Classification Using Domain Adaptation

ABSTRACT

Domain Adaptation by definition is learning from a source data distribution, a well performing model on a different (but related) target data distribution.
I aim to analyze domain adaptation as a means to create classification models using a well labeled source domain and a sparsely labeled target domain, assuming both domains are closely related. One characteristic of such a problem is that the target domain and the source domain have different data distribution, hence a regular classifier might not perform as well as it would have on a dataset having similar distribution. A major chunk of work on this project comes from the paper "A Two-Stage Approach to Domain Adaptation for Statistical Classifiers" [1].

Tools Used

Python with Scikit learn, as it provides an additional advantage of focusing more on the task of Domain Adaptation rather than the actual classification.

Datasets Used

1. BBC News – Dataset categorizing news articles. [5]
2. 20Newsgroup – Collection of approximately 20,000 newsgroup documents. [6]

INTRODUCTION

In standard classification, the labeled data on which we train our classifier and the data on which we want to make predictions come from the same domain and thus share the same distribution. However, in reality we often face the situation where we want to make predictions on a different domain, but the labeled examples we have in our training might not encapsulate the distribution of the target domain.
For example, during our presentation I saw a lot of students working on spam email classification. Most public data available on this topic is very general, but generally we would like spam filters to work on our personal email inbox which might have a completely different set of spams than the general dataset.

To solve the domain adaptation problem, intuitively, we want to make the most use of the labeled examples without over-fitting the source domains. We also want to exploit the unlabeled examples in the target domain to possibly pick up patterns that do not exist in the source domains. Following this intuition, the paper [1] to addresses the domain adaptation problem in two steps:

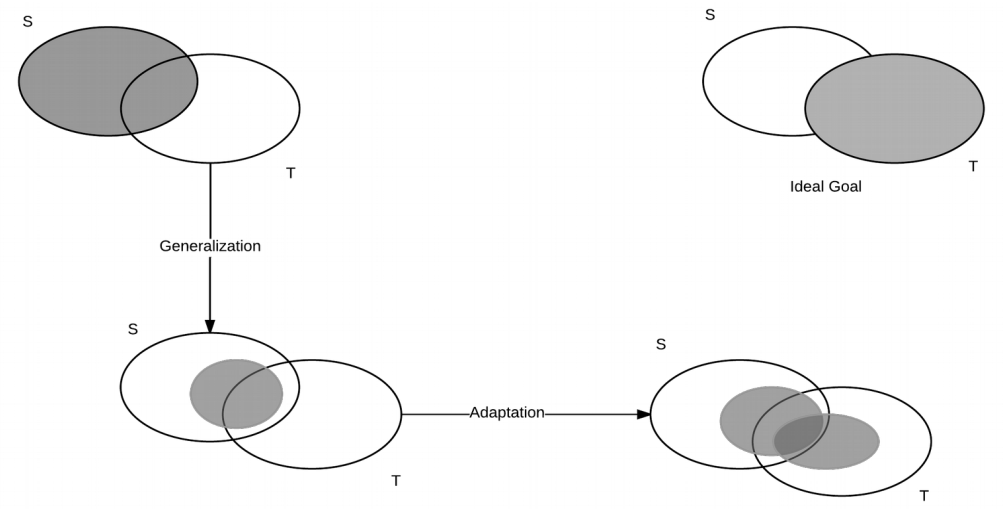
1. First find the common features that are important for both the source and the target domains.
2. Then identify useful features that are specific to the target domain but cannot be learned from the source domains, possibly through semi-supervised learning.

The paper proposed a two-stage approach to domain adaptation, where at the first stage we identify a set of generalizable features for the classification task and learn the appropriate weights for these generalizable features, and at the second stage, we pick up useful features specific for the target domain by using some examples from the target domain with pseudo labels.

My initial task was to find two datasets that would be clean and have some similarity among each other. Because there is a wide range of news data available I chose to go with this domain. My first choice for the dataset included BBC News as it had a data set which required virtually no preprocessing. Now that I had one dataset it was time to choose another which was similar to the BBC news dataset, I had heard a lot about the Reuters dataset and planned to use it but while processing the

dataset I found that this dataset had categories which I did not require further categories that I did want were not present. Next I found dataset by 20Newsgroup which essentially contains newsgroup documents partitioned into categories that are similar to the BBC News dataset.

THE TWO STAGE PROCESS



Domain Generalization

The first stage of our two-stage approach to domain adaptation is the domain generalization stage. We have shown that in order to make the model learned from the source domains useful in the target domain, we need to regularize the learning process such that the weight mass is mostly kept on the generalizable features.

There are two problems that need to be solved here:

Identifying the general features

As we choose source and target domains to be fairly similar we can be sure that there exists at-least some features that will be common among the two domains. The task here is to find those features.

Learning appropriate weights for these features

In standard supervised learning, we learn a single weight matrix w for all features using all the training examples. When the training data falls into several domains, however, we cannot expect the optimal weights to be the same for all domains. We thus introduce K different weight matrices, $\{w^k\}^K_{k=1}$, for the K training domains. In our case K=2.

