TRAINING DEEP NEURAL-NETWORKS BASED ON UNRELIABLE LABELS

Alan Joseph Bekker and Jacob Goldberger
Engineering Faculty, Bar-Ilan University, Israel

ABSTRACT

In this study we address the problem of training a neural network based on data with unreliable labels. We introduce an extra noise layer by assuming that the observed labels were created from the true labels by passing through a noisy channel whose parameters are unknown. We propose a method that simultaneously learns both the neural network parameters and the noise distribution. The proposed method is compared to standard back-propagation neural-network training that ignores the existence of wrong labels. The improved classification performance of the method is illustrated on several standard classification tasks. In particular we show that in some cases our approach can be beneficial even when the labels are set manually and assumed to be error-free.

Index Terms—deep-learning, back-propagation, noisy labels

1. INTRODUCTION

The presence of class label noise inherent to training samples has been reported to deteriorate the performance of even the best classifiers in a broad range of classification problems [1][2]. It was also observed that noisy labels tend to be more harmful than noisy attributes [3]. Finding noisy data is related to the data collection process. Typically, the labels used to train a classifier are assumed to be unambiguous and accurate. However, this assumption often does not hold since the labels provided by human judgments are subjective. Many of the largest image datasets have been extracted from social networks. Because these datasets images are labeled by non-expert users, building a consistent model on a precisely labeled training set is very tedious. Mislabeling examples have been reported even in critical applications such as biomedical datasets where the available data are restricted [4]. A very common approach with noisy datasets is to remove the suspect samples in a preprocessing stage or have them relabeled by a data expert [5]. However, these methods are not scalable and hold the risk of removing crucial examples that may be very significant for small datasets.

Robust noise variants have been proposed for the most common classifiers such as logistic-regression and SVM [6][7]. Natarajan et al. [8] proposed a generic unbiased estimator for binary classification with noisy labels. They developed a surrogate cost function that can be expressed by a weighted sum of the original cost functions, and provided asymptotic bounds for performance. Grandvalet and Bengio [9] addressed the problem of missing labels that can be viewed as an extreme case of noisy label data. They suggested a semi-supervised algorithm that encourages the classifier to predict the non-labeled labels with high confidence by adding a regularization term to the cost function.

In spite of the huge success of deep learning there are not many studies that have explicitly attempted to address the problem of Neural Net (NN) training using data with unreliable labels [10][11][12]. Larsen et al. assumed a single noise parameter that can be calculated by adding a new regularization term and cross validation. Mnih and Hinton [10] proposed a more realistic noise model that depends on the true label. However, they only considered the binary classification case. Sukhbaatar and Fergus [12] recently proposed adding a constrained linear layer at the top of the softmax layer, they have shown that only under some strong assumptions the linear layer can be interpreted as the transition matrix between the true and noisy (observed) labels and the softmax output layer as the true probabilities of the labels.

Unlike previous work, e.g. [13][12][10] in our framework we assume no clean data are available to estimate the noise parameters. We define a probabilistic model for the transformation from true labels to noisy labels and derive a learning scheme based on the EM algorithm. In the E-step we estimate the true labels and in the M-step we apply a back-propagation learning algorithm using the current label estimation. As a by-product of the algorithm we also obtain a parametric description of the noise distribution. The improved results of the proposed approach are demonstrated on several datasets.

2. TRAINING NN WITH NOISY LABELS

Assume we want to train a multi-class neural-network soft-classifier \( p(y = i|x; w) \) where \( x \) is the feature vector and \( w \) is the network parameter-set. We further assume that in the training process we cannot directly observe the correct label \( y \). Instead, we only have access to a noisy version of it denoted by \( z \). In our approach the noise generation is modeled by a parameter \( \theta(i, j) = p(z = j|y = i) \). The noise distribution is unknown and we want to learn it as part of the training phase. The probability of observing a noisy label \( z \) given the
feature vector $x$ is:

$$p(z = j|x; w, \theta) = \sum_{i=1}^{k} p(z = j|y = i; \theta)p(y = i|x; w)$$

where $k$ is the number of classes. The model is illustrated in the following diagram:

```
  x -> Neural-Network -> y -> noisy channel -> z
```

Assume the neural network classifier we are using is based on non-linear intermediate layers followed by a softmax output layer used for soft classification. Denote the non-linear function applied on an input $x$ by $h = h(x)$ and denote the softmax layer by:

$$p(y = i|x; w) = \frac{\exp(w_i^\top h)}{\sum_{j=1}^{k} \exp(w_j^\top h)}, \quad i = 1, \ldots, k$$

such that $w_1, \ldots, w_k$ are the softmax parameters which are subset of the entire network parameter set $w$.

```
  x -> w \u009C u -> non-linear function -> h -> u \u009C y -> soft-max
```

