Label-denoising Auto-Encoder for Classification with Inaccurate Supervision Information

Dong Wang
Department of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
#29 Yudao Street, Nanjing 210016, P.R.China
Email: dongwang@nuaa.edu.cn

Xiaoyang Tan
Department of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
#29 Yudao Street, Nanjing 210016, P.R.China
Email: x.tan@nuaa.edu.cn

***Abstract*—Label noise is not uncommon in machine learning applications nowadays and imposes great challenges for many existing classifiers. In this paper we propose a new type of auto-encoder coined label-denoising auto-encoder to learn a representation for robust classification under this situation. For this purpose, we include both the feature and the (noisy) label of a data point in the input layer of the auto-encoder network, and during each learning iteration, we disturb the label according to the posterior probability of the data estimated by a softmax regression classifier. The learnt representation is shown to be robust against label noise on three real-world data-sets.**

I. Introduction

In the ideal machine learning settings, a model with good generalization capability could be learnt if a sufficient number of labeled samples are available. However, this is rarely the case in real world applications, since the labels have to be obtained through huge amount of efforts with manually labeling or scientific experiments. To handle this problem, researchers have proposed many learning paradigms such as semi-supervised learning [1], transfer learning [2], multiple-task leaning [3] and so on. One basic idea behind these approaches is trying to bridge the gaps between unlabeled data or labels from other domains and the learning target such that the problem of insufficient label information could be alleviated. Despite the partial success in some fields, such connections are commonly built on a few assumptions that are hard to be verified (e.g., the clustering assumption, the manifold assumption), which prevents their wide applications in practice.

Recently, new techniques which allow us to obtain large amounts of label information cheaply have become available, such as crowdsourcing [4], harvesting data with weak labels through web searching [5], and so on. One major advantage of this type of methods is that it essentially relaxes the assumptions made by the aforementioned methods on the distribution of the obtained labels.

One limitation of these label acquiring techniques, however, lies in the fact that the label information obtained tends to be noisy while the performance of most commonly used classifiers such as SVM and logistic regression relies crucially on the correctness of this, although they usually have built-in mechanisms (e.g., regularization term) to tolerate some degree of data noise. But in general data noise is different from label noise in that data noise usually only causes a small move within the its neighborhood region in the feature space, while label noise could make an unpredictable large move to a random region. For example, the situation that a label flips from class i to j actually means that the corresponding data point jumps from the region of the j's class to that of the i's class. For SVM, boosting or other classifier whose loss function grows monotonously with the value of negative sample margin, this could result in an arbitrary large penalty.

Many works have been devoted to solve this problem recently and they can be roughly divided into three categories. The first type - perhaps the most intuitive type among them - is to pre-precess the data such that the data points whose labels are likely to be noise will be removed before feeding to classifier training [6] [7]. These methods could suffer from the disadvantage that some data points with clean labels could be removed as well.

Instead of removing these data points completely, the second type of methods tries to estimate the probability of their labels being noised and warns the classifier about this. The key issue here, therefore, is how to identify those suspicious points confidently. For this, in [8] a probabilistic model of a kernel Fisher Discriminant is presented in which an EM algorithm is proposed to update the probability of the data point being incorrectly labeled. This EM-type algorithm has inspired many later methods in which the true but unknown label of each data point is treated as latent variable and its posterior given the current label is estimated in a probabilistic framework [9] [10]. It has also been applied to learn robust distance metric using noisy labels [11]. Alternatively, a multiple instance learning-based method is proposed in [12] to cope with label noise, but it essentially has to estimate the most correctly labeled positive samples in a bag. Some heuristic strategy can also be adopted. For example, [13] takes boosting to detect the incorrect labels based on the observation that those data are likely to have a big weight.

The third type of methods tries to directly improve the robustness of the classifier against label noise, using various robust loss functions, training methods, or combining different base classifiers. In [14], a binary random variable is introduced to indicate if the label of current example is correct, based on which a stochastic programming problem is casted to learn the multiple kernel classifier, while [15] use truncated hinge loss for outlier reduction in SVM, since label noise usually results in outliers. Recently, [16] proposes a strategy to improve the robustness of SVMs based on a simple kernel matrix

correction, and [17] build a robust classifier taking into account the detected inconsistencies of the labels.

Our method is different from the above works in that we aim to learn a high level representation from the data with label noise for later robust classification. To this end, we propose a new type of auto-encoder method coined label-denoising auto-encoder. In particular, we feed both the training samples and their labels to the input layer of the network and try to reconstruct them both at the output layer. Inspired by the well known denoising auto-encoder [18], which corrupts the input feature with artificial noise, but instead of doing this on the data feature, we disturb the input label to its nearby labels according to some pre-estimated probability during each iteration. Like denoising auto-encoder, this can be thought of as a robust mechanism embedded into the learning process which effectively "denoises" the label noise. The learnt representation is shown to be robust against label noise on three real-world data-sets.

The remaining parts of this paper are organized as follows: In section II, we give a brief account on the auto-encoder neural network, and the proposed method is detailed in section III. The experimental results are presented in Section IV and we conclude the paper in section V.

II. Background

In this section, we first give a brief account on the auto-encoder neural network and its variant - denoising auto-encoder, which is closely related to the current work.

A. Auto-encoder Neural Network

An auto-encoder is a three-layered neural network, which is commonly used to learn a nonlinear feature representation from unlabeled data [19], [20], [21]. It can also be fine tuned to make the learnt representation more suitable for classification using labeled data after the unsupervised stage is finished by replacing its decoder layer with an output layer for label prediction [22].

The basic auto-encoder consists of an encoder and a decoder, whose weights can be trained with backpropagation [23], by setting the target values to be equal to the inputs. In particular, for an input matrix X, the active vector of its hidden layer can be denoted as $H = S(W_1X + b_1)$, where W_1 is the connection weight between input layer and hidden layer, b_1 is the bias to the hidden layer, and S is a nonlinear transform function (e.g., the sigmoid function). The output vector $\hat{X}$ is therefore calculated as $\hat{X} = S(W_2H + b_2)$, using the connection weights W_2 between the hidden layer and output layer and output bias b_2. The learning objective function is:

$$min\|\hat{X} - X\|^2 + \lambda\|W\|^2 \tag{1}$$

After training, the network can reconstruct the input data through hidden units, and the feature representation given by the hidden layer compactly models the information of input data. Note that the use of nonlinear transformation allows it to capture complex relationship within the data.

B. Denoising Auto-encoder

The denoising auto-encoder [18] proposed by Vincent et al. is a well-known variant of auto-encoder, which corrupts input x before feeding it into the auto-encoder, while constraining the network to faithfully reconstruct the original input (without corrupting). This yields the following objective function:

$$min \sum_{x \in D} E_{\tilde{x} \sim q(\tilde{x}|x)} L(x, g(f(\tilde{x}))) \tag{2}$$

where g and f are decoder and encoder respectively, L is some loss function, and $\tilde{x}$ is the disturbed version of x (according to the distribution $q(\tilde{x}|x)$). In general this can be implemented by sample it from a Gaussian noise, i.e., $\tilde{x}$ = x + ϵ, $\epsilon \sim$ N(0 ; σ^2I). This method actually imposes a regularization to the network such that it will behave more robustly when dealing with a future data point with data noise. But the degree to corrupt x should not be too much, because this may lead to overfitting. So we usually restrict $\tilde{x}$ to lie in the nearby region of x.

III. The Proposed Method

In this section we detail the proposed method for handing label noise data.

A. The Motivations

The goal of an ordinary classifier with clean label y is to learn a mapping f from data x to its label, i.e., $f(x) = y$. However, in the situation when the given label $\tilde{y}$ is noisy, this can not be done since we don't know the true label y. One way to surrogate this is to treat the true but known label as a latent variable and estimate the posterior probability given the data x and noisy label $\tilde{y}$, i.e., $p(y|\tilde{y}, x)$, by warping this in a maximal likelihood framework [8], [9]. However, the accuracy of this estimation depends strongly on the capability of the underlying probabilistic model and the quality and size of the data at hand.

In this paper, we adopted a surrogate strategy that directly learn a mapping from the input to the possible nosy label $\tilde{y}$, i.e., $\hat{f}(x, \tilde{y}) = \tilde{y}$. This looks ridiculous at the first sight since a noisy prediction is useless. However, our goal is not to use this mapping for final prediction but to learn a feature representation H for x with possibly noisy label y, such that $H = g(x, \tilde{y})$, where g is the feature mapping function. In other words, we do not directly reduce the gap between $\hat{f}$ and f, which is very difficult, but to learn a robust representation such that samples in each class could be more separable in the space of H. This is possible since such a representation is learnt from both the information from data features and labels - even the label part is noisy, the data feature part (we assume it is clean currently) still contribute to a reliable learning. Another way to look at this idea is that, since the noisy label is not directly used for the final decision making, we essentially reduce its possible negative influence on the performance.

In what follows we detail our method, which involves a preprocessing step followed by a training step.

B. Preprocessing for Label Noise

It has been shown that preprocessing is useful in handling data with label noise [6] [7], but too much preprocessing may be harmful since the data with clean label is likely to be filtered

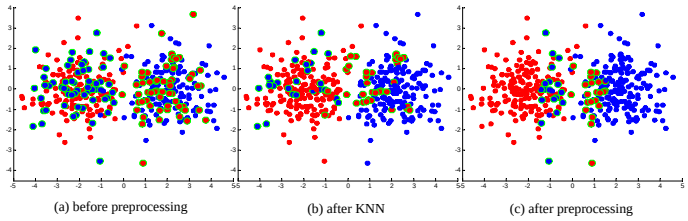


Fig. 1. The effect of preprocessing: The training set is with 20% random label noise. The data point with label noise is highlighted with green circle. Left: the original data distribution and their labels; Middle: the same but preprocessed by KNN rectification; Right: the final results after further preprocessed by softmax regression restoration.

during the procedure. Actually, it is not easy to judge the degree the data inclines to be noise. And even when a big training set contains some label noise, it is accompanied with more useful information than a small training set. Hence a careful preprocessing scheme is needed.

In this paper, we use a two-stage preprocessing strategy by a combination of KNN and softmax regression. In particular, we first use a KNN method to roughly clean the data set, based on the consideration that if the label of data is different from most of its neighbors, it is likely that its label is incorrect. Here we take a rather conservative strategy, i.e., only when more than 80% of one's K nearest neighbours have identical labels and when these labels are different from the one under consideration, we rectify its label to make it consist with those of its neighbours.

We then train a softmax regression classifier using the KNN-rectified data. For a training set $\{(x_i, y_i), i = 1, 2, ..., N\}$, $y_i \in \{1, 2, ..., K\}$. Given a test input x, we estimate the probability that $p(y = j|x)$ for each class:

$$p(y_i = j|x_i; \theta) = \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^{K} e^{\theta_l^T x_i}} \tag{3}$$

where θ_j is the parameter vector of the jth class. The model is estimated by maximal likelihood method with L2 regularization on the parameter vectors.

This serves two purposes: first, it assigns each sample a probability of its label being close to its true label (i.e., $p(y_i = j|x_i)$), which will be used later for label disturbance, and second, it provides us the information about where the most confusing regions lie in the feature space. Since we don't trust the KNN's rectifications in these regions (usually they are close to the boundaries of softmax decision plane), we restore these rectifications to the original labels. This can be done by looking at the posterior probability of each data point and the labels of those whose probability to its class is less than a threshold will be restored if they are previously flipped by the KNN rectifier. Furthermore, we can also use it to rectify the labels with high confidence value (higher than 0.8, for example).

Fig. 1 illustrates the effects of our preprocessing method. It can be seen that most of the label noise in the dense region is rectified by the KNN but those in the decision boundary remains.

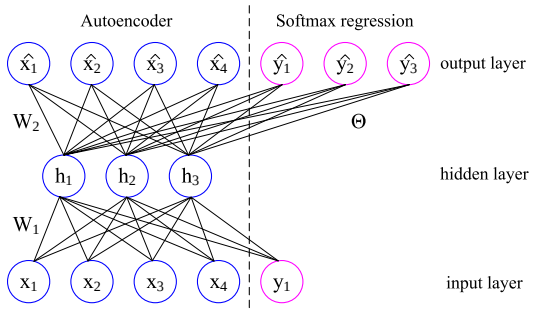


Fig. 2. The overall architecture of the proposed Label-denoising auto-encoder. The bias nodes are not shown for clear presentation.

C. Label-denoising Auto-encoder

In this section we detail our proposed label-denoising auto-encoder which aims to learn a robust feature representation despite of label noise approximating the aforementioned function $\hat{f}(x, \tilde{y}) = \tilde{y}$.

Labels by themselves can also be regarded as a feature. For example, if we label an object as a football, we know it is round and has some pentagon or hexagon grids on its surface. In this regards, label is essentially an abstraction of one class of things. Considering this, we add it to the input layer of the auto-encoder as an extra dimension of the data point.

Figure 2 illustrates a typical architecture of our Label-denoising auto-encoder (LDAE). We use one single node to represent a multi-class label in the input layer [1] and a 1-of-K coding scheme for the output layer. As a result, compared to the standard AE network, we simply add one label node to the input layer and K label nodes to the output layers. A softmax regression scheme is further adopted to estimate the accuracy of the label's reconstruction. The overall objective function of the network is as follows,

$$min_{W,\theta}\ J(X, W) + \lambda J(\tilde{Y}, \theta) + \gamma \|W\|^2 + \beta \|\theta\|^2 \tag{4}$$

$$J(X, W) = \sum_{i=1}^{n} \|\hat{x}_i - x_i\|^2 \tag{5}$$

$$J(Y, \theta) = -\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{1}\{\tilde{y}_i = k\} \cdot log\ \hat{y}_{ik} \tag{6}$$

where the matrix X denotes the training set with each sample x_i as its i-th row, $\mathbf{1}()$ is an indicator function, $\tilde{y}_i$ and $\hat{y}_i$ is respectively the disturbed and reconstructed label of the i-th sample. The k-th component of the i-th output label is denoted as $\hat{y}_{ik}$. Let S denote the sigmoid function, then the reconstructed data feature vector $\hat{x}_i$ can be estimated with,

$$\hat{x}_i = S(W_2 \times S(W_1 \times [x_i\ ,\ \tilde{y}_i] + b_1) + b_2) \tag{7}$$

Denote the parameter vector of the k-th category in the output layer as θ_k, then the corresponding label $\hat{y}_{ik}$ is estimated

[1] Other coding schemes in the input layer (e.g., 1-of-K) are possible, but we found that the current scheme leads to the best performance.

by,

$$\hat{y}_{ik} = C \times exp(\theta_k \times S(W_1 \times [x_i \ , \ \tilde{y}_i] + b_1)) \quad (8)$$

where C is a normalization constant such that $\sum_k \hat{y}_{ik} = 1$.

Recall that in the above formulation, $\tilde{y}_i$ denotes a random disturb on the original label y_i. This is implemented by sampling it based on a distribution given the current data point, i.e,

$$\tilde{y}_i \sim p(j|x_i) \quad (9)$$

where $p(j|x_i)$ is a discrete distribution table containing the posterior probability of the sample x_i belonging to each category. The table is obtained from our preprocessing stage (c.f., E.q. 3).

The traditional backpropagation [23] can be used for optimization of Eq. 4. Note that there is a tradeoff term (λ) involved here between two gradients concerning the reconstruction of data and the reconstruction of label, respectively. If λ is 0, this model degenerates to the standard auto-encoder. If it is set to ∞, we get multi-layer neural network.

It is worthy mentioning that in the traditional deep learning, the two processes are dependant but trained separately: firstly, large amounts of unlabeled data are used to train the stacked auto-encoder, and only on the second step, labeled data are introduced to fine-tuning the previous network. The two steps have different targets and the first step just provides a better initialization. While in our work the two are learned simultaneously with feature learning serving as regularization for the label reconstruction.

D. Discussions on the Label Disturbance

In our work, to get a label-robust feature representation, we disturb the input of label to its nearby label region during each iteration. In the denoising auto-encoder the input data is corrupted with a small random driving it to its nearby region, i.e., a small bias is added to each input feature. But in our case, the nearby region of a label is not its nearby value in the label space.

This issue is addressed by the softmax regression in our method, in which the nearby region of a label is determined by their posterior probability given the current data point. During each iteration the label of each point is determined dynamically according to this probability. As a consequence, the labels of those data with high probability would be less likely to change. This emphasizes that the label noise on the decision boundary should receive more treatment.

This denoising process is much in common with the process of de-noising autoencoder when coping with noise on data feature, with the difference in that under the presence of label noise we bring in the label robust mechanism through training process. There are mainly three reasons that we are very likely obtain more robust feature representation using our methods: First, we take more information into account when learning the representation; Second, we trained the model with an united objective function including both data and its label as illustrated by λ term of E.q. 4. In other words, even when one's label is mistakenly flipped, the feature learned from data can also help to alleviate the influence of such errors. Finally, the method effectively focuses on the most annoying label noises, i.e., those on the boundary. To disturb them means that we do not want to be always mislead by them and try to rectify them to more confident ones, hence it can also be seen as a label revising process.

E. Using the Label Denoising Autpo-encoder

After acquiring the features representations, we replace the decoder layer of the auto-encoder with a softmax layer while keeping the encoder layer fixed to train a classifier, then use it to make prediction for an incoming data point.

This means, however, we have to feed a 'noisy' label of the test sample to the network to estimate its 'clean' label (c.f., Figure 2). To do this, we may simply initialize its label with a random value. Alternatively, we may use another classifier (its particular implementation in this work is given in the experimental section) to assign one label for it, which we call pre-prediction. After this, the trained classifier can be used to make the final prediction. The overall flow chart of the proposed method is summarized in Algorithm 1.

Algorithm 1 The label-denoising auto-encoder method.

Input:

Training set: $\{(x_i, y_i)|\ i = 1, 2, ..., N\}$;
Testing set: $\{(x_i)|\ i = 1, 2, ..., N\}$;
Parameters: K of KNN, the number of hidden units n, the weight decay γ, the tradeoff parameter for label reconstruction λ

Output:

The prediction y of the test data x

———————- Training Stage

1: Preprocessing: Use KNN to clean the data as described in Section III-B. Then make a coarse classification by softmax regression and record all the posterior probability (E.q. 3);
2: Train a label-denoising autoencoder using the method described in Section III-C.
3: Train an autoencoder classifier based on the learnt representation using the training data.

———————- Test Stage

4: Do pre-prediction for the test data.
5: Make the prediction using the trained autoencoder classifier;

IV. Experiments and Analysis

To empirically validate the effectiveness of the proposed approach we conducted experiments on artificial toy data and real database: MINST, USPS and ORL database. In preprocessing we set K of KNN to be 5, and use the baseline auto-encoder (AE) + softmax classifier for pre-prediction. Note that in what follows we don't consider the probabilistic perturbation of x [18] but only the label y is disturbed probabilistically.

A. Experimental Results on Artificial Toy Data

To create our toy problem, we sampled 500 examples from two Gaussian distributions. We will focus on the label noises in the decision boundary since those regions are the places where label noise would be most likely to occur. In particular, we first estimate each data's posterior probability to its true class, and

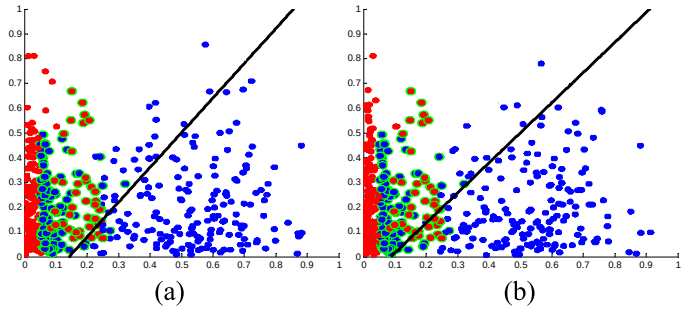


Fig. 3. Experimental results on the toy data. The data point with label noise is highlighted with green circle. (a) Feature space of auto-encoder (b) Feature space of label-denoising auto-encoder.

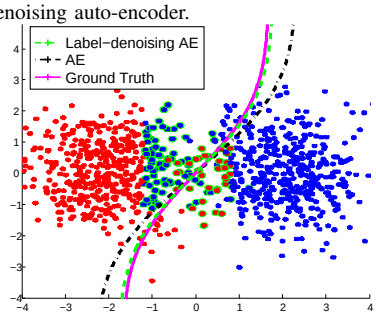


Fig. 4. The decision boundary of different models on the toy data. The data point with label noise is highlighted with green circle.

find those data points with posterior probability lower than 0.7 and flip their labels with a flipping probability of 0.8.

After this we use our label-denoising auto-encoder(LDAE) to learn the features with the number of hidden units set to be two. For comparison, we also train a standard auto-encoder with the same setting. Fig. 3 gives the representation learnt with the two models, respectively. One can see from the figure that our LDAE model gives better feature projection in terms of separability between two groups of samples. We then project the boundary back to the input space. Since the dimensions of the LDAE input space is 3, its boundary would be a curved surface. To make a clear visualization we only take the 2 dimensions of data such that the two model can be compared side by side. Fig. 4 gives the results. We can see that a biased boundary is acquired through AE, but our LDAE yields a more accurate boundary.

B. Experimental Results on the MINST Database

MINST [23] is a database of handwritten digits range from 0 to 9. The digits have been size-normalized and centered in a 28×28 image. It has a training set of 60000 examples, and a test set of 10000 examples. We random select 10000 data from training set to construct our noise set. For test set we use the standard one defined by the evaluation protocol of this dataset. The noise is injected randomly, i.e., changing the labels randomly to other categories, with a noise level varying from 10% to 70%.

Fig. 5(a) gives the results. We made several observations from this figure: 1) Our method ('LDAE') performs best among the compared ones consistently across various noise levels, and the 'softmax regression' performs worst in this experiment; 2) the KNN step has minor influence on this dataset but it does not deteriorate the performance as well ('LDAE without KNN'), possibly due to the conservative strategy we adopted; 3) inputting the original (noisy) label y instead of $\tilde{y}$ ('LDAE without label disturbance') performs worse than using disturbed label $\tilde{y}$ ('LDAE'), which shows that disturbing labels helps to improve the robustness performance against noise; and 4) embedding label information in the feature representation is useful (c.f., the performance difference between 'AE with label embedded' and 'AE without label embedding').

To highlight the effectiveness of the proposed method to embed label information in the feature representation, we compare it with another mechanism which uses the label information to adjust the network as well, that is, fine-tuning. The major difference between these two methods lies in that the fine-tuning is a two-step strategy which trains the network unsupervisedly then use label information to fine tune the learnt weights without changing the network architecture, while in our method the architecture of AE is modified by adding the label nodes in both input and output layer and the weights are trained in one step.

Fig. 5(b) shows the performance on the MINST dataset. One can see that if we don't embed the label information in the feature representation and simply use it in a fine-tuning way, the performance will deteriorate very fast with increasing amount of label noise injected. On the contrary, our method without fine-tuning achieves the best performance especially when the noise level is relatively high. This shows that the fine-tuning procedure is not robust against label noise, possibly due to the potential danger to reduce the quality of the learnt feature representation with the inaccurate supervision.

C. Experimental Results on USPS Database

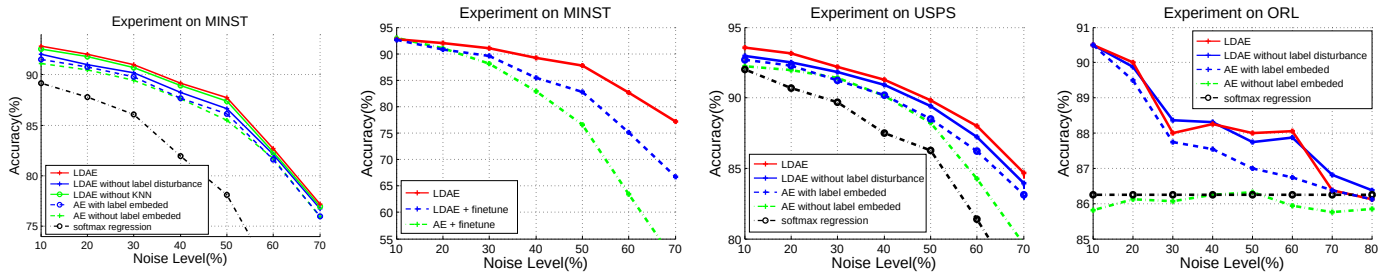
USPS is a database of handwritten digits range from 0 to 9. We downloaded the USPS digit database from a public link[2]. Each digit is resized to a 16 by 16 gray-level image. There are 1100 examples for each class. We randomly select 6000 of them as training set and 3000 as test set. And the noise level ranges from 10% to 70%. The number of hidden units here is 100. The preprocessing is performed before feature learning. Fig. 5(c) gives the results. One can see that the proposed LDAE method still performs best among the compared ones.

D. Experimental Results on ORL Database

The ORL database consists of 400 face images from 40 persons. For each person, there are 10 different gray scale images with a size of 92×112. We set aside two images of each individual (80 images in total) to make a pure data set to which we won't add noise. Then we then randomly choose 4 of each person to make a noisy training data set (160 images in total). Therefore we have a training set consisted of 240 images (among them 160 images have noisy labels), and the remaining 160 images are the testing data. The network has 100 hidden units.

In this dataset, we train a standard AE using all the training data and then turn it into a classifier by replacing the decoder layer with a softmax regression and train it using 80 clean training set. This is actually a standard semi-supervised learning scheme (denoted as "AE without label embedded" in

[2]http://www.cs.toronto.edu/ roweis/data/usps all.mat



(a) (b) (c) (d)

Fig. 5. Experiments results: (a) The performance of the label-denoising auto-encoder compared with baseline methods on the MINIST data set; (b) Performance compared with usual finetuning on the MINIST data set; (c) Performance comparison of different models on USPS database with label noise; (d) Experiment on ORL database.

Fig. 5(d)). The baseline softmax regression classifier is also trained with clean data. Fig. 5(d) gives the results. One can see that our proposed LDAE model outperforms this semi-supervised AE significantly.

V. Conclusion

In this paper we proposed a feature learning method in the presence of label noise, called label-denoising auto-encoder (LDAE). The architecture of LDAE allows it to be trained with noise labels together with the data features, which effectively alleviates the influence of inaccurate supervision information. We propose to optimize the model with dynamically disturbed labels to prevent noisy labels from misleading the direction of learning procedure. We show that the learnt feature representation is beneficial to the subsequent classification.

Acknowledgements

The authors want to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Science Foundation of China (61073112, 61035003, 61373060), Jiangsu Science Foundation (BK2012793), Qing Lan Project, Research Fund for the Doctoral Program (RFDP) (20123218110033).

References

[1] F. G. Cozman, I. Cohen, M. C. Cirelo *et al.*, "Semi-supervised learning of mixture models," in *ICML*, 2003, pp. 99–106.

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] T. Evgeniou, C. A. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," in *Journal of Machine Learning Research*, 2005, pp. 615–637.

[4] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[5] A. Bergamo and L. Torresani, "Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach," in *Advances in Neural Information Processing Systems*, 2010, pp. 181–189.

[6] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "A novel noise filtering algorithm for imbalanced data," in *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*. IEEE, 2010, pp. 9–14.

[7] S. Fefilatyev, M. Shreve, K. Kramer, L. Hall, D. Goldgof, R. Kasturi, K. Daly, A. Remsen, and H. Bunke, "Label-noise reduction with support vector machines," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 3504–3508.

[8] N. D. Lawrence and B. Schölkopf, "Estimating a kernel fisher discriminant in the presence of label noise," in *ICML*. Citeseer, 2001, pp. 306–313.

[9] C. Pal, G. Mann, and R. Minerich, "Putting semantic information extraction on the map: Noisy label models for fact extraction," in *Proceedings of the Workshop on Information Integration on the Web at AAAI*, 2007.

[10] J. Bootkrajang and A. Kabán, "Label-noise robust logistic regression and its applications," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 143–158.

[11] D. Wang and X. Tan, "Robust distance metric learning in the presence of label noise," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[12] T. Leung, Y. Song, and J. Zhang, "Handling label noise in video classification via multiple instance learning," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2056–2063.

[13] I. Cantador and J. R. Dorronsoro, "Boosting parallel perceptrons for label noise reduction in classification problems," in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*. Springer, 2005, pp. 586–593.

[14] T. Yang, M. Mahdavi, R. Jin, L. Zhang, and Y. Zhou, "Multiple kernel learning from noisy labels by stochastic programming," *arXiv preprint arXiv:1206.4629*, 2012.

[15] Y. Wu and Y. Liu, "Robust truncated hinge loss support vector machines," *Journal of the American Statistical Association*, vol. 102, no. 479, 2007.

[16] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise." *Journal of Machine Learning Research-Proceedings Track*, vol. 20, pp. 97–112, 2011.

[17] C. Bouveyron and S. Girard, "Robust supervised classification with mixture models: Learning from data with uncertain labels," *Pattern Recognition*, vol. 42, no. 11, pp. 2649–2658, 2009.

[18] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 1, p. 213, 2002.

[20] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.

[21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.