Averaged Stochastic Gradient Descent with Feedback for Large-Scale Continuous Activity Recognition

Xu Sun
MOE Key Laboratory of Computational Linguistics, Peking University

Abstract

Activity recognition has been studied with limited lab settings on small-scale datasets. These studies generally assume that the start and the end of each activity are known beforehand. In practice, however, different activities are present in a continuous temporal sequence with unknown boundaries. In this article, we propose a new online training method, referred to as *averaged stochastic gradient descent with feedback (ASF)*, for *continuous* activity recognition. We demonstrate reasonable convergence properties of ASF, and evaluate its performance using large-scale dataset collected from 100 persons. Our results demonstrate that as a practical solution, ASF outperforms a variety of existing training methods in both accuracy and robustness. The novelty of this paper is twofold: (1) An accurate and robust online training method; (2) Large-scale continuous activity recognition.

1 Introduction

Acceleration sensor based activity recognition is useful in practical applications (Bao and Intille, 2004; Präkkä et al., 2006; Ravi et al., 2005; Huynh et al., 2008). For example, in some medical programmes, researchers hope to prevent lifestyle diseases from being exacerbated. However, the traditional way of counseling is ineffective both in time and accuracy, because it requires many manual operations. In sensor-based activity recognition, an accelerometer is employed (e.g., attached on the wrist of people) to automatically capture the acceleration statistics (e.g., a temporal sequence of three-dimension acceleration data) in the daily life of counselees, and the corresponding categories of behaviors (activities) can be automatically identified with a certain level of accuracy.

Although there is a considerable literature on activity recognition, most of the prior work discusses activity recognition in a pre-defined limited environment (Bao and Intille, 2004; Präkkä et al., 2006; Ravi et al., 2005). For example, most of the prior work assumes that the beginning and ending time of each activity are known to the target recognizing system, and the produced system only performs simple classifications to the activity signals (Bao and Intille, 2004; Präkkä et al., 2006; Ravi et al., 2005). However, this is not the case for real-life activity sequences of human beings, in which different types of activities are performed one by one without an explicit segmentation on the boundaries. For example, people may first walk, and then take a taxi, and then take an elevator, in which the boundaries of the activities are unknown to the target activity recognition system. For this concern, it is important to develop a more powerful system that not only predicts the types of the activities, but also disambiguates the boundaries of those activities.

To this end, we have collected 100 users' real-life activity data continuously based on the 3-dimension acceleration data reported by sensors. We then further propose an accurate and robust online training method for continuous real-life activity recognition. Our proposed solution uses the popular structured classification models, conditional random fields (CRFs) and latent conditional random fields (LCRFs), to identify activity type while disambiguating the activity boundaries.

2 Related Work and Motivations

2.1 Human Activity Recognition

Activity recognition is an important topic and has many applications in practice (Kerr et al., 2011; Mandal and Eng, 2012; Banos et al., 2012; Keally et al., 2011; Junker, 2005; Bannach et al., 2008; Choudhury et al., 2008). Activity recognition is usually studied with limited lab settings and a small dataset. First, traditional lab settings assume that the start and the end of each activity are known beforehand. With this assumption, most of the prior work simplified the activity recognition task as a single-label classification problem (Bao and Intille, 2004; Präkkä et al., 2006; Ravi et al., 2005; Yin et al., 2005, 2008). Given a sequence of sensor signals, the activity recognition system predicts a single label (representing a type of activity) for the whole sequence. Ravi et al. (Ravi et al., 2005) used decision trees, support vector machines (SVMs) and K-nearest neighbors (KNN) models for classification. Bao and Intille (Bao and Intille, 2004) and Pärkkä et al. (Präkkä et al., 2006) used decision trees for classification. A few other works treated the task as a structured classification problem. Huynh et al. (Huynh et al., 2008) tried to discover latent activity patterns by using a Bayesian latent topic model. In this paper, unlike previous studies based on limited lab settings, we perform continuous activity recognition with activity boundaries unknown to the recognizer.

Second, most of the prior work of activity recognition used a relatively small dataset. For example, in Ravi et al. (Ravi et al., 2005), the data was collected from two persons. In Huynh et al. (Huynh et al., 2008), the data was collected from only one person. In Pärkkä et al. (Präkkä et al., 2006), the data was collected from 16 persons. Unlike previous studies based on small datasets, our study is based on a large-scale dataset collected from 100 persons.

2.2 Online Training

There are two major approaches for training log-linear models: batch training and online training. Standard gradient descent methods are normally batch training methods, in which the gradient computed by using all training instances is used to update the parameters of the model. The batch training methods include, for example, steepest gradient descent, conjugate gradient descent (CG), and quasi-Newton methods like Limited-memory BFGS (LBFGS) (Nocedal and Wright, 1999). The true gradient is usually the sum of the gradients from each individual training instance. Therefore, batch gradient descent requires the training method to go through the entire training set before updating parameters. Hence, the batch training methods are slow.

