Assignment2 2020 Spring Machine Learning

@Sungmoon Yoon 2012147567

Dataset

Activity recognition with healthy older people using a batteryless wearable sensor Data Set

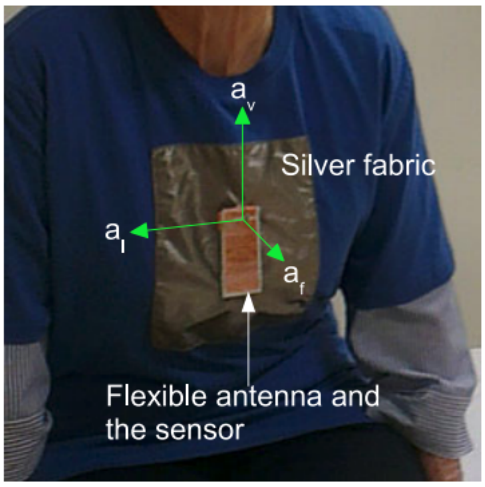
Activity recognition with healthy older people using a batteryless wearable sensor Data Set Abstract: Sequential motion data from 14 healthy older people aged 66 to 86 years old using a batteryless, wearable sensor on top of their clothing for the recognition of activities in clinical environments.
https://archive.ics.uci.edu/ml/datasets/Activity+recognition+with+healthy+older+people+using+a+batteryless+wearable+sensor

Background

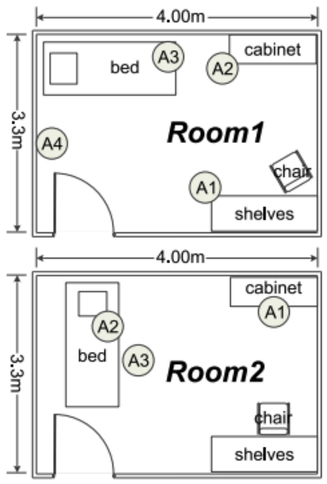
Getting out of bed and ambulating without supervision is identified as one of the major causes of patient falls in hospitals and nursing homes. An emerging generation of batteryless, lightweight, and wearable sensors are creating new possibilities for ambulatory monitoring. This data was investigated from use of a batteryless radio-frequency identification(RFID) tag response to analyze bad-egress movements.

Dataset Information

This dataset contains the motion data of 14 healthy order aged between 66 and 86 years old, performed broadly scripted activities using a batteryless, wearable sensor on top of their clothing at sternum level. Data is sparse and noisy due to the use of a passive sensor. Participants were allocated in two clinical room settings (S1 and S2). The setting of S1 (Room1) uses 4 RFID reader antennas around the room (one on ceiling level, and 3 on wall level) for the collection of data, whereas the room setting S2 (Room2) uses 3 RFID reader antennas (two at ceiling level and one at wall level) for the collection of motion data.



(a)



(b)

Dependent / Independent variables

Participants wore 4 sensors attached to the chest (3 accelerometers called w2isp sensors and 1 RFID tag), and an RFID reader was installed in the room to detect signals. The measured values of the three acceleration sensors and the signal strength values of the RFID reader are independent variables. Participants' actual behavior is a dependent variable.

Meaning of each column

The content of the file is as follows

- Column1: Time in seconds (real number).
- Column2: Acceleration reading in G for frontal axis (real number).
- Column3: Acceleration reading in G for vertical axis (real number).
- Column4: Acceleration reading in G for lateral axis (real number).
- Column5: ID of antena reading sensor. (interger)
- Column6: Received signal strength indicator (RSSI) (real number).
- Column7: Phase (real number).
- Column8: Frequancy (real number).
- Column9: Label of activity (integer) (1: sit on bed, 2: sit on chair, 3: lying, 4: ambulating)

In addition, gender of participant is included in the last charactor of file name eg: d1p33F (F: female)

Overall approach

Assignment Goal

The goal of this experiment is to distinguish whether or not the participant is lying in bed.

