Postprint of Handwritten Digit Recognition Algorithm Based on Jacobian Sparse Autoencoder

Authors: Wang Huiling, Song Wei

Date: 2018-05-20T00:00:00+00:00

Abstract

Due to significant variations in the edge contours of handwritten digits that result in diverse writing styles, we propose a Jacobian Sparse Autoencoder (JSAE) algorithm for handwritten digit recognition by incorporating sparse constraint terms and Jacobian regularization terms into the autoencoder to enhance recognition accuracy. The sparse constraint terms can effectively extract hidden structures within the data, while Jacobian regularization can characterize the edge features of data points, thereby improving the learning capability of the autoencoder algorithm and enabling more accurate extraction of the intrinsic features of samples. Experimental results demonstrate that JSAE achieves higher classification accuracy than the Autoencoder (AE) and Sparse Autoencoder (SAE) algorithms.

Full Text

Preamble

Title: Handwritten Digit Recognition Algorithm Based on Jacobian Sparse Auto-Encoder

Authors: Wang Huiling[1], Song Wei[1],[2] [1]School of Internet of Things Engineering, [2]Engineering Research Center of Internet of Things Technology Applications, Ministry of Education, Jiangnan University, Wuxi, Jiangsu 214122, China

Abstract: Due to significant variations in edge contours that lead to different handwriting styles, this paper proposes a Jacobian regularized sparse autoencoder (JSAE) algorithm for handwritten digit recognition to improve recognition accuracy. The algorithm incorporates both sparse constraints and Jacobian regularization into the auto-encoder framework. The sparse constraint term effectively extracts hidden structures from data, while Jacobian regularization can characterize the marginal features of data points and enhance the learning capability of the auto-encoder algorithm, thereby more accurately extracting

the essential features of samples. Experimental results demonstrate that JSAE achieves higher classification accuracy than both basic auto-encoders (AE) and sparse auto-encoders (SAE).

Keywords: handwritten digit recognition; Jacobian regularization; sparse constraint; auto-encoder; marginal feature

0 Introduction

Handwritten digit recognition has found increasingly widespread application in information technology, with researchers extending its use to various practical domains including large-scale data statistics, finance, taxation, banking, and mail sorting. Existing handwritten digit recognition algorithms can be categorized into two main types based on how they extract character features. The first category comprises structural feature-based algorithms that identify basic elements such as concave region features, contour features, and structural mutation point features within character images, employing template matching for automatic recognition. While these methods can intuitively describe character structure, they suffer from poor robustness to character deformation and noise. The second category consists of statistical feature-based algorithms that rely on representation, transformation, and learning from large sample sets. By estimating the feature space distributions of different sample categories, these algorithms train corresponding classifiers for unknown pattern classification. When training samples are sufficiently comprehensive, such algorithms demonstrate strong recognition capability.

Deep learning has evolved from artificial neural networks, with multi-layer perceptrons containing multiple hidden layers representing deep architectures. Deep learning discovers distributed data features by combining low-level features to form abstract high-level representations of object properties or characteristics. Unlike traditional neural networks that employ gradient descent methods—randomly initializing parameters, computing results, calculating errors, and making corrections through iterative training—deep learning addresses common issues such as data acquisition, local extrema, and gradient diffusion. In deep architectures, gradient descent applied to deeper layers causes residuals to become extremely small, leading to gradient diffusion and ineffective correction. Consequently, deep learning adopts layer-wise initialization through unsupervised learning.

Auto-encoder (AE) is a promising non-probabilistic machine learning algorithm that effectively learns representative features to discover potential explanatory factors hidden within massive datasets. In recent years, auto-encoders have attracted increasing attention and been applied to machine learning and signal processing tasks including domain adaptation, hashing, face alignment, human gesture recognition, speech information processing, and natural language processing. As an unsupervised learning algorithm, auto-encoder attempts to learn

a function that approximates the input with its output, extracting high-level features from input data. The model learns parameters by minimizing the error between input samples and their reconstructions.

