application of deep Learning networks to crime prediction

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Chao fang

Dr. Yi Shang, Advisor

December 2013

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

application of deep Learning networks to crime prediction

Presented by Chao Fang

A candidate for the degree of

Master of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Yi Shang

Dr. Dong Xu

Dr. Tim Trull

**ACKNOWLEDGEMENTS**

First, I would like to thank my advisor, Professor Yi Shang, for his patient advice, guidance, support and encouragement and for the opportunity to work on such an interesting deep learning research. His insightful views on doing research, warmhearted help, and resplendent smile will always inspire me and encourage me to overcome difficulties that I meet during study and life.

I would like to thank my father Weimin Fang and my mother Li Li for supporting me financially and mentally so that it is possible for me to study in America.

I appreciate the assistance and advice from the members of the Distributed Intelligent Computing Lab.

I would like to thank Professor Tim Trull and Professor Dong Xu for being a part of my committee and their advice on this thesis.

**TABLE OF CONTENTS**

[1. Introduction 7](#_Toc372621079)

[1.1 Crime prediction 7](#_Toc372621080)

[1.2 Related Work 8](#_Toc372621081)

[1.2.1 Crime Prediction 8](#_Toc372621082)

[1.2.2 Deep Belief Networks 9](#_Toc372621083)

[2. Methods 11](#_Toc372621084)

[2.1 Sparse Autoencoder 11](#_Toc372621085)

[2.1.1 Backpropagation Algorithm 12](#_Toc372621086)

[2.1.2 Autoencoders and Sparsity 14](#_Toc372621087)

[2.2 . Preprocessing: PCA and whitening 16](#_Toc372621088)

[2.3 . Softmax Regression 16](#_Toc372621089)

[2.4 . Self-Taught Learning 17](#_Toc372621090)

[2.5 . Deep learning: Stacked Autoencoder 19](#_Toc372621091)

[3. Performance study 23](#_Toc372621092)

[3.1 Comparison of Different Preprocessing Techniques 23](#_Toc372621093)

[3.2 Computation Time of Different Preprocessing Techniques 24](#_Toc372621094)

[3.3 Shallow Autoencoder vs. Deep Stacked Autoencoder 27](#_Toc372621095)

[3.4 Effects of Different Training Sizes 29](#_Toc372621096)

[3.5 Self-taught Learning vs. Semi-supervised Learning 30](#_Toc372621097)

[3.6 Stacked Autoencoders with Different Hidden Sizes 32](#_Toc372621098)

[4. Application 34](#_Toc372621099)

[4.1 Problem formulation 34](#_Toc372621100)

[4.2 Dataset 35](#_Toc372621101)

[4.3 Experimental results regions (50% chance crimes occur) 36](#_Toc372621102)

[4.4 Experimental results of predicting all 121 regions 36](#_Toc372621103)

[5. Future Work 45](#_Toc372621104)

[6. Summary 46](#_Toc372621105)

[7. Bibliography 47](#_Toc372621106)

**LIST OF FIGURES**

[Figure 2.1. An example of the autoencoder 14](#_Toc372632321)

[Figure 2.2. Autoencoder as feature extractor. 18](#_Toc372632322)

[Figure 2.3. Features/activations of the first half of an autoencoder. 18](#_Toc372632323)

[Figure 2.4. Self-taught learning model for supervised learning 19](#_Toc372632324)

[Figure 2.5. First layer of a stacked sparse autoencoder 21](#_Toc372632325)

[Figure 2.6. Second layer of a stacked sparse autoencoder 22](#_Toc372632326)

[Figure 2.7. Softmax classifer layer of a stacked sparse autoencoder classifier 22](#_Toc372632327)

[Figure 2.8. the final architecture of a stacked sparse autoencoder classifier. 22](#_Toc372632328)

[Figure 3.1. Preprocessing comparison 23](#_Toc372632329)

[Figure 3.2. Accuracy comparison 25](#_Toc372632330)

[Figure 3.3. Average time comparison 25](#_Toc372632331)

[Figure 3.4. Shallow vs. Deep SAE comparison 26](#_Toc372632332)

[Figure 3.5. Edge features 27](#_Toc372632333)

[Figure 3.6. Contours features 28](#_Toc372632334)

[Figure 3.7. Effects of Different Training Sizes 29](#_Toc372632335)

[Figure 3.8. Prediction Accuracy of digit 0-4 30](#_Toc372632336)

[Figure 3.9. Prediction Accuracy of digit 5-9 31](#_Toc372632337)

[Figure 3.10. SAE Performance with Different Hidden Sizes 32](#_Toc372632338)

[Figure 4.1. Chicago crime heatmap baseline 33](#_Toc372632339)

[Figure 4.2. Focused crime heatmap 34](#_Toc372632340)

[Figure 4.3. Prediction on picked regions 35](#_Toc372632341)

[Figure 4.4. Prediction heatmap of softmax 36](#_Toc372632342)

[Figure 4.5. Prediction heatmap of 1SAE 37](#_Toc372632343)

[Figure 4.6. Prediction heatmap of 2SAE 37](#_Toc372632344)

[Figure 4.7. Accuracy Comparison with FP 38](#_Toc372632345)

[Figure 4.8. Accuracy Comparison with Softmax 39](#_Toc372632346)

[Figure 4.9. Accuracy Comparison with 1SAE 40](#_Toc372632347)

[Figure 4.10. Accuracy Comparison with 2SAE 40](#_Toc372632348)

[Figure 4.11. Summary of average accuracy 41](#_Toc372632349)

**ABSTRACT**

Crimes are a major public concern in cities. Every day, a tremendous amount of law enforcement and policemen have been assigned to patrol and protect society from crimes of violence. With limited law enforcement resources in big city like Chicago, if crimes can be predicted rather than just investigated after they happen, then police officers can potentially be placed at the right time and location efficiently and achieve better crime prevention.

This thesis discuss the use of deep learning networks, an advanced machine learning technique, as applied to the task of predicting future crimes. First, an extensive set of experiments of various types of network structures, together with various pre-processing dimension-reduction techniques, are conducted on the MNIST handwritten digits image recognition dataset [1] to understand their performances and quality-cost trade-offs. Then, various deep learning networks are applied to future crime prediction based on historical crime records. Based on 13 years of crime data from the City of Chicago Crime Data Portal [2] , a region containing diverse crime activities is identified and used in the learning experiments.  The region is divided into 11 × 11 grids. The learning problem formulation predicts whether a crime will occur on a particular day, based on the crime activities in the previous day in all grids. The entire data set contains 4667 examples, derived from crime data of 4668 days. Extensive experimental results show that deep networks achieve better results than a basic prediction algorithm.

# . Introduction

## Crime prediction

We cannot predict where and when crimes will happen in the future with 100% certainty, but with the rapid development of advanced machine learning and statistical inference technologies, we can use patterns from historical crime data to predict future crimes with improved accuracy, just like in a weather forecast or earthquake prediction.

Predicting crimes before they happen is financially beneficial. If a crime can be predicted before it happens, it can save the city government from a tremendous cost in investigation, prosecution, and incarceration.