Online learning methods can significantly accelerated the training speed, compared with batch training methods (Bartlett et al., 2007; Bottou and LeCun, 2003; Duchi and Singer, 2009; Shalev-Shwartz and Tewari, 2009). A promising online training method is the stochastic gradient method, for example, the stochastic gradient descent method (SGD) and its extensions (Bottou, 1998; Spall, 2005; Langford et al., 2009; Lin, 2010). The parameters of the model are updated much more frequently, and much fewer iterations are needed before the convergence. For large-scale datasets, the SGD can be much faster than batch gradient based training methods. However, there are problems on the traditional SGD literature: First, the SGD method is sensitive to noise. The accuracy of the SGD training is limited when the data is noisy (for example, the data inconsistency problem that we will discuss in the experiment section). Second, the SGD method is not robust enough in some cases. It contains more hyper-parameters (e.g., not only regularization, but also learning rate) and it is quite sensitive to the tuning of hyper-parameters. In some cases, tuning hyper-parameters for SGD is not a easy task.

In this paper, we present a new online gradient-based learning method, the averaged SGD with feedback (ASF), for training log-linear models with convex or even non-convex objective functions. According to the experimental results, the proposed method is accurate and robust for training continuous activity recognizers.

3 Background of Methodology

We will review the technique details of conditional random fields, latent conditional random fields, and stochastic gradient descent training.

3.1 Conditional Random Fields

Conditional random fields (CRFs) are proposed as an alternative solution for structured classification by solving "the label bias pro-

blem" (Lafferty et al., 2001). Assuming a feature function that maps a pair of observation sequence $\mathbf{x}$ and label sequence $\mathbf{y}$ to a global feature vector $\mathbf{f}$, the probability of a label sequence $\mathbf{y}$ conditioned on the observation sequence $\mathbf{x}$ is modeled as follows (Lafferty et al., 2001):

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp\left[\mathbf{w}^\top \mathbf{f}(\mathbf{y}, \mathbf{x})\right]}{\sum_{\forall \mathbf{y}'} \exp\left[\mathbf{w}^\top \mathbf{f}(\mathbf{y}', \mathbf{x})\right]}, \quad (1)$$

where $\mathbf{w}$ is a parameter vector.

Typically, computing $\sum_{\forall \mathbf{y}'} \exp\left[\mathbf{w}^\top \mathbf{f}(\mathbf{y}', \mathbf{x})\right]$ is computationally intractable. This summation can be computed using dynamic programming techniques (Lafferty et al., 2001). To make the dynamic programming techniques applicable, the dependencies of labels are chosen to obey the Markov property. This has a computational complexity of $O(NK^M)$. N is the length of the sequence; K is the dimension of the label set; M is the length of the Markov order used by local features.

Given a training set consisting of n labeled sequences, $(\mathbf{x}_i, \mathbf{y}_i)$, for $i = 1 \ldots n$, parameter estimation is performed by maximizing the objective function,

$$L(\mathbf{w}) = \sum_{i=1}^{n} \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) - R(\mathbf{w}). \quad (2)$$

The first term of this equation represents a conditional log-likelihood of a training data. The second term is a regularizer for reducing overfitting. Since our preliminary experiments suggested that an L_2 prior performed slightly better than an L_1 prior in the tasks discussed in this paper, we employed an L_2 prior, $R(\mathbf{w}) = \frac{||\mathbf{w}||^2}{2\sigma^2}$. In what follows, we denote the conditional log-likelihood of each sample, $\log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$, as $\ell(i, \mathbf{w})$. The final objective function is as follows:

$$L(\mathbf{w}) = \sum_{i=1}^{n} \ell(i, \mathbf{w}) - \frac{||\mathbf{w}||^2}{2\sigma^2}. \quad (3)$$

3.2 Latent Conditional Random Fields

Real-world problems may contain hidden structures that are difficult to be captured by conventional structured classification models without latent variables. In such cases, models that exploit latent variables are advantageous in learning (Morency et al., 2007; Petrov and Klein, 2008; Sun et al., 2009b,a; Petrov, 2010, to appear). Therefore, as a representative structured classification model with latent variables, the discriminative probabilistic latent variable models (LCRFs) have become popular for performing a variety of tasks with hidden structures, e.g., vision recognition (Morency et al., 2007), syntactic parsing (Petrov and Klein, 2008; Petrov, 2010, to appear), abbreviation generation/recognition (Sun et al., 2009b), and biomedical named entity recognition (Sun et al., 2009a). For example, Morency *et al.* (Morency et al., 2007) demonstrated that LCRF models can learn latent structures of vision recognition problems efficiently, and outperform several widely-used conventional models, such as support vector machines (SVMs), conditional random fields (CRFs) and hidden Markov models (HMMs). Petrov and Klein (Petrov and Klein, 2008) reported on a syntactic parsing task that LCRF models can learn more accurate grammars than that of the conventional techniques without latent variables.

Given the training data, the task is to learn a mapping between a sequence of observations $\mathbf{x} = x_1, x_2, \ldots, x_m$ and a sequence of labels $\mathbf{y} = y_1, y_2, \ldots, y_m$. Each y_j is a class label for the j'th token of a word sequence, and is a member of a set $\mathbb{Y}$ of possible class labels. For each sequence, the model also assumes a sequence of latent variables $\mathbf{h} = h_1, h_2, \ldots, h_m$, which is unobservable in training examples.

