Project Report for Machine Learning

Zhige Li
Shanghai Jiaotong University
Shanghai, China
l-zhige@outlook.com

Xinpeng Sun
Shanghai Jiaotong University
Shanghai, China
1134790197@qq.com

ABSTRACT

In this project, we proposed three machine learning based algorithm for image classification problems in MNIST dataset. These three algorithms consider the problems in three complete different aspects. Specifically, the algorithms are integrated with graph embedding, variance auto encoder, and PCA developed convolutional neural network. All of these algorithms achieved great results.

KEYWORDS

Graph Embedding, PCA, AutoEncoder, MNIST, Classification

1 PROJECT DESCRIPTION

This section introduces the topic and the problem we need to address. MNIST is a hand-written image digit dataset, which consists of 10 different classes. The original black and white images from MNIST were size normalized to fit in a 45x45 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique. We need to build a classifier, which will be able to get high accuracy when predicting the specific digit class. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. -

2 MOTIVATION AND DIRECTION

Though our three algorithms are focused in three totally different aspects, actually there lies some common motivation and direction behind them. As we all know, a mature machine learning algorithm constitutes of two different parts: the unsupervised learning algorithm for learning some hidden features, and the supervised learning algorithm for fitting the function. Because nearly all the supervised learning algorithms are explained both detailedly and clearly, it is really hard for us to develop some brand new supervised learning algorithms. However, we could apply some unsupervised learning methods, which are various and rich, and then combine them together to get a good performance. The detailed methods and their motivations will be described below. They are created mostly enlightened on our previous research work.

Project Report, June 2018, SJTU

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 GRAPH EMBEDDING BASED ALGORITHMS

3.1 Background Knowledge

Existing graph embedding methods learn the node representation by maximizing the network probability in terms of local structures [2, 3, 6]. Specifically, given a network $G = (V, E)$, it seeks to maximize the log-probability of observing a network neighborhood $N(u)$ for a node u conditioned on its feature representation, given by g:

$$\max_{g} \sum_{u \in V} \log Pr(N(u)|g(u)) \tag{1}$$

It is essentially a skip-gram based learning architecture. Similar to some previous efforts, it can leverage random walks to generate co-occurrence corpus for skip-gram. With this method, the architecture will be able to preserve local neighborhoods of nodes and efficiently optimize this object using stochastic gradient descent(SGD) akin to backpropagation.

3.2 Motivation

Sometimes it will be difficult to solve the problem directly. So, it suddenly came to my mind, can we solve the problem in the graph embedding way. Specifically, instead of directly learning the class through a neural network, can we learn the similarity between samples. With the proper embeddings representing the correlation between different samples, we can then do the classification problem with some simple supervised learning method.

So in general, given the vector for all the samples, we could construct a graph. Now we could transform the image classification problem to a graph constructing problem. We regard each sample as an independent node. We expect that, at the best situation, after we generate the edge weight according to their embeddings, the nodes with the same label are connected closely, while nodes with different labels are expected to have no edge, as shown in Figure 1.

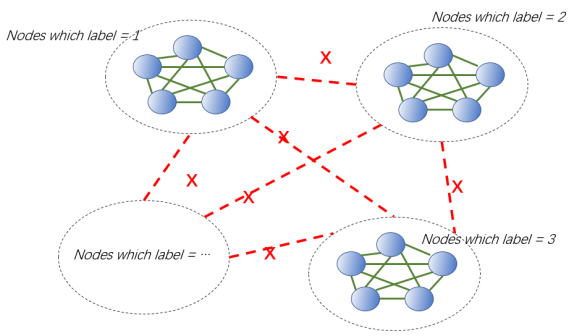

Figure 1: Expected Graph

So instead of viewing the MNIST dataset as a image classification problems, we can regard it as a graph embedding problem.The expression (1) can be defined as:

$$Pr(n_i|g(u)) = \frac{exp(g(n_i) \cdot g(u))}{\sum_{v \in V} exp(g(v) \cdot g(u))} \tag{2}$$

With the above assumptions, the objective in Expression 1 can be simplified to :

$$\max_{g} \sum_{u\ in V} \left[-logZ_u + \sum_{n_i \in N_S(u)} g(n_i) \cdot g(u) \right] \tag{3}$$

3.3 Algorithm

As shown in the figure 2, given the feature represented by each node, which is exactly the pixel of the hand-written digit, we plan to build a convolutional neural network, which is proved to be significantly useful in capturing features of the image and transform the input feature to embedding. After that, we applied the graph embedding based algorithm. Specifically, in this case, for each target input feature, we sample a batch of embeddings generated by the nodes whose labels are the same and then minimize the equation mentioned in Expression1. Besides, in order to minimize the Z_u in expression 3, we will also generate a batch of negative sampling. The algorithm is given as follows.

