Dual Degree Project Report

Learning multiple diverse classifiers for adapting to source-target distribution shift

Akshay Khare
M.Tech in Data Science
B.Tech in Ocean Engineering
Indian Institute of Technology Madras, Chennai 600036

Under the guidance of:
Dr. Harish Guruprasad Ramaswamy
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai 600036

March 6, 2020

1 Abstract

When the training data is an unbiased sample of an underlying distribution that is also followed by the test data, standard machine learning models perform well on the test data. However, if the training data is a biased sample, standard models struggle to generalize and do not perform well on the test data. In many real world applications, it is difficult to have access to labelled data. This necessitates the use of available data that is of a similar nature, but which suffers from a shift in the data distribution. The fields *domain adaptation* and *transfer learning* aim to help models generalize by overcoming the gap in the data distribution between train and test domains. Many successful approaches for domain adaptation so far require access to unlabelled data from the test distribution. However, availability of such labelled data is infeasible in many real world applications. We aim to develop an approach that does not require access to data from the test domain at all. Restricting our discussion at present to classification problems, our idea is to develop multiple "diverse" classifiers, each of which does well on the train data. We expect that at least one out of the multiple trained classifiers performs well on the test data. In other words, at least one of the multiple classifiers learns the task of interest. The way we develop "diverse" classifiers is by encouraging them to be orthogonal to each other. This work explores various ways of training multiple diverse classifiers and investigates their performance on different types of data that suffer from a shift between the train (source) and test (target) data distributions.

2 Introduction

Standard machine learning methods to learn the desired attributes are prone to learning the wrong attribute, i.e. attributes that are correlated with the attribute of interest in the training data, instead of the actual attribute of interest (Jayaraman, Sha, & Grauman, 2014). For instance, consider a dataset consisting of images of leaves. Here, the task is to classify the images into decayed or healthy leaves. The training data collected has a peculiar characteristic, that may not necessarily always hold true, that all the decayed leaf images are photographed outdoors and the healthy ones are photographed indoors, causing illumination difference in both the classes. Thus, achieving high training accuracy is no indication that the classification model has learned the task of distinguishing decayed leaves from healthy ones. The model may take an easier way out and end up learning the task of classifying based on difference in illumination. This can be seen by using a test set where the relation between healthiness of the leaves and illumination level, as present in the training set, does not hold. Here is a typical example of a healthy and decayed leaf in such a biased training data.

(a) Typical example of a decayed leaf in the training data

(b) Typical example of a healthy leaf in the training data

The challenge is that standard classification models can associate an attribute with any direction in the feature space that happens to separate the positive and negative instances in the training data. This often results in learning attributes that are correlated with the attribute of interest (Jayaraman et al., 2014). If the training data is an unbiased sample of an underlying distribution that even the test data follows, then the learned classification function performs well on the test data. However, if the training data consists of biased samples, standard machine leaning classifiers struggle to do well on the test data. *Domain adaptation* and *transfer learning* are the fields in machine learning which aim to bridge this gap in the distribution of the observed and actual data.

Deep neural networks have achieved state-of-the-art performance in a wide variety of application areas. But, these leaps in performance rest on the basis of availability of huge amount of labelled training data. For problems lacking labelled data, it may be still possible to obtain training sets that are big enough for training large-scale deep models, but that suffer from a shift in data distribution from the actual data encountered at test time. For instance, suppose we want our model to perform object detection on real world traffic videos. However, we do not have access to labelled real world traffic videos to train our model on. In this situation, we might consider using labelled, but synthetic videos that can be obtained from other sources, say from the field of computer games.
There have been successful attempts to build models that overcome this data distribution change between the train and test data. However, such approaches require access to data from the test distribution, labelled or unlabelled. What if data from the test distribution is unavailable? For instance, suppose a fruit-grower would like to use a pre-trained model that helps him distinguish decayed leaves from healthy ones. It would be impractical in such a case to access the images from test data distribution every time before making predictions on new leaf images. In this study, we aim to develop an approach that does not require access to data from the test distribution at all.

