Sparse Autoencoder for One Class Classification

Yu Zhao
yzhao@jhu.edu

Fan Yang
yfan@jhu.edu

Abstract

Autoencoder neural network is an unsupervised learning algorithm, setting the target values to be equal to input. It's always used to learn a compact representation of the input. Especially, if a sparsity constraint is imposed on the hidden units, we have one variant: sparse autoencoder. One class classification is to predict whether sample is positive or negative with no negative sample in training.

In our experiment, sparse autoencoder is implemented and tuned to perform one class classification on common used handwritten digits dataset MNIST. False positive rate of 16.08% and false negative rate of 15.25% are achieved.

1 Introduction

Traditional classification tasks try to distinguish sample between two or more classes while the training sets contain data from all classes. Nevertheless, sometimes it's hard or even impossible to gather such datasets. For example, one would like to tell whether a fruit is an apple or not while he or she only has the knowledge of apple. This is the problem of one class classification a.k.a OCC. In our experiment, we try out the method of sparse autoencoder to tackle OCC.

1.1 One class classification

Let's put above problem formally. Assume that in one class classification, only one class of data, i.e. the target class is available. In no means, data from other class (often called outlier[1]) could be used in training phase. This is the essential difference between conventional classification problem and OCC.

One problem for one class classification is that it is hard to draw decision boundary given one class data only. Recall classification algorithm taught in class. Our goal is to draw the decision boundary to keep the balance between complexity of model and error rate. In one class classification, people would like to minimize the chance of accepting the outlier and maximize the chance of accepting the target. However, we could neither estimate the distribution of outlier nor tell if decision boundary is tight. Moreover, it's hard to decide which features should be used to model the boundary.

Nevertheless we still try to define error in OCC. In table 1, all possible situations of classifying a sample in OCC are shown.

Table 1: Four situations of one class classification

	data from target class	data from outlier class
classified as target class	true positive e_{tp}	false positive e_{fp}
classified as outlier class	false negative e_{fn}	true negative e_{tn}

[1] so sometimes one class classification is called outlier/novelty detection

The fraction of target class which is accepted by classifier (classified as positive class) is e_{tp}.The fraction of target class which is rejected by classifier (classified as negative class) is e_{fn}. It's usually called type I error in statistics ie. false negative ratio. Similarly, the fraction of outlier class which is accepted by classifier is e_{tp} and the fraction of target class which is accepted by classifier is e_{tp} which contributes to type II error. Notice that $e_{tp} + e_{fn} = 1, e_{tn} + e_{fp} = 1$.

Several metrics are usually used to measure the performance of OCC[8].

$$\text{Precision} = P = \frac{e_{tp}}{e_{tp} + e_{fp}}$$

$$\text{Recall} = R = \frac{e_{tp}}{e_{tp} + e_{fn}}$$

$$F = (1 + \beta^2)\frac{PR}{\beta^2 P + R}$$

β controls relative importance of false positive and false negative for F-score. If $\beta > 1$, false positive is more important. If $\beta = 1$, they are equally important . If $\beta < 1$, false negative is more important.

1.2 One class classification methods

Several methods are used to train one class classifier. Generally, they could be categorized into three groups[8].

1. Density methods

   These methods try to estimate the distribution of data on several assumption directly and set a threshold[3]. For instance, simple normal distribution could be assumed and classifier would reject if x is far deviated from mean.

2. Boundary methods

   One of the most common OCC algorithm is one class SVM[2]. Instead of maximizing margin between classes, one class SVM manages to separate all data from the origin of the feature space (defined by kernel function) and maximizes the distance from this hyperplane (decision boundary) to the origin. Other boundary methods are similarly trying to maintain a tight decision boundary.

3. Reconstruction methods

   The reconstruction methods interestingly do not construct classifier directly, but model and fit the data. We assume more compressed/compact representation of data could be learned and difference between representation and data is small. Since the outlier do not satisfy the assumptions about the target distribution, they should be represented worse and their reconstruction error should be high. Autoencoder and PCA are usually used for OCC[1].

1.3 Autoencoder

Autoencoder neural network[4] is an unsupervised learning algorithm that applies back propagation, letting the target values to be equal to the inputs. That is, the autoencoder tries to learn $h(x) \approx x$

The simplest autoencoder has 3 layers: input, hidden layer and output. It seems that if hidden layer has more units than input, then a pretty trivial identity function would be learned. But if constraints are imposed then structure within data could be discovered. For instance, if hidden layer has only half of units with regard to input units, then hidden layer actually could learn to compress the information from input.

Loss of autoencoder is defined as square loss

$$L = \frac{1}{N}\sum_{i=1}^{N} ||h(x_i) - x_i||^2$$

Back propagation of autoencoder is the same with neural network

$$\text{gradInput} = \text{gradOutput} \times z_i'.$$

We use the same notation as hw5 where gradInput is the gradient w.r.t. input and gradOutput is the gradient w.r.t. output.

Like neural network, dropout layer could also be used in autoencoder to prevent overfitting.