The LCRF model is defined as follows (Morency et al., 2007):

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \triangleq \sum_{\mathbf{h}} P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \mathbf{w}) P(\mathbf{h}|\mathbf{x}, \mathbf{w}),$$

where $\mathbf{w}$ represents the parameter vector of the model. To make the training efficient, a restriction is made for the model: for each label, the latent variables associated with it have no intersection with the latent variables from other labels (Morency et al., 2007; Petrov and Klein, 2008). This simplification is also a popular practice in other latent conditional models, including hidden-state conditional random fields (HCRF) (Quattoni et al., 2007). Each h is a member in a set $\mathbb{H}(y)$ of possible latent variables for the class label y, and $\mathbb{H}(y^j) \cap \mathbb{H}(y^k) = \varnothing$ if $y^j \neq y^k$. $\mathbb{H}$ is defined as the set of all possible latent variables; i.e., the union of all $\mathbb{H}(y)$ sets: $\mathbb{H} = \cup_{y \in \mathbb{Y}} \mathbb{H}(y)$. In other words, h can have any value from $\mathbb{H}$, but $P(y|h)$ is zero except for only one of y in $\mathbb{Y}$. The disjoint restriction indicates a

discrete simplification of $P(\mathbf{y}|\mathbf{h},\mathbf{x},\mathbf{w})$:

$$P(\mathbf{y}|\mathbf{h},\mathbf{x},\mathbf{w}) = 1 \quad if \quad \mathbf{h} \in \mathbb{H}(y_1) \times \ldots \times \mathbb{H}(y_m)$$
$$P(\mathbf{y}|\mathbf{h},\mathbf{x},\mathbf{w}) = 0 \quad if \quad \mathbf{h} \notin \mathbb{H}(y_1) \times \ldots \times \mathbb{H}(y_m)$$

where m is the length of the labeling: $m = |\mathbf{y}|$. The formula $\mathbf{h} \in \mathbb{H}(y_1) \times \ldots \times \mathbb{H}(y_m)$ indicates that the latent-labeling $\mathbf{h}$ is a latent-labeling of the labeling $\mathbf{y}$, which can be more formally defined as follows:

$$\mathbf{h} \in \mathbb{H}(y_1) \times \ldots \times \mathbb{H}(y_m) \iff h_j \in \mathbb{H}(y_j),\ j = 1, \ldots, m.$$

Since sequences that have any $h_j \notin \mathbb{H}(y_j)$ will by definition have $P(\mathbf{y}|h_j,\mathbf{x},\mathbf{w}) = 0$, the model can be simplified as:

$$P(\mathbf{y}|\mathbf{x},\mathbf{w}) \triangleq \sum_{\mathbf{h} \in \mathbb{H}(y_1) \times \ldots \times \mathbb{H}(y_m)} P(\mathbf{h}|\mathbf{x},\mathbf{w}). \quad (4)$$

The item $P(\mathbf{h}|\mathbf{x},\mathbf{w})$ is defined by the usual conditional random field formulation:

$$P(\mathbf{h}|\mathbf{x},\mathbf{w}) = \frac{\exp\left[\mathbf{w}^\top \mathbf{f}(\mathbf{h},\mathbf{x})\right]}{\sum_{\mathbf{h}' \in \mathbb{H} \times \ldots \times \mathbb{H}} \exp\left[\mathbf{w}^\top \mathbf{f}(\mathbf{h}',\mathbf{x})\right]}, \quad (5)$$

where $\mathbf{f}(\mathbf{h},\mathbf{x})$ is a global feature vector.

Given a training set consisting of n labeled sequences, $(\mathbf{x}_i,\mathbf{y}_i)$, for $i = 1 \ldots n$, parameter estimation is performed by optimizing the objective function,

$$\begin{aligned} L(\mathbf{w}) &= \sum_{i=1}^{n} \log P(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w}) - R(\mathbf{w}) \\ &= \sum_{i=1}^{n} \ell(i,\mathbf{w}) - \frac{||\mathbf{w}||^2}{2\sigma^2}. \end{aligned} \quad (6)$$

3.3 Stochastic Gradient Descent

Training LCRFs is quite challenging. Traditional gradient-based batch training methods, such as Limited-memory BFGS (LBFGS) (Nocedal and Wright, 1999), are intolerably slow on LCRFs (Petrov and Klein, 2008; Sun et al., 2009a). For example, (Petrov and Klein, 2008) highlighted both time and memory cost problems on training LCRFs for natural language parsing. Also, (Sun et al., 2009a) showed that the training of LCRFs is computationally expensive on large scale problems. On a corpus of 10-thousand sentences, the training of LCRFs took several days.

The stochastic gradient descent (SGD) training method is quite successful on conventional models (Tsuruoka et al., 2009; Vishwanathan et al., 2006; Shalev-Shwartz et al., 2007; Bottou, 1998; Spall, 2005), including CRFs and support vector machines. The SGD uses a small randomly-selected subset of the training samples to approximate the gradient of the objective function given by Equation 3. The number of training samples used for this approximation is called the batch size. By using a smaller batch size, one can update the parameters more frequently and speed up the convergence. The extreme case is a batch size of one, and it gives the maximum frequency of updates, which we adopt in this work. Then, the model parameters are updated in such a way:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \gamma_k \nabla_{\mathbf{w}_k} \left[\ell(i,\mathbf{w}_k) - \frac{||\mathbf{w}_k||^2}{2n\sigma^2} \right], \quad (7)$$

where k is the update counter, γ_k is the learning rate, and n is the number of data samples in a training dataset. A proper learning rate can guarantee the convergence of the SGD method (Bottou, 1998; Spall, 2005). A typical convergent choice of learning rate can be found in Collins *et al.* (Collins et al., 2008):

$$\gamma_k = \frac{\gamma_0}{1 + k/n},$$

where γ_0 is a constant. This scheduling guarantees ultimate convergence (Bottou, 1998; Spall, 2005). In this paper we adopt this learning rate schedule for the SGD.