3 Literature review

3.1 On domain adaptation and transfer learning

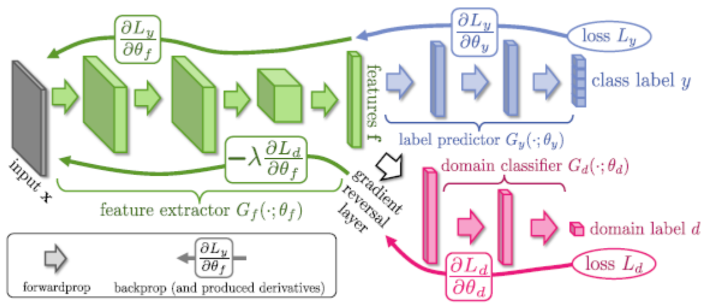
The aim of domain adaptation and transfer learning is to develop models that can generalize in spite of learning from 'biased' training data. A domain is defined by a combination of 3 parts, the input space, the output space and the associated probability distribution. The domain of the population of interest, that we want to make predictions on is called *target domain.* Whereas, the domain of the available labelled training data is called the *source domain.* Inputs are subsets of the N-dimensional real space $\mathbb{R}^N$. Outputs belong to $\mathbb{R}$ in case of regression problems or are categorical in case of classification settings ({-1, 1} for the binary case). A domain change refers to the change in at least one of the three components, namely the input space, output space and the probability distribution (Ben-David et al., 2010). Transfer learning refers to the general case where the domains are free to vary in any of the three components. Domain adaptation is defined as a particular case where the input and output spaces remain the same but the associated distribution changes (Ben-David et al., 2010).

There have been attempts to perform domain adaptation by re-weighting training samples (Chen et al., 2015). In this approach, the re-sampling weights are directly inferred by matching the distribution between the training and test sets in feature space in a non-parametric manner.

There are approaches that explicitly perform feature transformation that maps the source distribution to the target distribution (Pan et al). This approach proposes a method called transfer component analysis. Transfer component analysis learns some transfer components across domains in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD). In the subspace spanned by these transfer components, data distributions in different domains are close to each other.
Some other approaches are highlighted below.

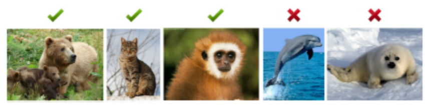
3.2 "Adversarial" domain adaptation

There are approaches that attempt to match the feature space distributions by modifying the feature representation itself instead of re-weighting or geometric transformation (Ganin & Lempitsky, 2015), (Tzeng, Hoffman, Saenko, & Darrell, 2017). The approach by (Ganin & Lempitsky, 2015) makes use of both the available labelled data from the source distribution as well as the unlabelled data from the target distribution. This approach focuses on combining domain adaptation and deep feature learning within one training process (deep domain adaptation). The goal is to embed domain adaptation into the process of learning representation, so that the final classification decisions are made based on features that are both discriminative and invariant to the change of domains, i.e. have the same or very similar distributions in the source and the target domains. In this way, the obtained feed-forward network can be applicable to the target domain without being hindered by the shift between the two domains.


$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

3.3 Regularization based approaches

Approaches by (Jayaraman et al., 2014), (Jayaraman & Grauman, 2014) employ regularization to make models generalize in spite of biased training data. The approach by (Jayaraman et al., 2014) avoids learning correlated attributes by attempting to decorrelate semantic visual attributes by resisting the urge to share features. Methods for attribute learning typically follow the standard discriminative learning pipeline that has been successful in visual recognition problems. Using training images labelled by the attributes they exhibit, low-level image descriptors are extracted, and used to independently train a discriminative classifier for each attribute in isolation. The problem with this is that this standard approach is prone to learning image properties that are correlated with the attribute of interest, rather than the attribute itself. Suppose you are tasked with learning the attribute present in the first three images, but absent in the others. There are many plausible hypotheses for the attribute: brown? furry? has-ears? land-dwelling? Standard methods