Crime prevention [3] is the attempt to reduce [crime](http://en.wikipedia.org/wiki/Crime) and criminals. It is applied specifically to efforts made by [governments](http://en.wikipedia.org/wiki/Government) to reduce crime, enforce the law, and maintain [criminal justice](http://en.wikipedia.org/wiki/Criminal_justice). Crime prevention is an important research area and most crime prediction techniques are developed from sociological studies and field experts. Machine learning technology has made significant progress in the past few decades and has been applied to many fields, involving finance, military, agriculture, education, etc. In recent years, police forces have been enhancing their traditional method of crime reporting with new technological advancements to increase their output by efficiently recording crimes to aid their investigation [4]. In today’s world, computers are playing a major role in the investigation of all types of crime from those that are considered as volume crime (burglary, vehicle crime etc.) to major crime such as fraud, drug trafficking, or murder etc.[5]. Recently, several commercial products have been developed based on large data sets and machine learning techniques, such as PredPol [6] and CommandCentral [7], which can predict where and when the future crimes are likely to happen.

## Related Work

### Crime Prediction

Crime prediction has become an active research topic over the last decade and increasingly crime prediction software has been used by law enforcement and police officers to predict and stop crimes. This is due to two reasons according to Gorr and Harries [8]: a) criminality of places can be established based on theories like routine activities [9][10][11], the ecology of crime, and hot spots [12], and b) Geographical Information Systems (GIS) had become an important tool for police agencies. As a result, mapping of crimes and identification of hotspots have become a regular practice. Liu and Brown [13] suggest that the probability of a new criminal incident at some location is depended on incidents in the near past of the same type, some demographic, and proximity features such as distance to the nearest highway. They compare two density models, with and without the extra features and conclude that the model performs significantly better with the extra features. Brown [14] proposed k-means and the nearest neighbor approach to clustering spatial data of crimes to find "hotspot" areas in a city. The spatial clustering methods are often used in "hotspot analysis" [15]. Artificial intelligence techniques also have been successfully used in predicting future crimes and the unsolved past crimes. Oatley et. al [16] used Bayesian networks to predict if a site will be revictimised in the coming period. In the same application they used a Kohonen network to match crimes to a known offender. Sentient software [17] developed in collaboration with Dutch Police has a forecasting technique based on fuzzy matching. Generally speaking, there has been some remarkable research on predicting unsolved crimes using crime classification, clustering, and artificial intelligence.

### Deep Belief Networks

The concept of Deep Belief Networks was first introduced by Hinton et al [18]. The basic idea is to stack layers of Restricted Boltzmann Machines (RBM). RBM is a type of stochastic neural works capable of representing a joint probability distribution of training examples. It trains each layer independently, which is much more efficient and achieves better results than training all layers of a deep multi-layer neural network at once. DBNs have been applied to many problems including acoustic modeling, image classification, object recognition, face recognition, motion detection and dimensionality reduction, and obtained outstanding results [19].

Specifically, Zhou et al. [20] introduce a method of using labeled and unlabeled data to perform greedy layer-wise unsupervised learning followed by fine-tuning the whole deep architecture by gradient-descent based supervised learning. Their experimental results show that DBNs perform excellently on image classification tasks, especially for the harder ones.

Lin et al. [21] address the problem of face recognition when there is a non-linearity in the data due to pose variations. The paper discusses the use of the deep architecture to learn the relationship between low resolution and high resolution images to give highly accurate results.

Lee et al. [22] propose efficient Convolutional Deep Belief Networks on full-sized, high-dimensional images. Probabilistic max-pooling units are integrated into the generative, hierarchical models. Results show that the method is able to learn high-level visual features from unlabeled image data.

# . Methods

## Sparse Autoencoder

Although supervised learning has been widely used in artificial intelligence researches, one problem is that training features are usually manually specified, which is time-consuming, difficult, and may not be the best. The performance of a classifier is limited by the quality of extracted features as its input: a good feature set leads to high accuracy classification, whereas a bad feature set leads to bad results. Furthermore, real-world problems usually are large scale, rendering labor-intensive, hand-engineering approaches unsuitable and impractical.

A sparse autoencoder [23] [24] [25] [26] is a special type of feed-forward neural network. It has three layers: an input layer, a hidden layer, and an output layer. The input layer has the same number of neurons as the output layer. The output neurons try to reconstruct the values of the input neurons. The activation of the hidden neurons is sparse, meaning only a small fraction of the neurons are activated for any given input. A sparse autoencoder can be trained using the classical backpropagation algorithm or other local optimization methods.

This project focuses on stacked sparse autoencoders (SSAs) as a deep learning method. Several sparse autoencoders are trained separately and then stacked together. In an unsupervised learning setting, an SSA can learn features from unlabeled data automatically. SSAs have been used in many applications, including speech recognition, spam filtering system, pattern recognition, and self-driving cars, etc.

### Backpropagation Algorithm

The backpropagation algorithm is an efficient implementation of gradient descent to optimize neural network weights based on a set of training examples. Given a training set of *m* training samples for a single training sample, the error function is . For *m* training samples, the total error or cost function with weight regularization is：

|  |  |  |
| --- | --- | --- |
|  |  |  |

The first term is the average sum-square error and the second is weight regularization used to prevent overfitting. The weight regularization parameter is to balance the two terms. The cost formula can be used for both classification and regression problems. For classification, to represent two classes. For regression, the output is within the range of .

The goal of supervised training is to minimize the non-convex error function . Traditional optimization algorithms such as gradient descent can be used to find local optima, . One step of the gradient descent that updates is as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

where is the learning rate. The back-propagation algorithm computes the partial derivatives efficiently as follows.

Given a training example of a pair of input and output, ,

1. Perform a feed-forward pass to compute the activations of the units (neurons) at each layer, .
2. For each unit in layer , set:

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. For , set:

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. Compute the partial derivatives:

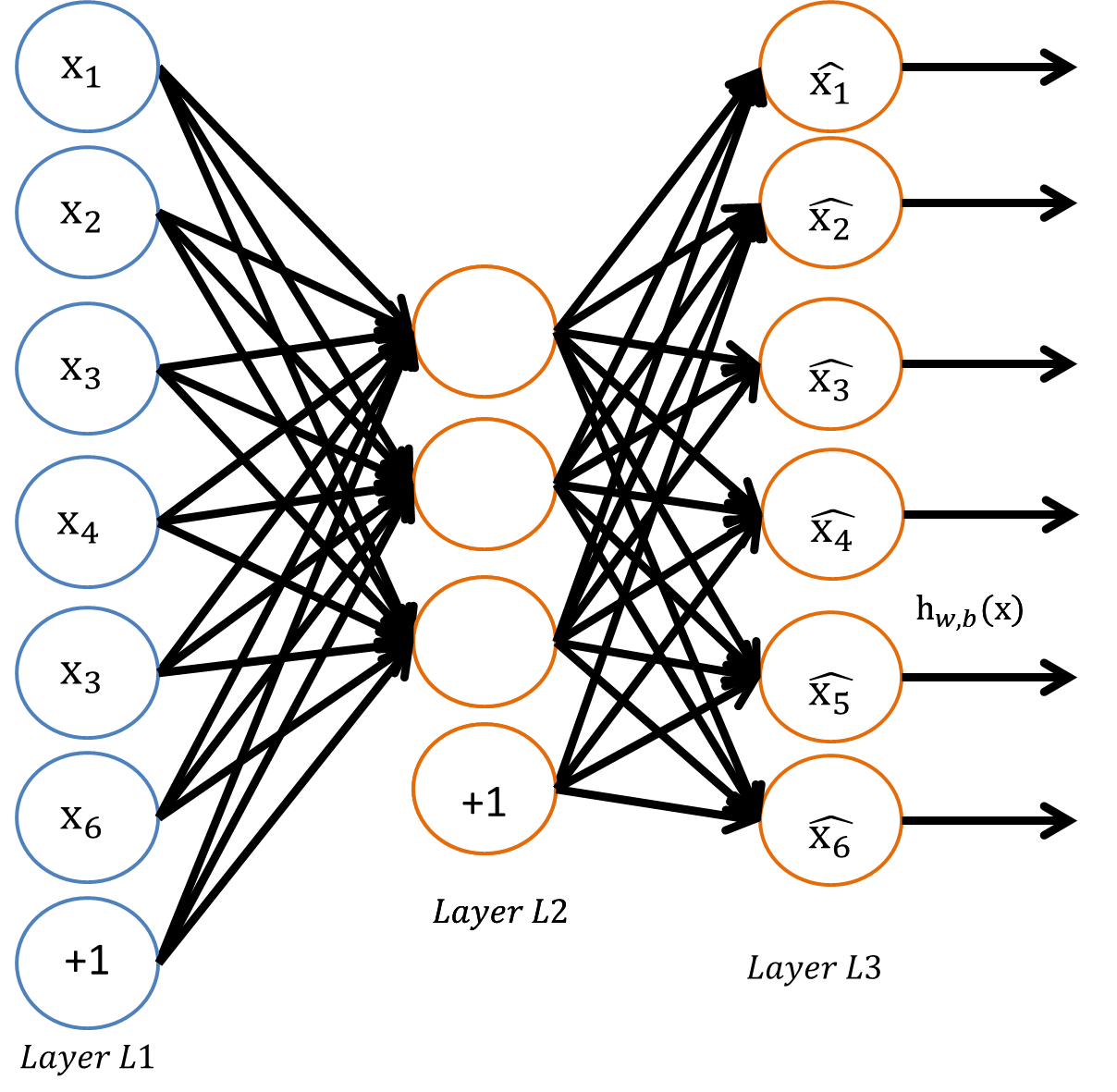
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

So the overall cost function will be:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

### Autoencoders and Sparsity

Given a set of unlabeled training samples , an autoencoder neural network tries to match its output to its input. Here is an example of a one-hidden-layer autoencoder:



**Figure 2.1. An example of the autoencoder**

The autoencoder is trying to learn an identity function: , which means the output is similar to input . When the number of hidden units is less than the number of input units, the hidden units represent a compressed version of the input. For example, when the input is a 28×28 image (784 pixels) and the number of hidden nodes is 200, this network is trained to learn a compressed representation in 200 dimensions of the 784-dimension input data.

If the input data are independent and identical distributed (IID), then the compression task is difficult. However, if the input data are correlated, then the autoencoder can discover some of the correlations. Autoencoder is similar to PCA in terms of both trying to learn a low-dimensional representation (i.e., feature extraction) of a high-dimensional data. Autoencoder is much flexible than PCA because it is a non-linear mapping rather than linear mapping.

In sparse autoencoders, even if the number of nodes in the hidden layer is larger than that of the input layer, feature extraction can still be achieved by using sparsity constraint on the hidden units. If a neuron is active, then its output value is close to 1; if inactive, then close to 0. In sparse autoencoders, the hidden neurons are inactive most of the time. Let represent the specific input x led to the activation of hidden unit ,

|  |  |  |
| --- | --- | --- |
|  |  |  |

represents the average activation of hidden unit over all training examples. The sparsity is controlled by a sparsity parameter, e.g., 0.1, so that The following penalty term is added to the cost objective function to penalize deviating from :

|  |  |  |
| --- | --- | --- |
|  |  |  |

where is the number of hidden nodes. This term is based on the concept of Kullback-Leibler (KL) divergence, a function to measure the difference between two distributions [5] [6] [7]. The penalty term can also be written as:

|  |  |  |
| --- | --- | --- |
|  |  |  |

where

Now, the overall cost function is:

|  |  |  |
| --- | --- | --- |
|  |  |  |

where is defined previously and controls the weight of the penalty term.

## . Preprocessing: PCA and whitening

Principal Component Analysis (PCA) is an algorithm to reduce input data dimension. An image is highly redundant because adjacent pixels are highly correlated. For images in the MNIST handwritten digits image recognition data set [1] that have 784 dimensions, on average, using PCA to retain 95% information of the images can reduce the image to 121 dimensions.

After using PCA, whitening can be used to normalize data variance across all dimensions. There are two commonly used techniques, PCA whitening and ZCA whitening.

## . Softmax Regression

Softmax model is similar to logistic regression, which is a supervised learning algorithm. Logistic regression can only classify two classes while Softmax can classify more than two classes. Softmax classifier can be used in conjunction with the previous discussed deep learning/unsupervised feature learning method.

The error function of a software classifier is as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |

A weight term is added to penalize large values of the parameters. The cost function is strictly convex so there is a unique optimal solution . The derivative of the error function is:

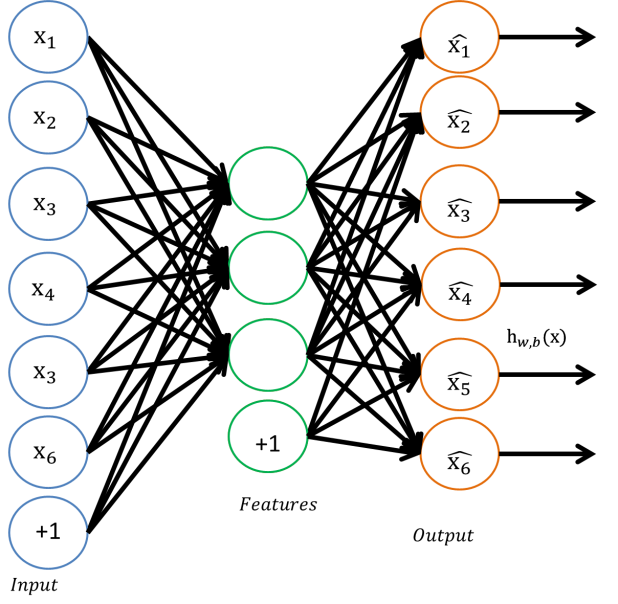
|  |  |  |
| --- | --- | --- |
|  |  |  |

## . Self-Taught Learning

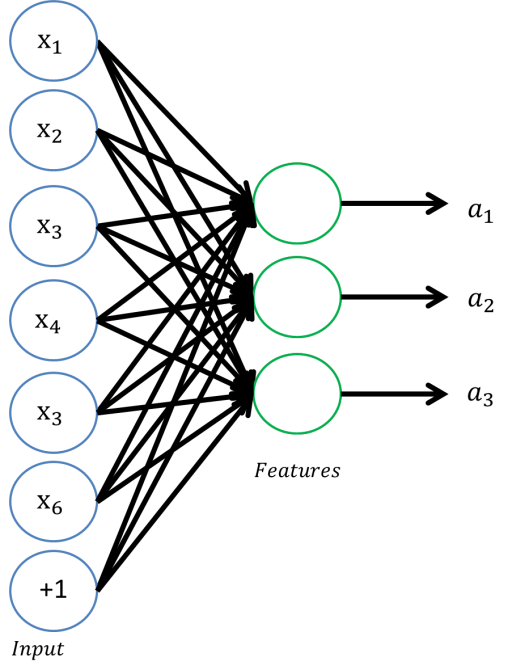
Compare to semi-supervised learning, self-taught learning is more general because the unlabeled data does not need to be drawn from the same distribution as labeled data .