1.3.1 Sparse autoencoder

Even if the number of hidden units are large, we could still impose sparsity constraint on the hidden units to learn the compress of data. Formally, let z_j denote the activation of hidden unit j

$$\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N} z_j(x_i)$$

Note that x_i here is not neural network's input. Constraint $\hat{\rho}_j = \rho$ would be enforced. ρ is a sparsity parameter, whose value is close to zero. In other words, the average activation of each hidden unit j to be close to ρ. To satisfy this constraint, the hidden unit's activations must mostly be near 0.

Therefore, ReLU as activation usually would not be used in sparse autoencoder because it doesn't have upper bound, resulting big fluctuation of gradient if activation is much greater than ρ. Furthermore, input data would better be normalized beforehand.

Loss of sparse autoencoder is

$$L = \frac{1}{N}\sum_{i=1}^{N} ||h(x_i) - x_i||^2 + \beta \left\{ \sum_j \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log \frac{1-\rho}{1-\hat{\rho}_j} \right\}$$

The second term is the sum over all hidden units and β controls the weight of sparsity. For back propagation, backward method of hidden layer should be modified

$$\text{gradInput} = \left(\text{gradOutput} + \beta\left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i}\right)\right) z_i'.$$

1.3.2 Sparse autoencoder for OCC

Autoencoder was not intentionally developed for OCC so the empirical threshold t has to be obtained using the training data. More specifically, if reconstruction error is greater than t then classifier rejects, otherwise accepts.

2 Experiments

We use the sample skeleton of neural network as hw5 and for datasets, a different dataset[2] from hw2 is used.

2.1 Dataset

Hand written digits MNIST is used for one class classification. In our experiment, digit 5 is preserved and other digits are excluded in training phase. The number of training samples is 4506, each one of which is a $28 \times 28 = 784$ vector and each entry is scaled from 0 to 1. For test data, there are 892 digits 5 and 9108 outliers.

2.2 Parameters settings

After tuning the parameters, we set the sparsity parameter $\rho = 0.1$ and $\beta = 1$. For F-score, we consider false positive and false negative equally important. The number of hidden units are 196 so compression of factor is 4. For gradient descent setting, batch size is 256 and sgd with momentum is used.

In our experiment, we choose hyper parameters to minimize the loss and once they are set, we choose t to maximize F-score.

[2] http://deeplearning.net/data/mnist/

2.3 Results

Given above model and parameters setting, we train our autoencoder and below is the test results

Table 2: Test results

	data from target class	data from outlier class
classified as target class	751	1489
classified as outlier class	141	7619

So Precision = 83.74 Recall = 84.20 F = 83.97, false positive rate = 16.08% and false negative rate = 15.25%. The first ten original images of digit 5 and reconstructed images are shown below.

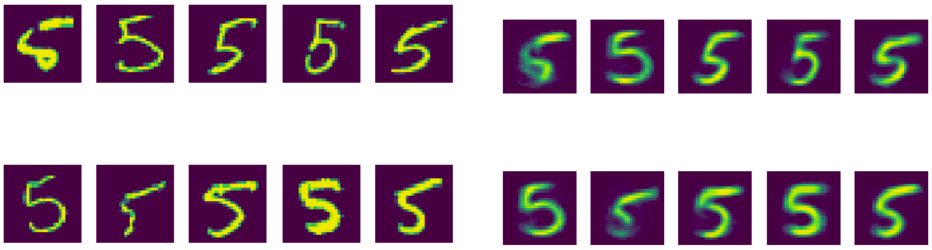

Figure 1: original images and reconstructed images

2.3.1 Visualisation autoencoder

We might also want to know what autoencoder has learned. In particular, parameter of weight W of first layer would mainly determine what would be computed by hidden units. Andrew Ng [4] shows input which maximally activates hidden unit i is setting pixel p_{ij} to be:

$$p_{ij} = \frac{W_{ij}}{\sqrt{\sum_{j=1}^{100}(W_{ij})^2}}$$

The following shows the first 25 units' results:

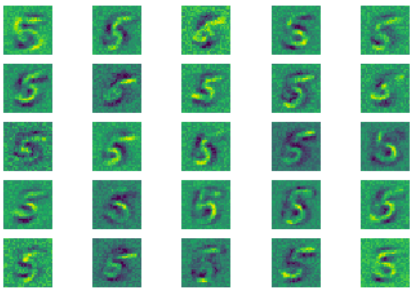

Figure 2: visualization of trained autoencoder

3 Conclusion

Autoencoder could be used to learn the compressed representation of data and especially if we set the threshold, autoencoder would perform one class classification naturally. But what does the

sparsity constraint function here? We discover it would force each unit to learn the most significant part of information especially when the number of hidden units is large. It does sound like L1 regularization especially L1 usually leads to sparsity. If the sparsity parameter ρ is set very close to zero, then sparsity would increase, which may also perform like dropout method with dropout rate close to 1.

Nevertheless, dropout layer is kind of artificial induced sparsity in training phase and in evaluation phase, dropout does not always lead to sparsity. As for visualization of weight, we could found each hidden unit does learn the correlation of the data. To be more specific, hidden unit use edges to distinguish the samples ie. the input producing the most significant response is the one that aligns in the same direction as the weight vector.

References

[1] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523, 1995.

[2] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[3] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[4] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[6] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[7] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[8] D Martinus and J Tax. *One-class classification: Concept-learning in the absence of counterexamples*. PhD thesis, PhD thesis, Delft University of Technology, 2001.