Classifying Eye Movements with Machine Learning Techniques

Brian Fakhoury

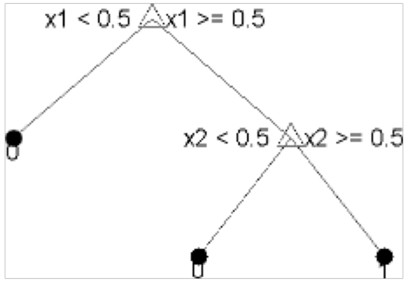
Data

- Raw readout from eye-tracker, asci format:
  - e.g.
    - SFIX R 901858
    - 901858   785.6 377.7 628.0 789.4 398.8 1365.0 .
    - 901860   785.5 377.8 628.0 789.3 398.6 1365.0 .
- We clean the data to produce the training set:
  - [901858, 785.6, 377.7, 0]
  - Time, x-axis, y-axis, type
- We also produce a similar untagged set for testing purposes
  - [901858, 785.6, 377.7]
  - Time, x-axis, y-axis

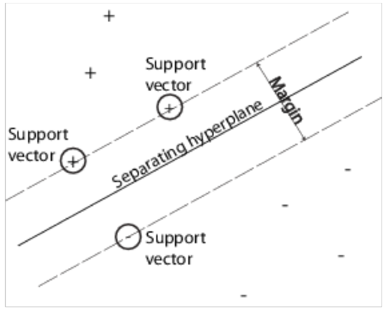
```
MSG 4150176 !CAL CALIBRATION HV9 LR LEFT    GOOD
MSG 4150177 !CAL CALIBRATION HV9 LR RIGHT   GOOD
INPUT   4153296 120
MSG 4165879 !CAL VALIDATION HV9 LR LEFT  GOOD ERROR 0.71 avg. 1.34 max
MSG 4165879 !CAL VALIDATION HV9 LR RIGHT GOOD ERROR 0.65 avg. 1.36 max
INPUT   4175458 120
INPUT   4176004 120
MSG 4178072 DRIFTCORRECT LR REPEATING due to large error
MSG 4179532 DRIFTCORRECT LR LEFT  at 640,360  OFFSET 2.19 deg.  -17.5,
MSG 4179532 DRIFTCORRECT LR RIGHT at 640,360  OFFSET 0.44 deg.  -7.3,-
MSG 4180093 RECCFG P 500 2 1
MSG 4180093 GAZE_COORDS 0.00 0.00 1279.00 719.00
MSG 4180094 !MODE RECORD P 500 2 1
START   4180108     LEFT    RIGHT   SAMPLES EVENTS
PRESCALER   1
VPRESCALER  1
PUPIL   AREA
EVENTS  GAZE    LEFT    RIGHT   RATE     500.00 TRACKING    P   FILTER
SAMPLES GAZE    LEFT    RIGHT   RATE     500.00 TRACKING    P   FILTER
INPUT   4180108 120
4180108   639.1   368.3  2001.0   639.9   359.1  1626.0 .
4180110   638.9   368.7  2002.0   639.8   359.8  1626.0 .
4180112   639.2   368.9  2004.0   639.7   359.9  1626.0 .
4180114   639.2   368.5  2004.0   639.7   360.1  1626.0 .
SFIX L   4180116
SFIX R   4180116
4180116   639.1   368.8  2003.0   639.7   360.3  1623.0 .
4180118   639.0   368.9  2006.0   639.6   360.6  1621.0 .
4180120   638.8   369.0  2010.0   639.8   361.8  1621.0 .
4180122   638.9   371.4  2012.0   640.1   362.3  1622.0 .
4180124   638.9   371.2  2012.0   640.6   362.9  1622.0 .
4180126   639.7   371.2  2011.0   640.5   363.0  1623.0 .
4180128   640.0   371.4  2011.0   641.0   363.4  1623.0 .
MSG 4180130 SYNCTIME
4180130   640.2   372.0  2001.0   640.9   363.5  1623.0 .
4180132   640.2   372.5  2001.0   641.0   363.7  1621.0 .
4180134   640.0   372.6  2001.0   640.7   364.4  1620.0 .
4180136   640.1   372.4  2004.0   641.1   365.2  1621.0 .
4180138   640.1   372.7  2004.0   641.0   365.6  1620.0 .
4180140   640.1   372.7  2004.0   641.1   365.5  1621.0 .
4180142   640.0   373.8  2001.0   640.8   366.4  1620.0 .
4180144   640.0   375.1  2001.0   640.8   367.4  1620.0 .
4180146   640.5   375.7  1996.0   640.8   368.0  1619.0 .
4180148   639.9   374.9  1994.0   640.8   367.8  1618.0 .
4180150   639.3   375.3  1993.0   640.8   368.5  1618.0 .
4180152   639.6   375.1  1995.0   641.0   368.6  1616.0 .
4180154   639.9   375.1  1994.0   641.0   368.6  1619.0 .
4180156   640.1   375.1  1995.0   640.9   368.6  1621.0 .
4180158   639.6   375.3  1995.0   641.3   369.1  1624.0 .
4180160   639.6   375.4  1995.0   641.2   369.2  1627.0 .
4180162   639.6   375.5  1994.0   641.0   369.2  1630.0 .
4180164   640.3   375.0  1993.0   640.9   369.2  1629.0 .
4180166   640.0   374.7  1995.0   640.9   369.2  1629.0 .
4180168   639.9   374.4  1998.0   641.0   369.0  1630.0 .
4180170   639.8   374.0  1997.0   641.2   368.2  1631.0 .
4180172   639.3   373.8  1997.0   641.2   367.3  1626.0 .
4180174   639.4   373.5  1997.0   641.1   367.1  1628.0 .
4180176   639.4   372.8  1997.0   641.4   366.9  1625.0 .
4180178   639.5   373.0  1996.0   641.7   365.5  1622.0 .
4180180   639.7   372.3  1995.0   641.6   364.8  1622.0 .
4180182   639.9   371.7  1994.0   641.6   364.4  1622.0 .
4180184   640.0   371.2  1997.0   641.4   364.4  1615.0 .
4180186   639.9   370.9  1994.0   641.1   364.3  1617.0 .
4180188   639.9   370.3  1991.0   640.9   363.4  1616.0 .
4180190   639.4   369.9  1983.0   640.7   363.0  1616.0 .
```