A common quote used in the machine learning industry, “sometimes it’s not who has the best algorithm that wins; it’s who has the most data.” Getting more labeled data for supervised learning can be expensive when labeling data samples involves hand-engineering. Self-taught learning and unsupervised feature learning is a way to learn features from massive amounts of unlabeled data. Even though a single unlabeled example is less informative than a single labeled example, a large amount of unlabeled data can help achieve better performance with only a limited amount of labeled training data. From unlabeled data, high-level features representing the data can be extracted. Later, for a classification task, the learned high-level features and a small amount of labeled data could be sufficient to train a good classifier quickly.

Given an unlabeled training set with unlabeled examples, a sparse autoencoder as in Figure 2.2 is first trained to get of this model. Then for an input , the corresponding activations at the hidden layer form the features representing the original input, as shown in Figure 2.3.



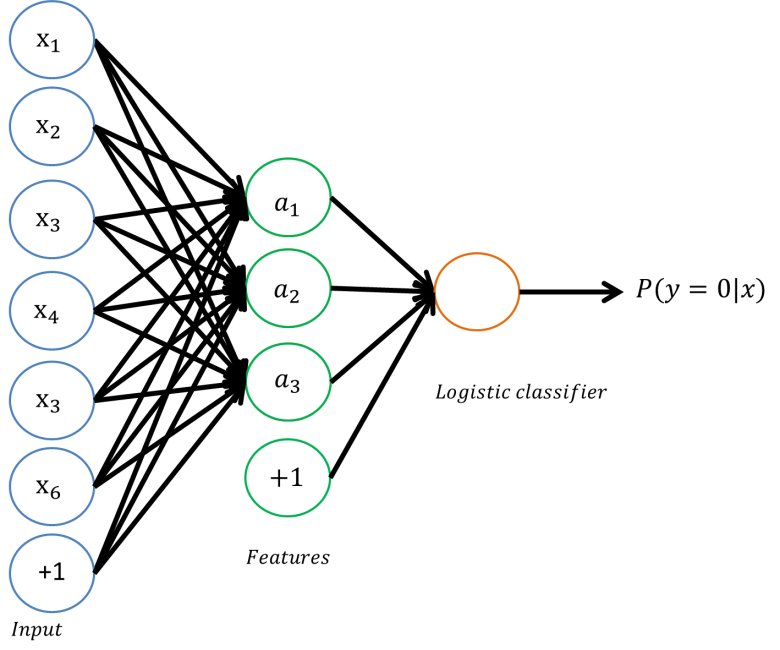
**Figure 2.2. Autoencoder as feature extractor.**



**Figure 2.3. Features/activations of the first half of an autoencoder.**

For a classification task with output labels given labeled training examples , the activations at hidden layer can be used to represent original inputs. That means we can feed into the autoencoder and get the activation and replace the original input with Then, the training set becomes

Finally, we can train a classifier such as SVM, logistic regression, softmax, etc. using the new training set by mapping features to labels The final classifier is shown in Figure 2.4.



**Figure 2.4. Self-taught learning model for supervised learning**

## . Deep learning: Stacked Autoencoder

In the self-taught learning model in Figure 2.4, the parameters are trained in two stages: 1) the weights of the first layer are the autoencoder weights between input and hidden unit activation and 2) the weights of the second layer weights are the logistic regression weights between hidden layer and the logistic classifier.

The overall classifier is a multi-layer neural network, which can be refined (or fine-tuned) by using labeled training data to further reduce the training error. Usually this step is referred to as the fine-tuning step and the previous steps are pre-training.

A deep neural network contains multiple hidden layers, which can extract more complex or higher-level features because each hidden layer performs additional feature extraction based on the previous layer’s output. An efficient method for training a deep network is to train each layer separately. A stacked autoencoder consists of multiple layers of sparse autoencoders in which the outputs of each layer are connected to the inputs of the successive layer. Now assume a stacked autoencoder with layers and be the parameters of the kth autoencoder. The encoding step is running then encoding stack of each layer in forward order:

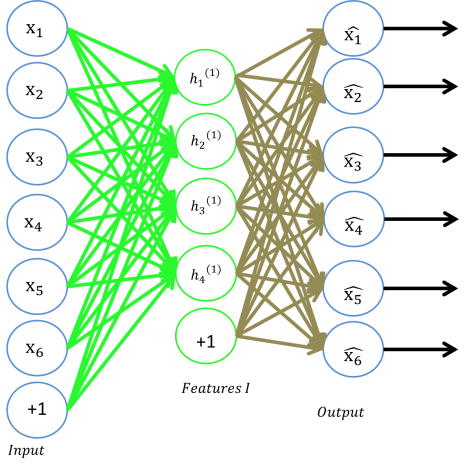
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

The decoding step is running the decoding stack of each layer in reverse order:

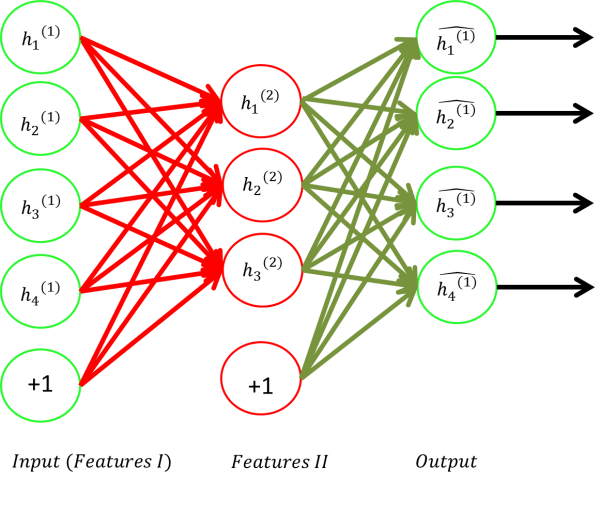
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

The activations of the hidden units represent the input in terms of higher level features.

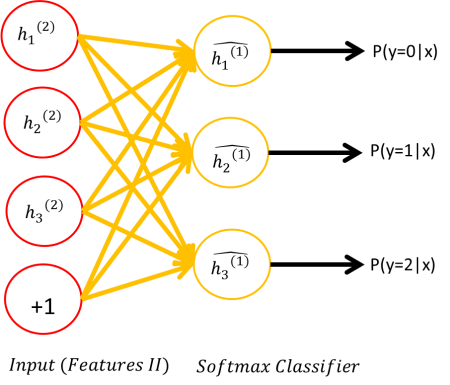
A stacked autoencoder is usually trained in a greedy layer-wise fashion. The first layer (input layer) is trained using raw input to get its parameters , as the sample shown in Figure 2.5. The first layer parameters transform raw input into the activations of the hidden units. Then, the second layer is trained to get its parameters using the outputs of the first layer, as the example shown in Figure 2.6. More hidden layers can be stacked in the same way. In the end, to use the stacked autoencoders to construct a classifier, the outputs of the last hidden layer units become the inputs of a softmax classifier, as the example shown in Figure 2.7. The softmax layer is trained on its own using the training examples. As the example shown in Figure 2.8, a deep network classifier consists of multiple autoencoder layers and one softmax layer.