Pre-condition in this assignment

This data set was measured in two hospital rooms of the same size, with different arrangements of furniture and sensors. In this task, I would like to focus on selecting only the dataset collected from the first hospital room to achieve a specific goal.

Clssification Algorithm: SVM (support vector machine)

Classifying data is a common task in machine learning. Assuming that the given data points belong to two classes respectively, the goal is to determine which of the two classes the new data points belong to. Given a set of data belonging to either category, the SVM algorithm creates a non-stochastic binary linear classification model based on the given data set to determine which category the new data belongs to. When behavior changes occur, acceleration changes. The amount of change in the lying motion is greater. The data classifies the classes into four movements, but the class was divided into lying and not, as the goal was to capture the behavior of an elderly patient in bed. The reason I chose SVM is because when I plotted only the magnitude of the acceleration, I decided that I could divide the two classes by just changing the size.

Featuring engineering ideas

Hypothesis1, relation with acceleration.

As long as the participant does not perform constant acceleration movement, the acceleration increases when there is a change in movement. Therefore, the acceleration is small when lying in bed. From the given data, we can simply calculate the acceleration force using the equation below.

$$acc_n = |a_{nf} + a_{nv} + a_{nl}|$$

Drawing the acceleration force in chronological order is as follows.

Hypothesis2, weight at each axis.

A person's behavior is directional. For example, if you try to move forward, there will be a moment when af is high. In order to distinguish the lying behavior from the other behavior, weight was adjusted by adding alpha, beta, and gamma coefficients in the calculation of the magnitude of the acceleration force defined in Hypothesis 1. Each coefficient is the square of the median acceleration of each axis in all lying motions recorded in the dataset. If the lying motion is not measured, the default is 1. The coefficients were squared to ensure a difference from the values.

$$\alpha = median(|a_f|), (a \in D_{lying})$$

$$\beta = median(|a_v|), (a \in D_{lying})$$

$$\gamma = median(|a_l|), (a \in D_{lying})$$

$$acc_n = |\alpha^2 a_{nf} + \beta^2 a_{nv} + \gamma^2 a_{nl}|$$

Hypothesis3, acc per gap with before acc.

I can get clustered data, but not all datasets are satisfactory. Since the x-axis is time, the line that actually separates the two groups is drawn parallel to the x-axis. However, since it is possible to lie down on the bed several times, it is not easy to separate the two into one straight line. Therefore, if you look at the difference from the last data, you can see the instantaneous acceleration change.

$$newfeature_n = \\ (x_n, y_n) = \{x_n | x_n = |acc_n - acc_{n-1}|, y_n | y_n = acc_n\}$$

Program source codes

Github repositoty

GodMoonGoodman/Real-Time-Bed-egress-Recognition-in-Older-People

Sequence Learning with Passive RFID Sensors for Real Time Bed-egress Recognition in Older People - GodMoonGoodman/Real-Time-Bed-egress-Recognition-in-Older-People

https://github.com/GodMoonGoodman/Real-Time-Bed-egress-Recognition-in-Older-People

Representative source codes

- step1.py (hypothesis1)

```
from lib.plot import color_shape
from data.load import origin_data_load as load_data, DATA_PATH
from lib.const import *
from lib.hypothesis import acceleration

from os import listdir
import sys
import matplotlib.pyplot as plt

activity_labels = [LABEL_LYING, LABEL_SIT_ON_BED, LABEL_SIT_ON_CHAIR, LABEL_AMBULATING]

def draw_chart(file_name, save = False):
  # x - axis data
  # key is label
  # value is array of time(n)
  x = {
    '1': [],
    '2': [],
    '3': [],
    '4': [],
  }
  # y - axis data
  # key is label
  # value is array of acc(n)
  y = {
    '1': [],
    '2': [],
    '3': [],
    '4': []
  }

  rows = load_data(file_name)

  for row in rows:
    label = str(row['label'])
    time = row['time']
    af = row['a_f']
    av = row['a_v']
    al = row['a_l']
    acc = acceleration(af, av, al)

    x[label].append(time)
    y[label].append(acc)

  plt.xlabel('time')
  plt.ylabel('acc')
  plt.title('data: {}'.format(file_name))

  for label in [str(label) for label in activity_labels]:
    plt.plot(x[label], y[label], color_shape[label])

  if save:
    SAVE_PATH = 'figures/hypothesis1/'
    file_path = '{}{}.png'.format(SAVE_PATH, file_name)
    plt.savefig(file_path, dpi=300)
    plt.clf()

    print('{} was saved'.format(file_path))
  else:
    plt.show()


if __name__ == "__main__":
  cmd = sys.argv[1]
  if cmd == '-d':
    file_name = sys.argv[2]
    draw_chart(file_name, save = '-s' in sys.argv)
  elif cmd == '-a':
    file_names = [f for f in listdir(DATA_PATH)]
    for file_name in file_names:
      draw_chart(file_name, save = '-s' in sys.argv)
```

