Large Scale Real-Life Action Recognition Using Conditional Random Fields with Stochastic Training

Xu Sun[1], Hisashi Kashima[1], Ryota Tomioka[1], and Naonori Ueda[2]

[1] Department of Mathematical Informatics, The University of Tokyo
{xusun,kashima,tomioka}@mist.i.u-tokyo.ac.jp
[2] NTT Communication Science Laboratories, Kyoto, Japan
ueda@cslab.kecl.ntt.co.jp

Abstract. Action recognition is usually studied with limited lab settings and a small data set. Traditional lab settings assume that the start and the end of each action are known. However, this is not true for the real-life activity recognition, where different actions are present in a continuous temporal sequence, with their boundaries unknown to the recognizer. Also, unlike previous attempts, our study is based on a large-scale data set collected from real world activities. The novelty of this paper is twofold: (1) Large-scale non-boundary action recognition; (2) The first application of the averaged stochastic gradient training with feedback (ASF) to conditional random fields. We find the ASF training method outperforms a variety of traditional training methods in this task.

Keywords: Continuous Action Recognition, Conditional Random Fields, Online Training.

1 Introduction

Acceleration sensor based action recognition is useful in practical applications [1,2,3,4]. 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 action 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 (actions) can be automatically identified with a certain level of accuracy.

Although there is a considerable literature on action recognition, most of the prior work discusses action recognition in a pre-defined limited environment [1,2,3]. It is unclear whether or not the previous methods perform well in a more natural real-life environment. For example, most of the prior work assumes that the beginning and ending time of each action are known to the target recognizing system, and the produced system only performs simple classifications to the

J.Z. Huang, L. Cao, and J. Srivastava (Eds.): PAKDD 2011, Part II, LNAI 6635, pp. 222–233, 2011.

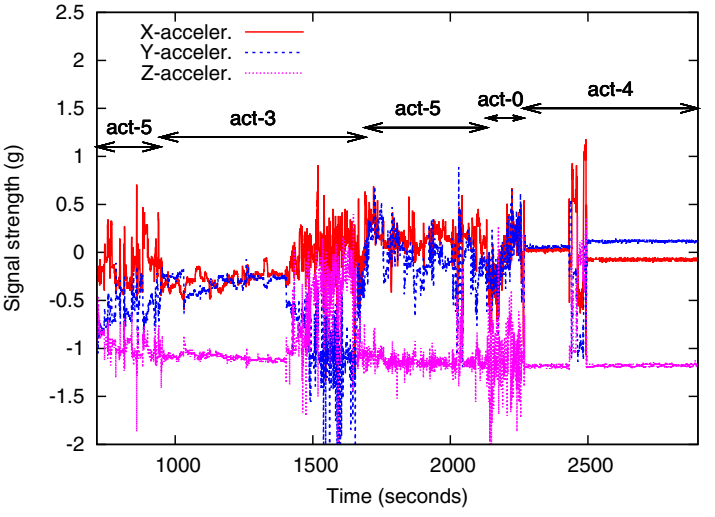

Fig. 1. An example of real-life continuous actions in our data, in which the corresponding 3D acceleration signals are collected from the attached sensors. See Section 5 for the meaning of the 'g' and action types, act-0 to act-5.

action signals [1,2,3]. However, this is not the case for real-life action sequences of human beings, in which different types of actions 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 actions are unknown to the target action recognition system. An example of real-life actions with continuous sensor signals is shown in Figure 1. For this concern, it is necessary and important to develop a more powerful system not only to predict the types of the actions, but also to disambiguate the boundaries of those actions.

With this motivation, we collected a large-scale real-life action data (continuous sensor-based three-dimension acceleration signals) from about one hundred people for continuous real-life action recognition. We adopt a popular structured classification model, conditional random fields (CRFs), for recognizing the action types and at the same time disambiguate the action boundaries. Moreover, good online training methods are necessary for training CRFs on a large-scale data in our task. We will compare different online training methods for training CRFs on this action recognition data.

2 Related Work and Motivations

Most of the prior work on action recognition treated the task as a single-label classification problem [1,2,3]. Given a sequence of sensor signals, the action recognition system predicts a single label (representing a type of action) for the whole

sequence. Ravi et al. [3] used decision trees, support vector machines (SVMs) and K-nearest neighbors (KNN) models for classification. Bao and Intille [1] and Pärkkä et al. [2] used decision trees for classification. A few other works treated the task as a structured classification problem. Huynh et al. [4] tried to discover latent activity patterns by using a Bayesian latent topic model.