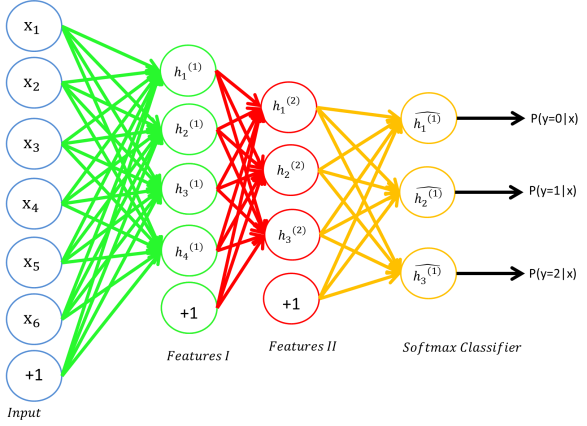
**Figure 2.5. First layer of a stacked sparse autoencoder**



**Figure 2.6. Second layer of a stacked sparse autoencoder**



**Figure 2.7. Softmax classifer layer of a stacked sparse autoencoder classifier**



**Figure 2.8. the final architecture of a stacked sparse autoencoder classifier.**

# . Performance study

## Comparison of Different Preprocessing Techniques

PCA, PCA-whitening and ZCA-whitening are three different ways to pre-process raw data. This experiment compares their effects on softmax classifiers’ classification accuracy based on the MNIST dataset.

There are 60,000 training samples and 10,000 test examples. Input size is 784, corresponding to the 28 × 28 pixel input image size. In PCA, variance retention rate of 90%, 95%, and 99% corresponds to reducing the input data to 65, 121, and 301 dimensions, respectively. The result is shown in Figure 3.1.

**Figure 3.1. Preprocessing comparison**

It uses input generated by PCA with different variance retention rates, PCA followed by whitening, and ZCA following by whitening based on the MNIST dataset.

The result shows that the accuracy increases slightly as the retention rate in PCA increases. The accuracy of PCA is nearly the same as that of PCA-whitening. ZCA-whitening is very poor for low retention rate and is about the same as the two other methods on raw data, i.e., when the retention rate is 1. Thus ZCA whitening does not work well on reduced dimensions and should be used with all dimensions of the data. Whitening has little effect on classification accuracy on this data set.

## Computation Time of Different Preprocessing Techniques

In this experiment, the computation time and classification accuracy is tested with and without PCA pre-processing using stacked neural network. The raw dataset are of high dimensional, and each experiment requires nearly two hours for a two stacked autoencode plus softmax to run. PCA can improve the experiment time and has some small tolerate error on the classification accuracy.

First, one layer of autoencoder and softmax system is used to test PCA. The hidden layer has 200 nodes. Four sets of the experiment are performed. Retaining rate is set to 90%, 95%, 99%, and 100%, respectively. In each set, experiments are performed ten times. The training set has 60000 samples and the testing set has 10000 samples. Two layers of autoencoder and softmax are then tested using PCA. Both hidden layer are set to 200. Similar experiments are performed as one layer of autoencoder and softmax. The comparison classification accuracy result is shown in Figure 3.2. The average experiment time is shown in Figure 3.3. In the Figure 3.3 and the figures afterwards, “bft” refers to before-fine-tuning, and “aft” refers to after-fine-tuning.

**Figure 3.2. Accuracy comparison**

Results from one and two autoencoder plus Softmax classifiers using input generated by PCA with different variance retention rates based on the MNIST dataset

**Figure 3.3. Average time comparison**

Results from one and two autoencoder plus Softmax classifiers using input generated by PCA with different variance retention rates based on the MNIST dataset

The results show that the after-fine-tuning accuracy of retain 99% PCA is almost the same as that one without PCA. The average running time of the experiment with PCA is shorter than comparing to that of the experiment on raw data without PCA.

Using PCA with high retention rate (99%) can have nearly as good performance as the one without PCA, but the experiment time can be highly reduced as trade-off.

## Shallow Autoencoder vs. Deep Stacked Autoencoder

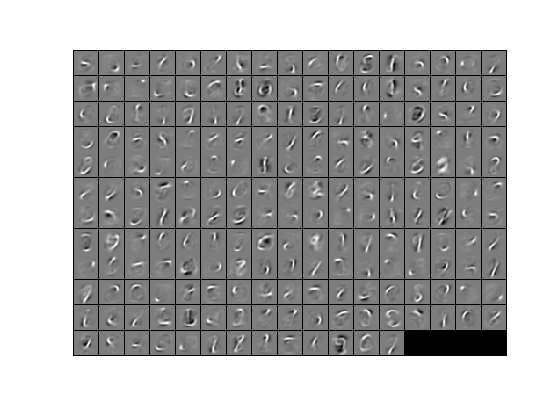
In this experiment, the performance of different classification models is tested as follows: a supervised learning classifier (Softmax), one layer of neural network plus softmax, and two layers of neural network plus softmax. The performance comparison result is shown in Figure 3.6.

**Figure 3.4. Shallow vs. Deep SAE comparison**

Results from different model: softmax, one-layer autoencoder and two-layer stacked autoencoder using input of MNIST dataset

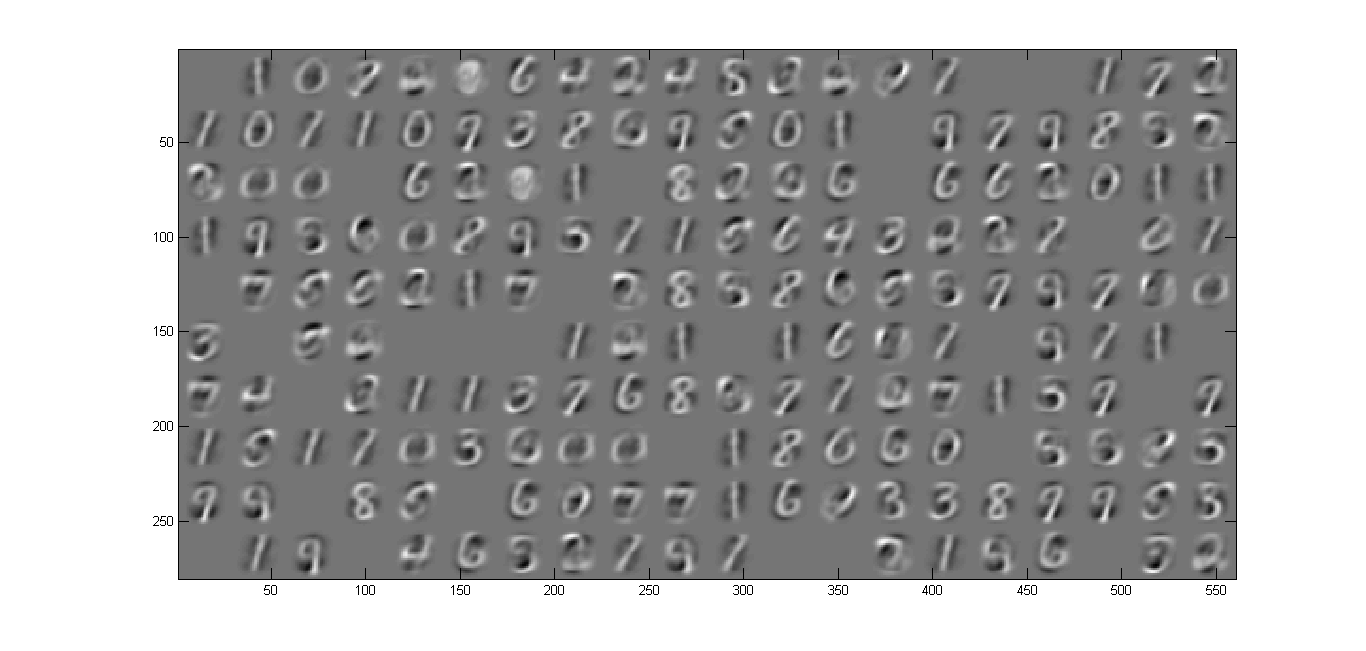
The results show that deeper networks have higher classification accuracy. After fine-tuning, the accuracy is higher than before fine-tuning.

