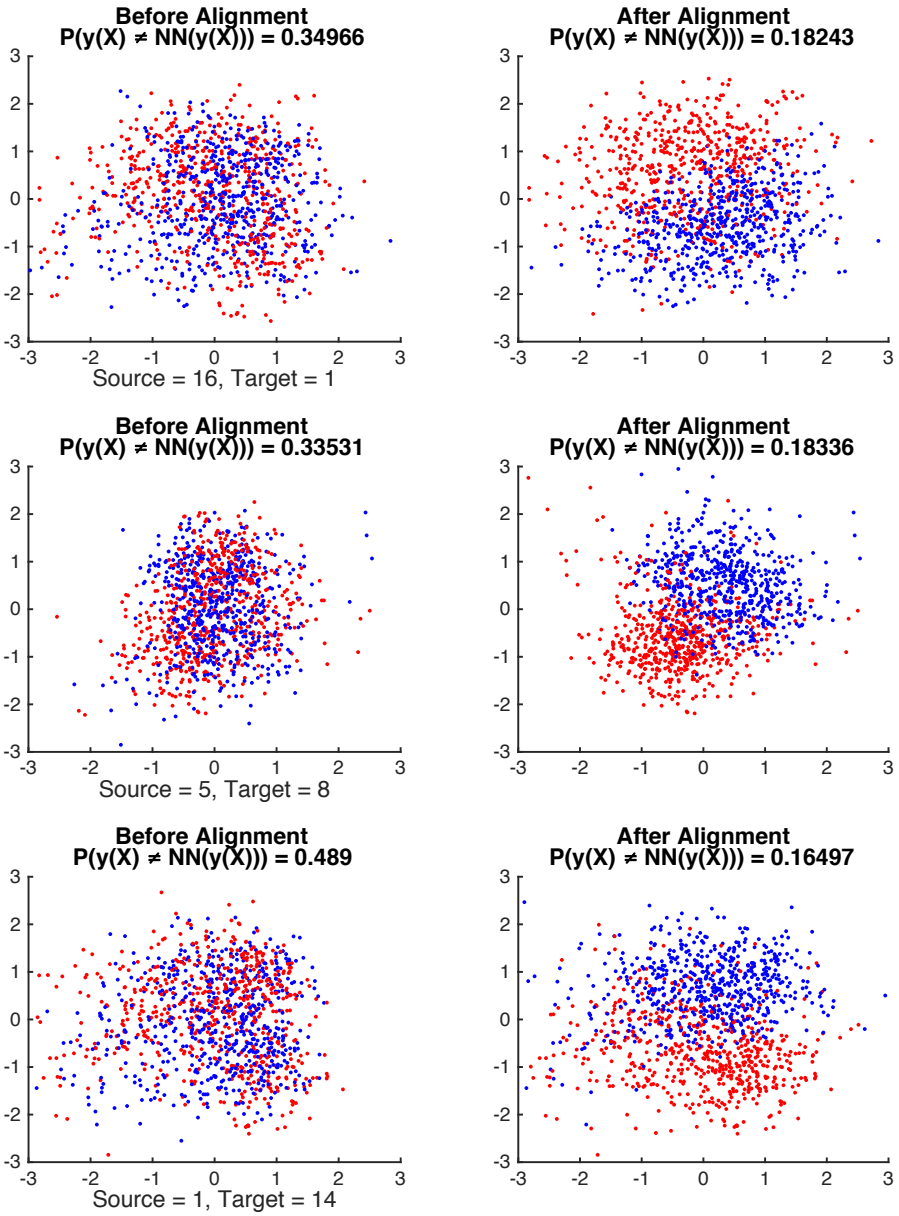

Figure 6: Three source-target pairs before and after rotational alignment. This example shows the first two principal components of the data as described in the second MEG experiment.

Conclusions

There are several big assumptions made by our algorithm. First, we're assuming that the high-dimensional classifier, h, is able to uncover enough global structure in the data to do better than guessing on the target data. If this is violated, we can't expect to be able to align the data well.

The amount of global structure available to h is really a function of the number of sources available and how similar the datasets are to one another. The MEG data, for example, has sources that are very dissimilar, but there are enough of them for h to still uncover some structure. On the other hand, in the spam experiment, the datasets are similar enough that h would perform better than guessing on a target even if there was only one source available.

Second, we assume that the low-dimensional classifier trained on a fixed source subject, f_j, is the right shape, meaning that there is some way to orient the target data such that f_j classifies it well. Of course, if this is violated, the algorithm probably won't be able to perform.

One advantage to our procedure is the flexibility of the classifiers and dimensionality reduction technique used. For example, it might be useful to improve h by allowing it to be trained according to stacked generalization, covariate shift, or any other transfer learning procedure.

Our technique lifts out unwanted variance in the data caused by differing datasets by finding a rotational transformation, which doesn't disturb the pairwise sample relationships in either the source or the target data. Overall, the presented algorithm offers an intuitive and powerful solution to transductive transfer learning.

Bibliography

[1] Olivetti E., Kia S.M., and Avesani P. MEG decoding across subjects. In *Pattern Recognition in Neuroimaging, 2014 International Workshop on*, pages 1–4, 2014.

[2] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[3] D.H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[4] Z.J. Koles, M.S. Lazar, and StevenZ. Zhou. Spatial patterns underlying population differences in the background eeg. *Brain Topography*, 2(4):275–284, 1990.

[5] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[6] J.L. Crassidis, R. Alonso, and Junkins J.L. Optimal attitude and position determination from line-of-sight measurements. *Journal of Astronautical Sciences*, 48(2):391–408, 2000.

[7] F.L. Markley. Attitude determination using vector observations and the singular value decomposition. *The Journal of the Astronautical Sciences*, 36(3):245–258, 1988.