In the training phase we are given $n$ feature vectors $x_1, \ldots, x_n$ with corresponding unreliable labels $z_1, \ldots, z_n$ which are viewed as noisy versions of the correct hidden labels $y_1, \ldots, y_n$. The log-likelihood of the model parameters is:

$$L(w, \theta) = \sum_{i=1}^{n} \log \sum_{i=1}^{k} p(z_i|y_t = i; \theta)p(y_t = i|x_t; w))$$

Based on the training data, the goal is to find both the noise distribution $\theta$ and the Neural Network parameters $w$ that maximize the likelihood function. Since the random variables $y_1, \ldots, y_n$ are hidden, we apply the EM algorithm to find the maximum-likelihood parameter set. The EM-auxiliary function is:

$$Q(w_0, \theta_0, w, \theta) = \sum_{t=1}^{n} \sum_{i=1}^{k} p(y_t = i|x_t, z_t; w_0, \theta_0) \log p(y_t = i|x_t; w, \theta)$$

(1)

such that $w_0, \theta_0$ are the current parameter values, and we are looking for $w, \theta$ that maximize the EM auxiliary function. From the auxiliary function we can easily derive the update procedure of the EM algorithm. In the E-step of each iteration of the EM algorithm we estimate the hidden true data labels based on the noisy labels and the current parameters:

$$c_{ti} = p(y_t = i|x_t, z_t; w_0, \theta_0)$$

(2)

$$= \frac{p(z_t|y_t = i; \theta_0)p(y_t = i|x_t; w_0)}{\sum_{j} p(z_t|y_t = j; \theta_0)p(y_t = j|x_t; w_0)}$$

$$= \frac{\theta_0(i, z_t) \exp(u_{ti}^\top h_0(x_t))}{\sum_{j} \theta_0(j, z_t) \exp(u_{tj}^\top h_0(x_t))}$$

such that $u_{10}, \ldots, u_{k0}$ and $h_0(x)$ are the components of the current parameter value $w_0$. In the M-step we update both the NN and the noisy channel parameters. The updated noise distribution has a closed-form solution:

$$\theta(i, j) = \frac{\sum_{t} c_{ti} 1(z_t = j)}{\sum_{t} c_{ti}}, \quad i, j \in \{1, \ldots, k\}$$

(3)

The $k \times k$ matrix $\theta$ is a confusion matrix between the soft estimates of the true label $\{c_{ti}|i = 1, \ldots, k\}$ and the noisy labels $z_t$. To find the updated NN parameter $w$ we need to maximize the following function:

$$S(w) = \sum_{i=1}^{n} \sum_{t=1}^{k} c_{ti} \log p(y_t = i|x_t; w)$$

(4)

which is a soft-version of the likelihood function of the fully observed case, based on the current estimate of the true labels.

We next derive the back-propagation equations for the score function (4), which is obtained by the EM algorithm, to get a better understanding of the learning process. In the case that the true labels $y_1, \ldots, y_n$ are given, the partial derivatives of the likelihood function $S(w) = \sum_{t} \log p(y_t|z_t; w)$ with respect to the softmax parameters are:

$$\frac{\partial S}{\partial w_i} = \sum_{t=1}^{n} (1(y_t = i) - p(y_t = i|x_t; w)) h(x_t)$$

(5)

In our case, where only noisy labels $z_1, \ldots, z_n$ are provided, the partial derivatives of the function (4) that we maximize in the M-step are:

$$\frac{\partial S}{\partial w_i} = \sum_{t=1}^{n} (p(y_t = i|x_t, z_t; w_0, \theta_0) - p(y_t = i|x_t; w)) h(x_t)$$

(6)

Comparing Eq. (5) and (6) we see that the back-propagation learning algorithm in the case of noisy labels is very similar to the fully observed label case. The only difference is that instead of using the labels values we use estimated labels values that were computed in the E-step based on the current network and noise parameter values. Another consequence of comparing Eq. (5) and (6) is that the computational complexity of the back-propagation algorithm is the same in both cases.
network training and update the noise parameters. For example, we can use standard methods for neural-networks which are iterative methods and we can alternate between the two steps. The EM Algorithm iterates between the two steps:

**E-step:** Estimate true labels based on the current parameter values \( \theta \):

\[ c_{ti} = p(y_t = i|x_t, z_t; w, \theta) \]

**M-step:** Update the noise parameter \( \theta \):

\[ \theta(i, j) = \frac{\sum_t c_{ti} 1(z_t = j)}{\sum_t c_{ti}} \]

and train a NN to find \( w \) that maximizes the following function:

\[ L(w) = \sum_{i=1}^{n} \sum_{i=1}^{k} c_{ti} \log p(y_t = i|x_t; w) \]