4 Noise Analysis in Activity Recognition

Before presenting a reasonable solution for continuous activity recognition, it is important to analyze characteristics and difficulties in this task. Since noise is a major reason that limits the performance of an activity recognizer, we will focus on noise analysis. To our knowledge, there is no previous work on such kind of analysis in continuous activity recognition. We found the data of continuous activities are much more noisy than other real-world tasks, and the noise is from the very nature of activity recognition: inconsistency of persons and inconsistency of axes.

4.1 Inconsistency of Persons

Since our dataset is based on 100 persons, it will be interesting to analyze the consistency of activity patterns among different persons. As we can see in Figure **??**, the personal data distributions are

not even from one to another. Some persons (e.g., person-IDs from #0 to #40) have relative small amount of collected data. Since it is inefficient to test consistency among 100 persons, we pick 5 representative persons (the top-5 persons with the largest size of data) for the consistency test. A CRF model is used as the activity recognizer, and the SGD method is used for training the CRF recognizer. We first merge the training data of the five persons for training a unified recognizer. On the other hand, we use the training data of each person to train a personalized recognizer for each person, and we can get five personalized recognizers in total. Then, we compare the unified and personalized recognizers by evaluating prediction accuracies on each person.

In Figure **??**, we show the curves of accuracies on varying the number of training iterations of the SGD-Merged and the SGD-Personalized. *SGD-Merged* is the unified recognizer trained by SGD: all the personal training data is merged for training. *SGD-personalized* is the personalized recognizer trained by SGD. The left figure shows the curves of overall accuracies of the different recognizers. The overall accuracies are averaged over the test data of all five persons. The middle figure shows the curves of personal accuracies of the unified recognizer. *SGD-Merged-1* means the unified recognizer is evaluated on the test data of the *Person-1*. Finally, the right figure shows the curves of the personalized recognizers. *SGD-1-1* means the personalized recognizer is trained on the training data of the *Person-1*, and evaluation is on the test data of the *Person-1*.

As can be seen from the left figure, in contrast to our expectation, the unified recognizer performed slightly worse than personalized recognizers in terms of overall accuracies. This is interesting because the unified recognizers used more training data than the personalized recognizers. More details are shown in the middle and the right figure. As we can see, the curves of the unified recognizer (in the middle figure) is much more unstable than the curves of the personalized recognizers (in the right figure). The severe accuracy fluctuations of the unified recognizer indicate that activity patterns of different persons can be very different and even opposite. The activity patterns of different persons are not as consistent as our expectation.

From this experimental analysis, we can see that it is a challenge to train a unified activity recognizer for different persons, because of the inconsistency of personal activity patterns. On the other hand, note that, building a unified activity recognizer is still a more practical solution than building multiple personalized activity recognizers, simply because many new users do not have their own training data at all. In such cases, there is no way to build personalized activity recognizers for the new users.

4.2 Inconsistency of Axes

Another significant source of noise in activity recognition is the axis rotation problem: the rotation of the *x-axis*, *y-axis*, and *z-axis* in the collected data. Since different people attached the iPod accelerometer with a different rotation of iPod accelerometer, the *x-axis*, *y-axis*, and *z-axis* faced the risk of inconsistency in the collected data. Take an extreme case for example, while the *x-axis* may represent a horizontal direction for an instance, the same *x-axis* may represent a vertical direction for another instance. As a result, the acceleration signals of the same axis may face the problem of inconsistency. A candidate solution to keep the signal consistency is to keep a standard rotation of accelerometers when collect the data. However, we try to avoid such typical lab settings, because this will make the collected data "unnatural". For example, people put accelerometers (e.g., in iPod or iPhone) randomly in their pocket in daily lives.

5 A New Online Training Method

We have shown that the continuous activity recognition task contains very noisy training data. Therefore, a robust and accurate online training method is crucial to achieve good performance in this task. As we have introduced, the SGD optimization method is a popular and successful training method in usual circumstances. However, in an unusual circumstance, such as noisy training data, the SGD method faces difficulties in training parameters. In addition, given a noisy dataset, training a LCRF model is even more challenging, because of the incorporation of latent variables in LCRFs. Since the parameters of latent variables are totally unobservable in the training stage, the approximate gradient (of latent variables) used at each update is much more noisy than on traditional models. As a result, the fluctuation of approximate gradient on LCRFs is especially severe in

noisy dataset. In such an extremely noisy situation, the performance of the traditional stochastic training methods is limited.

To alleviate the training problem in continuous activity recognition, we will present the averaged stochastic gradient descent training method, and more importantly, we will show that a reasonable feedback schedule is the key to make the averaged SGD being robust and accurate.

5.1 Averaged SGD with Feedback (ASF)

The naive version of averaged SGD is inspired by the averaged perceptron technique (Collins, 2002). Let $\mathbf{w}^{iter(c),sample(d)}$ be the parameters after the d'th training example has been processed in the c'th iteration over the training data. We define the *averaged parameters* at the end of the iteration c' as:

$$\overline{\mathbf{w}}^{iter(c')} \triangleq \frac{\sum_{c=1...c',\, d=1...n} \mathbf{w}^{iter(c),sample(d)}}{nc'}. \tag{8}$$

It is noteworthy that the parameter averaging can be performed on-the-fly as follows:

$$\overline{\mathbf{w}}^{t} = \frac{t-1}{t}\overline{\mathbf{w}}^{t-1} + \frac{1}{t}\mathbf{w}^{t},$$

where t is a count of the weight updates. Hence, there is no need to store the previous weight vectors.