- step2.py (hypothesis2)

```
from lib.plot import color_shape
from data.load import origin_data_load as load_data, DATA_PATH
from lib.const import *
from lib.median import median
from lib.hypothesis import acceleration_coef

from os import listdir
import sys
import matplotlib.pyplot as plt

activity_labels = [LABEL_LYING, LABEL_SIT_ON_BED, LABEL_SIT_ON_CHAIR, LABEL_AMBULATING]

def draw_chart(file_name, save = False):
  # x - axis data
  # key is label
  # value is array of time(n)
  x = {
    '1': [],
    '2': [],
    '3': [],
    '4': [],
  }
  # y - axis data
  # key is label
  # value is array of acc(n)
  y = {
    '1': [],
    '2': [],
    '3': [],
    '4': []
  }

  # used for calculating alpha, beta, gamma coefficient
  acceleration_of_lying =  {
    'a_f': [],
    'a_v': [],
    'a_l': []
  }

  # alpha, beta, gamma coefficient
  median_acceleration_of_lying = {
    'a_f': 1,
    'a_v': 1,
    'a_l': 1
  }

  # import dataset
  rows = load_data(file_name)

  # collect acceleraton datas of only lying activity
  for row in list(filter(lambda row: row['label'] == LABEL_LYING, rows)):
    acceleration_of_lying['a_f'].append(row['a_f'])
    acceleration_of_lying['a_v'].append(row['a_v'])
    acceleration_of_lying['a_l'].append(row['a_l'])

  # calculate coefficient, median of each axis acceleration
  median_acceleration_of_lying['a_f'] = median(acceleration_of_lying['a_f'])
  median_acceleration_of_lying['a_v'] = median(acceleration_of_lying['a_v'])
  median_acceleration_of_lying['a_l'] = median(acceleration_of_lying['a_l'])

  for row in rows:
    label = str(row['label'])
    time = row['time']
    af = row['a_f']
    av = row['a_v']
    al = row['a_l']
    acc = acceleration_coef(
      af = af,
      av = av,
      al = al,
      alpha = median_acceleration_of_lying['a_f'],
      beta = median_acceleration_of_lying['a_v'],
      gamma = median_acceleration_of_lying['a_l']
    )

    # x-axis : time sequence
    x[label].append(time)
    # y-axis : improved acceleration
    y[label].append(acc)

  plt.xlabel('time')
  plt.ylabel('acc')
  plt.title('data: {}'.format(file_name))

  for label in [str(label) for label in activity_labels]:
    plt.plot(x[label], y[label], color_shape[label])

  if save:
    SAVE_PATH = 'figures/hypothesis2/'
    file_path = '{}{}.png'.format(SAVE_PATH, file_name)
    plt.savefig(file_path, dpi=300)
    plt.clf()

    print('{} was saved'.format(file_path))
  else:
    plt.show()


if __name__ == "__main__":
  cmd = sys.argv[1]
  if cmd == '-d':
    file_name = sys.argv[2]
    draw_chart(file_name, save = '-s' in sys.argv)
  elif cmd == '-a':
    file_names = [f for f in listdir(DATA_PATH)]
    for file_name in file_names:
      draw_chart(file_name, save = '-s' in sys.argv)
```