There is no need, of course, to fully train the NN model on each EM iteration. Both EM and back-propagation algorithms are iterative methods and we can alternate between them. For example, we can use standard methods for neural-network training and update the noise parameter \( \theta \) after few passes over the training set. The EM algorithm is a greedy optimization procedure and is notoriously known to be sensitive to the starting point. Hence a good initialization of the model parameters is important to achieve good results. We can use the following strategy to initialize the EM parameters. We first train the NN using standard methods ignoring the fact that the labels are noisy. The obtained NN parameter set \( w \) is used as an initial value the EM iteration. We then compute the confusion matrix on the train set and used it as an initial value for the noise parameter set \( \theta \):

\[ \theta(i, j) = \frac{\sum_t 1(z_t = j) p(y_t = i|x_t; w)}{\sum_t p(y_t = i|x_t; w)}, \quad i, j \in \{1, ..., k\} \]

The proposed method, which we dub the Noisy Labels Neural-Network (NLNN) algorithm, is summarized in Table 1.

<table>
<thead>
<tr>
<th>Table 1. The Noisy Labels Neural-Network (NLNN) algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Data-points ( x_1, ..., x_n \in R^d ) with corresponding noisy labels ( z_1, ..., z_n \in {1, ..., k} ).</td>
</tr>
<tr>
<td><strong>Output:</strong> Neural-network parameters ( w ) and noise parameters ( \theta ).</td>
</tr>
</tbody>
</table>

In this section, we evaluate the robustness of deep learning to training data with noisy labels with and without explicit noise modeling. We used two data-sets with injected label noise in our experiments. The first is the MNIST database of handwritten digits, which consists of \( 28 \times 28 \) images. The dataset has 60k images for training and 10k images for testing. The second dataset is used for phoneme classification and is based on the TIMIT acoustic-phonetic continuous speech corpus which has 1.5M frames for training and 500K frames for testing. The input features are the Mel frequency cepstral coefficients (MFCCs) of the signal, powered by delta and delta delta coefficients. Context frames were added to the current time frame as in Mohamed et al. \[15\]. Overall each speech frame is represented by 351 features. This is the standard feature set for phoneme classification and is known to provide the best classification results.

A similar network architecture and hyper-parameters were used for both datasets. We used a two hidden layer NN comprised of 500 and 300 neurons. The learning scheme is based on the TIMIT acoustic-phonetic continuous speech corpus which has 1.5M frames for training and 500K frames for testing. The input features are the Mel frequency cepstral coefficients (MFCCs) of the signal, powered by delta and delta delta coefficients. Context frames were added to the current time frame as in Mohamed et al. \[15\]. Overall each speech frame is represented by 351 features. This is the standard feature set for phoneme classification and is known to provide the best classification results.

In the first type of dataset, we used the TIMIT database of speech with noise modeling. We used two data-sets with injected label noise in our experiments. The first is the MNIST database of handwritten digits, which consists of \( 28 \times 28 \) images. The dataset has 60k images for training and 10k images for testing. The second dataset is used for phoneme classification and is based on the TIMIT acoustic-phonetic continuous speech corpus which has 1.5M frames for training and 500K frames for testing. The input features are the Mel frequency cepstral coefficients (MFCCs) of the signal, powered by delta and delta delta coefficients. Context frames were added to the current time frame as in Mohamed et al. \[15\]. Overall each speech frame is represented by 351 features. This is the standard feature set for phoneme classification and is known to provide the best classification results.

A similar network architecture and hyper-parameters were used for both datasets. We used a two hidden layer NN comprised of 500 and 300 neurons. The learning scheme is based on a back-propagation algorithm with an adaptive learning rate combined with momentum. The learning rate was initialized to 0.01. It was then increased in each epoch by multiplying the learning rate by 1.05 if the new likelihood score exceeds the old likelihood score by more than 4%. Otherwise, the learning rate is kept. If the likelihood score was less than the old likelihood, the learning rate was decreased by multiplying the learning rate by 0.7. The momentum was set to 0.5. The number of training epochs was set to 100 for the naive approach and for the NLNN initialization that was described above. These settings were kept fixed for all the experiments described below.

We generated two types of noisy data from clean data by stochastically changing some of the labels. In the first type, we used the TIMIT database of speech with noise in our experiments. The first is the MNIST database of handwritten digits, which consists of \( 28 \times 28 \) images. The dataset has 60k images for training and 10k images for testing. The second dataset is used for phoneme classification and is based on the TIMIT acoustic-phonetic continuous speech corpus which has 1.5M frames for training and 500K frames for testing. The input features are the Mel frequency cepstral coefficients (MFCCs) of the signal, powered by delta and delta delta coefficients. Context frames were added to the current time frame as in Mohamed et al. \[15\]. Overall each speech frame is represented by 351 features. This is the standard feature set for phoneme classification and is known to provide the best classification results.

