Artificial Intelligence Cource Project

CS410

Ruijie Wang
515021910338
Wjerry5@sjtu.edu.cn

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

I. INTRODUCTION

II. METHOD

A. Data Processing

Since the given dataset is a large p small n problem, dimensionality reduction is required at first. In this part, we will introduce two dimensionality reduction methods: PCA and Autoencoder. Later we compare both their performances and their individual result under different dimension using a same SVM classifier.

1) PCA: Principal components analysis(PCA) aims to perform a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. Suppose we are given data matrix X with size $n \times p$, where n is the quality of sampling and p is the dimension of feature. The PCA algorithm is shown as follow:

- Normalize the data : $x = x - \sum_i^N x_i$;
- Calculate the covariance matrix $C = \frac{X^T X}{n-1}$. Then diagonalize it : $C = VLV^T$, where V is the matrix of eigenvectors and L is diagonal matrix with eigenvalues $_i$.
- Choose d rows ,the sum of whose variance reach some percentage of original variance. The criterion should be followed Eq(1).

$$\frac{\sum_i^d \lambda_i}{\sum_i^n \lambda_i} \geq Threshold \tag{1}$$

The final p is determined by $Threshold$. PCA is based on the idea that the feature has large variance but the noise has low variance.

2) Autoencoder: An autoencoder neural network is an unsupervised learning algorithm that applies back propagation, setting the target values to be equal to the inputs. The structure of autoencoder is show as Fig(1):

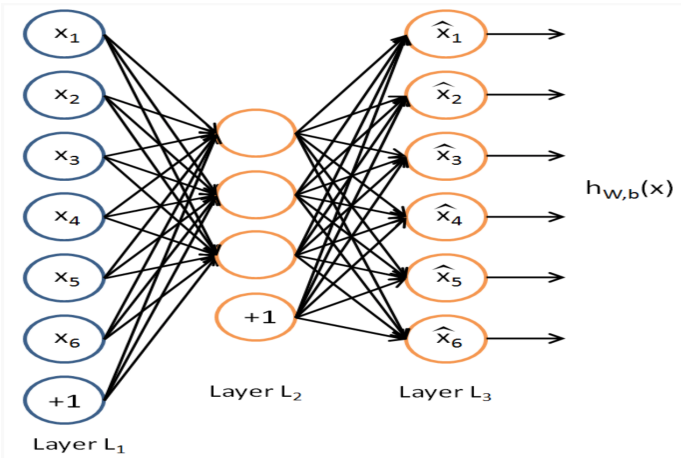

Fig. 1. structure of autoencoder

Now suppose we have a set of unlabeled training examples $X = [x_1, x_2, x_3, \dots]$, we want $X = H_{W,b}(X)$. Then the lower-dimension output of hidden layer L2 is the encoder of the raw data. This output is a nonlinear dimensionality reduction, which may overcome the weakness of PCA that it is only a linear dimensionality reduction method.

B. Classical Methods

1) Logistic Regression: Logistic regression is a classical supervised linear classifier. Generally speaking, logistic regression firstly calculates the

boundary among different classes, then it can predict the possibility of test data class based on the calculated data boundary.

Suppose now that we are given all of the training dataset annotated with labels, while $\mathbf{x}$ denotes the feature vector and $\mathbf{Y}$ denotes label, then the calculated class possibility is defined as Eq(2):

$$P(Y = y_i|x_i) = \frac{e^{\omega x_i + b}}{1 + \sum_i e^{\omega x_i + b}} \tag{2}$$

where ω denotes the regression weights and b denotes the regression bias. Based on the labels training data, we can determine the value of ω and b using Maximum Likelihood Estimation(MLE) method. Assume $h_\omega(x_i)$ denoting the possibilities, the log likelihood function is defined as Eq(3):

$$L(\omega) = \sum_i^N [y_i log h_\omega(x_i) + (1 - y_i) log(1 - h_\omega(x_i))] \tag{3}$$

Various gradient-based optimize algorithm can be used to determine the value of ω and b. In our work, we use logistic regression in both multiple classification task and binary classification task and we compare its performance under various optimization and regularization methods.

