Transfer of Representation based on Dictionary Learning

Kha-Dinh Luong

I. DESCRIPTION OF ALGORITHMS

1) Overview: The algorithms that we investigated in this section, Self-taught Learning [2] and Multi-task Feature Learning [1], learn a dictionary to store interesting representations that can be applied across various learning tasks. Both are based on the assumptions that there are similar features across domains and it would be useful if we can represent them using some learned common features. The learned features need not be, and will often not be, a subset of the set of features in the domains, similar to the idea in dimensionality reduction methods. However, the ways that the algorithms define, learn, and apply the dictionaries are different.

Of the two algorithms that we looked at, only Self-taught Learning will be implemented and tested on various experiments.

2) Multi-task Feature Learning: Training a new model for a new learning task is time consuming. Many times, a new learning task is not much different from tasks we have already learned. The key idea behind this algorithm is that, given a set of similar learning tasks, we can learn them at the same time as a single training task. The algorithm accomplish that by learning a common representation dictionary for all the common tasks' features.

Assuming there are T learning tasks, each with an associated concept function f_t, then, assuming that the tasks share some common features, each function f_t can be written as:

$$f_t(x) = \sum_{i=1}^{d} a_{it} h_i(x),\ t \in \mathbb{N}_T$$

where d is the dimension of the examples, h_i are the features, and a_{it} are the regression parameters of the features. The main assumption is that each task function is a linear combination of the features, in which most of the features have zero regression parameters. As a result, the algorithm will learn a sparse dictionary of the features. The authors consider only linear features so each feature h_i, in turn, can be written as:

$$h_i(x) = \langle u_i, x \rangle$$

The authors then define the dictionary learning task as a minimization problem of the following loss function:

$$\mathcal{E}(A, U) = \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle a_t, U^T x_{ti} \rangle) + \gamma \|A\|_{2,1}^2$$

where A is a $d \times T$ matrix of regression parameters a_{it} and U is a $d \times d$ matrix of entries u_i. However, the above problem is non-convex and the $2,1-norm$ of matrix A is not continuous. If we let $w_t = \sum_i a_{it} u_i$, then the original function f_t can be rewritten as:

$$f_t(x) = \langle w_t, x \rangle$$

With this result, after a series of algebraic transformation and proof, the authors were able to transform the initial non-convex minimization problem into an equivalent convex minimization problem of the function:

$$\mathcal{R}(W, D) = \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti} \rangle) + \gamma \sum_{t=1}^{T} \langle w_t, D^+ w_t \rangle$$

where D is a positive semi-definite matrix of size d and D^+ is the pseudo-inverse of D. Solving the above optimization is, however, non-trivial. Another major contribution of the paper was coming up with an algorithm to iteratively solve the problem. The algorithm is algebraically complicated that it is not suitable to explain in details in the context of a brief overview. We have actually attempted to re-implement the algorithm, however, our programming was not able to run as efficiently and correctly as we want and thus will not be included in this report.

3) Self-taught Learning: The main assumption behind this algorithm is similar tasks share some common representations. Even tasks that are not apparently related may have some similar features. As the result, even the representations learned from a collection of disorganized unlabeled data may be useful for later supervised learning tasks.

To illustrate the idea, let us consider a classifier trying to learn the concept $Lion$. The classifier is expected to be able to learn features that have great correlation with $Lion$ such as $SharpTeeth$, $HorizontalBody$, $FourLegs$, and $OneTails$. Even though the learner has never seen a $Lion$, it is still able to extract these features if it has seen a lot of animals, some of which have $SharpTeeth$, $HorizontalBody$, $FourLegs$, and $OneTails$. When the learner sees a $Lion$, it can fit the $Lion$ into the feature space that it is familiar with. The important point is, when observing other animals, the learner need not know the labels of those animals, because the focus is not to identify what the animals are, but to identify the features.