Most of the prior work of action recognition used a relatively small data set. For example, in Ravi et al. [3], the data was collected from two persons. In Huynh et al. [4], the data was collected from only one person. In Pärkkä et al. [2], the data was collected from 16 persons.

There are two major approaches for training conditional random fields: 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) [5]. 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 on training CRFs.

A promising fast online training method is the stochastic gradient method, for example, the stochastic gradient descent (SGD) [6,7]. The parameters of the model are updated much more frequently, and much fewer iterations are needed before the convergence. For large-scale data sets, the SGD can be much faster than batch gradient based training methods. However, there are problems on the current SGD literature: (1) The SGD 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). (2) The SGD is not robust. It contains many hyper-parameters (not only regularization, but also learning rate) and it is quite sensitive to them. Tuning the hyper-parameters for SGD is not a easy task.

To deal with the problems of the traditional training methods, we use a new online gradient-based learning method, the averaged SGD with feedback (ASF) [8], for training conditional random fields. According to the experiments, the ASF training method is quite robust for training CRFs for the action recognition task.

3 Conditional Random Fields

Many traditional structured classification models may suffer from a problem, which is usually called "the label bias problem" [9,10]. Conditional random fields (CRFs) are proposed as an alternative solution for structured classification by solving "the label bias problem" [10]. 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 [10,11,12]:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) = \frac{\exp[\mathbf{\Theta} \cdot \mathbf{f}(\mathbf{y}, \mathbf{x})]}{\sum_{\forall \mathbf{y}} \exp[\mathbf{\Theta} \cdot \mathbf{f}(\mathbf{y}, \mathbf{x})]}, \tag{1}$$

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

Typically, computing $\sum_{\forall \mathbf{y}} \exp \mathbf{\Theta} \mathbf{f}(\mathbf{y}, \mathbf{x})$ could be computationally intractable: it is too large to explicitly sum over all possible label sequences. However, if the dependencies between labels have a linear-chain structure, this summation can be computed using dynamic programming techniques [10]. To make the dynamic programming techniques applicable, the dependencies of labels must be chosen to obey the Markov property. More precisely, we use Forward-Backward algorithm for computing the summation in a dynamic programming style. 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,

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

The first term of this equation represents a conditional log-likelihood of a training data. The second term is a regularizer for reducing overfitting. In what follows, we denote the conditional log-likelihood of each sample $\log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{\Theta})$ as $L_s(i, \mathbf{\Theta})$, and therefore:

$$L(\mathbf{\Theta}) = \sum_{i=1}^{n} L_s(i, \mathbf{\Theta}) - R(\mathbf{\Theta}). \tag{3}$$

3.1 Stochastic Gradient Descent

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 1, and it gives the maximum frequency of updates, which we adopt in this work. Then, the model parameters are updated in such a way:

$$\mathbf{\Theta}^{k+1} = \mathbf{\Theta}^{k} + \gamma_k \frac{\partial}{\partial \mathbf{\Theta}} [L_s(i, \mathbf{\Theta}) - R(\mathbf{\Theta})],$$

where k is the update counter and γ_k is the learning rate. A proper learning rate can guarantee the convergence of the SGD method [6,7]. A typical convergent choice of learning rate can be found in Collins *et al.* [13]:

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

where γ_0 is a constant. This scheduling guarantees ultimate convergence [6,7]. In this paper we adopt this learning rate schedule for the SGD.

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 learning rate, $\gamma \longleftarrow \frac{\gamma_0}{1+b/Z}$, is only for theoretical analysis. In practice we can simply set $\gamma \leftarrow 1$, i.e., remove the learning rate.

Procedure ASF-train
Initialize $\mathbf{\Theta}$ with random values
$c \longleftarrow 0$
for $b \longleftarrow 1$ to m
. $\gamma \longleftarrow \frac{\gamma_0}{1+b/Z}$ with $Z \gg n$, or simply $\gamma \leftarrow 1$
. **for** 1 to b
. $\mathbf{\Theta} \longleftarrow$ SGD-update($\mathbf{\Theta}$)
. $c \longleftarrow c + b$
. $\mathbf{\Theta} \longleftarrow \overline{\mathbf{\Theta}}^{iter(c)}$ in Eq. 4
Return $\mathbf{\Theta}$

Procedure SGD-update($\mathbf{\Theta}$)
for 1 to n
. select a sample j randomly
. $\mathbf{\Theta} \longleftarrow \mathbf{\Theta} + \gamma \frac{\partial}{\partial \mathbf{\Theta}} L_s(j, \mathbf{\Theta})$
Return $\mathbf{\Theta}$