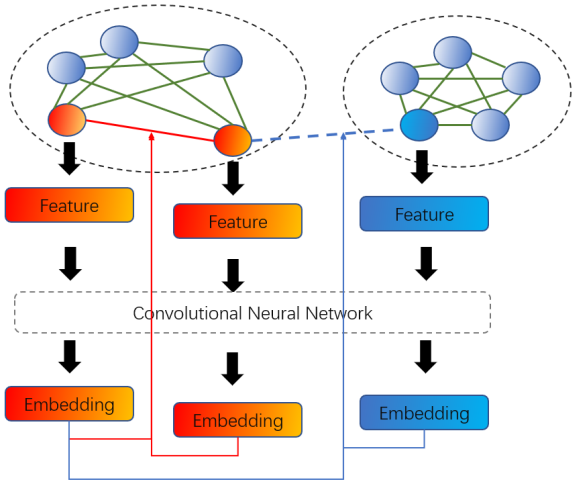

Figure 2: Model Framework

4 PCA DEVELOPED CONVOLUTIONAL NEURAL NETWORK

4.1 Motivation

As we all know, PCA is a widely used method for feature extraction [5]. However, with the development of the convolutional neural network, the PCA is hardly used in image classification problem now. But, can we seek some methods to combine them in order to get a good performance? Looking back the process of the neural network, we find that the value of the filter is optimized through back propagation, so here came something, that can we directly apply PCA to decide the value of the filter? Inspired by **wavelet scattering networks**(ScatNet) [1, 4], where the convolutional filters in ScatNet are prefixed, they are simply wavelet operators, hence no learning is needed at all. Somewhat surprisingly, such a pre-fixed filter bank, once utilized in a similar multistage architecture of ConvNet or DNNs, has demonstrated superior performance over ConvNet and DNNs in several challenging vision tasks such as handwritten digit and texture recognition. Thus with the significant information capture ability of the image by the convolutional neural network(here means no matter the image is upside down or get larger or smaller, the CNN can still recognize it), and the great denoising techniques by PCA, the information can be greatly saved. Then with this unsupervised learning method, we could apply the SVM classifier to evaluate. The basic framework is showed in Figure 3.

Algorithm 1 Graph Embedding based Classification

procedure TRAIN(Single Target x, Similar Sampling Set X', Different Sampling Set X'', Convolutional Neural Network cnn, hyper parameter λ)
  $Loss = cnn(X') * cnn(x)^T - \lambda * cnn(X'') * cnn(x)^T$
  $StochasticGradientDescent(Loss)$

procedure DIGIT2VEC(Training Digit Image Set X_{train}, Corresponding Label y_{train}, Test set X_{test}, Test label y_{test}, KNN Classifier knn, Convolutional Neural Network cnn)
  for all $x_i \in X_{train}$ **do**
    Sample set X', where each $x_j \in X', y_j = y_i$
    Sample Set X'', where each $x_k \in X'', y_k = y_i$
    $Train(x_i, X', X'', cnn)$
  $Embedding_{train} = cnn(X_{train})$
  $Embedding_{test} = cnn(X_{test})$
  $knn.fit(Embeddding_{train}, y_{train})$
  $predict = knn.predict(Embeddingtest)$
  Return $predict$

4.2 Algorithm

The detailed algorithm is shown in Algorithm 2. The Processing operation means some data processing techniques. The easiest way

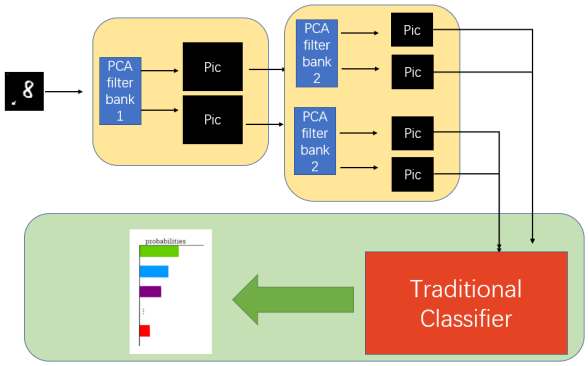

Figure 3: Model Framework

Algorithm 2 PCA Developed Convolutional Neural Network

procedure TRAIN(Training Set $X \in R^{N\times m\times n}$, where m is figure height, n is the figure width. Filter Size: $k_1 \times k_2$. Filter Number: F. Layer Number: L)
  $I_1 = X$
  for i from 1 to L **do**
    Block Sampling I_i and get matrix:$X' \in R^{N\times m'n'\times k_1k_2}$, where $m' = m - \lceil \frac{k_1}{2} \rceil$, $n' = n - \lceil \frac{k_2}{2} \rceil$.
    Padding X' and reshape X' to matrix X'', where $X'' \in R^{k_1k_2\times Nmn}$
    $w^i = [q_1, q_2, \cdots, q_{F_i}]$, where q_t denotes the tth eigenvector for matrix $X''(X'')^T$
    for j from 1 to F_i **do**
      $I_{i+1} = []$
      $I_{i+1}.append(I_i \odot q_j)$, where $\odot$ denotes operation of convolution.
  $Feature = Processing(I_{-1})$
  Classifier with traditional classifiers based on $Feature$.

is simply to reshape the matrix to a vector. Also, it can be done with binary mapping, hashing, etc.