However, since the generalizable features behave similarly in the two domains, we expect the weights for them to be similar across the domains. Hence there is no need to transform the weights for the different domain in our case.

Domain Adaptation

After the first stage of domain generalization, we will obtain a generalized feature vector, which represents the set of generalizable features learned from the source domains, as well as a set of weights for these generalizable features. In the second stage of domain adaptation, our goal is to pick up those features that are specifically useful for the target domain, but cannot be learned from the source domains. A sensible way to achieve this goal is to include some labeled instances from the target domain in the learning process.

Formally, let A' be the optimal feature selection matrix that we have obtained in the first domain generalization stage, and let $\{x^t_i , y'^t_i\}^m_{i=1}$ be the set of examples in the target domain that have been predicted with the highest probabilities $p(y'^t_i \mid x^t_i)$, where y'^t_i is the predicted label of x^t_i. Now we can use this new "adapted" domain to build our model to predict the documents in the target domain accurately.

ANALYSIS

The Categories I chose to work on are "Politics", "Sports" and "Technology". Although the 20Newsgroup dataset had further fine grained data like "sport.baseball" and "sport.hockey" I could easily group these together. Analyzing the datasets after cleaning a few notable features to mention are that surprisingly 20Newsgroup dataset had far more documents than BBC.

1. BBC Dataset size = 1329 documents
2. 20Newsgroup Dataset size = 13406 documents

After this information I chose 20Newsgroup as my source domain and BBC as my target domain. After doing a thorough read of the Literature Review on Domain Adaptation[2] and the paper Two stage approach to Domain Adaptation for statistical classifiers[2] I formulated the steps I would need to perform,

1. Extract features from source domain.
2. Train a classification model on the source domain, say A.
3. Run A on the target domain. Note results.
4. Extract features from target domain.
5. Find features common in source and domain, say F_common.
6. Use F_common as a feature set to train the same classification model on the source domain, say B.
7. Run B on target domain. Note results.
8. Take most confident predictions of B and move these target documents to the training set and add their features to F_common.
9. Train the same model using the new feature set on the source+confident target predictions as training set, say C.
10 Run C on the target domain. Note results.

As from my reading steps 5-6 is the domain generalization stage and steps 7-9 is the domain adaptation stage.
For my document classification model I selected unigram and bigram counts as feature sets.
I wanted to analyze the question, Does domain adaptation help me improve my accuracy, precision and recall? I needed a way to use different classification techniques and verify that my result is consistent, without having to change a lot in terms of code, to do this I incorporated Scikit-learn a python package which can be used to train and run various classification algorithms.
Scikit-learn provides a pipeline which allows us to set the features and the classification algorithm we want to use, we then provide this pipeline with the training data and train it. Next we predict on the target domain using the same pipeline. Sckit-learn also provides a nice feature that prints the performance metrics in a clean way.

IMPLEMENTATION

Classification Techniques Used
Logistic Regression with Stochastic Gradient Descent and Multinomial Naive Bayes

Feature Vector
Unigram and Bigram.

Source Domain
20Newsgroup

Target Domain
BBC News

Extracting features
We extract unigram and bigram counts as features and remove any stop words. We do this in scikit-learn using the code below

```
vectorizer =  CountVectorizer(ngram_range=(1, 2),stop_words='english')
X_source_train = vectorizer_source.fit_transform(data.data)
```

Line 1 initializes a CountVectorizer with required parameter and line 2 executes this on the data source.
We use this method to extract features fro both the source and the target domain.

Finding Common Feature Set
Python provides a Data Structure call set() using which we can easily to set operations like intersection and union

1. We get feature names for both source and target

```
feature_names_target = vectorizer_target.get_feature_names()
feature_names_source = vectorizer_source.get_feature_names()
```

2. Transform this list to a python set and perform an intersection operation

```
a = set(feature_names_target)
b = set(feature_names_source)
feature_names_common = list(a.intersection(b))
```

Train Model using the common feature set
1. Build a pipeline

```
text_clf = Pipeline([('vect', CountVectorizer(ngram_range=(1, 2),
                              vocabulary=feature_names_common,
                              stop_words='english')),
        ('clf',      classifier)
)]
```

2. Fit the source data, that is, the labeled data.

```
text_clf = text_clf.fit(data_source.data, data_source.target)
```

3. Use this model to predict the target domain documents

```
predicted = text_clf.predict(data_target.data)
```

Using the common feature set model get the top 10% documents in the Target domain that have high prediction score.
We pick the top 10% documents from the above model and add them to the source domain and update the features by including all features that exist in these documents.

Train a Model on this new feature set and new Source domain
Doing this makes our model more specific towards the target domain and hence provides better predictions on the target domain.

RESULTS

20Newsgroup Dataset size = 13406 documents
BBC Dataset size = 1329 documents

A. Training on 20Newsgroup dataset and predicting BBC News documents using feature only present in 20Newsgroup

# of Training Docs	13406
Size of feature vector	1453879
Accuracy	73.664409

	PRECISION	RECALL	F1-SCORE
POLITICS	0.56	1.00	0.72
SPORT	0.99	0.44	0.61
TECH	0.93	0.85	0.89
AVG/TOTAL	0.84	0.74	0.73

Here we can see that using a model trained only on the source domain gives an accuracy of only 73% which suggests the need of some way to improve on this.