- step3.py (hypothesis3)

```
from lib.plot import color_shape
from data.load import origin_data_load as load_data, DATA_PATH
from lib.const import *
from lib.median import median
from lib.hypothesis import acceleration_coef

from os import listdir
import sys
import matplotlib.pyplot as plt

activity_labels = [LABEL_LYING, LABEL_SIT_ON_BED, LABEL_SIT_ON_CHAIR, LABEL_AMBULATING]

def draw_chart(file_name, save = False):
  # x - axis data
  # key is label
  # value is array of time(n)
  x = {
    '1': [],
    '2': [],
    '3': [],
    '4': [],
  }
  # y - axis data
  # key is label
  # value is array of acc(n)
  y = {
    '1': [],
    '2': [],
    '3': [],
    '4': []
  }

  acceleration_of_lying =  {
    'a_f': [],
    'a_v': [],
    'a_l': []
  }

  median_acceleration_of_lying = {
    'a_f': 1,
    'a_v': 1,
    'a_l': 1
  }

  # import dataset
  rows = load_data(file_name)

  # collect acceleraton datas of only lying activity
  for row in list(filter(lambda row: row['label'] == 3, rows)):
    acceleration_of_lying['a_f'].append(row['a_f'])
    acceleration_of_lying['a_v'].append(row['a_v'])
    acceleration_of_lying['a_l'].append(row['a_l'])

  # calculate coefficient, median of each axis acceleration
  median_acceleration_of_lying['a_f'] = median(acceleration_of_lying['a_f'])
  median_acceleration_of_lying['a_v'] = median(acceleration_of_lying['a_v'])
  median_acceleration_of_lying['a_l'] = median(acceleration_of_lying['a_l'])

  before_row = None

  for row in rows:
    label = str(row['label'])
    time = row['time']
    af = row['a_f']
    av = row['a_v']
    al = row['a_l']
    acc = acceleration_coef(
      af = af,
      av = av,
      al = al,
      alpha = median_acceleration_of_lying['a_f'],
      beta = median_acceleration_of_lying['a_v'],
      gamma = median_acceleration_of_lying['a_l']
    )

    # record for before_row['acc']
    row['acc'] = acc

    if before_row != None:
      time_gap = row['time'] - before_row['time']
      acc_gap = abs(row['acc'] - before_row['acc'])

      if time_gap > MIN_TIME_GAP:
        # x-axis : time sequence
        x[label].append(acc_gap)
        # y-axis : improved acceleration
        y[label].append(acc)

    before_row = row

  plt.xlabel('time')
  plt.ylabel('acc')
  plt.title('data: {}'.format(file_name))

  for label in [str(label) for label in activity_labels]:
    plt.plot(x[label], y[label], color_shape[label])

  if save:
    SAVE_PATH = 'figures/hypothesis3/'
    file_path = '{}{}.png'.format(SAVE_PATH, file_name)
    plt.savefig(file_path, dpi=300)
    plt.clf()

    print('{} was saved'.format(file_path))
  else:
    plt.show()


if __name__ == "__main__":
  cmd = sys.argv[1]
  if cmd == '-d':
    file_name = sys.argv[2]
    draw_chart(file_name, save = '-s' in sys.argv)
  elif cmd == '-a':
    file_names = [f for f in listdir(DATA_PATH)]
    for file_name in file_names:
      draw_chart(file_name, save = '-s' in sys.argv)
```