Fig. 2. The major steps of the ASF training

4 Averaged SGD with Feedback

Averaged SGD with feedback (ASF) is a modification and extension of the traditional SGD training method [8]. The naive version of averaged SGD is inspired by the averaged perceptron technique [14]. Let $\mathbf{\Theta}^{iter(c),sample(d)}$ 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:

$$\overline{\mathbf{\Theta}}^{iter(c')} \triangleq \frac{\sum_{c=1...c',\ d=1...n} \mathbf{\Theta}^{iter(c),sample(d)}}{nc'}. \tag{4}$$

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

The ASF periodically resets the parameters $\mathbf{\Theta}$ by using the averaged parameters $\overline{\mathbf{\Theta}}$. 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 $\mathbf{\Theta}$ is highly noisy, so that the feedback operation to $\mathbf{\Theta}$ 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, the ASF adopts a schedule of *linearly slowing-down feedback*: the number of iterations increases linearly in each period, as the training goes on.

Figure 2 shows the steps of the ASF. We denote $\mathbf{\Theta}^{b,c,d}$ 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 $\mathbf{\Theta}^{b,c,d}$ more simply as $\mathbf{\Theta}^{b,cn+d}$ where n is the number of samples in a training data. Similarly, we use $g^{b,cn+d}$ to denote $\frac{\partial}{\partial \mathbf{\Theta}} L_s(d, \mathbf{\Theta})$ in the c'th iteration of the b'th period. Let $\gamma^{(b)}$ be the learning rate in the b'th period. Let $\overline{\mathbf{\Theta}}^{(b)}$ be the averaged parameters produced by the b'th period. We can induce the explicit form of $\overline{\mathbf{\Theta}}^{(1)}$:

$$\overline{\mathbf{\Theta}}^{(1)} = \mathbf{\Theta}^{1,0} + \gamma^{(1)} \sum_{d=1\ldots n} \frac{n-d+1}{n} g^{1,d}. \tag{5}$$

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

$$\begin{aligned} \overline{\mathbf{\Theta}}^{(2)} = \mathbf{\Theta}^{1,0} &+ \gamma^{(1)} \sum_{d=1\ldots n} \frac{n-d+1}{n} g^{1,d} \\ &+ \gamma^{(2)} \sum_{d=1\ldots 2n} \frac{2n-d+1}{3n} g^{2,d}. \end{aligned} \tag{6}$$

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

$$\overline{\mathbf{\Theta}}^{(b)} = \mathbf{\Theta}^{1,0} + \sum_{i=1\ldots b} (\gamma^{(i)} \sum_{d=1\ldots in} \frac{in-d+1}{ni(i+1)/2} g^{i,d}). \tag{7}$$

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 [6]. The ASF guarantees to achieve *almost sure convergence* [8]. The averaged parameters produced at the end of each period of the optimization procedure of the ASF training are "almost surely convergent" towards the optimum $\mathbf{\Theta}^*$ [8]. On the implementation side, there is no need to keep all the gradients in the past for computing the averaged gradient $\overline{\mathbf{\Theta}}$: we can compute $\overline{\mathbf{\Theta}}$ on the fly, just like the averaged perceptron case.

5 Experiments and Discussion

We use one month data of the ALKAN dataset [15] for experiments. This is a new data, and the data contains 2,061 sessions, with totally 3,899,155 samples

Table 1. Features used in the action recognition task. For simplicity, we only describe the features on *x-axis*, because the features on *y-axis* and *z-axis* are in the same setting like the *x-axis*. $\mathbb{A} \times \mathbb{B}$ means a Cartesian product between the set $\mathbb{A}$ and the set $\mathbb{B}$. The time interval feature *do not* record the absolute time from the beginning to the current window. This feature only records the time difference between two neighboring windows: sometimes there is a jump of time between two neighboring windows.