attempting to learn "furry" from such images are prone to learn "brown" instead, or some combination of correlated properties. This issue is worsened by the fact that different visual attributes may occupy the same spatial region in an image. For instance, a "brown" object may also be "round" and "shiny". Thus, an image search user querying "furry" would be very much frustrated if the system return images that are "brown" instead due to training data correlations.
This work aims to decorrelate attributes at the time of learning by proposing a multi-task learning framework that encourages each attribute classifier to use a disjoint set of image features to make its predictions. This idea of feature competition is central to this approach. Whereas conventional models train each attribute classifier independently, and therefore are prone to re-using image features for correlated attributes, this multi-task approach resists the urge to share. Moreover, since some attributes naturally should share features, we leverage side information about the attributes' semantic relatedness to encourage feature sharing among closely related properties (e.g., reflecting that "red" and "brown" are likely to share).
This method takes as input images labelled according to the presence/absence of each attribute, as well as a set of attribute "groups" reflecting those that are mutually semantically related. As output, it produces one binary classifier for each attribute. Attributes in the same group are encouraged to share low-level feature dimensions, while unrelated attributes compete for them. The key to this approach is to jointly learn all attributes in a vocabulary, while enforcing a structured sparsity prior that aligns feature sharing patterns with semantically close attributes and feature competition with semantically distant ones.

3.4 Learning diverse classifiers

The work by (Ross, Pan, & Doshi-Velez, 2018) is closely related to our idea to develop multiple diverse classifiers. If there exist multiple accurate classifiers for a given dataset, standard machine learning algorithms are likely to discover s complex model that combines them. This work attempts to recover maximal set of distinct but accurate models for a dataset such that these models are simpler and more interpretable than the more complex ones. The outputs of this ensemble of diverse models are statistically independent over small perturbations of inputs. This work is also related to methods of creating ensembles that encourage diversity during training. Typically these methods obtain diverse ensembles by training models on different datasets or different weightings of the same dataset as in boosting (Schapire, 1999), bagging (Breiman, 1996), or explicitly incorporating a term in the loss function that encourages diversity of training predictions, as in Negative Correlation Learning (Liu & Yao, 1999).

4 Problem statement

The goal is develop an approach that does not require access to data from the test distribution and still perform well at test time by training a model on data of a similar nature, but with a shifted data distribution. Suppose we want to learn a particular attribute of interest, for instance, differentiating between odd and even digit images, or distinguishing animals with or without fur, or classifying leaf images into decayed or healthy, etc. However, the data we have at hand has characteristics of multiple different attributes, one of which is of our interest. Thus, training a single model using standard classification algorithms is no guarantee that the model will learn the attribute of interest.
Our idea is to develop multiple "diverse" classifiers on the available training data such that each classifier performs well on the training data, and at the same time learns qualitatively different aspects of the training data. By training multiple classifiers, we expect that there exists a classifier among these multiple trained classifiers that performs well on both the training data as well as the test data, or in other words, learns the attribute of interest.

5 Current approach

Our idea is to develop multiple diverse classifiers, such that each classifier learns qualitatively different aspects of the training data. The way we encourage the multiple classifiers to be different is by making them orthogonal to each other. We attempt to achieve this orthogonality by modifying the loss function in a way that encourages the training process to learn orthogonal classifiers. We do this modification in 2 ways. One method is by simply adding a constant multiple of the standard inner product of the weight vectors to the loss function. Another way is by using a "spatially-aware" inner product instead of the standard inner product. Such an inner product, in case of image data, encourages the different classifiers to focus on spatially distant aspects in the images.
Let us suppose we want to train n diverse classifiers. We treat n as a hyper-parameter that is not learned during training. At present, we restrict our discussion to binary classifiers and a linear model (logistic regression). Let L be the binary cross entropy loss function corresponding to the loss of a single logistic regression classifier. Let the weight vectors of the n classifiers be denoted by W_i where $i \in \{1, 2, ...n\}$ and $W_i \in \mathbb{R}^{N \times 1}$, where N is the dimension of the feature space. Modifying the loss function using standard inner product to learn diverse classifiers is done as follows:

$$\min_{W_1,\dots W_n} \sum_{i=1}^{n} L(W_i) + \lambda_1 \sum_{i=1}^{n} \sum_{j=i+1}^{n} (W_i^T W_j)^2 + \lambda_2 \sum_{i=1}^{n} \|W_i\|_2$$

Modifying the loss function using "spatially-aware" inner product to learn diverse classifiers is done as follows:

$$\min_{W_1,\dots W_n} \sum_{i=1}^{n} L(W_i) + \lambda_1 \sum_{i=1}^{n} \sum_{j=i+1}^{n} (W_i^T M W_j)^2 + \lambda_2 \sum_{i=1}^{n} \|W_i\|_2$$

The data we use for our experiments has images with dimension of 28 $\times$ 28. Thus, the matrix M has dimension of 784 $\times$ 784. Each element in M is associated with 2 pixels in an image. To encourage focus on spatially distant aspects of an image, an element of M has a large value if the associated 2 pixels are close to each other. We first create a tensor M of size 28 $\times$ 28 $\times$ 28 $\times$ 28 as follows.