The idea consists of 2 steps. In the first step, given an unlabeled dataset, or multiple datasets, the learner must, in an unsupervised manner, identify the important features within those examples. In the second steps, the learner must apply its knowledge about the features to learn an supervised learning problem.

The authors translated this idea into an algorithm that first learn a sparse dictionary of features from a set of unlabeled source domain data. After that, the target domain examples are represented using this dictionary via sparse encoding. The resulting encoding is used to represent the labeled examples when learning a classifier. The algorithm steps are as the followings.

a) Representation learning step: Learn a sparse dictionary of the unlabeled data by solving the optimization problem:

$$\text{minimize}_{b,a} \sum_i \|x_u^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta \|a^{(i)}\|_1$$
$$\text{s.t. } \|b_j\|_2 \leq 1, \ \forall j \in 1, \dots s$$

where x is an example and the subscript u is for unlabeled, b is a set of basis vectors and a is a set of corresponding activations for the vectors. Each unlabeled example is therefore encoded as a linear combination of the learned basis vectors. This first step returns the sets of basis vectors and activations.

b) Transferring of representation step: In this step, the labeled data is transformed using the representation learned in step 1, i.e. , produce an encoding for each labeled example from the set of basis vectors and activations we obtained. To do this, the model solve the next optimization problem:

$$\hat{a}(x_l^{(i)}) = \arg\min_{a^i} \|x_l^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta \|a^{(i)}\|_1$$

where x is an example and the subscript l is for labeled. $\hat{a}(x_l^{(i)})$ is the set of activations for a particular example, and will be used to represent that example during later learning task.

c) Learn a classifier : In this step, the model learn a classifier for the labeled examples based on their new representations. Any classification algorithm can do the job. In our implementation, we chose SVM as our base learner for the algorithm.

II. EXPERIMENTS

Datasets: House Number, MNIST, Fashion MNIST

Pairs of datasets are used to test the algorithm, each takes turn being the source domain and the target domain. When learning the classifier for each experiment, we apply parameter selection for the SVM classifier using the following ranges of values:

Kernel : (linear, polynomial, rbf)

Degree : (2,3) , for polynomial kernel

C : (0.001, 0.01, 0.1, 0.5, 1, 2, 3) , the trade-off constant

Gamma : (0.001, 0.01, 0.1, 0.5, 1, 2, 3)

The reported result of each experiment is the result obtained from the best set of parameters. In each experiment, a set of 1000 unlabeled data is randomly generated to learn the dictionary. A training set of size 2500 and a test set of size 500 of labeled data are randomly generated with equal amount of examples from each of the 10 classes. Cross-validation with 5 folds was applied.

First, we collected baseline results when learning each dataset without transfer learning. The result is shown on $Table\ I$.

TABLE I
BASELINE PERFORMANCE

Dataset	MNIST	Fashion MNIST	House Number
Accuracy	91.1 ± 0.3	81.7 ± 0.4	33.9 ± 0.5

Then, we collected the results for pairwise experiments, as shown in $Table\ II$ (The unlabeled data is in the left column).

The experiments above showed that choosing an appropriate dataset to learn the dictionary has a positive effect on the classification performance on the target domain. To be specific, using MNIST or House Number as the source domain does not improve the classification performance of Fashion MNIST. Using Fashion House Number as the source domain has negative