To obtain better generalized features, researchers have proposed various regularized auto-encoders to capture underlying data distribution structures. These generally fall into four categories: sparse auto-encoders, denoising auto-encoders, contractive auto-encoders, and Laplacian auto-encoders. Sparse auto-encoders incorporate sparsity constraints on hidden layer nodes to learn data features, obtaining more meaningful representations of input data dimensions. Denoising auto-encoders minimize denoising reconstruction error to reconstruct true original inputs from randomly corrupted data. Contractive auto-encoders optimize the sum of reconstruction error and a contraction penalty term, minimizing the objective function to make learned feature representations as invariant as possible to input data. Laplacian auto-encoders use Laplacian regularization to enhance the local preservation performance of encoders learned at data points.

Inspired by these representative auto-encoder concepts, this paper integrates Jacobian regularization and sparsity constraints into the auto-encoder algorithm, proposing a novel auto-encoder called Jacobian Sparse Auto-Encoder (JSAE). The proposed JSAE offers two advantages: (a) Incorporating Jacobian regularization into the encoding process enables JSAE to preserve edge feature structures, making feature extraction more complete and information utilization more comprehensive; (b) Adding sparsity constraints to the encoding process provides JSAE with strong robustness to noise while effectively extracting hidden structures from data, achieving better classification performance.

[Figure 1: see original paper] shows the auto-encoder structure diagram, and [Figure 2: see original paper] illustrates the auto-encoder model. To evaluate effectiveness, extensive experiments were conducted on handwritten digit datasets, comparing JSAE with auto-encoders and sparse auto-encoders. Results demonstrate JSAE' s superior accuracy.

1 Auto-Encoder (AE)

The auto-encoder concept was proposed in the late 1980s. A basic auto-encoder consists of a three-layer neural network comprising an input layer, hidden layer, and output layer, with identical numbers of neurons in the input and output layers. [Figure 1: see original paper] depicts this structure, where Layer L1 is the input layer, Layer L2 is the hidden layer, and Layer L3 is the output layer. Neurons between input-hidden and hidden-output layers are connected by different weights, with each hidden and output layer neuron connected to a bias term. Hidden layer node counts can be set according to specific requirements.

[Figure 2: see original paper] presents the auto-encoder model, where function f maps input vector x to obtain intermediate representation y, and function g maps this intermediate representation to obtain reconstructed data z. The auto-encoder adjusts and optimizes network weights by minimizing the loss function

$L(x, z)$. The auto-encoder training algorithm uses backpropagation, with the loss function comprising reconstruction error terms. Let $a_i^{(w,b)}(x)$ denote the activation value of the i-th neuron, J_{wd} the weight decay term aimed at reducing weight magnitude to prevent overfitting, and λ the coefficient.

The auto-encoder attempts to learn a function making the output approximate the input for feature extraction. The model learns parameters $\theta = \{W, b\}$ by minimizing the error between input samples and their reconstructions:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{n} L(x^{(i)}, g(f(x^{(i)})))$$

where n is the number of training samples, $x^{(i)}$ is the i-th training sample, f and g are encoding and decoding functions respectively, and L is a loss function such as traditional squared error.

2 Jacobian Sparse Auto-Encoder (JSAE)

This paper integrates sparsity constraints and Jacobian regularization into the auto-encoder algorithm, proposing a novel auto-encoder called Jacobian Sparse Auto-Encoder (JSAE). Similar to the human brain where only certain neurons are stimulated by specific inputs while most remain inhibited, this work introduces sparse constraints to ensure the hidden layer contains only a few non-zero elements or elements significantly greater than zero. This makes hidden layer neuron activations satisfy certain sparsity requirements, enabling more effective discovery of internal structures and patterns hidden within input data. Consequently, sparse representations often prove more effective than alternative representations.

In handwritten digit images, pixels with dramatic changes—namely edge contours—are particularly important and contain rich intrinsic image information. The Jacobian regularization term computes a Jacobian matrix of input data, performing local linearization and spatial contraction to obtain image edge features. This extracts information about shape and reflectance or transmittance from images, providing invariance and robustness to small input variations. In summary, JSAE offers two advantages: First, incorporating Jacobian regularization into encoding preserves edge feature structures; second, adding sparsity constraints provides robustness to noise while effectively extracting hidden data structures, enabling superior application performance.