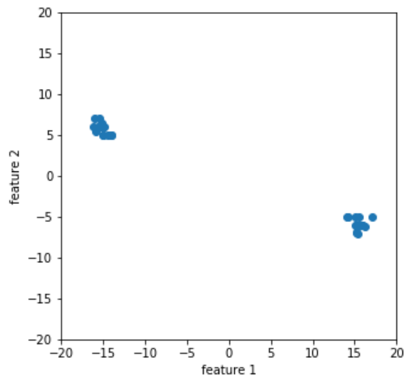
$$M[i_1, i_2, j_1, j_2] = e^{\frac{-((i_1-j_1)^2+(i_2-j_2)^2)}{2\sigma^2}}$$

where $i_1, i_2, j_1, j_2 \in \{0, 1, 2, ...27\}$. M is then reshaped to 784 $\times$ 784.
The idea behind using the "spatially-aware" inner product can be understood more clearly from the following example.
Let us suppose, for the want of simplicity, that our image is of dimension 10 $\times$ 10. In the first case, the learned weight vectors corresponding to the two classifiers are $W_1 = [1, 0, 0, ...0]$ and $W_2 = [0, 1, 0, ...0]$. In the second case, let the learned vectors for the 2 classifiers be $W_1 = [1, 0, 0, ...0]$ and $W_2 = [0, 0, 0, ...1]$.
As we can see, the learned classifiers in the first case learn spatially very close aspects of the image as compared to the second classifier. However, in both the cases the standard inner product has the value 0. Since we want the models to focus on spatially distant parts of images, the "spatially-aware" inner product is a better choice since its value in the first case (0.9950) is higher than the value in the second case (0.4449) These two values are obtained assuming $\sigma = 10$.

6 Experiments to learn multiple diverse classifiers by modifying loss function using inner products

6.1 Experiment 1

In order to assess the validity of our approach to learn multiple orthogonal classifiers, we first experiment on a simple toy dataset. The two dimensional toy datasets is shown below.

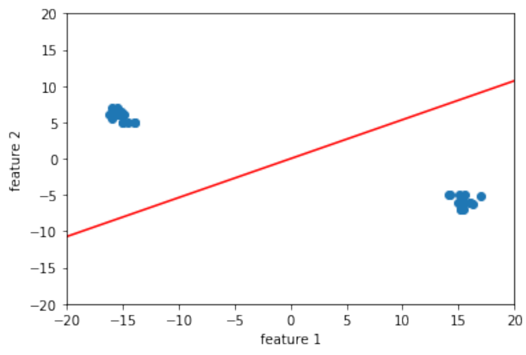


For different combinations of number of classifiers and regularization, we expect the model to behave in the following manner.

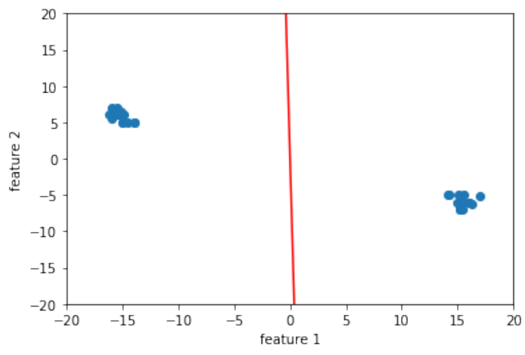
- **1 classifier, no regularization**: In this case, we expect the model to learn a slanting decision boundary where it gives importance to both the features for the classification task. Since there are no constraints, we expect the model to learn the slanting decision boundary because it separates the data by a maximum margin.