A deep neural network can compactly represent a significantly larger set of functions than shallow networks. By using a deep network, in the case of images, the first layer will learn to group pixels together to detect edges, as shown in Figure 3.7. The second layer will group edges together to detect contours, as shown in Figure 3.8.



**Figure 3.5. Edge features**

Represented by the hidden units in the first layer of a deep learning network



**Figure 3.6. Contours features**

Represented by the hidden units in the second layer of a deep learning network

## Effects of Different Training Sizes

This experiment shows how the size of training set affect classification performance is tested on two-hidden-layer stacked deep learning networks.

MNIST digits 0~9 is the experiment data. 60000 samples are used as unlabeled data. The labeled training data size is 60, 600, 6000, and 60000 respectively. 10000 samples are used to be test dataset. Both hidden layers are set 200. The classification performance using different size of training set is shown in Figure 3.9.

**Figure 3.7. Effects of Different Training Sizes**

Results from two stacked autoencoder plus Softmax classifiers using different training data size generated from the MNIST dataset

The results show that the before and after-fine-tuning accuracy is increasing as the training data size increases. In 60 training data case, the after-fine-tuning accuracy is lower than before-fine-tuning accuracy. The fine-tuning technique should be used when we have a large labeled training set; in this way, fine-tuning can significantly improve the performance of the classifier. However, if there is a large unlabeled dataset and only a relatively small labeled training set, then fine-tuning is significantly less likely to be helpful.

## Self-taught Learning vs. Semi-supervised Learning

The first experiment is to recognize digit 0 to 4. Images of digit 5 to 9, about 30000 samples, are used as unlabeled data for feature learning. Then 14700 images of digit 0 to 4 are used as labeled training data to train the softmax classifier layer. Another 14700 images of digit 0 to 4 are used as labeled testing set. Both hidden layers have 200 nodes. Then, a similar experiment is performed for recognizing digit 5 to 9, while using images of digit 0 to 4 as unlabeled data. The recognition accuracy for digit 0 to 4, the first experiment, is shown in Figure 3.10. and the recognition accuracy for digit 5 to 9, the 2nd experiment, in Figure 3.11.

**Figure 3.8. Prediction Accuracy of digit 0-4**

Results of predicting digit 0-4 using different deep learning model using input generated from the MNIST dataset

**Figure 3.9. Prediction Accuracy of digit 5-9**

Results of predicting digit 5-9 using different deep learning model using input generated from the MNIST dataset

The results show that the accuracy of using two layer stacked autoencoder to predict digit 5-9 is lower than that of predicting 0-4. The same phenomenon appears in the one layer autoencoder. The accuracy of predicting digit 5-9 is lower than that of predicting digit 0-4. Self-taught learning is general and powerful. The unlabeled data set is drawn from a different distribution as labeled data. Digit 5-9 is more complex than digit 0-4, so the average accuracy is lower.

## Stacked Autoencoders with Different Hidden Sizes

This experiment shows how a hidden layer size of an autoencoder affects the classification accuracy is tested. The hidden layer of the stacked autoencoder can extract high level features from the input layer. The sparsity term is also used to control the nodes in hidden layer being activated.

In this experiment, two stacked autoencdoer plus Softmax classifier model is used. The number of neural nodes in input layer is the same as the dimension of the input data. The training data has 60000 samples and the testing data has 10000 samples. PCA with retaining rate of 99% is used to preprocess data. Both hidden layer nodes are set as: 100, 200, 300, 400, and 500 respectively. The sparsity term is 0.1. The accuracy result of different hidden layer size shown in Figure 3.12.

**Figure 3.10. SAE Performance with Different Hidden Sizes**

Prediction accuracy of two layer stacked autoencoder with different hidden size using input generated by PCA with retention rates 99% based on the MNIST dataset.

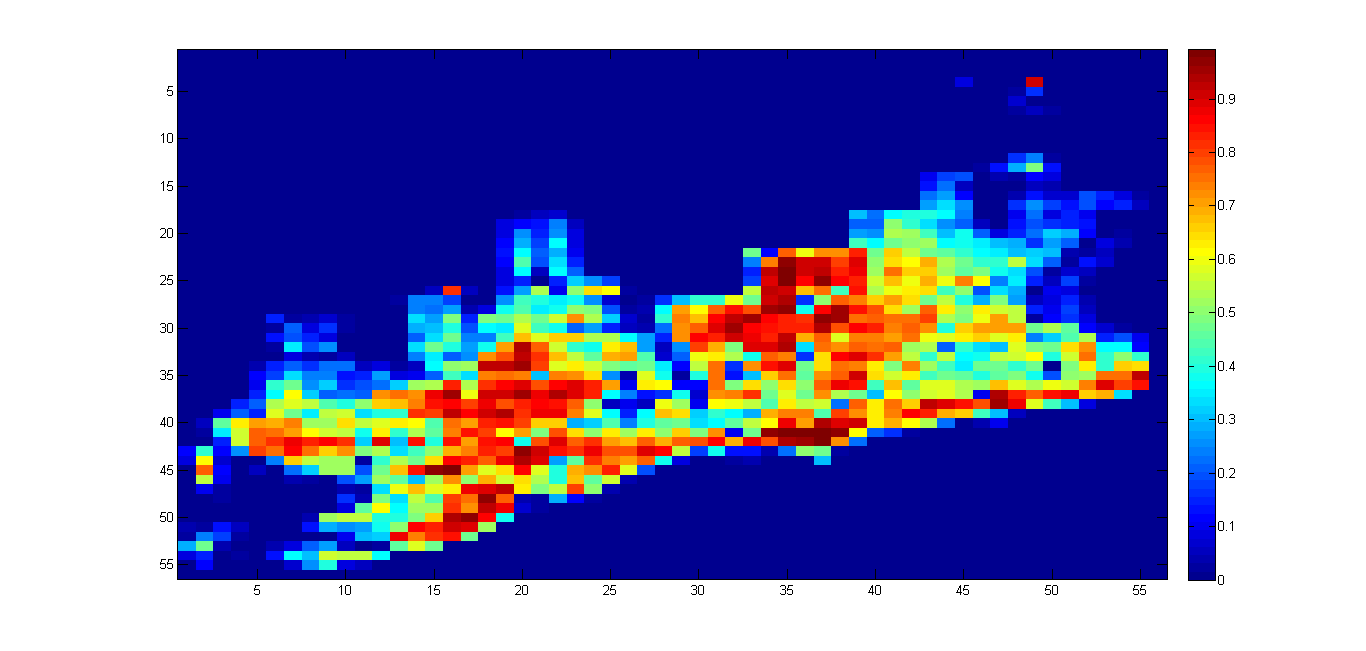
The results show that a hidden size of 200 has the highest accuracy. If the hidden size is too small, the accuracy is as low as random guessing. If hidden size is too high, the accuracy has not much improvement, but the experiment requires much longer running time.