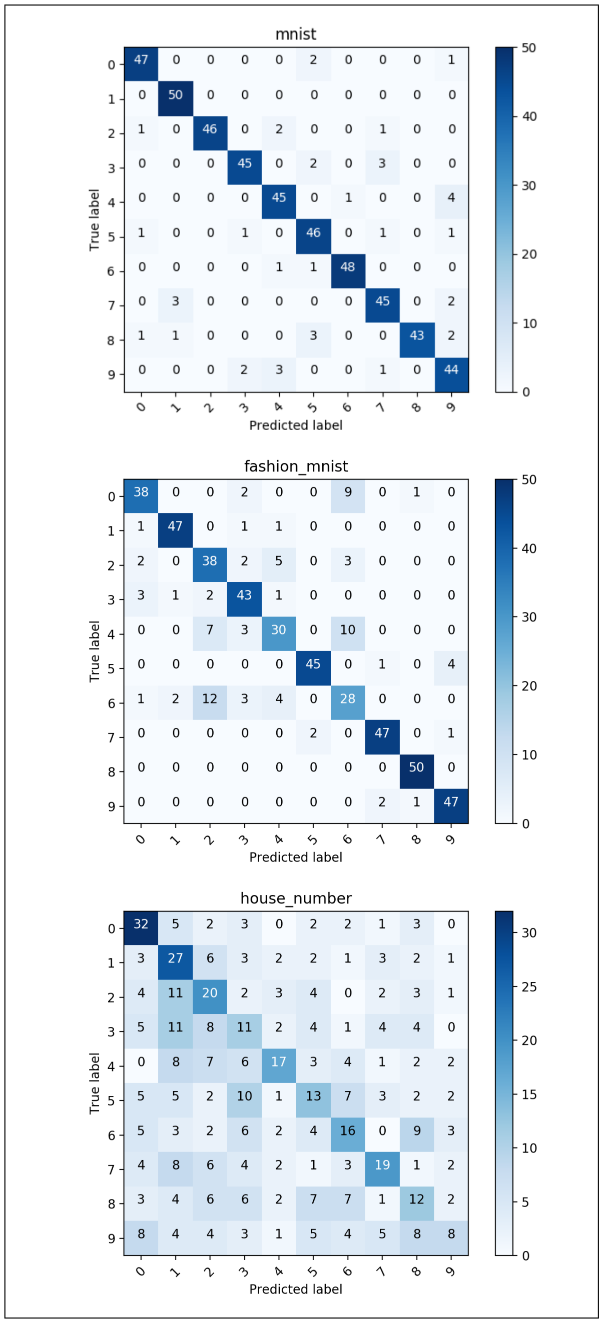

Fig. 1. Baseline Confusion Matrices

TABLE II

SELF-TAUGHT LEARNING RESULTS

	Mnist	Fashion Mnist	House Number
Mnist		80.7 ± 0.8	40.5 ± 1.2
Fashion Mnist	91.4 ± 0.2		42.4 ± 1.6
House Number	86.9 ± 0.7	82.0 ± 0.6	

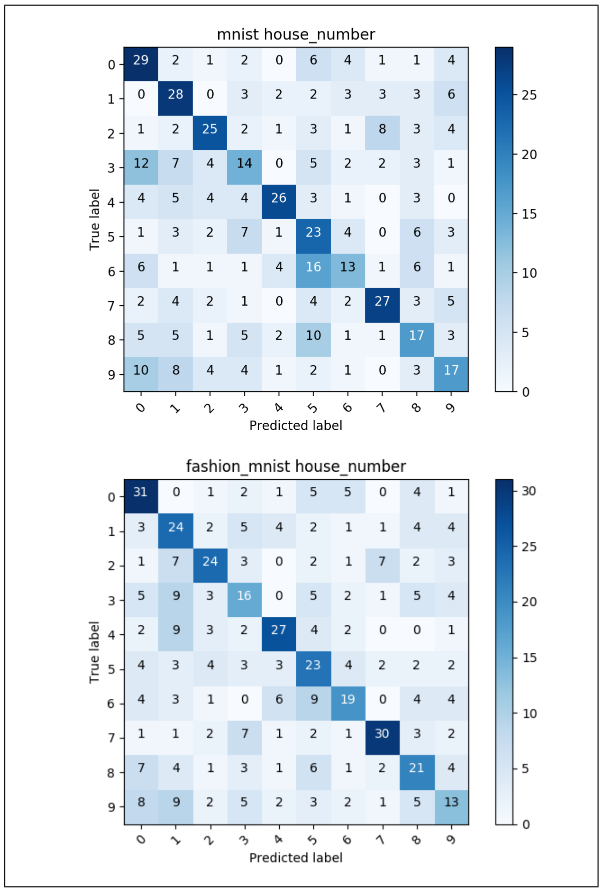

Fig. 2. Self-taught Learning Performance on House Number

effect on the classification of MNIST. However, the classification performance of House Number improve by about 6% when using MNIST as the source domain and by about 8% when using Fashion MNIST as the source domain. With the unlabeled - labeled pair the yielded the largest improvement, Fashion MNIST - House Number, we conducted another experiment to see the effect of increasing the amount of unlabeled data on the classification performance.

Using the same setup as before, this time, we varied the amount of unlabeled data (200, 500, 1000 - the dictionary size was kept the same). The results are shown in $Table\ III$.

Interestingly, smaller amount of unlabeled data yielded better classification result on the target domain. This may be due to overfitting when the dictionary started picking up the irrelevant details from the unlabeled data set. To be more specific, when there was more unlabeled data, the dictionary became less sparse as the learner started including features from the unlabeled domain that is not relevant to the target domain.

TABLE III

AMOUNT OF UNLABELED DATA VERSUS PERFORMANCE

Unlabeled data size	200	500	1000
Performance	45.2 ± 0.7	44.8 ± 1.0	42.4 ± 1.6

III. INTERPRETATION OF RESULTS

The obtained results show that self-taught learning did not improve the classification performance on the MNIST and Fashion MNIST datasets. Using House Number as the unlabeled data for classifying MNIST actually hurt the performance.

However, there is a significant improvement of at least 6% on the House Number dataset.

A possible explanation for these observations may be came up by looking at the complexity of the classification tasks, where:

$$\text{MNIST} < \text{Fashion MNIST} < \text{House Number}$$

A human learns by picking up simple concepts first and then applying those concepts to learn more complex concepts. That intuition may holds in this case. Applying a more complex representation to a less complex concept may not improve the overall performance (House Number $\rightarrow$ Fashion MNIST) or even hurt the overall performance (House Number $\rightarrow$ MNIST). In the other hand, applying a less complex representation to a more complex concept may improve the performance (MNIST $\rightarrow$ House Number and Fashion MNIST $\rightarrow$ House Number).

Another thing that we can think about is what the algorithm is actually doing. Consider 2 learning tasks: classification of $Shoes$ and classification of $Boat$. We know that $Shoes$ and $Boat$ have similarities in term of their shapes, and learning the representation of $Shoes$ beforehand may be beneficial for learning the $Boat$ classifier. However, only some features of the $Shoes$ are actually useful for learning $Boat$ (the general oval shape); other features such as shoe laces are not relevant. For that reason, we would want to pick only the relevant features. The self-taught learning algorithm that we evaluated, in the contrary, pick all the features. Essentially, the algorithm first learns all the useful features for classifying $Shoes$ and then fit the $Boat$ into the shoe representation. As the result, when learning to classify $Boat$, the algorithm may not learn the boat concept, but the things that look like shoes concept.

In that sense, it is reasonable for the overall performance to go down when we try to fit a more complex concept into a less complex concept.

IV. FURTHER RESULTS TO COMPARE WITH OTHER TRANSFER LEARNING APPROACHES

We prepared another set of experiments to get results that we can used to compare with other algorithms that we implemented. Pairwise experiments with 200 unlabeled examples, 600 labeled training examples, 3-fold cross validation, and parameter selection were conducted. For each experiment, the average performance of 3 folds and the performance for each class in each fold were returned. We used the result with the highest performance for each dataset for comparison with other algorithms.

REFERENCES

[1] Argyriou, A., Evgeniou, T., & Pontil, M. (2007). Multi-task feature learning. In Advances in neural information processing systems (pp. 41-48).

[2] Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007, June). Self-taught learning: transfer learning from unlabeled data. In Proceedings of the 24th international conference on Machine learning (pp.759-766). ACM. Chicago