- SVM.py (Verification of hypothesis3)

```
from lib.plot import color_shape
from data.load import origin_data_load as load_data, DATA_PATH
from lib.const import *
from lib.median import median
from lib.hypothesis import acceleration_coef

from os import listdir
import sys
import matplotlib.pyplot as plt

from sklearn.svm import SVC
import numpy as np

activity_labels = [LABEL_LYING, LABEL_SIT_ON_BED, LABEL_SIT_ON_CHAIR, LABEL_AMBULATING]

def draw_chart(file_name, save = False):
  # x - axis data
  x = []
  # y - axis data
  y = []

  acceleration_of_lying =  {
    'a_f': [],
    'a_v': [],
    'a_l': []
  }

  median_acceleration_of_lying = {
    'a_f': 1,
    'a_v': 1,
    'a_l': 1
  }

  rows = load_data(file_name)

  # only lying
  for row in list(filter(lambda row: row['label'] == LABEL_LYING, rows)):
    acceleration_of_lying['a_f'].append(row['a_f'])
    acceleration_of_lying['a_v'].append(row['a_v'])
    acceleration_of_lying['a_l'].append(row['a_l'])

  median_acceleration_of_lying['a_f'] = median(acceleration_of_lying['a_f'])
  median_acceleration_of_lying['a_v'] = median(acceleration_of_lying['a_v'])
  median_acceleration_of_lying['a_l'] = median(acceleration_of_lying['a_l'])

  before_row = None

  train_points = []
  labels = []

  classifier = SVC(kernel = 'linear')

  for row in rows:
    label = str(row['label'])
    time = row['time']
    af = row['a_f']
    av = row['a_v']
    al = row['a_l']
    acc = acceleration_coef(
      af = af,
      av = av,
      al = al,
      alpha = median_acceleration_of_lying['a_f'],
      beta = median_acceleration_of_lying['a_v'],
      gamma = median_acceleration_of_lying['a_l']
    )

    row['acc'] = acc

    if before_row != None:
      time_gap = row['time'] - before_row['time']
      acc_gap = abs(row['acc'] - before_row['acc'])

      if time_gap > MIN_TIME_GAP:
        x.append(acc_gap)
        y.append(acc)

        train_points.append([acc_gap, acc])
        labels.append('r' if row['label'] == LABEL_LYING else 'y')

    before_row = row
  try:
    classifier.fit(train_points, labels)
  except ValueError:
    return

  predicts = classifier.predict(train_points)
  success_count = 0
  for i, v in enumerate(predicts):
    if v == labels[i]:
      success_count += 1

  accuracy = round(success_count / len(predicts), 2)

  plt.scatter(x, y, c=labels, s=30, cmap=plt.cm.Paired, marker="x")
  ax = plt.gca()
  xlim = ax.get_xlim()
  ylim = ax.get_ylim()
  xx = np.linspace(xlim[0], xlim[1], 30)
  yy = np.linspace(ylim[0], ylim[1], 30)
  YY, XX = np.meshgrid(yy, xx)
  xy = np.vstack([XX.ravel(), YY.ravel()]).T
  Z = classifier.decision_function(xy).reshape(XX.shape)
  ax.contour(XX, YY, Z, colors='k', levels=[-1,0,1], alpha=0.5, linestyles=['--', '-', '--'])

  plt.xlabel('time')
  plt.ylabel('acc')
  plt.title('data: {}, accuracy: {}'.format(file_name, str(accuracy * 100) + '%'))

  if save:
    SAVE_PATH = 'figures/result/'
    file_path = '{}{}.png'.format(SAVE_PATH, file_name)
    plt.savefig(file_path, dpi=300)
    plt.clf()

    print('{} was saved'.format(file_path))
  else:
    plt.show()


if __name__ == "__main__":
  cmd = sys.argv[1]
  if cmd == '-d':
    file_name = sys.argv[2]
    draw_chart(file_name, save = '-s' in sys.argv)
  elif cmd == '-a':
    file_names = [f for f in listdir(DATA_PATH)]
    for file_name in file_names:
      draw_chart(file_name, save = '-s' in sys.argv)
```

Execution results

💡 The execution data was randomly selected as one of the data with the most significant change in each step. Running with other data gave similar results.

💡 **The result chart for all datasets has been uploaded to Github**. README.md explains what command can get the chart.

- sample dataset for drawing chart: d1p55F
- color set
  - red: lying
  - green : sit on the bed
  - blue : sit on chair
  - magenta : ambulating
- mean of accuracy (all dataset) : **97.25%**

Figure from Hypothesis1: relation with acceleration.