# . Application

## Problem formulation

In this chapter, deep learning network is applied into a real world problem: crime prediction. Chicago historical data is used to train a deep belief network and then use the model to predict future crimes.

In order to get an overview crime distribution of Chicago, the heatmap of crime frequency is plotted. The map is divided into 56×56 grids. The timeline is divided into 4668 days. The plot of Chicago heatmap baseline (frequency) is shown in Figure 4.1.

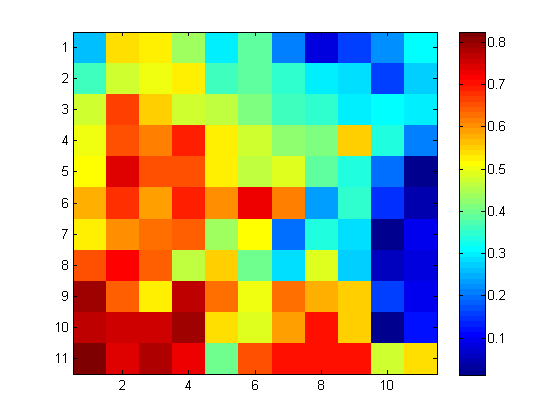


**Figure 4.1. Chicago crime heatmap baseline**

the map is 56×56 pixels, timeline is divied into 4668 days

Instead of the whole crime map, a small heatmap is used in the prediction. This is because the darker regions of the map have no crime evidence over the thirteen years. There is no need to predict on those no-crime zones.

A small focused crime heatmap of 11×11 grids over 4668 days is picked. The regions on which there is 50% chance that crime may happen over 4668 days is used to do the first experiment. It is more reasonable to predict on those fair-game regions. The focused crime heatmap on region {20:30, 40:50} of the whole crime heatmap is shown in Figure 4.2.



**Figure 4.2. Focused crime heatmap**

Pick region {20:30, 40:50} of the whole crime heatmap, map is 11×11 pixels and timeline is divied into 4668 days

## Dataset

The input data features are 11×11 pixels. Crime data has distributed in 4668 days so 4667 samples with 4667 targets is created. On the focused crime heatmap, the regions on which there is 50% chance that crime may happen are picked. After filtering, there are nine such regions, so they are set to be target region respectively. That is to say, 11×11 crime regions are used in time *t* as input data. The picked nine crime regions in time *t*+1 become targets, respectively. The train data has 3734 samples and the testing data has 934 samples. The input size is 11×11 pixels. If using autoencoder, the hidden layer size is 200. Sparsity term is set to 10%.

## Experimental results regions (50% chance crimes occur)

The crime prediction average accuracy on nine picked region use three different deep belief network model is shown in Figure 4.3.

**Figure 4.3. Prediction on picked regions**

Prediction on picked regions of the focused cirme heatmap, where crimes have 50% chance to happen

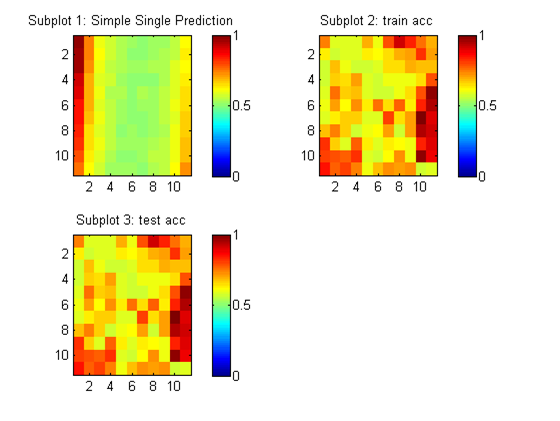
Result shows that the accuracy of softmax, one stacked autoencoder plus softmax and two stacked autoencder plus softmax are nearly same as the statistic baseline.

## 4.4 Experimental results of predicting all 121 regions

Next, crime prediction experiments are performed on all 121 focused regions. Three different system models: Softmax, one layer stacked autoencoder plus softmax, and two-layer stacked autoencoder plus softmax.

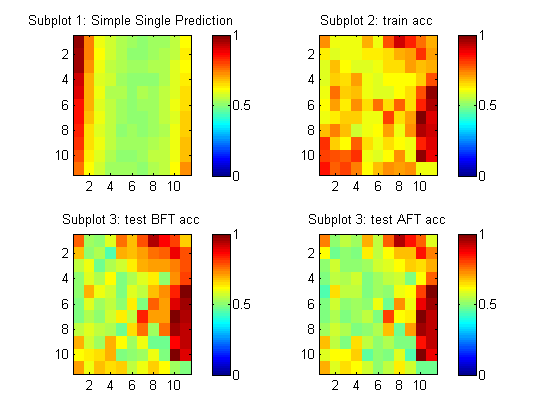
In the first model, 80% labeled crime data to train the softmax and use rest 20% to test are used. The softmax prediction accuracy with simple single prediction is shown is Figure 4.4.

Then, one layer autoencoder and softmax system is trained. The prediction accuracy heatmap is shown in Figure 4-5. After that, two layer autoencoder and softmax system are trained. The prediction accuracy heatmap is shown in Figure 4.6.



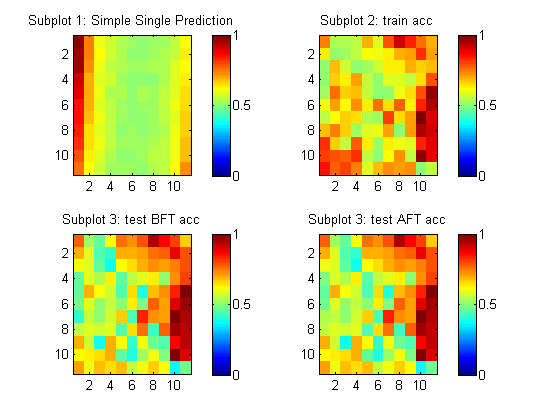
**Figure 4.4. Prediction** **heatmap of softmax**

Simple single prediction and softmax prediction accuracy of 121 regions picked from the heatmap



**Figure 4.5. Prediction** **heatmap of 1SAE**

Simple single prediction and 1SAE prediction accuracy of 121 regions picked from the heatmap.



**Figure 4.6. Prediction** **heatmap of 2SAE**

Simple single prediction and 2SAE prediction accuracy of 121 regions picked from the heatmap.

In order to visualize and compare the result in detail information, graph comparing the baseline with different prediction accuracies from different prediction system is plotted in Figure 4.7.

**Figure 4.7. Accuracy Comparison with FP**

Baseline, simple single location and frequency prediction accuracy of 121 regions picked from the heatmap

Figure 4.7. shows that in low frequency region, frequency prediction accuracy is higher than simple single prediction accuracy. In higher dimension, the simple single location prediction accuracy is same as frequency prediction.

The single location and Softmax prediction accuracy is plotted in Figure 4.8.