[Figure 3: see original paper] shows the JSAE framework for classification. The JSAE cost function is:

$$J_{JSAE} = J_{AE} + J_{wd} + J_{sparse} + J_{jacobi}$$

where λ, β, and ν control the relative importance of each term. The first term J_{AE} is the reconstruction error between training samples and their reconstructions. The second term J_{wd} prevents weight overfitting. The third term J_{sparse} is the sparsity penalty that discovers important structures in input data even with numerous hidden neurons. The fourth term J_{jacobi} describes edge features of handwritten digit recognition data points, where the Jacobian matrix is computed from training samples using the Jacobian function.

Let ρ_j denote the penalty factor for hidden neuron j, where $\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m} a_j^{(2)}(x^{(i)})$ represents the average activation of hidden neuron j, ρ is the sparsity parameter (typically a small value close to 0), and $KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log \frac{1-\rho}{1-\hat{\rho}_j}$ is the KL divergence that increases monotonically as the difference between $\hat{\rho}_j$ and ρ grows. [Figure 4: see original paper] shows the penalty factor curve as $\hat{\rho}_j$ varies.

The input vector is represented as $x \in [0,1]^n$. The encoding mapping to the hidden layer is $y = f(x) = s(Wx + b)$ where $s(x) = 1/(1+e^{-x})$ is the sigmoid function, W is the weight matrix of size $n \times s$, and b is the bias vector. The intermediate output y is then mapped to reconstruct vector $z = g(y) = W'y + b'$ where W' is a weight matrix of size $s \times n$, typically constrained such that $W' = W^T$.

When the hidden layer contains many neurons, sparsity constraints are added to ensure learning of more local and information-rich structures. Before each iteration, the average activation of hidden neurons across all training samples is computed:

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m} a_j^{(2)}(x^{(i)})$$

where m is the number of samples, i indexes samples, j indexes hidden layer neurons, and $a_j^{(2)}(x^{(i)})$ is the activation of neuron j for sample i.

Under sparsity constraints, with sparsity parameter ρ (typically close to 0), the KL divergence is used:

$$KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log \frac{1-\rho}{1-\hat{\rho}_j}$$

This term equals 0 when $\hat{\rho}_j = \rho$ and increases monotonically as the difference between $\hat{\rho}_j$ and ρ grows.

The Jacobian regularization term is:

$$J_{jacobi} = \frac{1}{2m}\sum_{i=1}^{m} \left\| J_f(x^{(i)}) \right\|_F^2$$

where $J_f(x^{(i)})$ is the Jacobian matrix computed from training samples.

This paper uses gradient descent to minimize the cost function, updating W and b through:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)$$

where α is the learning rate. The key step involves computing partial derivatives via backpropagation. Derivatives for the entire cost function $J(W, b)$ are derived from the cost for a single example $J(W, b; x, y)$:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x^{(i)}, y^{(i)}) \right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J(W, b; x^{(i)}, y^{(i)})$$

The JSAE partial derivatives are:

$$\frac{\partial J_{JSAE}}{\partial W} = \frac{\partial J_{AE}}{\partial W} + \frac{\partial J_{wd}}{\partial W} + \frac{\partial J_{sparse}}{\partial W} + \frac{\partial J_{jacobi}}{\partial W}$$

$$\frac{\partial J_{JSAE}}{\partial b} = \frac{\partial J_{AE}}{\partial b} + \frac{\partial J_{wd}}{\partial b} + \frac{\partial J_{sparse}}{\partial b} + \frac{\partial J_{jacobi}}{\partial b}$$

The complete algorithm process is described as follows:

Input: N input samples $\{x_1, x_2, ..., x_N\}$

Process: a) Forward propagation to compute hidden layer output y and reconstruction z b) Compute reconstruction error term J_{AE} c) Compute weight decay term J_{wd} to prevent overfitting d) Compute sparsity penalty term J_{sparse} to enforce sparse hidden layer activations e) Compute Jacobian regularization term J_{jacobi} to capture edge features f) Randomly initialize weights W and biases b, perform forward propagation g) Use gradient descent to minimize the cost function and obtain optimal weights

Output: Encoding parameters $\theta = \{W, b\}$ and decoding parameters $\theta' = \{W', b'\}$

The time complexity analysis shows that computing J_{AE} is $O(snm)$, computing J_{wd} and its derivatives is $O(sn)$, computing J_{sparse} and its derivatives is $O(sn)$,

and computing J_{jacobi} and its derivatives is $O(snm)$. Therefore, the total time complexity per iteration for JSAE is $O(snm)$.