- **1 classifier with regularization**: In this case, we expect the model to give more importance to feature 1. This is because the model faces a trade-off between learning the maximum margin and shrinking the magnitude of the weight vector. Hence, we expect the weight corresponding to feature 2 to be much smaller as compared to that corresponding to feature 1. This is because classification based on feature 1 provides a larger margin than classification based on margin 2.

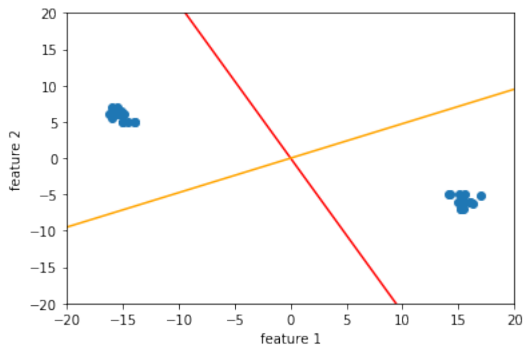
- **2 classifiers with orthogonality constraint and no regularization**: In this case, we expect the 2 classifier weight vectors to be orthogonal to each other and at the same time have a high margin for classification. Thus, learning two slanting and orthogonal boundaries satisfies these requirements.
- **2 classifiers with orthogonality constraint and L2 regularization**: In this case, the models not only need to be orthogonal, but also need to be parsimonious in terms of magnitudes of the weight vectors. Hence, we expect the two classifiers to learn the horizontal and vertical decision boundaries.



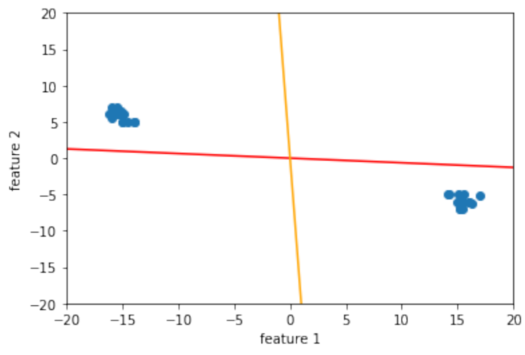
(a) 1 classifier without regularization



(b) 1 classifier with L1 regularization



(a) 2 orthogonal classifiers without regularization



(b) 2 orthogonal classifiers with L2 regularization

6.2 Experiment 2

In this experiment, the task is to distinguish odd digits from even digits. We use the MNIST data for this. To create the biased source data, we add a white patch to the top left corner for even digits and a white patch to the top right corner for odd digits. Before adding the patches, we make the MNIST digits harder to learn by adding Gaussian noise to the original MNIST images. This is because we find out that the original MNIST data is too simple to learn since even a single classifier learns both the illumination pattern as well as the digit pattern well. This defeats our objective to learn diverse classifiers. Hence, we make the MNIST digits pattern harder to learn.

- Source data: MNIST (with Gaussian noise of standard deviation 100) with patches

- Target data: MNIST (with Gaussian noise of standard deviation 100) without patches

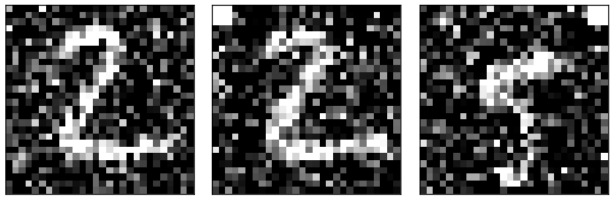


Figure 1: Source and target domain images for this experiment

We train a single classifier on the source data and check its performance on the target data to get a baseline performance. Such a classifier achieves an accuracy of **100%** on the source data and an accuracy of **75%** on the target data. This indicates that the model primarily learns to classify images based on patches pattern. This is confirmed by visualising the learned weight.

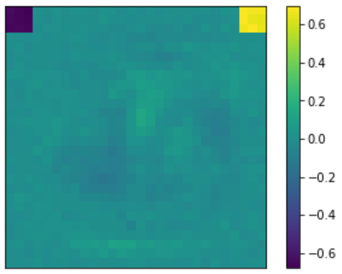


Figure 2: Visualising the learned weight when a single classifier is trained on source domain

We now train 4 orthogonal classifiers on the source data and test their performance on the target data. Each of the classifier achieves a training accuracy of **100%** and a testing accuracy of **76%**, **75%**, **73%**, **75%** respectively. The classifiers take an easy way out and learn the patches pattern instead of the digits pattern.

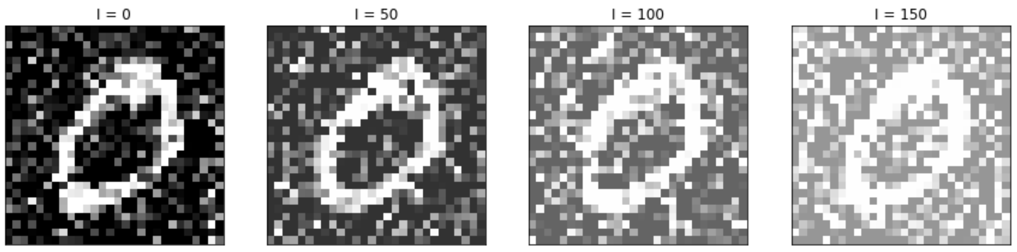
C1-C2	**C1-C3**	**C1-C4**	**C2-C3**	**C2-C4**	**C3-C4**
0.455	0.463	0.448	0.470	0.469	0.463

Table 1: Pairwise cos of the angle between the weight vectors

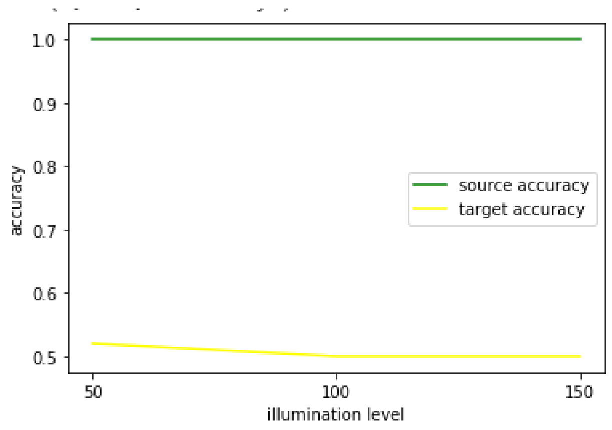
Thus, for the source target shift using patches, developing multiple orthogonal classifiers does not improve the model performance over training a single model.

6.3 Experiment 3

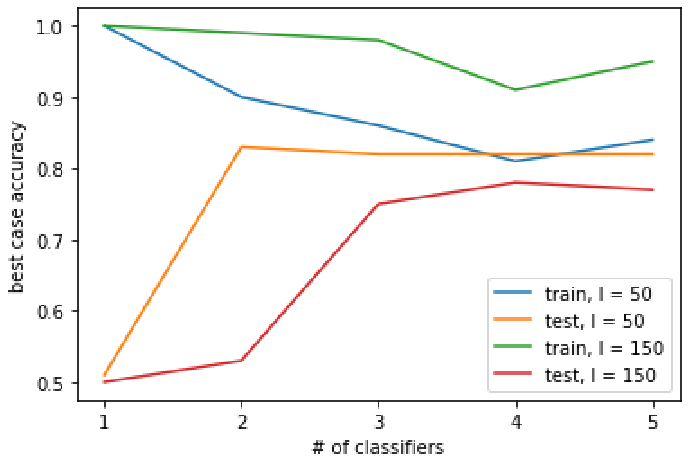
In this experiment, instead of biasing the data with the help of left and right patches, we generate a bias using illumination changes. The images with even digits are made brighter than the images with odd digits. This is simply achieved by increasing the pixel values in the image with even digits by a fixed number. We experiment with 3 values of this fixed number (I) - 50, 100, 150. The images on the newly formed dataset look as follows.



Note, now the dataset with class-wise changed illumination is the source data and the data with no illumination changes is the target data. To gauge the effect of adding the illumination on classifier performance, we train a single linear classifier on the source data and test it on the target data. The results are as follows. Thus, as is clear from the above figure that the single linear model trained is not able to learn the task of



distinguishing between even and odd digits.
Now to test performance of our model, we train 4 linear orthogonal classifiers on the source data and test these trained models on both the source and target datasets. The results are shown in the figure below.
On the x axis, we plot the number of classifiers trained, and on the y axis we plot the best case train and test accuracies among the multiple classifiers trained.