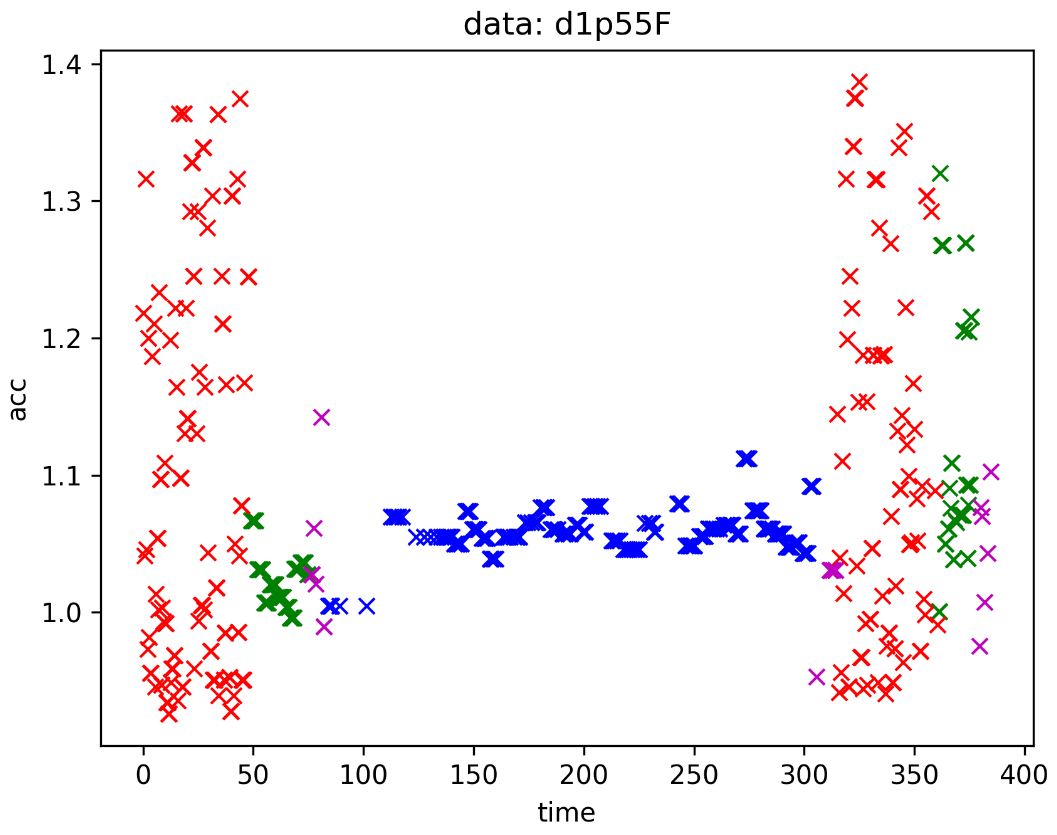


Figure from Hypothesis2: weight at each axis.