However, a straightforward application of parameter averaging is not adequate. A potential problem of traditional parameter averaging is that the model parameters $\mathbf{w}$ receive no information from the averaged parameters: the model parameters $\mathbf{w}$ are trained exactly the same like before (SGD without averaging). $\mathbf{w}$ could be misleading as the training goes on. To solve this problem, a natural idea is to reset $\mathbf{w}$ by using the averaged parameters, which are more reliable. We propose a refined version of averaged SGD by further applying a *"periodic feedback"*.

We periodically reset the parameters $\mathbf{w}$ by using the averaged parameters $\overline{\mathbf{w}}$. The interval between a feedback operation and its previous operation is called a *training period* or simply a *period*. It is important to decide when to do the feedback, i.e., the length of each period should be adjusted reasonably as the training goes on. For example, at the early stage of the training, the $\mathbf{w}$ is highly noisy, so that the feedback operation to $\mathbf{w}$ should be performed more frequently. As the training goes on, less frequent feedback operation would be better in order to adequately optimize the parameters. In practice, we adopt a schedule of *linearly slowing-down feedback*, and we will show the reasonable convergence properties of this scheduling in Section 5.2.

Notes m is the number of periods when the ASF reaches the convergence;
b is the current number of period;
c is the current number of iteration;
n is the number of training samples;
The decaying rate, $\gamma \leftarrow \frac{\gamma_0}{1+b/Z}$, is only for theoretical analysis. In practice we can simply set $\gamma \leftarrow 1$, i.e., remove the decaying rate.

Procedure ASF-train
  Initialize $\mathbf{w}$ with random values
  $c \leftarrow 0$
  for $b \leftarrow 1$ to m
  . $\gamma \leftarrow \frac{\gamma_0}{1+b/Z}$ with $Z \gg n$, or simply $\gamma \leftarrow 1$
  . **for** 1 to b
  . $\mathbf{w} \leftarrow$ SGD-update($\mathbf{w}$)
  . $c \leftarrow c + b$
  . $\mathbf{w} \leftarrow \overline{\mathbf{w}}^{iter(c)}$ in Eq. 8
  Return $\mathbf{w}$

Procedure SGD-update($\mathbf{w}$)
  for 1 to n
  . select a sample j randomly
  . $\mathbf{w} \leftarrow \mathbf{w} + \gamma \nabla_{\mathbf{w}} \ell(j, \mathbf{w})$
  Return $\mathbf{w}$

Figure 1: The ASF training.

In what follows, we call the proposal as *averaged SGD with feedback (ASF)*. Figure 1 shows the steps of the ASF. Now, we analyze the averaged parameters produced by each period. We denote $\mathbf{w}^{b,c,d}$ as the model parameters after the d'th sample is processed in the c'th iteration of the b'th period. Without making any difference, we denote $\mathbf{w}^{b,c,d}$ more simply as $\mathbf{w}^{b,cn+d}$ where n is the number of samples in a training data. Similarly, we use $g^{b,cn+d}$ to denote $\nabla_{\mathbf{w}} \ell(d, \mathbf{w})$ in the c'th iteration of the b'th period. Let $\gamma^{(b)}$ be the decaying rate in the b'th period. Let $\overline{\mathbf{w}}^{(b)}$ be the averaged parameters produced by the b'th period. We can induce

the explicit form of $\overline{\mathbf{w}}^{(1)}$:

$$\overline{\mathbf{w}}^{(1)} = \mathbf{w}^{1,0} + \gamma^{(1)} \sum_{d=1\ldots n} \left(\frac{n-d+1}{n} g^{1,d} \right). \tag{9}$$

When the 2nd period ends, the parameters are again averaged over all previous model parameters, $\mathbf{w}^{1,0}, \ldots, \mathbf{w}^{1,n}, \mathbf{w}^{2,0}, \ldots, \mathbf{w}^{2,2n}$, and it can be expressed as:

$$\overline{\mathbf{w}}^{(2)} = \mathbf{w}^{1,0} + \gamma^{(1)} \sum_{d=1\ldots n} \left(\frac{n-d+1}{n} g^{1,d} \right) + \gamma^{(2)} \sum_{d=1\ldots 2n} \left(\frac{2n-d+1}{3n} g^{2,d} \right). \tag{10}$$

Similarly, the averaged parameters produced by the b'th period can be expressed as follows:

$$\overline{\mathbf{w}}^{(b)} = \mathbf{w}^{1,0} + \sum_{i=1\ldots b} \left[\gamma^{(i)} \sum_{d=1\ldots in} \left(\frac{in-d+1}{ni(i+1)/2} g^{i,d} \right) \right]. \tag{11}$$

The procedure of deriving Eq.11 is sketched in Appendix.

5.2 Convergence Analysis

The best possible convergence result for stochastic learning is the *"almost sure convergence"*: to prove that the stochastic algorithm converges towards the solution with probability 1 (Bottou, 1998). We will show that the proposed method guarantees to achieve *almost sure convergence*.