C. Deep Learing Based Methods

Deep learning has been shown as a successful machine learning method for a variety of tasks. In our work, we use a deep neural network with fully-connected multilayer structure both on PCA-processed data and on raw data with **greedy layer-wise pre training**. Besides, we add L2 regularization in each fully-connected layer to avoid overfitting and control the model complexity.

1) network structure: Since the features do not have any locality or direct relationship, we dont use convolutions.

- **nn on pca-processed data**
  In this model, firstly we use pca-processed data (500-dimension) as our network input. Then we add hidden layer1(256-dimension), hidden layer2(512-dimension), hidden layer3(256-dimension), finally a output layer. The structure is shown as Fig(2) network i. For weight initialization, we initialize each neurons weight vector as a random vector sampled from a multi-dimensional gaussian. For bias initialization, we set the bias vectors all to 0. We initialize our weight with the consideration that the size of our network is limited and it can be trained easily according to the network ii.
- **nn on raw data**
  In this model, we use the raw data as the network inputs. Then we add hidden layer1(8192-dimension), hidden layer2(512-dimension), hidden layer3(128-dimension), finally a output layer. The structure is shown as Fig(2) network ii. Since we did not use the low-dimension data, the training become much harder according to out experiment. So the network initialization is very import. We use greedy layer-wise pre training , which is a pre-train method based on the idea of autoencoder, to initialize the network.

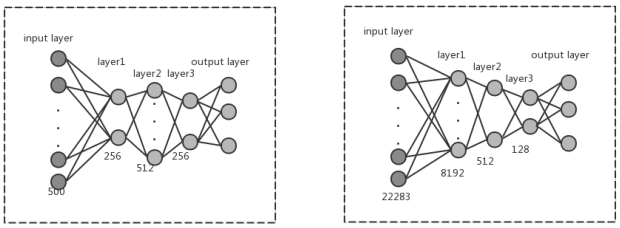

Fig. 2. a.network i; b, network ii

2) Regularization: We use L2 regularization in each fullt-connected layer.It can be implemented by penalizing the squared magnitude of all parameters directly in the objective. For every weight w in the network, we add the term $1/2w$ to the objective, where is the regularization strength. This can help control the capacity of neural networks to prevent overfitting.

Furthermore, we employ Dropout to prevent overfit- ting. While training, dropout is implemented by only keep- ing a neuron active with some probability p, or setting it to zero otherwise.

3) Pre-train: In network ii, the training get very hard because of the high dimension of the network. We use **greedy layer-wise pre training** to pre-train. It will determine the initial weights of each

layer layer by layer with the idea of autoencoder. Since each layer has been local optimal solution, the training get much easier .

More specifically, to determine weights of first hidden layer, we use the network shown as Fig(1). Once we get $X = h_{\omega,b}(X)$, it means the hidden layer can encode the input properly, at the same time it can pass these extracted feature to the following layer. Seemingly, when we initialize the second hidden layer, we use the output of the first hidden layer as the input of network in Fig(1). Thus we can determine all the weights. The process is show as Fig(3). The network ivin Fig(3) is we want, so to determine the 3 layers weights, we use network i-network iii(which are same as Fig(1)) to train them.

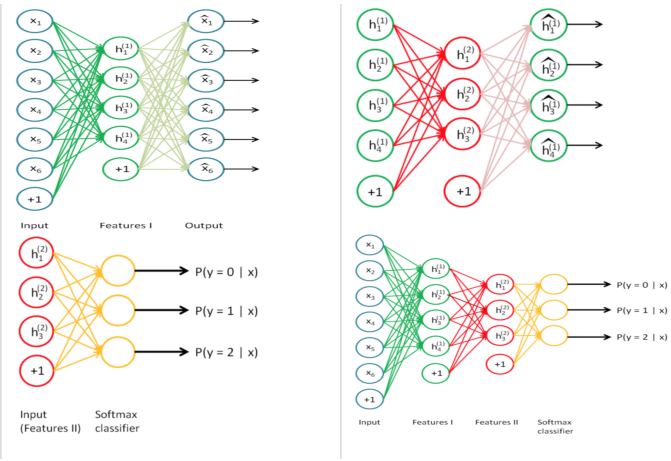

Fig. 3. a.network i; b, network ii; c, network iii; d, network iv

III. RESULT