B. Training on 20Newsgroup dataset and predicting BBC News documents using intersection of feature sets from 20Newsgroup and BBC News.

# of Training Docs	13406
Size of feature vector	39845
Accuracy	88.036117

	PRECISION	RECALL	F1-SCORE
POLITICS	0.75	1.00	0.86
SPORT	0.96	0.89	0.92
TECH	1.00	0.75	0.86
AVG/TOTAL	0.90	0.88	0.88

In the above results we can see using only a subset feature that exist in both domains drastically reduces my feature vector size but gives a better performance than using the complete feature vector set.

C. Training on 20Newsgroup dataset and predicting BBC News documents using intersection of feature sets from 20Newsgroup and BBC News + features from confidently predicted documents in step B.

# of Training Docs	13406 + 131
Size of feature vector	70537
Accuracy	90.067720

	PRECISION	RECALL	F1-SCORE
POLITICS	0.79	1.00	0.88
SPORT	0.96	0.83	0.94
TECH	1.00	0.77	0.87
AVG/TOTAL	0.92	0.90	0.90

The above results represent the performance after adapting features from the target domain to the source domain, as theorized the scores for this have increased hence validating the claims of the experiment.

In the next section we will verify that results drawn using Multinomial Naive Bayes classifier are consistent when using a different classification algorithm

Using Logistic Regression with stochastic gradient descent

A. Training on 20Newsgroup dataset and predicting BBC News documents using feature only present in 20Newsgroup

# of Training Docs	13406
Size of feature vector	1453879
Accuracy	78.555305

	PRECISION	RECALL	F1-SCORE
POLITICS	0.66	0.96	0.78
SPORT	0.88	0.89	0.88
TECH	0.92	0.48	0.63
AVG/TOTAL	0.82	0.79	0.77

B. Training on 20Newsgroup dataset and predicting BBC News documents using intersection of feature sets from 20Newsgroup and BBC News.

# of Training Docs	13406
Size of feature vector	39845
Accuracy	84.875847

	PRECISION	RECALL	F1-SCORE
POLITICS	0.78	0.95	0.86
SPORT	0.87	0.92	0.90
TECH	0.93	0.65	0.77
AVG/TOTAL	0.86	0.85	0.84

C. Training on 20Newsgroup dataset and predicting BBC News documents using intersection of feature sets from 20Newsgroup and BBC News + features from confidently predicted documents in step B.

# of Training Docs	13406
Size of feature vector	70537
Accuracy	86.531226

	PRECISION	RECALL	F1-SCORE
POLITICS	0.85	0.93	0.89
SPORT	0.88	0.89	0.88
TECH	0.86	0.77	0.81
AVG/TOTAL	0.87	0.87	0.86

We can see a similar trend when using Logistic Regression(MaxEnt), hence supporting the claim that adapting target domain features into the source and modifying our model results in better target specific prediction.

FUTURE WORK

I have only used two classifiers in my experiment, which can be increased to give us a better overall picture. Further reading, suggested various optimization techniques for weighting the feature vector based on the data distribution in the different domains, this could be incorporated in this experiment.

In my experiment, as I use only unigrams and bigrams as features taking a generalized feature set is as easy a intersection of sets. But for more complex features we need some method to find a good set of generalized features. In paper[1] there are two optimization techniques to do so, which I did not explore too much in detail, hence this could be something to work on in future.

Here I showed how domain adaptation compares to using regular classification techniques using 20Newsgroup and BBC News dataset. Further future work could also be exploring the same idea in a more generalized form.

CONCLUSION

In conclusion, we can say that domain adaptation can be a very useful technique when building classification models as seen by the results.

Further using domain adaptation we can create models from very large datasets and adapt them to counterparts that do not have such a huge labeled dataset, giving us better predictive results on the target domain. This can be especially useful in spam-ham email classification where we could use a generalized model and build target models based on user's spam emails.

REFERENCES

[1] A Two-Stage Approach to Domain Adaptation for Statistical Classifiers [Jing Jiang, ChengXiang Zhai] - https://pdfs.semanticscholar.org/85e1/b1db620d528bce1a74bf4bf63fef7b768e6f.pdf

[2] Frustratingly Easy Domain Adaptation - [Hal Daume´ III] - http://www.umiacs.umd.edu/~hal/docs/daume07easyadapt.pdf

[3] A Literature Survey on Domain Adaptation of Statistical Classifiers - [Jing Jiang] - http://www.mysmu.edu/faculty/jingjiang/papers/da_survey.pdf

[4] Literature Survey: Domain Adaptation Algorithms for Natural Language Processing - [Qi Li] http://nlp.cs.rpi.edu/paper/qisurvey.pdf

[5] BBC News Dataset- http://mlg.ucd.ie/files/datasets/bbc-fulltext.zip

[6] 20 Newsgroup Dataset - http://qwone.com/~jason/20Newsgroups/20news-19997.tar.gz