**Figure 4.8. Accuracy Comparison with Softmax**

Baseline, simple single location and Softmax prediction accuracy of 121 regions picked from the heatmap

Figrue 4.8. shows that the training accuracy of softmax is lower than test accuracy in high frequency region. This contradicts to convention, because usually the training accuracy should be higher than the test accuracy. One possible reason is that in this case, the softmax is trained and suitable to classify high frequency region rather than low frequency region.

Simple single location and one layer autoencoder and two layer stacked autoencoder prediction accuracy is plotted in Figure 4.9 and Figure 4.10 respectively.

**Figure 4.9. Accuracy Comparison with 1SAE**

Baseline, simple single location and 1SAE + Softmax prediction accuracy of 121 regions picked from the heatmap

**Figure 4.10. Accuracy Comparison with 2SAE**

Baseline, simple single location and 2SAE + Softmax prediction accuracy

As we can see from the above two plots, the stacked autoencoder has test accuracy lower than training accuracy in the high frequency region comparing to the low frequency region.

The summary of average accuracy of all 121 regions is plotted in Figure 4.11.

**Figure 4.11. Summary of average accuracy**

Results from all regions using different prediction methods of 121 regions picked from the heatmap

Figure 4.11 shows that the two layer stacked autoencoder accuracy is better than one layer stacked autoencoder. The deep learning network has better prediction results than simple single location prediction. Two layer stacked autoencoder accuracy is worse than baseline prediction accuracy. The reason is baseline prediction uses all data set to make a prediction. The input data contains time information range from 4000 days. However, the stacked autoencoder system developed in this example takes the input data of each individual day.

For future experiments, the model can be designed as to take input of multiple days, especially if prediction wants to be made on a certain day of the week (e.g. Tuesday), then Monday, last Monday data can be used to train the autoencoder. Another idea is to train seven different autoencoders used for seven days. And specific autoencoder can be used to make prediction on that specific day.

# . Future Work

The performance of deep belief network can be further studied. In this research, gray scale pictures are used as input and used to predict. Voice input, video input or other color pictures may be used as input in future experiment. In this research, PCA is used to perform linear dimension reduction. Future studies can try different methods of dimension reduction. The classifier used in this thesis is softmax. Future studies can use other supervised classifier such as SVM or kNN. Based on different problem, different depth deep belief network can be used. The MNIST data are two dimension digits and two-layer autoencoder plus softmax is sufficient to extract high level features (penstroke, contour) to perform well in prediction tasks. A deeper network like three, four, even five layer deep network also may be used in future works. From our experiment results, the after-fine-tuning accuracy from two-layer network doesn’t improve much comparing with that one-layer network. There is no need to build deeper network since it will be even more time consuming. In future experiments, different dataset may also be used to test the performance: psychological data, gene data, etc. The models can be developed and tested accordingly. Self-taught learning is more general than semi-supervised learning. In the future studies, more unlabeled data can be used to train the sparse autoencoder and let it learn high-level features for classification use purpose.

# . Summary

First, an extensive set of experiments of various types of deep network structures, together with various pre-processing dimension-reduction techniques, are conducted on the MNIST handwritten digits image recognition data set to understand their performances and quality-cost trade-offs.

Various deep learning networks are applied to future crime prediction based on historical crime records. Extensive experimental results show that deep networks achieve better results than a basic prediction algorithm.

# 7. Bibliography

1. MNIST handwritten digits image recognition dataset. Avaible via WWW: <http://yann.lecun.com/exdb/mnist/>
2. City of Chicago Crime Data Portal. Avaiable via WWW: <https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2>
3. Barthe, Emmanuel. Crime Prevention Publicity Campaigns. U.S. Department of Justice. 2006, p.9.
4. Adderley, R., and Musgrove, P.B., (1999) Data Mining at the West Midlands Police: A Study of Bogus Official Burglaries, BCS Specialist Group on Expert Systems, ES99, London, Springer – Verlag, pp191-203, 1999.
5. Grover, Vikas, Richard Adderley, and Max Bramer. "Review of current crime prediction techniques." Applications and Innovations in Intelligent Systems XIV. Springer London, 2007. 233-237.
6. PredPol. Crime prediction software, Available via WWW: <http://www.predpol.com/>
7. CommandCentral Predictive. Crime prediction software, available via WWW: <http://www.publicengines.com/products/commandcentral-predictive.php>
8. Gorr, W. and R. Harries, Introduction to crime forecasting. InternationalJournal of Forecasting, 2003. 19(4): p. 551-555.
9. Brantingham, P. and P. Brantingham, Patterns in Crime. 1984, NewYork: Macmillan Publishing Company.
10. Clarke, R.V. and D.B. Cornish, Modeling Offenders’ Decisions: A Framework for Research and Policy, in Crime and Justice: An Annual Review of Research, M. Tonry and N. Morris, Editors. 1985, University of Chicago Press: Chicago.
11. Clarke, R.V. and M. Felson, Introduction: Criminology, Routine Activity, and Rational Choice, in Routine Activity and Rational Choice, R.V. Clarke and M. Felson, Editors. 1993, Transaction Publishers: New Brunswick London.
12. Bowers, K.J., S.D. Johnson, and K. Pease, Prospective hot-spotting: The future of crime mapping? British Journal of Criminology, 2004. 44(5): p. 641-658.
13. Lin, S. and D.E. Brown, An outlier-based data association method for linking criminal incidents. Decision Support Systems, 2004. 41(2006): p. 604-615.
14. Liu, H. and D. Brown, Criminal incident prediction using a pointpattern-based density model. International Journal of Forecasting, 2003.19(4): p. 603-622.
15. Oatley, G. and E. Brian, Crimes analysis software: pins in maps, clustering and bayes net prediction. Expert Systems with Applications, 2003. 25(4): p. 569 - 588.
16. Oatley, G.C. and B.W. Ewart, Applying the concept of revictimization: using burglars' behaviour to predict houses at risk of future victimization. International Journal of Police Science and Management, 2003. 5(2).
17. Sentinent, Data Detective. 2007, Sentinent Information Systems: Amsterdam, The Netherlands.
18. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313, 504-507
19. Mohamed, A.; Dahl, G.E.; Hinton, G.; , "Acoustic Modeling Using Deep Belief Networks," Audio, Speech, and Language Processing, IEEE Transactions on , vol.20, no.1, pp.14-22, Jan. 2012
20. Shusen Zhou; Qingcai Chen; Xiaolong Wang; , "Discriminative Deep Belief Networks for image classification," Image Processing (ICIP), 2010 17th IEEE International Conference on , vol., no., pp.1561-1564, 26-29 Sept. 2010
21. Miaozhen Lin; Xin Fan; , "Low resolution face recognition with pose variations using deep belief networks," Image and Signal Processing (CISP), 2011 4th International Congress on , vol.3, no., pp.1522-1526, 15-17 Oct. 2011
22. [Honglak Lee , Roger Grosse , Rajesh Ranganath , Andrew Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”, Proceedings of the 26th Annual International Conference on Machine Learning, p.609-616, June 14-18, 2009, Montreal, Quebec, Canada](http://dl.acm.org/citation.cfm?id=1553453&CFID=98389300&CFTOKEN=74985722)
23. Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. Science 2006.
24. Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. Greedy Layer-Wise Training of Deep Networks. NIPS 2006
25. Pascal Vincent, Hugo Larochelle, Yoshua Bengio and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. ICML 2008.
26. UFLDL Tutorial. Available via WWW: <http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial>