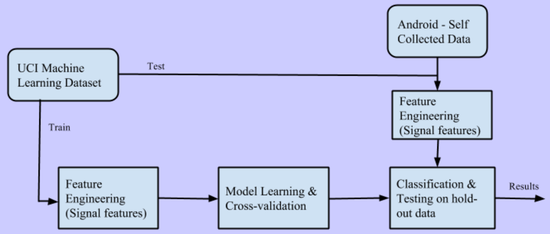
Mobile Activity Detection

Alejandro Carbonara & Pulkit Bhuwalka
Carnegie Mellon University, Machine Learning

> Abstract

Our project aims to be able to categorize human movements using the sensors typically found on a phone. We first worked with a dataset collected by another project. Using these sets, we recreated the functions they used to generate feature vectors, and tested various learning mechanisms. We have created a small application to collect mobile sensor data from our phones, and plan to use that to create our own dataset. We found that we could get good classification using a SVM and a Gaussian Markov Model.

> Method



We first tested our learning algorithms over some clean prepared data. We found several datasets online (UCI Machine Learning Repo and Kaggle) which have data from sensors annotated with activity, along with some precomputer feature vectors. After some deliberation, we chose to focus on the UCI dataset. We experimented running a support vector machine over the data. This dataset should serve as a good starting point, as it tested wheter our code works on older data. We didn't want bugs in our collection to give us trouble at the start of our project.

We modelled the time series information by taking fixed-width duration readings. Each portion gives us 128 readings in the X, Y, and Z direction for our accelerometer and gyroscope. These readings are used to compute a large number of features, creating a feature vector with 561 elements (simulating the generated feature vectors that were included with the UCI data).

This feature vector is used to train a SVM which can classify multiple classes using the 1v1 trick. We were able to get a good classification by working with a linear soft margin SVM.

While the SVM does a good job of clasifying the data, we wanted to further increase the accuracy. One of the intuitions behind it was using past state to determine the current state. We thus modeled the data as an HMM with multiple Gaussian outputs which increased our accuracy upto 4%

> Phone Data

We created an Android application that collects sensor data, performs the same form of fixed-window calculation mentioned above and writes it out to a file. The application can be instructed to collect data for all the 6 activities. The application also had options to collect data with varying frequency, window and overlap sizes.

Unfortunately, due to differences in sensor types, the data we collected was not able to be classified by our mechanisms. Our readings were differently scaled than the readings from the UCI set, and we were not able to figure out how to translate them into a a classifiable format. Furthermore, we were unable to build up a sufficiently large dataset to learn on our own data.

> Features

The UCI raw data was sampled at 50Hz with a fixed-width window with 50% overlap, 128 readings per window. The readings gave us readings for body acceleration (accelerometer), gravitational acceleration (accelerometer), and rotation (gyroscope). Each row was with one of the following labels: WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING
Training Samples - 7352
Testing Samples - 2947
Number of subjects - 30

The process we used to turn the readings into feature vectors:

1. Noise removal Median filter + 3rd order low-pass butterworth filter (Corner frequency of 20Hz)
2. Body and Gravity Component Separation Butterworth filter (Corner frequency 0.3 Hz)
3. Signal Generation (Using One Attribute from each column)

BodyAcc-XYZ		Raw		Time Domain
GravityAcc-XYZ	X	Jerk	X	Frequency Domain
BodyGyro-XYZ		Magnitude		

4. Variable estimations (over the generated signals)

mean	std	max	min	mad (Median absolute deviation)
correlation	energy	iqr	entropy	arCoeff (Autoregresion coefficients)
maxInds	meanFreq	skewness	kurtosis	sma (Signal magnitude area)
angle	maxInds (index of max frequency)			bandsEnergy (Energy of binned freqs)

5. Normalization - All the values are normalized to [-1, 1]

> SVM 1v1 Trick

Our first approach to classifying our data was to simply run a support vector machine. We used the LIBSVM library, mainly because it had many features built in that were not in the matlab svm implementation. One major feature it had was that it had built in support for SVMs with multiple category classification. The comparison is done using "1-against-1" classification. For this mechanism, we first solve the pairwise classification problem using a linear soft margin SVM.

Once we have computed each of these pairwise classifications, we use the "voting" strategy to pick a class for each point. In this strategy, the class selected is the one that is chosen by the most pairwise classifications.

We ran five-fold cross validation to check whether we selected the best mechanism to model our data. Based on this data, we set our soft margin variable c to be 1000.

> SVM Results

Running this on our hold out data, we got an accuracy of 93.38%. The confusion matrix for our classification was:

490	11	10	0	0	0	WALKING
1	460	34	2	0	0	WALKING_UPSTAIRS
5	0	376	0	0	0	WALKING_DOWNSTAIRS
0	0	0	396	39	0	SITTING
0	0	0	93	493	0	STANDING
0	0	0	0	0	537	LAYING

The (i,j)th position is the number of items classified i that were really j. The six labels in order are: WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING. From this, we can see that the various walking motions were frequently confused, and standing was often confused with sitting.

> Continuous Hidden Markov Models (with Gaussian outputs)

A Hidden Markov Model (HMM) is a natural fit for our project given that sensor output is time series data. Users generate a stream of sensor acceleration readings when they switch between different states. We consider the actual activities that a user is performing (WALKING, STANDING etc.) to be the hidden states of the model and the observed sensor readings to be output that is emmitted by the model. Each 561 dimension feature vector that we observe is considered to be one output created by the activity. We assume a gaussian distribution over each one of these 561 features. This means that we learn a mean and variance of each feature dimension conditional on each hidden state. Our steps to generate the model are as follows:

1. The prior probability of the model is given a random value and assigned across the 6 states.
2. The model is trained using the observed emissions (Xtrain) and hidden states (Ytrain) that are known to us. We directly train the model by computing the MLEs from the observations.
3. Once we have learned the model, the sequence of the test data is predicted by using the Viterbi path algorithm.

> HMM Results

The algorithm is run over the test data using various sets of observations together. We notice that the HMM helps increase the accuracy upto 97.8% With no background information (each test point individually) the accuracy is only 94.3% but it increaases by 2-4% when considered together.

Accuracy (in %)	Observations per sequence
94.3	1
94.67	5
96.2	10
97.12	20
97.05	50
97.29	100
97.79	All (3k)

Confusion Matrix

487	1	8	0	0	0	WALKING
0	470	1	0	0	0	WALKING_UPSTAIRS
5	42	373	0	0	0	WALKING_DOWNSTAIRS
0	0	0	463	27	1	SITTING
0	0	0	0	532	0	STANDING
0	0	0	0	0	537	LAYING

The (i,j)th position is the number of items classified i that were really j. Similar to the SVM case, the various walking motions were frequently confused, and standing was often confused with sitting.

> Conclusions

We were able to get good classification of our data using both SVM and HMM models. Even without taking the temporal aspects into consideration, the Hidden Markov Model with a Gaussian output was able to classify our data better than a featurized SVM.
We were unable to run our models on our personal phones, due to the conditions of collection being too different to train off of the UCI data.