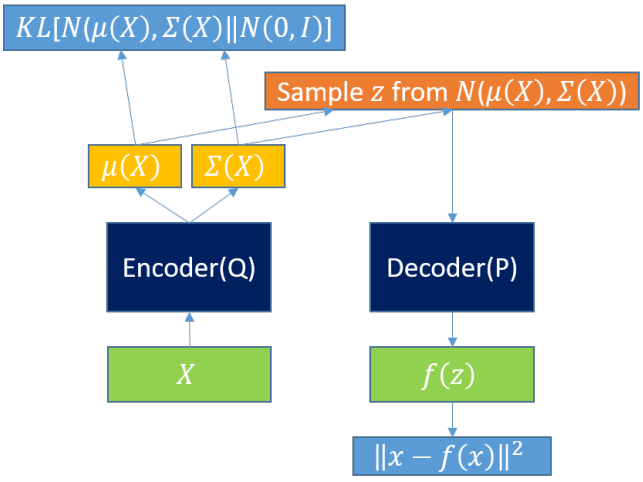

Figure 4: VAE Framework

5 VARIATIONAL AUTO-ENCODER BASED ALGORITHM

5.1 Background Knowledge

VAE is one of the most popular unsupervised complex probability distribution learning. The biggest feature of VAE is to imitate the learning and prediction mechanism of automatic coding machine and encode and decode between measurable functions. The mathematical basis is that for a target probability distribution, given any kind of probability distribution, there is always a differentiable measurable function can map it to another probability distribution which is randomly close to the target probability distribution. The training framework of VAE is to encode the samples to hidden variables and then decode the hidden variables, and then minimize the reconstruction loss. The purpose of training is to learn the mapping function of the encoder and the mapping function of the decoder, so the training process is actually performing variational inference to look for a certain function to optimize the target. The training framework is shown in Figure 4.

Algorithm 3 VAE Based Classification

procedure TRAIN(Training Data Set $X \in R^{N\times m\times n}$, where m is figure height, n is the figure width. Training Label Set Y, the number of labels is p. Divide the X into p subsets according their labels from $X_1 to X_p$. Loss Feature $L \in R^{N\times p}$)
  for i from 1 to p **do**
    use X_i to train AE_i with VAE
    for j from 1 to N **do**
      $L[j][i]$ = the loss of $X[j]$ for AE_i
    for k from 1 to N_{test} **do**
      $L_{test}[k][i]$ = the loss of $X_{test}[k]$ for AE_i
  $svm.fit(L, Y)$
procedure PREDICT
  $predict = svm.predict(L_{test}, Y_{test})$
  Return $predict$

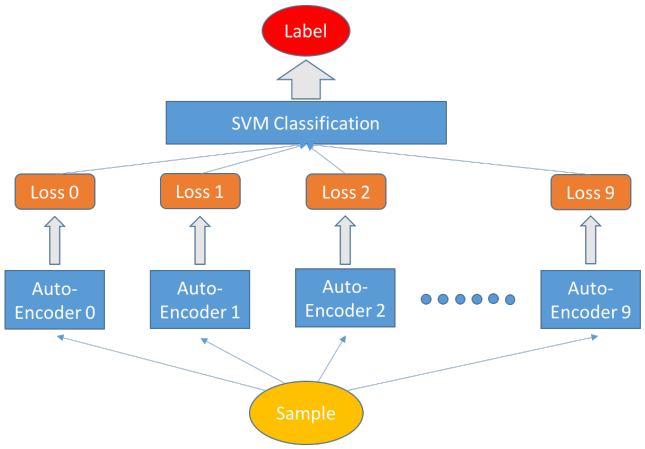

Figure 5: VAE Training Framework

5.2 Motivation

VAE has many good characters as an unsupervised learning method. At the same time, the loss of the Auto-Encoder can be used to do pattern recognition, such as catastrophe forecast of financial data, which is a very novel idea. So we want to use this idea for reference and combine VAE with traditional supervised learning method(such as SVM) to solve this classification problem.