5.2.1 Convex Case

We first introduce the general convexity assumption: everywhere in the parameter space, the opposite of the gradient must point toward a unique minimum $\mathbf{w}^*$. Such a strong assumption is only valid for a few simple learning algorithms. Nevertheless, the assumption usually holds within the final convergence region because the cost function is locally convex in many practical applications.

If a stochastic update is convergent, it means that either the gradients or the learning rates must vanish near the optimum (Bottou, 2004). According to (Bottou, 2004), it is reasonable to assume that the variance of the stochastic gradient does not grow faster than the norm of the real gradient itself. Also, it is reasonable to assume that $||\nabla_{\mathbf{w}} \ell(i, \mathbf{w})||^2$ behaves quadratically within the final convergence region. Both assumptions are conveniently expressed as follows:

$$\mathrm{E}_i(||\nabla_{\mathbf{w}} \ell(i, \mathbf{w})||^2) \leq A + B(\mathbf{w} - \mathbf{w}^*)^2, \tag{12}$$

where $A \geq 0$ and $B \geq 0$. Based on the assumptions, the *convergence theorem* has been given (Bottou, 2004): two conditions on the learning rate are sufficient conditions for the *almost sure convergence* of the SGD to the optimum $\mathbf{w}^*$. The two conditions on the learning rate are as follows (Bottou, 2004):

$$\sum \gamma_k = \infty \quad and \quad \sum \gamma_k^2 < \infty. \tag{13}$$

Too fast decaying rate may make the SGD fail to reach the optimum if it is far away, while too slow decaying rate may make the SGD keep oscillating around.

With those preparations, we have the following theorem for the ASF:

Theorem 1: *Let the optimization procedure be defined in Figure 1. Given the convex assumption and the assumption Eq. 12, the averaged parameters produced at the end of each period of the optimization procedure are "almost surely convergent" towards the optimum* $\mathbf{w}^*$.

The proof of Theorem 1 is sketched in Appendix. As we discussed before, although the convex assumption is a strong one, in practice the assumption usually holds within the final convergence region for non-convex cost functions. From another point of view, for non-convex cost functions, the convergence is probably not a big issue for practitioners because normally the training has to be terminated at a certain number of iterations in practice (Tsuruoka et al., 2009).

5.2.2 Non-convex Case

When the loss function is non-convex (e.g., the case of LCRFs), the convergence analysis is more difficult. Instead of proving that the model weights converge, we prove the loss function $-L(\mathbf{w})$ and its gradient $-\nabla_{\mathbf{w}} L(\mathbf{w})$ converge[1].

The convergence results rely on the following assumptions (Bottou, 1998), in addition to the assumptions of Eq. 12 and Eq. 13: First, the loss function $-L(\mathbf{w})$ is three times differentiable with continuous derivatives. The loss function has a

[1]The objective function $L(\mathbf{w})$ and its gradient also converge. However, following traditions on convex analysis, we stick on the convergence of loss function.

lower-bound that is non-negative. Second, when the norm of the weights $\mathbf{w}$ is larger than a value D, $-\mathbf{w}\nabla_{\mathbf{w}}L(\mathbf{w})$ is bounded:

$$\inf_{\mathbf{w}^2>D} -\mathbf{w}\nabla_{\mathbf{w}}L(\mathbf{w}) > 0. \tag{14}$$

Third, when the norm of the weights $\mathbf{w}$ is smaller than E with $E > D$, the norm of the update term is bounded:

$$\sup_{\mathbf{w}^2<E} ||\mathbf{g}|| \leq K. \tag{15}$$

With those preparations, we have the following theorem for general online optimization with non-convex loss function:

Theorem 2: *Let the optimization procedure be defined in Figure 1. Suppose we have the assumptions of Eq. 12, Eq. 13, and the three additional assumptions. Then, the loss function $-L(\mathbf{w})$ at the end of each period of the optimization procedure converges almost surely to $-L(\mathbf{w})^{\infty}$, and its gradient $-\nabla_{\mathbf{w}}L(\mathbf{w})$ converges almost surely to* $\mathbf{0}$.

The proof can be derived by extending the proof of (Bottou, 1998). Since the proof can be derived in a similar way following the Theorem 1, we omit the details of the proof. The convergence of the gradient $-\nabla_{\mathbf{w}}L(\mathbf{w})$ to $\mathbf{0}$ is the most interesting result. In other words, for the difficult case of non-convex loss functions, the algorithm will converge to *extremal points* of loss function. Extremal points include local optimums.

Appendix

Derivation of Eq. 11

We follow the denotations of $\mathbf{w}^{b,cn+d}$, $g^{b,cn+d}$, $\gamma^{(b)}$ and $\overline{\mathbf{w}}^{(b)}$ defined in Section 5. First, based on the optimization procedure defined in Figure 1, we analyze the resulting parameters at the end of each period. For the 1st period, the parameters after each update is as follows:

$$\begin{aligned}
&\mathbf{w}^{1,1} = \mathbf{w}^{1,0} + \gamma^{(1)}g^{1,1},\\
&\mathbf{w}^{1,2} = \mathbf{w}^{1,1} + \gamma^{(1)}g^{1,2} = \mathbf{w}^{1,0} + \gamma^{(1)}g^{1,1} + \gamma^{(1)}g^{1,2},\\
&\cdots\\
&\mathbf{w}^{1,n} = \mathbf{w}^{1,n-1} + \gamma^{(1)}g^{1,n}\\
&\quad = \mathbf{w}^{1,0} + \gamma^{(1)}g^{1,1} + \gamma^{(1)}g^{1,2} + \cdots + \gamma^{(1)}g^{1,n}.
\end{aligned}$$

