Final Report: combining signal transformation to learning models for human activity recognition

Tianyi Li, Dan Luo
Stanford University
{tianyil, dluo0123} @ stanford.edu

1 Introduction

In machine learning and deep learning fields, analysing sound, image data is not an easy task and are often dealt with large scale models such as convolutional neural networks or recurrent neural networks.

However, signal processing techniques can be used to detect components of interest in measured signals such as sound, image and biological measurements and they are more easy to implement in general. In this project, we try to combine signal processing techniques with machine learning models in order to create simpler models with higher accuracy.

The task we worked on, human activity recognition, is a multi-class classification problem. The inputs are 3-axis accelerometer signals recorded when humans are doing one of six different activities (walking, walking upstairs, walking downstairs, sitting, standing, and laying). The output would be one of the six given activities. We first apply signal processing techniques including discrete Fourier transform (DFT), discrete cosine transform (DCT), wavelet transform (WT), high-pass filtering to the signals. Then feed the data into k-NN, SVM, gradient boosting models to output a predicted activity.

2 Related Work

In [1], the authors developed a modified multiclass Support Vector Machine algorithm based on fixed-point arithmetic and achieved 89.3% accuracy on the same dataset. Naive Bayes method is used to create a model with accuracy of 86.6% on different dataset for human activities recognition with more features in [3]. A decision tree model is proposed in [4] with similar performance. In [6], the authors presents the Transition-Aware Human Activity Recognition system which outperforms state-of-the-art baseline works by combining the probabilistic output of consecutive activity predictions of a SVM with a heuristic filtering approach.

3 Dataset Description

The experiments were carried out with a group of 30 volunteers within an age bracket of 19-48 years. They performed a protocol of activities composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). All the participants were wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution. The experimenters captured 3-axial linear acceleration at a constant rate of 50Hz using the embedded accelerometer of the device. The experiments were video-recorded to label the data manually. The obtained dataset was randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). More details can be found at `http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones`

4 Methods

We intend to explore whether and how signal pre-processing improves the performance of machine learning algorithms. Here, we briefly introduce the signal pre-processing methods and machine learning models chosen for the project.

4.1 Signal Pre-processing Method

Four common signal pre-processing methods are used: discrete fourier transform (DFT), discretecosine transform (DCT), wavelet transform (WT), and high-pass filter on z-axis. The DFT decomposes a function of time (a signal) into its constituent frequencies. So it might help the model to ignore phase different (such as different starting times of doing activities) when classifying the signal. DCT is similar to DFT, but with a different set of basis that uses only real-valued functions. The advantage WT has over Fourier transforms is temporal resolution: it captures both frequency and location information (location in time).

Gravity has a significant component along z-axis and it should not be helpful in classifying activities, we apply high-pass filter on z-axis to remove gravitational components.

4.2 Machine Learning Algorithms

We use three machine learning algorithms: k-nearest-neighbor (k-NN), support vector machines (SVM) and gradient boosted trees for the multi-class classification task. The choices of algorithms are motivated by two reasons. First, since the data is labeled with categorical information, supervised classification algorithms are most suitable. Second, we are agnostic about how signals are related to activities, thus nonparametric algorithms, which do not rely on specific functional assumption, are better choices.

4.2.1 k-NN

In k-NN classification, an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). k-NN algorithm involves three main parameters: the number of neighbors, the metric used to determined distance, the weight of neighbors. In our implementation, we use the classic Euclidean metric and uniform weights; we try several different values for the number of neighbors and find k=1 has the best performance at large. k-NN model is built through [5].

4.2.2 SVM

A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

In our implementation, we use SVM with $l1$ regularization. The reason is that our dataset is bascially non-linearly separable so the crude SVM might be susceptible to outliers. Our SVM optimization is formulated as follows:

$$\begin{aligned} &\min_{\gamma,\omega,b} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}\xi_i \\ &s.t. y^{(i)}(\omega^T x^{(i)} + b) \geq 1 - \xi_i,\ i = 1, \dots n \\ &\xi_i \geq 0,\ i = 1, \dots n \end{aligned}$$

We experiment with different C, which governs how much regularization is exerted, and different kernels. It turns out RBF kernel with $C = 25$ performs well at large. SVM models are built through [5].

4.2.3 Gradient Boosting Tree

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models. In our implementation, the weak prediction model is decision tree.