3 Experimental Results and Analysis

Experiments were conducted on MATLAB R2014b with the following system configuration: Intel(R) Core i3-3240 CPU at 3.40 GHz, 4 GB memory, Microsoft Windows 7 operating system. The experimental datasets used were MNIST handwritten digit recognition dataset, Pen-Based Recognition of Handwritten Digits dataset, and USPS dataset.

3.1 Experimental Datasets

Dataset Specifications: - **MNIST Handwritten Digits**: 60,000 training images and 10,000 test images, 10 classes (digits 0-9), each image 28×28 pixels with normalized pixel values - **Pen-Based Recognition of Handwritten Digits**: 10,992 samples with 16 attributes and 10 classes - **USPS Dataset**: 9,298 handwritten digit images (16×16 pixel grayscale values, normalized) with 7,291 training and 2,007 test samples

3.2.1 MNIST Handwritten Digits

MNIST contains 60,000 training and 10,000 test handwritten digit images across 10 classes (Arabic digits 0-9). Each image has 28×28 pixels with normalized values. [Figure 5: see original paper] shows 72 randomly displayed images from the MNIST database.

To validate the proposed method, experiments were conducted using 1,500, 2,000, 2,500, 3,000, 3,500, and 4,000 training images, with the original 10,000 test images. A softmax classifier was used to classify learned encodings. JSAE was compared against basic auto-encoders and sparse auto-encoders.

[Figure 6: see original paper] shows classification accuracy of the three auto-encoders on the handwritten digit database. Due to MNIST' s large size, experiments were run 10 times with averages reported. Training samples were randomly selected while test sample numbers remained constant, causing smaller databases to show lower accuracy than the original. Across all six datasets, JSAE achieved superior classification accuracy on MNIST subsets compared to AE and SAE. By describing data point marginal features and effectively extracting hidden data structures, JSAE demonstrates more pronounced performance advantages, with accuracy increasing as training dataset size grows.

With 2,000 training samples, JSAE parameters λ, β, and ν were investigated. Parameters were varied according to rules $\{10^e | e = -6, -5, -4, -3, -2, -1, 0, 1, 2\}$. Extensive experiments showed ν variations did not affect classification accuracy because the Jacobian regularization term describes data point edge features through linearization of input data, with summation making each learned feature locally invariant. Thus, only λ and β needed discussion to reflect

parameter effects on results. [Figure 7: see original paper] illustrates JSAE classification accuracy under different λ and β values, showing stable results above 90%, with higher accuracy on the original database.

The impact of training iterations on classification accuracy was also examined. Experiments were conducted at iteration counts of 250, 350, 450, 550, 650, and 750, comparing JSAE with AE and SAE. [Figure 8: see original paper] shows that JSAE mostly achieved higher accuracy than AE and SAE at the same iteration counts, peaking at 450 iterations, while AE and SAE peaked at 150 iterations, demonstrating JSAE' s advantage with more iterations.

3.2.2 Pen-Based Recognition of Handwritten Digits Dataset

The Pen-Based Recognition of Handwritten Digits dataset contains 10,992 samples (7,494 training, 3,498 test) from the UCI Machine Learning Repository. To avoid experimental randomness, 5,496 samples were randomly selected as training data and 5,496 as test data for each experiment. [Figure 9: see original paper] compares classification accuracy across different iteration counts, showing JSAE significantly outperformed the other two auto-encoders, maintaining accuracy above 96% and reaching 97.53% at 350 iterations.

3.2.3 USPS Dataset

The USPS dataset uses the USPS handwritten digit recognition database (http://www.datatang.com/data/11927) containing 16×16 pixel grayscale images with normalized values. The dataset includes 9,298 handwritten digit images (7,291 training, 2,007 test). [Figure 10: see original paper] compares classification accuracy across different iteration counts, showing JSAE maintained relatively stable accuracy around 96%, consistently outperforming SAE and AE.