A few interesting observations from the above plot are as follows.

- The best case accuracy on the target domain increases initially and then becomes stagnant as the number of classifiers increases.
- As the best case target accuracy increases, the best case source accuracy decreases.
- For a given value of the number of classifiers developed, the best case target accuracy in case of $I = 150$ is less than that for $I = 50$. We expect this since the alternative attribute of illumination is much stronger in the former case than the latter.

Training multiple diverse classifiers achieves a significant improvement in performance on the target domain as compared to training a single model.
Thus, for the source target shift with illumination changes, we see that our approach of training multiple models achieves a significantly improved performance on the target domain over a single model trained on the source domain.

7 Future work

- The next step in our work is to graduate to non-linear, deep models. Here, our idea is to replace the weight vectors used in the inner product term in the modified loss function by gradients of the output with respect to the inputs.
- We intend to try out other ways of developing "diverse" classifiers using metrics such as conicity (Chandrahas, Sharma, & Talukdar, 2018) and though negative correlation learning (Liu & Yao, 1999).
- We aim to investigate effect of regularization such as drop-out in developing multiple diverse classifiers.

8 Training multiple diverse deep models

We now graduate from linear model to a deep model. We consider a Convolutional Neural Network (CNN) with a single convolution layer with single or multiple filters, and a single fully connected hidden layer.

8.1 Comparative analysis of inner product inspired vs conicity inspired diversity

In this experiment, we train multiple diverse classifiers on the MNIST data with added patch bias as explained in Experiment 2. We encourage diversity by two different ways, namely, the standard inner product

and conicity. Training diverse classifiers by modifying the loss function using standard inner product is explained in detail in section 5.
Before formulating conicity, let us first understand the metric 'alignment to mean' (ATM). The alignment to mean of a vector $\mathbf{v}$ belonging to a set of vectors $\mathbf{V}$ is defined as the cosine similarity between $\mathbf{v}$ and mean of all vectors in $\mathbf{V}$.

$$\mathrm{ATM}(\mathbf{v}, \mathbf{V}) = \mathrm{cosine}(\mathbf{v}, \frac{1}{|\mathbf{V}|} \sum_{x \in \mathbf{V}} x)$$

Conicity is defined as the mean ATM of all vectors in $\mathbf{V}$.

$$\mathrm{Conicity}(\mathbf{V}) = \frac{1}{|\mathbf{V}|} \sum_{\mathbf{v} \in \mathbf{V}} \mathrm{ATM}(\mathbf{v}, \mathbf{V})$$

We train multiple diverse classifiers by encouraging diversity using conicity as follows. Conicity of a set of vectors achieves a maximum value of 1 when all vectors are aligned in the same direction. The value of conicity decreases as the vectors start to spread out.

Let us suppose we want to train n diverse classifiers. We treat n as a hyper-parameter that is not learned during training. Considering the case of a Convolutional Neural Network (CNN) classifier, let L be the binary cross entropy loss function corresponding to the loss of a CNN classifier. Let the weight vectors of the n classifiers be denoted by W_i where $i \in \{1, 2, ...n\}$ and $W_i \in \mathbb{R}^{N \times 1}$, where N is the dimension of the feature space. Let all the weight vectors W_i belong to the set W. Modifying the loss function using conicity to learn diverse classifiers is done as follows:

$$\min_{W_1,...W_n} \sum_{i=1}^{n} L(W_i) + \lambda\, \mathit{conicity}(W)$$

The variation in accuracy with the number of classifiers trained for the two approaches to encourage diversity is shown in the plot below. We see from the above plot that encouraging diversity using conicity

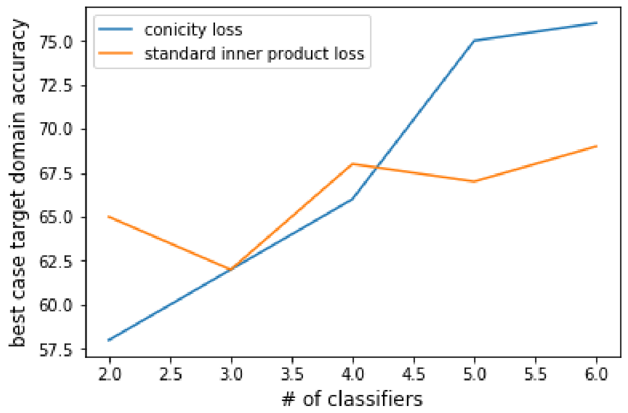


Figure 3: Comparing the best case target domain accuracy between conicity and standard inner product based diversity

gives better results in general over using standard inner product to encourage diversity.

In all of the below experiments, the task is to classify odd vs even digits. The source data is obtained by biasing the MNIST dataset with added Gaussian noise using label dependent white patches as explained in section 6.2.

8.2 Experiment 4.1

The CNN architecture in this experiment consists of a single convolution layer with a single 5×5 filter and a fully connected layer.

8.3 Experiment 4.2

The CNN architecture in this experiment consists of a single convolution layer with a single 3×3 filter and a fully connected layer.

8.4 Experiment 4.3

The CNN architecture in this experiment consists of a single convolution layer with a single 7×7 filter and a fully connected layer.

8.5 Experiment 4.4

The CNN architecture in this experiment consists of a single convolution layer with two 7×7 filters and a fully connected layer.

Description of new data set. Until till now, we corrupted the data by addition of white patches at either the top left or top right corner of an image, the location of the patch determined by the label for the image. Now, we decide to experiment by inducing a different type of bias in the data. We add a randomly (uniform) located vertical (10×3) or horizontal (3×10) patch in the MNIST data with added Gaussian noise for a the case of an odd digit image or even digit image respectively.

8.6 Experiment 5.1

New data set. The CNN architecture in this experiment consists of a single convolution layer with a single 5×5 filter and a fully connected layer.

8.7 Experiment 5.3

New data set. The CNN architecture in this experiment consists of a single convolution layer with a single 3×3 filter and a fully connected layer.

8.8 Experiment 5.3

New data set. The CNN architecture in this experiment consists of a single convolution layer with a single 7×7 filter and a fully connected layer.

8.9 Experiment 5.4

New data set. The CNN architecture in this experiment consists of a single convolution layer with two 5×5 filters and a fully connected layer.

References

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010, May). A theory of learning from different domains. *Mach. Learn.*, *79*(1-2), 151–175. Retrieved from `https://doi.org/10.1007/s10994-009-5152-4` doi: 10.1007/s10994-009-5152-4

Breiman, L. (1996, Aug 01). Bagging predictors. *Machine Learning*, *24*(2), 123–140. Retrieved from `https://doi.org/10.1023/A:1018054314350` doi: 10.1023/A:1018054314350

Chandrahas, Sharma, A., & Talukdar, P. (2018, July). Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 122–131). Melbourne, Australia: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/P18-1012` doi: 10.18653/v1/P18-1012

Chen, Q., Huang, J., Feris, R., Brown, L. M., Dong, J., & Yan, S. (2015, June). Deep domain adaptation for describing people based on fine-grained clothing attributes. In *2015 ieee conference on computer vision and pattern recognition (cvpr)* (p. 5315-5324). doi: 10.1109/CVPR.2015.7299169

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd international conference on international conference on machine learning - volume 37* (pp. 1180–1189). JMLR.org. Retrieved from `http://dl.acm.org/citation.cfm?id=3045118.3045244`

Jayaraman, D., & Grauman, K. (2014). *Zero shot recognition with unreliable attributes.*

Jayaraman, D., Sha, F., & Grauman, K. (2014). Decorrelating Semantic Visual Attributes by Resisting the Urge to Share. In *Cvpr.*

Liu, Y., & Yao, X. (1999, December). Ensemble learning via negative correlation. *Neural Netw.*, *12*(10), 1399–1404. Retrieved from `http://dx.doi.org/10.1016/S0893-6080(99)00073-8` doi: 10.1016/S0893-6080(99)00073-8

Ross, A. S., Pan, W., & Doshi-Velez, F. (2018). *Learning qualitatively diverse and interpretable rules for classification.*

Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the 16th international joint conference on artificial intelligence - volume 2* (pp. 1401–1406). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from `http://dl.acm.org/citation.cfm?id=1624312.1624417`

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017, July). Adversarial discriminative domain adaptation. In *The ieee conference on computer vision and pattern recognition (cvpr).*