Gradient boosting at the m-th step would fit a decision tree $h_m(x)$ to seudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m,\ldots,}R_{J_m}$ and predicts a constant value in each region. Therefore, $h_m(x)$ can be written as the sum:

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} 1_{R_{J_m}}(x),$$

where b_{jm} is the value predicted in the region R_{jm}. Then the coefficients b_{jm} are multiplied by some value γ_m, chosen using line search so as to minimize the loss function, and the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x),$$
$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

where $L(\cdot, \cdot)$ is the loss function. Gradient boosting model is implemented through [2].

4.2.4 Dimension reduction

In addition to the three algorithms, we also experiment with two dimension-reduction techniquess: Fisher's linear discriminant analysis (LDA) and principal component analysis (PCA). We want to check whether they are helpful in reducing the scale of models without hurting much performance. In our implementation, we reduce the data to 20-dimension. We implemented Fisher's LDA with our own code, and PCA by Scikit-learn.

5 Experiments

In this section, we present some preliminary results. Performance of models are evaluated by their accuracy on the test dataset. For each of the three algorithms, we present the results on six types of data: the raw data (`raw`), the data with discrete fourier transform (`DFT`), the data with high-pass filter on z-axis (`filtered`), the data with both discrete fourier transform and high-pass filter on z-axis (`DFT_filtered`), the data with discrete cosine transform (`DCT`), and the data with wavelet transform (`WT`). For gradient boosting method, considering it being a very powerful model, we include both the magnitudes and the angles of DFT frequency components for dataset `DFT` and `DFT_filted`, thus the dimension of the data is doubled from 384 to 768.

5.1 k-NN

For the number of neighbors k, we try 1, 3 and 5. The results are presented in the following table. Both discrete fourier transform and high-pass filter on z-axis can significantly improve the performance of k-NN algorithms. The best performance here is achieved by k-NN on DFT transformed data with k=3 or 5.

The number of neighbors	raw	DFT	filtered	DFT_filtered	DCT	WT
1	0.7233	0.8285	0.7816	0.8123	0.7152	0.8285
3	0.6456	**0.8301**	0.7104	0.8074	0.6392	0.7735
5	0.6117	**0.8301**	0.6812	0.8025	0.5922	0.7557

Compared with data in time domain, data in frequency domain results in a better result. We also observe that in the frequency domain, signals with and without filtering work similar accuracy. The reason might be that, the low frequency components give similar values of magnitudes of frequency components while they might have time shift in the time domain leading to low accuracy in the time domain.

5.2 SVM

The choices of the kernel type and penalty parameter C of the error term are very important to the performance of SVM. We use RBF kernel because it produces the best overall performance. We are agnostic about the optimal C so we try a wide range of parameters, 1, 5 and 25. The results are presented in the following table. The best performance is achieved by SVM on WT transformed data with penalty parameter C being 25.

C	raw	DFT	filtered	DFT_filtered	DCT	WT
1	0.6926	0.8172	0.7152	0.7929	0.6100	0.8285
5	0.7702	0.8382	0.7799	0.8333	0.7184	**0.8673**
25	0.8025	0.8592	0.8188	0.8463	0.7896	**0.8673**

For different C, discrete fourier transform and wavelet transform consistently improves the performance of SVM. Especially, when the performance of SVM on the raw is relatively bad, the improvement is more salient. However, high-pass filter on z-axis has little effect. We conclude that such a filter technique have some advantage when the model is very simple (underfitting) and may not help when the model becomes more complex.

5.3 Gradient Boosting

Since gradient boosting is a very powerful algorithm, its performance on the raw data is already good. On top of that, discrete fourier transform can still significantly improve the performance. High-pass filter on z-axis has little effect.

raw	DFT	filtered	DFT_filtered	DCT	WT
0.8188	**0.8657**	0.7961	0.8641	0.7848	0.8350

5.4 Analysis of Confusion Matrix

To provide more intuition on how signal pre-processing improve the performance, we briefly analyze the confusion matrices. Here, we use the confusion matrices of k-NN with k=1. Clearly, the main improvement brought by signal pre-processing is that the k-NN algorithm is less likely to mistake the first and second types of activities for the third and fourth ones. Discrete fourier transform even completely eliminates such mistakes. This also explains why high-pass filter on z-axis does not improve the performance further in addition to discrete fourier transform.