At the end of the 1st period, the parameters are averaged as follows:

$$\begin{aligned}
\overline{\mathbf{w}}^{(1)} &= \frac{\sum_{d=1\ldots n}\mathbf{w}^{1,d}}{n}\\
&= \mathbf{w}^{1,0} + \gamma^{(1)}\sum_{d=1\ldots n}\left(\frac{n-d+1}{n}g^{1,d}\right).
\end{aligned}$$

Then, at the beginning of the 2nd period, we initialize the model parameters with averaged parameters: $\mathbf{w}^{2,0} \leftarrow \overline{\mathbf{w}}^{(1)}$. In the 2nd period, the parameters are updated based on the synchronized parameters, with a new decaying rate and the iterations increased to 2. When the 2nd period ends, the parameters are again averaged over all previous model parameters, $\mathbf{w}^{1,0}, \ldots, \mathbf{w}^{1,n}, \mathbf{w}^{2,0}, \ldots, \mathbf{w}^{2,2n}$:

$$\begin{aligned}
\overline{\mathbf{w}}^{(2)} &= \frac{\sum_{d=1\ldots n}\mathbf{w}^{1,d} + \sum_{d=1\ldots 2n}\mathbf{w}^{2,d}}{n+2n}\\
&= \frac{n\overline{\mathbf{w}}^{(1)} + \sum_{d=1\ldots 2n}\mathbf{w}^{2,d}}{3n}\\
&= \frac{n\overline{\mathbf{w}}^{(1)} + [2n\overline{\mathbf{w}}^{(1)} + \gamma^{(2)}\sum_{d=1\ldots 2n}(2n-d+1)g^{2,d}]}{3n}\\
&= \mathbf{w}^{1,0} + \gamma^{(1)}\sum_{d=1\ldots n}\left(\frac{n-d+1}{n}g^{1,d}\right)\\
&\quad + \gamma^{(2)}\sum_{d=1\ldots 2n}\left(\frac{2n-d+1}{3n}g^{2,d}\right).
\end{aligned}$$

In a similar way, it is straightforward to conclude that:

$$\begin{aligned}
\overline{\mathbf{w}}^{(b)} =& \mathbf{w}^{1,0} +\\
&\sum_{i=1\ldots b}\left[\gamma^{(i)}\sum_{d=1\ldots ni}\left(\frac{ni-d+1}{n\sum_{k=1\ldots i}k}g^{i,d}\right)\right].
\end{aligned} \tag{16}$$

□

Proof of theorem 1

In Eq. 16, notice that the $\overline{\mathbf{w}}^{(b)}$ can be explicitly expressed as the form $\overline{\mathbf{w}}^{(b)} = \mathbf{w}^{1,0} + \overline{\gamma}_1 g_1 + \overline{\gamma}_2 g_2 + \cdots + \overline{\gamma}_i g_i$, where $g_1 \ldots g_i$ represents the respective gradients in the right side of Eq. 16 and $\overline{\gamma}_1 \ldots \overline{\gamma}_i$ are the corresponding factors of those gradients.

Now, we try to prove that the decaying rates $\overline{\gamma}_1 \ldots \overline{\gamma}_i$ satisfy the conditions: $\sum \overline{\gamma}_i = \infty$ and

$\sum \overline{\gamma}_i^2 < \infty$ (see Eq. 13). First,

$$\begin{aligned}
&\lim_{b\to\infty} \sum \overline{\gamma}_i \\
&= \lim_{b\to\infty} \sum_{i=1\ldots b} \left(\gamma^{(i)} \sum_{d=1\ldots ni} \frac{ni-d+1}{n\sum_{k=1\ldots i} k} \right) \\
&= \lim_{b\to\infty} \sum_{i=1\ldots b} \left(\frac{\gamma_0}{1+i/n} \sum_{d=1\ldots ni} \frac{ni-d+1}{n\sum_{k=1\ldots i} k} \right) \\
&= \lim_{b\to\infty} \sum_{i=1\ldots b} \frac{\sum_{d=1\ldots ni}(ni-d+1)}{ni\sum_{k=1\ldots i} k} \\
&= \lim_{b\to\infty} \sum_{i=1\ldots b} \frac{1}{i} \\
&= \infty
\end{aligned} \tag{17}$$

Similarly,

$$\begin{aligned}
&\lim_{b\to\infty} \sum \overline{\gamma}_i^2 \\
&= \lim_{b\to\infty} \sum_{i=1\ldots b} \left[\left(\frac{\gamma_0}{1+i/n} \right)^2 \sum_{d=1\ldots ni} \left(\frac{ni-d+1}{n\sum_{k=1\ldots i} k} \right)^2 \right] \\
&= \lim_{b\to\infty} \sum_{i=1\ldots b} \frac{1}{i^3} \\
&< \infty
\end{aligned} \tag{18}$$

Then, given the convex assumption and the assumption Eq. 12, with the two conditions, Eq. 17 and Eq. 18, we can apply the *convergence theorem* (Bottou, 2004; Spall, 2005) (see Section 5.2) to prove the *almost sure convergence* of $\overline{\mathbf{w}}^{(b)}$ towards the optimum $\mathbf{w}^*$. The proof is based on a relaxation of convex assumption to the case of the new update term, g_i, in ASF training.