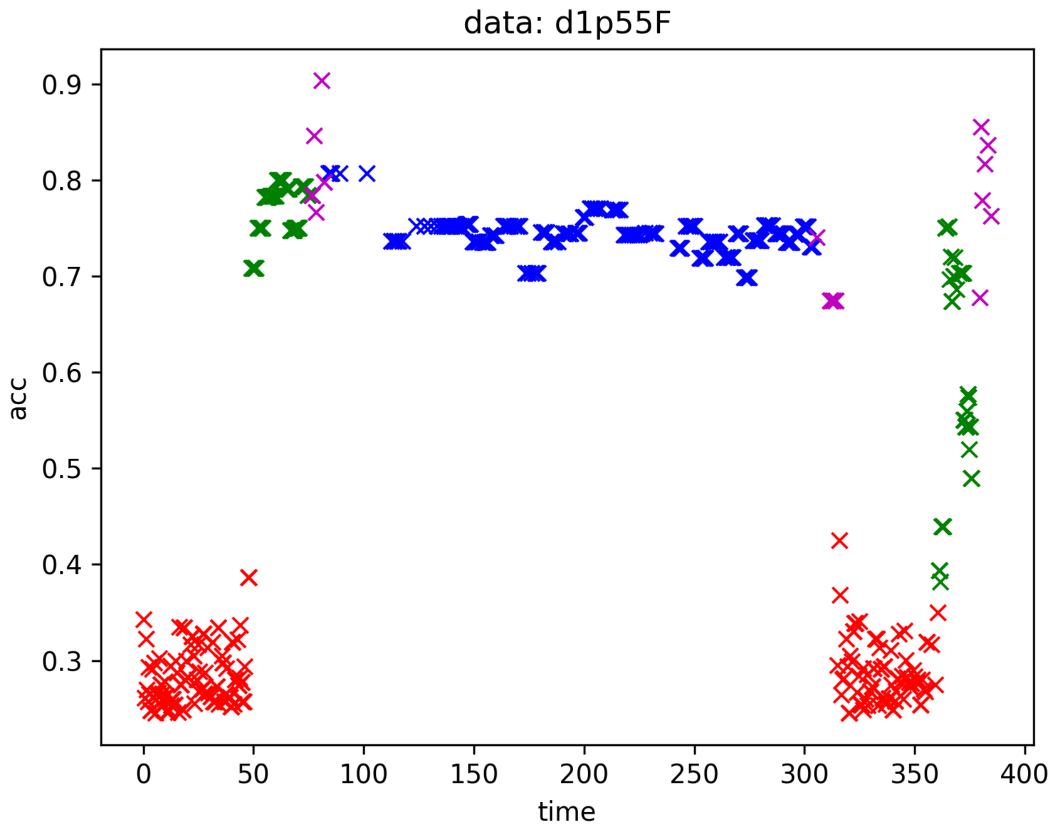


Figure from Hypothesis3: acc per gap with before acc.