Classification algorithms and hypotheses

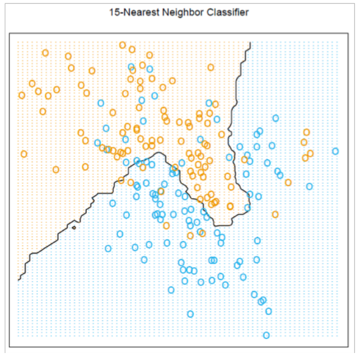
- Decision tree
  - Reason by binary steps. Each step is some threshold determined by some function of our features. MatLab seems to use some simple cost function, and thus decides the steps based off only numerical data
  - This might not be useful where our x and y data are only relative to our time data, and no explicit connection is specified between this data.



- Support Vector Machine
  - Attempts to create some hyperplane that distinguishes between 2 explicit categories.
  - Our data has 3 categories (fixation, normal, and saccade) so we use an algorithm that uses multiple SVM's to scale to 3 categories, so it might not perform well if there is no distinct separation between any 2 classes.



- Nearest Neighbor classifier
  - Attempts to model the data on the basis of nearby data. Closer data points will be weighted stronger should they be of the same category.
  - I expect this to be the best choice as this model can more easily abstract away from specific location of eye movement



MATLAB setup

1. produce_training_set.m

   Performance script for sanitizing data

   1. Feed in raw .asc file, output .csv with data cleaned
      1. Depending on the first parameter of the call {type Boolean}, data will have extra column with feature tag

2. cross_test_train.m

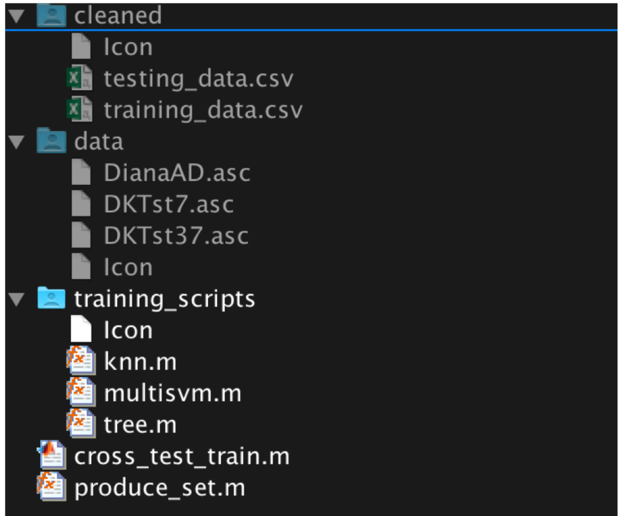
   Evaluation script for testing algorithms

   1. Searches for appropriately named .csv data files (two of them), one with tagged data, one without
   2. Choose learning algorithms present in ./training_scripts
   3. Evaluates algorithm and returns % accuracy
      1. Trains on trained set, tests on untagged set