□

References

David Bannach, Oliver Amft, and Paul Lukowicz. 2008. Rapid prototyping of activity recognition applications. *IEEE Pervasive Computing* 7(2):22–31.

Oresti Banos, Miguel Damas, Hctor Pomares, Alberto Prieto, and Ignacio Rojas. 2012. Daily living activity recognition based on statistical feature quality group selection. *Expert Syst. Appl.* 39(9):8013–8021.

Ling Bao and Stephen S. Intille. 2004. Activity recognition from user-annotated acceleration data. In *Pervasive Computing*. Springer, Lecture Notes in Computer Science 3001, pages 1–17.

Peter L. Bartlett, Elad Hazan, and Alexander Rakhlin. 2007. Adaptive online gradient descent. In *Proceedings of NIPS'07*. MIT Press.

Léon Bottou. 1998. Online algorithms and stochastic approximations. *Online Learning and Neural Networks. Saad, David. Cambridge University Press* .

Léon Bottou. 2004. Stochastic learning. *Advanced Lectures on Machine Learning* pages 146–168.

Léon Bottou and Yann LeCun. 2003. Large scale online learning. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, Dirk Haehnel, Beverly Harrison, Bruce Hemingway, Jeffrey Hightower, Predrag Klasnja, Karl Koscher, Anthony LaMarca, James A. Landay, Louis LeGrand, Jonathan Lester, Ali Rahimi, Adam Rea, and Danny Wyatt. 2008. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing* 7(2):32–41.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 1–8.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *J. Mach. Learn. Res. (JMLR)* 9:1775–1822.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research* 10:2873–2898.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*. COLING '10, pages 358–366.

T. Huynh, M. Fritz, and B. Schiele. 2008. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, pages 10–19.

Holger Junker. 2005. *Human activity recognition and gesture spotting with body-worn sensors.*. Ph.D. thesis, Phd Thesis, ETH Zurich.

Matthew Keally, Gang Zhou, Guoliang Xing, Jianxin Wu, and Andrew Pyles. 2011. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, SenSys '11, pages 246–259.

Wesley Kerr, Anh Tran, and Paul R. Cohen. 2011. Activity recognition with finite state machines. In *IJCAI'11*. pages 1348–1353.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*. pages 282–289.

John Langford, Lihong Li, and Tong Zhang. 2009. Sparse online learning via truncated gradient. *Journal of Machine Learning Research* 10:777–801.

Xiao Lin. 2010. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research* 11:2543–2596.

Bappaditya Mandal and How-Lung Eng. 2012. Regularized discriminant analysis for holistic human activity recognition. *IEEE Intelligent Systems* 27(1):21–31.

Louis-Philippe Morency, Ariadna Quattoni, and Trevor Darrell. 2007. Latent-dynamic discriminative models for continuous gesture recognition. In *Proceedings of CVPR'07*. pages 1–8.

Jorge Nocedal and Stephen J. Wright. 1999. Numerical optimization. *Springer* .

Slav Petrov. 2010, to appear. Products of random latent variable grammars. In *Proceedings of NAACL'10*.

Slav Petrov and Dan Klein. 2008. Discriminative log-linear grammars with latent variables. In *Advances in Neural Information Processing Systems 20 (NIPS)*. pages 1153–1160.

Juha Präkkä, Miikka Ermes, Panu Korpipää, Jani Mäntyjärvi, Johannes Peltola, and Ilkka Korhonen. 2006. Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine* 10(1):119–128.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(10):1848–1852.

Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. 2005. Activity recognition from accelerometer data. In *Proceedings of AAAI'05*. pages 1541–1546.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of ICML'07*. pages 807–814.

Shai Shalev-Shwartz and Ambuj Tewari. 2009. Stochastic methods for l1 regularized loss minimization. In *Proceedings of ICML'09*.

James C. Spall. 2005. Introduction to stochastic search and optimization. *Wiley-IEEE* .

Xu Sun, Hisashi Kashima, and Naonori Ueda. 2013. Large-scale personalized human activity recognition using online multitask learning. *IEEE Trans. Knowl. Data Eng.* 25(11):2551–2563.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009a. Latent variable perceptron algorithm for structured classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*. pages 1236–1242.

Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2008. Modeling latent-dynamic in shallow parsing: A latent conditional model with imrpoved inference. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*. pages 841–848.

Xu Sun, Naoaki Okazaki, and Jun'ichi Tsujii. 2009b. Robust approach to abbreviating terms: A discriminative latent variable model with global information. In *Proceedings of the ACL'09*. Suntec, Singapore, pages 905–913.

Xu Sun and Jun'ichi Tsujii. 2009. Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation. In *Proceedings of EACL'09*. pages 772–780.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 477–485.

S.V.N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic meta-descent. In *International Conference on Machine Learning (ICML)*. pages 969–976.

Jie Yin, Dou Shen, Qiang Yang, and Ze-Nian Li. 2005. Activity recognition through goal-based segmentation. In *Proceedings of AAAI'05*. pages 28–34.

Jie Yin, Qiang Yang, and Junfeng Pan. 2008. Sensor-based abnormal human-activity detection. *IEEE Trans. Knowl. Data Eng. (TKDE)* 20(8):1082–1090.