Signal strength features:
$\{s_{i-2},\ s_{i-1},\ s_i,\ s_{i+1},\ s_{i+2},\ s_{i-1}s_i,\ s_i s_{i+1}\} \times \{y_i,\ y_{i-1}y_i\}$
Time interval features:
$\{t_{i+1} - t_i,\ t_i - t_{i-1}\} \times \{y_i,\ y_{i-1}y_i\}$
Mean, standard deviation, energy, covariance features:
$m_i \times \{y_i,\ y_{i-1}y_i\}$
$d_i \times \{y_i,\ y_{i-1}y_i\}$
$e_i \times \{y_i,\ y_{i-1}y_i\}$
$\{c_{x,y,i}, c_{y,z,i}, c_{x,z,i}\} \times \{y_i,\ y_{i-1}y_i\}$

(in a temporal sequence). The data was collected by *iPod* accelerometers with the sampling frequency of 20HZ. A sample contains 4 values: {*time (the seconds past from the beginning of a session), x-axis-acceleration, y-axis-acceleration, z-axis-acceleration*}, for example, {539.266(s), 0.091(g), -0.145(g), -1.051(g)}[1]. There are six kinds of action labels: `act-0` means *"walking or running"*, `act-1` means *"on an elevator or escalator"*, `act-2` means *"taking car or bus"*, `act-3` means *"taking train"*, `act-4` means *"up or down stairs"*, and `act-5` means *"standing or sitting"*.

5.1 How to Design and Implement Good Features

We split the data into a training data (85%), a development data for hyper-parameters (5%), and the final evaluation data (10%). The evaluation metric are sample-accuracy (%) (equals to recall in this task: the number of correctly predicted samples divided by the number of all the samples). Following previous work on action recognition [1,2,3,4], we use acceleration features, mean features, standard deviation, energy, and correlation (covariance between different axis) features. Features are extracted from the iPod accelerometer data by using a window size of 256. Each window is about 13 seconds long. For two consecutive windows (each one contains 256 samples), they have 128 samples overlapping to each other. Feature extraction on windows with 50% of the window overlapping was shown to be effective in previous work [1]. The features are listed in Table 1. All features are used without pruning. We use exactly the same feature set for all systems.

[1] In the example, 'g' is the *acceleration of gravity*.

The mean feature is simply the averaged signal strength in a window:

$$m_i = \frac{\sum_{k=1}^{|w|} s_k}{|w|},$$

where $s_1, s_2, \ldots$ are the signal magnitudes in a window. The energy feature is defined as follows:

$$e_i = \frac{\sum_{k=1}^{|w|} s_k^2}{|w|}.$$

The deviation feature is defined as follows:

$$d_i = \sqrt{\frac{\sum_{k=1}^{|w|}(s_k - m_i)^2}{|w|}},$$

where the m_i is the mean value defined before. The correlation feature is defined as follows:

$$c_i(x, y) = \frac{covariance_i(x, y)}{d_i(x)d_i(y)},$$

where the $d_i(x)$ and $d_i(y)$ are the deviation values on the i'th window of the x-axis and the y-axis, respectively. The $covariance_i(x, y)$ is the covariance value between the i'th windows of the x-axis and the y-axis. In the same way, we can define $c_i(y, z)$ and $c_i(x, z)$.

A naive implementation of the proposed features is to design several real-value feature templates representing the mean value, standard deviation value, energy value, and correlation value, and so on. However, in preliminary experiments, we found that the model accuracy is low based on such straightforward implementation of real features. A possible reason is that the different values of a real value (e.g., the standard deviation) may contain different indications on the action, and the difference of the indications can not be directly reflected by evaluating the difference of their real values. The most easy way to deal with this problem is to split an original real value feature into multiple features (can still be real value features). In our case, the feature template function automatically splits the original real value features into multiple real value features by using a heuristic splitting interval of 0.1. For example, the standard deviations of 0.21 and 0.31 correspond to two different feature IDs, and they correspond to two different model parameters. The standard deviations of 0.21 and 0.29 correspond to an identical feature ID, with only difference on the feature values. In our experiment, we found splitting the real features improves the accuracy (more than 1%).

It is important to describe the implementation of edge features, which are based on the label transitions, $y_{i-1}y_i$. For traditional implementation of CRF systems (e.g., the HCRF package), usually the edges features contain only the information of y_{i-1} and y_i, without the information of the observation sequence (i.e., $\mathbf{x}$). The major reason for this simple implementation of edge features is for reducing the dimension of features. Otherwise, there can be an explosion of edge features in some tasks. For our action recognition task, since the feature

Table 2. Comparisons among methods on the sensor-based action recognition task. The number of iterations are decided when a training method goes to its empirical convergence state. The *deviation* means the standard deviation of the accuracy over four repeated experiments.