5.3 Algorithm

First, we split the training data into 10 subsets according to their labels. Then we use VAE to train 10 auto-encoders respectively using

these 10 subsets. Next, we put all samples into the 10 auto-encoders respectively and for each sample we will get 10 loss values. We will take the 10 loss values as the new features for the samples. Until this step, the new training data has 10 dimensions of continuous value features so we can use SVM to train a classification. The procedure is shown in Figure 5 and the detailed algorithm is shown in Algorithm 3.

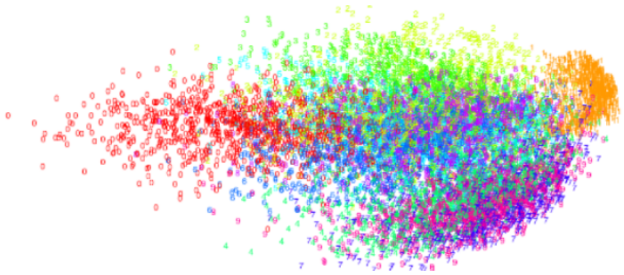

Figure 6: Visualization without Graph Embedding Optimization Method

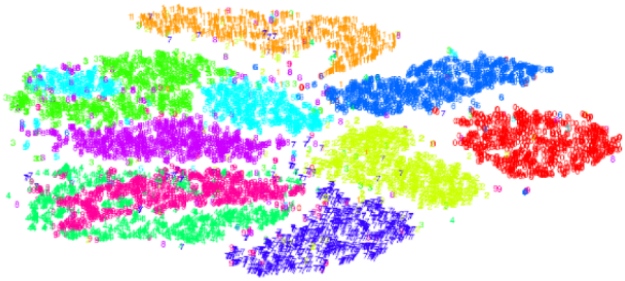

Figure 7: Visualization with Graph Embedding Optimization Method

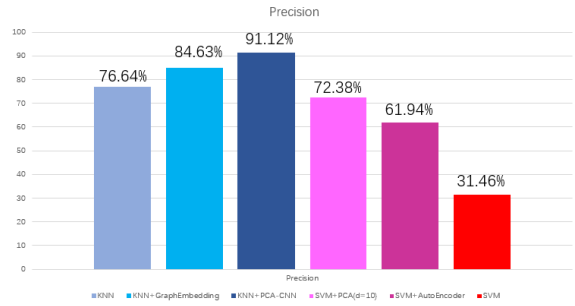

Figure 8: Precision Result

6 EXPERIMENT AND COMPARISON

6.1 Comparison

Three of the methods all develop some unsupervised learning methods to extract features from the image. Because the traditional classifier is hard to recognize the object as for its complicated information such as angle, size, etc. So the graph embedding method and PCA-CNN combined methods all conduct some strategies based on CNN net. These two methods are then expected to get good scores. The auto-encoder based method simply relies on the pattern recognition generated by training loss. However, loss is a rough metric and can contain much noise. Also, the pattern recognition impact of auto-encoder is hardly discussed in research area. So though the result might seem not well enough, the thought can be referenced for other area and further developed.

6.2 Experiment

For the graph embedding based method, we could clearly see the embedding effectiveness. Here we use the t-sne method to do the visualization. If we directly visualize the data, the result is shown in Figure 6. But after the sampling based work, the result is shown in Figure 7. We could see that the samples are distributed in a more regular pattern.

The final precision result is shown in Figure 8. We could see that with graph embedding method achieve nearly 10% than simply directly applying K Nearest Neighborhood algorithm. With PCA-CNN combined method will gain nearly over 15% compared to normal KNN. The PCA-CNN and Graph Embedding methods all used KNN as their supervised learning metric because they all mapped the input feature to very high dimension, and SVM could not deal with high dimensional problems as we can see its accuracy is only 30%. But one thing that SVM obtain such low accuracy actually results from the large dimension of the input data. This could be proved from the fact that, with PCA transformed the original image to 10-dimensional vector, the accuracy suddenly jump to 72%. The Auto-Encoder based algorithm actually do not beat the PCA based algorithm, but based on the fact that loss is a rough metric and it is a pretty brand new test, the 61% accuracy still means that the loss do reflect some knowledge of the pattern of the input data.

So in conclusion, we propose three different approaches, which view the problem from three completely different aspects. All of the methods achieve nice results and the thoughts behind the approaches can be enlightening in other future directions.

REFERENCES

[1] Joan Bruna and Stephane Mallat. 2013. Invariant Scattering Convolution Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1872–1886.
[2] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. 2016 (2016), 855–864.
[3] Bryan Perozzi, Rami Alrfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. (2014), 701–710.
[4] Laurent Sifre and Stephane Mallat. 2013. Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1233–1240.
[5] L. I. Smith. 2002. A Tutorial on Principal Components Analysis. *Information Fusion* 51, 3 (2002), 52.
[6] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1225–1234.