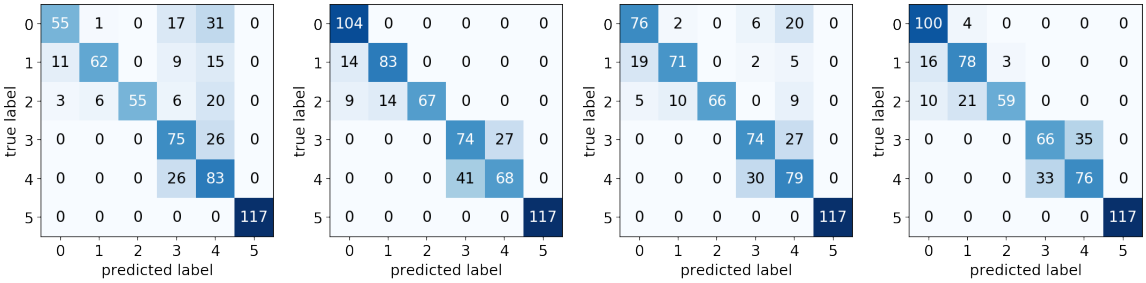

Figure 1: Confusion matrices for raw, DFT, filted, and DFT_filted

5.5 Dimension Reduction

In this subsection, we implement LDA and PCA on the raw data and pre-processed data. We want to check whether they are helpful in reducing the scale of models without hurting much performance. Since the previous analysis indicates DFT and WT transform can significantly improve prediction performance, we focus on the raw data, the data with DFT and the data with WT.

		k-NN	SVM	XGB
LDA	Raw	0.6359	0.6375	0.6317
	DFT	0.7961	0.7702	0.8204
	WT	N/A	N/A	N/A
PCA	Raw	0.7702	0.8317	0.7945
	DFT	08269	0.8722	**0.8883**
	WT	0.8333	0.8851	0.8171

According to the above table, we find PCA performs better than LDA in two senses. First, compared to the original data, PCA does not lose much predictability while LDA does. Second, PCA works much faster so has broader availability. Actually, implementing LDA on the data with WT is so slow that it cannot finish in a reasonable length of time.

6 Conculusion

Our preliminary experiments imply that signal preprocessing can effectively improve the performance of machine learning algorithms. Moreover, even though different machine learning algorithms have very distinct performance on the raw data, they have quite similar performance on the processed data. The improvement brought by signal processing is more salient for relatively simple algorithms. In addition, signal preprocessing can be paired with dimension reduction techniques to create simpler models with higher performance.

It should also be noticed that though k-NN and SVM algorithms can be kernelized, but the processing technique we implemented cannot be written into a kernel. A DFT can be written as a matrix multiplication $X = Fx$, where x is the original signal, X is the vector of frequency components, and $F \in \mathbb{C}^{n\times n}$ is the DFT matrix

$$F = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{j2\pi/n} & e^{(2)j2\pi/n} & \dots & e^{(n-1)j2\pi/n} \\ 1 & e^{j4\pi/n} & e^{(2)j4\pi/n} & \dots & e^{(n-1)j4\pi/n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{j(n-1)\pi/n} & e^{(2)j(n-1)\pi/n} & \dots & e^{(n-1)j(n-1)\pi/n} \end{bmatrix} \tag{1}$$

Then, getting the magnitude of frequency components X is equivalent to get the magnitude of complex numbers $|X|$. Such an operation does not corresponding to any kernel since its corresponding kernel matrix is not positive semi-definite.

7 Contributions

Coding: Tianyi 60%, Dan 40%
Report Writing: Tianyi 40%, Dan 60%

Our code can be found at: `https://github.com/IllidanUUU/CS229.git`

References

[1] Davide Anguita et al. “Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic.” In: *J. UCS* 19.9 (2013), pp. 1295–1314.

[2] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[3] Luciana C Jatoba et al. “Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity”. In: *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2008, pp. 5250–5253.

[4] Uwe Maurer et al. *Activity recognition and monitoring using multiple sensors on different body positions*. Tech. rep. Carnegie-mellon Univ Pittsburgh Pa School Of Computer Science, 2006.

[5] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[6] Jorge-L Reyes-Ortiz et al. “Transition-aware human activity recognition using smartphones”. In: *Neurocomputing* 171 (2016), pp. 754–767.