Methods	Accuracy	Iteration	Deviation	Training time
ASF	**58.97%**	60	0.56%	0.6 hour
Averaged SGD	57.95%	**50**	**0.28%**	**0.5** hour
SGD	55.20%	130	0.69%	1.3 hour
LBFGS (batch)	57.85%	800	0.74%	8 hours

dimension is quite small, we can combine observation information of $\mathbf{x}$ with label transitions $y_{i-1}y_i$, and therefore make "rich edge features". We simply used the same observation templates of node features for building rich edge features (see Table 1). We found the rich edge features significantly improves the prediction accuracy of CRFs.

5.2 Experimental Setting

Three baselines are adopted to make a comparison with the ASF method, including the traditional SGD training (SGD), the SGD training with parameter-averaging but without feedback (averaged SGD), and the popular batch training method, limited memory BFGS (LBFGS).

For the LBFGS batch training method, which is considered to be one of the best optimizers for log-linear models like CRFs, we use the OWLQN open source package [16][2]. The hyper-parameters for learning were left unchanged from the default settings of the software: the convergence tolerance was 1e-4; and the LBFGS memory parameter was 10.

To reduce overfitting, we employed an L2 prior $R(\mathbf{\Theta}) = \frac{||\mathbf{\Theta}||^2}{2\sigma^2}$ for both SGD and LBFGS, by setting $\sigma = 5$. For the ASF and the averaged SGD, we did not employ regularization priors, assuming that the ASF and the averaged SGD contain implicit regularization by performing parameter averaging. For the stochastic training methods, we set the γ_0 as 1.0. We will also test the speed of the various methods. The experiments are run on a Intel Xeon 3.0GHz CPU, and the time for feature generation and data input/output is excluded from the training cost.

5.3 Results and Discussion

The experimental results are listed in Table 2, and the more detailed results of the respective action categories are listed in Table 3. Since recognizing actions from real-life continuous signals require the action-identification and the boundary-disambiguation at the same time, it is expected to be much more difficult than

[2] Available online at: `https://www.cs.washington.edu/homes/galen/` or `http://research.microsoft.com/en-us/um/people/jfgao/`

Table 3. Comparisons among methods on different action labels

action types	*walk/run*	*on elevat.*	*car/bus*	*train*	*stairs*	*stand/sit*	*overall*
# samples	2,246	37	3,848	2,264	221	5,275	13,891
Acc.(%) ASF	62.09	0	77.07	25.43	0	62.31	58.97
Acc.(%) Averaged SGD	60.84	0	76.35	25.88	0	61.36	57.95
Acc.(%) SGD	62.40	0	73.75	15.31	1.50	58.08	55.20
Acc.(%) LBFGS	51.66	0	76.41	22.98	1.02	66.57	57.85

the previous work on simply action-identification. An additional difficulty is that the data is quite noisy. The number of iterations are decided when a training method goes to its empirical convergence state[3].

Note that, the ASF training achieved better sample-accuracy than other online training methods. The ASF method is relatively stable among different iterations when the training goes on, while the SGD training faces severe fluctuation when the training goes on. The averaged SGD training reached its empirical convergence state faster than the ASF training. The ASF training converged much faster than the SGD training. All of the online training methods converged faster than the batch training method, LBFGS.

In Figure 3, we show the curves of sample-accuracies on varying the number of training iterations of the ASF, the averaged SGD, and the traditional SGD. As can be seen, the ASF training is much more stable/robust than the SGD training. The fluctuation of the SGD is quite severe, probably due to the noisy data of the action recognition task. The robustness of the ASF method relates to the stable nature of the averaging technique with feedback. The ASF outperformed the averaged SGD, which indicates that the feedback technique is helpful to the naive parameter averaging. The ASF also outperformed the LBFGS batch training with much fewer iteration numbers (therefore, with much faster training speed), which is surprising.

5.4 A Challenge in Real-Life Action Recognition: Axis Rotation

One of the tough problems in this action recognition task is 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. We suppose this is an important reason that prevented the experimental results reaching a higher level of accuracy. A candidate solution to keep the consistency is to tell the people to adopt a standard rotation when

[3] Here, the empirical convergence state means an empirical evaluation of the convergence.