3. ./training_scripts

   Implements callable training scripts

   1. Training data is passes in and a trained model is returned to cross_test_train for accuracy testing



```
%% Produce training set

% NOTE: using only left eye data for simplicity
% future change might be to double the dimmension space
% with right eye data on the same row

functionproduce_set(tagged_flag)
  disp("Creating training set...");
  file = input('name of session list-->','s');
  fid = fopen(['./data/'file'.asc']);

  whiletrue
    line = fgetl(fid);
    ifcontains(line,'SYNCTIME')
      break;
    end
  end

  left_data = [];

  iftagged_flag

    whiletrue
      data = strsplit(char(fgetl(fid)));
      current_type = 0;

      if(strcmp(data{1},'SFIX')), current_type = 1;
      elseif(strcmp(data{1},'EFIX')),continue;
      elseif(strcmp(data{1},'SSACC')), current_type = 2;
      elseif(strcmp(data{1},'ESACC')),continue;
      elseif(strcmp(data{1},'SBLINK')), current_type = -1;
      elseif(strcmp(data{1},'EBLINK')),continue;
      end

      if(length(data) == 8)
        left_data = [left_data; str2double(cell2mat(data(1,1))), str2double(cell2mat(data(1,2))), str2double(cell2mat(data(1,3))), current_type];
      end

      if~feof(fid) == 0,break;end
    end

  else
    whiletrue
      data = strsplit(char(fgetl(fid)));

      if(length(data) == 8)
        left_data = [left_data; str2double(cell2mat(data(1,1))), str2double(cell2mat(data(1,2))), str2double(cell2mat(data(1,3)))];
      end

      if~feof(fid) == 0,break;end
    end
  end

  file_name = file;
  iftagged_flag, filename = file_name +"_training_data";
  else, filename = file_name +"_testing_data";end
  csvwrite("./data/"+ filename +".csv", left_data);
end
```

```
%% Train and test script
clearvars
clc
% use training scripts from
% training scripts folder
addpath('training_scripts/');
disp("Running analysis script///");
%% test knn model

% produce a cleann set of data for training
% and a separate one for testing
% uses different trial data
produce_set(true);
produce_set(false);
% read the train and test sets
dataset_train = csvread('./cleaned/training_data.csv');
dataset_test = csvread('./cleaned/testing_data.csv');

disp("Data registered in ./cleaned");
disp("Cleaned data read.");

% Pass function handles
model_handles = {@knn @multisvm @tree};
for i=1:length(model_handles)
  test_model(model_handles{i}, dataset_train, dataset_test);
end

function test_model(name, dataset_train, dataset_test)

  % use the given script (./training_scripts) to train a model on the train set
  trainfn_model = name(dataset_train);

  % produce an unlabeled matrix for the returned prediction function
  test_set = dataset_test(:,1:3);

  % run the prediction function
  prediction_set = trainfn_model.predict(test_set);

  % get the actual answers from known data
  actual_set = dataset_test(:,4);

  % calulate accuracy with logic comparison normalized
  score = 100 * sum(prediction_set == actual_set) / length(prediction_set);

  fprintf("The %s model scored %f%%\n", func2str(name), score);
end
```

Statistical analysis of experiment

- Each model should be trained on one dataset and tested against another dataset to truly evaluate if the model was able to extract pertinent features at abstraction of one level at least.
- Once we have a trained model and test it against our test data, we evaluate each decision against known answers for the testing set. We then calculate the accuracy as a percentage of the total number of data points that were classified correctly.

Results

Testing set

Training set

	DianaAD	DKTst7	DKTst37
DianaAD	n/a	The knn model scored 78.582760% The multisvm model scored 78.582760% The tree model scored 52.564781%	The knn model scored 74.453723% The multisvm model scored 74.418808% The tree model scored 30.771917%
DKTst7	The knn model scored 0.207425% The multisvm model scored 84.485693% The tree model scored 40.655249%	n/a	The knn model scored 78.582760% The multisvm model scored 78.582760% The tree model scored 65.732417%
DKTst37	The knn model scored 0.207425% The multisvm model scored 0.015956% The tree model scored 34.435166%	The knn model scored 0.052373% The multisvm model scored 0.017458% The tree model scored 27.207658%	n/a

Immediate conclusions

- Training on large datasets (DianaAD, DKTst7) yields better invariant performance for knn, multisvm
  - Tree model is not noticeably affected by training set size
  - Therefore, model choice should be accounted for on basis of available data
- Therefore, if we want to create future accurate models for applications, we want to ensure large datasets first and foremost.
- Multisvm has the most consistent performance. Not statistically provable as we have small sample size, but should be want to tune a multisvm, we can expect good performance in this use case.

Onward

- Successful feature selection of multisvm shows us that more powerful abstraction machines like deep neural networks can, with high certainty, discover more intriguing data hidden in eye-movement.
- From a feature classification of saccades or fixations, the next step might be to classify environmental factors such as brightness, alertness, calmness, image complexity, etc…