4 Conclusion

Learning effective features for handwritten digit recognition is crucial for achieving good classification performance. This paper improves upon auto-encoders by proposing a Jacobian Sparse Auto-Encoder (JSAE) that uses Jacobian regularization to describe data point marginal features, linearizes input data to extract the most meaningful features from handwritten digit images, and employs sparse constraints to effectively extract hidden data structures. By constraining the hidden layer, JSAE learns a more complex nonlinear function with strong capability for learning dataset features, enabling more accurate extraction of sample essential characteristics.

Experiments on MNIST, Pen-Based Recognition of Handwritten Digits, and USPS datasets demonstrate JSAE' s superior classification accuracy compared to auto-encoders and sparse auto-encoders, validating the theoretical approach. The analysis of training dataset size effects shows that more training data yields

higher accuracy. The investigation of cost function parameters identifies optimal values for maximum accuracy, providing practical guidance. The study of training iteration counts confirms that iterations significantly impact classification accuracy.

References

[1] Ren D, Chen X. Principle and application of handwritten digit recognition [J]. Computer Era, 2007 (3): 45-46.

[2] Song Z, Yu R. Research on handwritten digit classification based on deep learning [J]. Journal of Chongqing Technology and Business University, 2015, 32 (8): 49-53.

[3] Chen J. Research on free handwritten digit recognition algorithm based on combined structural features [J]. Computer Engineering and Applications, 2013, 49 (5): 179-184.

[4] Wu Z, Zhu G, Huang G. Handwritten digit recognition method based on image recognition technology [J]. Industrial Control Computer, 2011, 21 (12): 48-51.

[5] Shi H, Hu X, Liu R. Research on handwritten digit character recognition based on heuristic GA-SVM [J]. Computer Technology and Development, 2012, 22 (10): 5-9.

[6] Shi H. Support vector machine and its application in handwritten digit recognition [D]. Chongqing: Chongqing University, 2013: 1-30.

[7] Yang X, Zhang T, Xu C. Cross-domain feature learning in multimedia [J]. IEEE Trans on Multimedia, 2014, 17 (1): 64-78.

[8] Chen M, Xu Z, Weinberger K. Marginalized denoising auto-encoders for domain adaptation [J]. Computer Science, 2012.

[9] Carreira-Perpiñán M A, Raziperchikolaei R. Hashing with binary auto-encoders [J/OL]. (2015). http://eecs.ucmerced.edu.

[10] Zhang J, Shan S, Kan M, et al. Coarse-to-fine auto-encoder networks (CFAN) for real-time face alignment [C]// Proc of Computer Vision ECCV 2014: 1-16.

[11] Li S Z, Yu B, Wu W. Feature learning based on SAE-PCA network for human gesture recognition in RGBD images [J]. Neurocomputing, 2015, 151 (151): 107-114.

[12] Deng L, Seltzer M, Yu D, et al. Binary coding of speech spectrograms using a deep auto-encoder [C]// Proc of Interspeech. 2010.

[13] Liou C Y, Cheng C W, Liou J W, et al. Autoencoder for words [J]. Neurocomputing, 2014, 139 (139): 84-96.

[14] Schölkopf B, Platt J, Hofmann T. Efficient learning of sparse representations with an energy-based model [C]// Advances in Neural Information Processing Systems. 2006: 1137-1144.

[15] Vincent P, Larochelle H, Lajoie I. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion [J]. Journal of Machine Learning Research, 2010, 11 (12): 3371-3408.

[16] Rifai S, Vincent P, Muller X. Contractive auto-encoders: explicit invariance during feature extraction [C]// Proc of ICML. 2011.

[17] Jia K, Sun L, Gao S. Laplacian auto-encoders [J]. Neurocomputing, 2015, 160 (C): 250-260.

[18] Yang Y, Zhang W. Image automatic annotation algorithm based on deep learning [J]. Data Acquisition and Processing, 2015, 30 (1): 88-97.

[19] Guo H, Li G, Qiu B. Application of denoising auto-encoder in image recognition [J]. Journal of Jinling Institute of Technology, 2015, 31 (3): 32-35.

[20] Liu W, Ma T, Tao D. HSAE: a Hessian regularized sparse auto-encoders [J]. Neurocomputing, 2016, 187: 59-65.

Note: Figure translations are in progress. See original paper for figures.