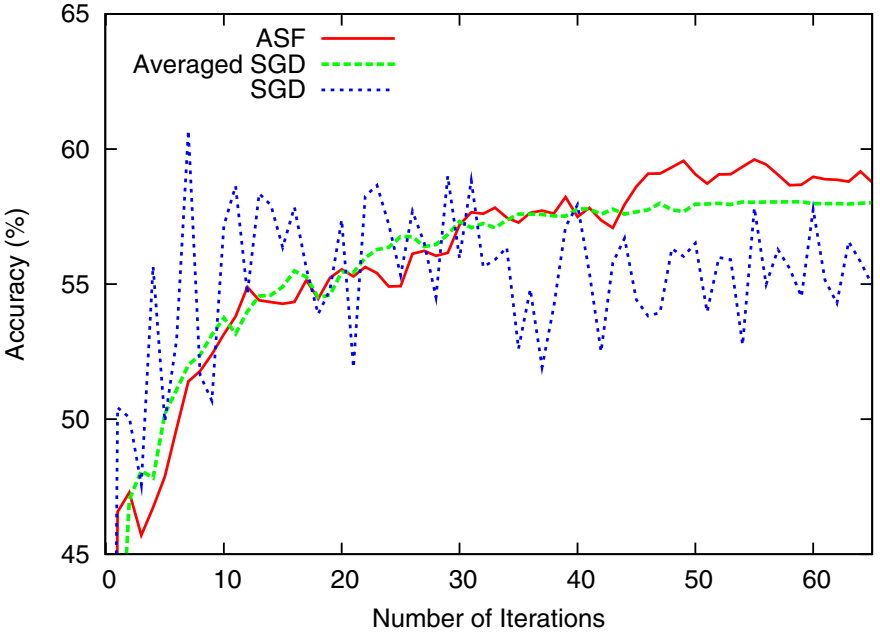

Fig. 3. Curves of accuracies of the different stochastic training methods by varying the number of iterations

collecting the data. However, this method will make the collected data not "natural" or "representative", because usually people put the accelerometer sensor (e.g., in iPod or iPhone) randomly in their pocket in daily life.

6 Conclusions and Future Work

In this paper, we studied automatic non-boundary action recognition with a large-scale data set collected in real-life activities. Different from traditional simple classification approaches to action recognition, we tried to investigate real-life continuous action recognition, and adopted a sequential labeling approach by using conditional random fields. To achieve good performance in continuous action recognition, we presented how to design and implement useful features in this task.

We also compared different online optimization methods for training conditional random fields in this task. The ASF training method demonstrated to be a very robust training method in this task with noisy data, and with good performance. As future work, we plan to deal with the axis rotation problem through a principled statistical approach.

Acknowledgments

X.S., H.K., and N.U. were supported by the FIRST Program of JSPS. We thank Hirotaka Hachiya for helpful discussion.

References

1. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
2. Präkkä, J., Ermes, M., Korpipää, P., Mäntyjärvi, J., Peltola, J., Korhonen, I.: Activity classification using realistic data from wearable sensors. IEEE Transactions on Information Technology in Biomedicine 10(1), 119–128 (2006)
3. Ravi, N., Dandekar, N., Mysore, P., Littman, M.L.: Activity recognition from accelerometer data. In: AAAI, pp. 1541–1546 (2005)
4. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 10–19. ACM, New York (2008)
5. Nocedal, J., Wright, S.J.: Numerical optimization. Springer, Heidelberg (1999)
6. Bottou, L.: Online algorithms and stochastic approximations. In: Saad, D. (ed.) Online Learning and Neural Networks. Cambridge University Press, Cambridge (1998)
7. Spall, J.C.: Introduction to stochastic search and optimization. Wiley-IEEE (2005)
8. Sun, X., Kashima, H., Matsuzaki, T., Ueda, N.: Averaged stochastic gradient descent with feedback: An accurate, robust and fast training method. In: Proceedings of the 10th International Conference on Data Mining (ICDM 2010), pp. 1067–1072 (2010)
9. Bottou, L.: Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole. PhD thesis, Université de Paris XI, Orsay, France (1991)
10. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning (ICML 2001), pp. 282–289 (2001)
11. Daumé III, H.: Practical Structured Learning Techniques for Natural Language Processing. PhD thesis, University of Southern California (2006)
12. Sun, X.: Efficient Inference and Training for Conditional Latent Variable Models. PhD thesis, The University of Tokyo (2010)
13. Collins, M., Globerson, A., Koo, T., Carreras, X., Bartlett, P.L.: Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. J. Mach. Learn. Res. (JMLR) 9, 1775–1822 (2008)
14. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of EMNLP 2002, pp. 1–8 (2002)
15. Hattori, Y., Takemori, M., Inoue, S., Hirakawa, G., Sudo, O.: Operation and baseline assessment of large scale activity gathering system by mobile device. In: Proceedings of DICOMO 2010 (2010)
16. Andrew, G., Gao, J.: Scalable training of L_1-regularized log-linear models. In: Proceedings of ICML 2007, pp. 33–40 (2007)