A similar network architecture and hyper-parameters were used for both datasets. We used a two hidden layer NN comprised of 500 and 300 neurons. The learning scheme is based on the TIMIT acoustic-phonetic continuous speech corpus which has 1.5M frames for training and 500K frames for testing. The input features are the Mel frequency cepstral coefficients (MFCCs) of the signal, powered by delta and delta delta coefficients. Context frames were added to the current time frame as in Mohamed et al. \[15\]. Overall each speech frame is represented by 351 features. This is the standard feature set for phoneme classification and is known to provide the best classification results.

A similar network architecture and hyper-parameters were used for both datasets. We used a two hidden layer NN comprised of 500 and 300 neurons. The learning scheme is based on a back-propagation algorithm with an adaptive learning rate combined with momentum. The learning rate was initialized to 0.01. It was then increased in each epoch by multiplying the learning rate by 1.05 if the new likelihood score exceeds the old likelihood score by more than 4%. Otherwise, the learning rate is kept. If the likelihood score was less than the old likelihood, the learning rate was decreased by multiplying the learning rate by 0.7. The momentum was set to 0.5. The number of training epochs was set to 100 for the naive approach and for the NLNN initialization that was described above. These settings were kept fixed for all the experiments described below.
type we did not distinguish between classes when intentionally mislabeling data, and rather choosing new labels from a uniform distribution over the labels. The original label $i$ is randomly changed to a different label $j$ with probability $\theta(i,j) = p/(k-1)$ where $k$ is the number of classes and $p$ is the percentage of incorrect labels we want to create in the training data. In the second type we converted each label with probability $p$ to a different label according to a predefined permutation. The labels of the test data remained, of course, unperturbed to validate and compare our method to the regular approach.

**Fig. 1.** MNIST test data classification accuracy as a function of fraction of noisy labels with uniform noise.

**Fig. 2.** MNIST test data classification accuracy as a function of fraction of noisy labels with permutation type noise.

**MNIST:** When training the NLNN model on the MNIST data, we first initialize the network parameters $w$ with a 100 epochs of back propagation using the noisy data and then we run the EM algorithm until the likelihood converges (this took less than 10 iterations). In each iteration we train the NN with the current estimated labels for 50 epochs. In each iteration of the EM we start the back-propagation training with random NN parameters. We found that this strategy works better than starting the NN training with the parameters obtained in the previous EM iteration.

Figure 1 and Figure 2 show comparative test errors results as a function of the fractions of noise for the two noise types described above. We compared the performance of several approaches. We first implemented the naive NN training algorithm (ignoring the existence of noise) and compared it to our model (NLNN) and to a variant of the NLNN model where the correct noise distribution is given and is not updated during the EM iterations (NLNN true $\theta$). Figure 1 shows that NLNN significantly outperformed the standard network training approach for every noise ratio and achieved the same results as a model where the known distribution was given. Figure 2 shows the accuracy of the MNIST dataset as a function of the permutation type noise. For a high noise level the NLNN model accuracy dropped compared to the NLNN true $\theta$ but still achieved better performance than the naive NN.

**Fig. 3.** Phoneme classification accuracy on the TIMIT dataset as a function of fraction of noisy labels.

**TIMIT:** In this dataset we conducted an experiment using a uniform noise distribution. Figure 3 shows the phoneme classification accuracy on the test data as a function of the noise level. We show the performance of the naive model, our proposed NLNN model and NLNN model with the true noise distribution $\theta$. Figure 3 shows that the NLNN model achieves significantly better classification accuracy than a naive neural network model that ignores the presence of noisy labels. In this case of phoneme classification we even obtained a slightly better result in the noise-free case where we used the correct phoneme labels provided by the TIMIT dataset. This is an example of a situation where manual annotation is subjective and is not error-free. In the case of speech data, manual phoneme annotation can be noisy because phoneme boundaries are not defined exactly and co-articulation acoustic effects that occur when moving from one phoneme to the next. Classification performance decreased significantly near phoneme boundaries.

To conclude, in this paper we proposed an algorithm for training neural networks based solely on noisy data where the noise distribution is unknown. We showed that we can reliably learn the noise distribution from the noisy data without using any clean data which, in many cases, is not available. The algorithm can be easily incorporated into existing deep learning implementations. Our results encourage collecting more data at a cheaper price, since mistaken data labels can be less harmful to performance. One possible future research direction would be generalize our learning scheme to cases where both the features and the labels are noisy.
4. REFERENCES