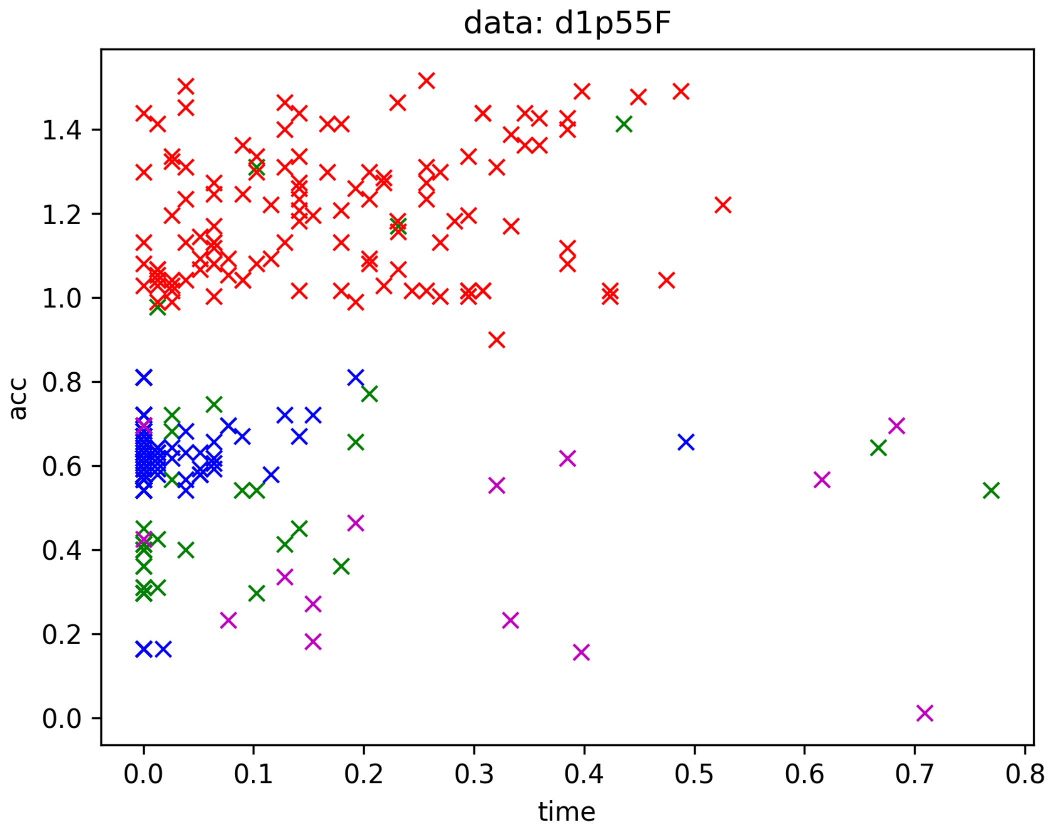
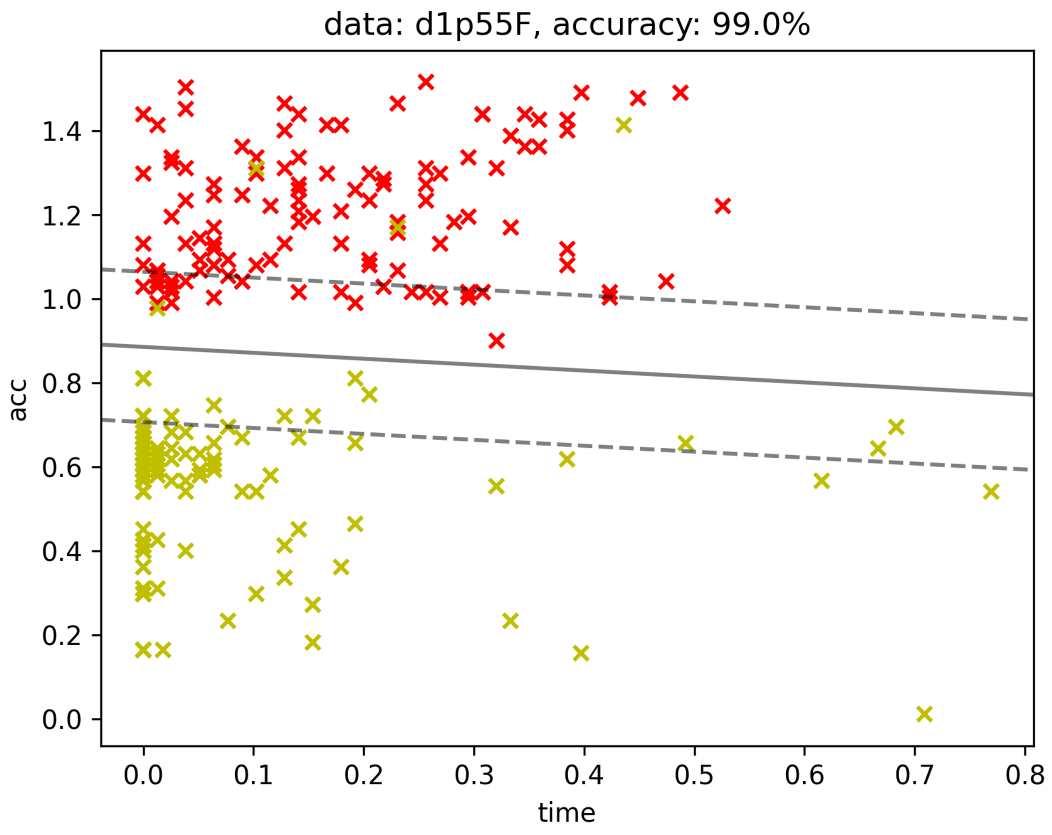


Figure from SVM (Red: lying, Yellow: other activities)



References

- [Relavant Paper from Dataset] Wickramasinghe, A., Ranasinghe, D. C., Fumeaux, C., Hill, K. D., Visvanathan, R. (2016), 'Sequence Learning with Passive RFID Sensors for Real Time Bed-egress Recognition in Older People,' in IEEE Journal of Biomedical and Health Informatics , vol.PP, no.99, pp.1-1
- UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Activity+recognition+with+healthy+older+people+using+a+batteryless+wearable+
- Wikipedia (https://ko.wikipedia.org/wiki/서포트_벡터_머신)