Discrete Gaze-Estimation using Machine Learning*

Ahmad Salim Al-Sibahi (asal@itu.dk)

IT University of Copenhagen

Nicolai Skovvart (nbsk@itu.dk)

IT University of Copenhagen

December 12, 2012

*Supervisor: Dan Witzner Hansen

Abstract

Abstract goes here

Contents

1 Introduction

1.1 What is Machine Learning?

In today's connected world, data is becoming more and more interesting. Whether it is social information for advertising, film-rankings for leisure or medical histories for health, collecting and classifying data has become more of a norm than an exception. While most useful data contains patterns it is sometimes hard or infeasible to use statistical and/or formal methods to find a useful classification. In those cases, one could resort to the use of Machine Learning.

Machine Learning is the idea of pattern recognition on sample datasets, such as to enable classification of specific data classes. More precisely it is about finding representative data fitting a particular probability distribution to learn or 'train' on, such that any given data point coming from that probability distribution can be processed with a relatively high accuracy. In many ways Machine Learning, therefore, relates to the concept of *generalization*, which specifies the measure of accuracy between in-sample errors which are calculated during training, and out-of-sample errors which represents the general data. A Machine Learning model can be said to perform well, when the accuracy achieved during training can be generalized.

1.2 Investigating Machine Learning

To prepare for the experimental work done in this project, we have taken an on-line course on Machine Learning called "Learning from Data" by Dr. Yaser Abu-Mostafa[1]. The course is what we have used the majority of the time on with relation to the project, and consists of 18 lectures covering introductory techniques and theory. These lectures are composed of general concepts such as the learning problem and overfitting, techniques and models such as linear models and support vector machines, and mathematical theory regarding testing and generalization. We will summarise the most important concepts in Section 3.1 - General Overview On Machine Learning.

1.3 Applying Machine Learning to Real-World Data

In order to gain practical experience of the application of Machine Learning to real-world problems, we will perform a practical experiment which will the form the basis of the dataset used. The core of the experiment relates to that if given a fixed arrangement of lights it is possible to detect which light the person is looking at, solely by using a still image of the eye.

1.3.1 Processing Data

To get a workable set of data, we will use an appearance-based extraction method called Principal Component Analysis (see Section 3.2 for details). The analysis will allow us to get a low-dimensional reduced space, that captures most of the actual variation and in that way be able to work with huge numbers of data without much loss of information. Finally we will attempt to run multiple types of machine learning algorithms, to examine if it is possible to find a suitable model that allows generalized gaze estimation.

In short, this report will explain the concepts of machine learning utilized in this project, the experiment in more detail, how the results can be evaluated, possible threats to validity and the final conclusions.

2 Problem Description

In this project we will determine under what conditions it is possible to perform gaze estimation using machine learning.

Gaze estimation is the process of identifying where an eye is looking. It has a lot of applications, for example in giving disabled people the ability to type with, and control devices using, their eyes. Especially people with Amyotrophic Lateral Sclerosis (ALS) can make use of this technology. It is also closely related to eye tracking where image analysis is used to identify eyes and eye features such as eye corners, center of the iris, glints in the eye and so on. We will not be using eye tracking as it is outside the scope of this project.

Using machine learning for gaze estimation can be problematic, as images of faces (or eyes) are high-dimensional data, making learning complex and time-consuming. There are a few approaches to reducing this complexity. One could reduce the input space by converting the image to grayscale, isolating the eye-pixels from the rest of the image, and scaling the resulting image to a size where learning is feasible. One could also reduce the input space by extracting features of the eye such as the positions of the corners of the eye and the centre of the pupil. In this project we attempt both approaches.

When gathering test data there are also many factors to consider. Test subjects should be placed similarly in such a way that their eyes are clearly visible to the camera and at a distance where glints in the eyes from infra-red lights are visible. The angle of the head and eyes can also impact learning immensely, as they make the eyes look very different compared to a front-facing eye. To combat this, one could attempt to rotate the eye, but this may cause other problems. Lighting is also very important and should be consistent during testing.

When processing the initial test data, it can be beneficial to histogram equalize the images to increase the contrast. This makes potential lighting issues have less of an impact.

3 Theory

3.1 General Overview On Machine Learning

This section serves as a summary of general machine learning concepts which are recommended to better understand the rest of the report. The theory originates mainly from the machine learning course[1] and the accompanying text book[2].

3.1.1 Types of Machine Learning

As mentioned in the Introduction (see Section 1), machine learning is a way of picking a fitting set of data to train on and calculate a model, such that similar data can be evaluated using the same model with high accuracy. Generally speaking there are three popular types of machine learning: supervised learning, reinforcement learning and unsupervised learning. The type of learning is often for a project is often chosen on different sets criteria such as the type of data available, and the knowledge about how the data is grouped. More often than not the choice will lie on one single solution, something which will become clear in the following paragraphs.

Supervised Learning To use supervised learning, the sample dataset used for learning must not only contain some valid input, but also the corresponding "correct" output. The challenge therefore primarily lies on finding the right way of grouping or evaluating the input such that the corresponding desired output can be achieved, by calculation on the model. In many ways, this often makes Supervised Learning the most desired method, as there it is often easier to find a fitting solution which corresponds to what is observed on real-world data, than when the desired output is completely unknown. It is also why, supervised learning has the most different techniques such as linear classification and support vector machines (see Section 3.1.2).

Reinforcement Learning Reinforcement learning is in many ways similar to supervised learning in that it requires both input data and corresponding output to be used. The main difference is that while supervised learning requires the desired output to be correct, reinforcement only requires a grading of such output. This technique is therefore often used, when there is no holistic overview regarding the distribution of data but where machine learning is till required.

Example 1 *Imagine a hospital where data is gathered on persons with an epidemical illness with multiple stages, and the medical personnel would want to determine what factors could be causing such illness. As symptoms often can be misinterpreted on early stages, it is unknown if the outcome is entirely correct but a grading of how ill a patient is can be used to aid the algorithm.*

In many ways reinforcement learning, also has to handle the grading additionally to finding a fitting model. This usually makes the models more complex, and as such harder to generalize properly[3][4].

Unsupervised Learning The last popular type of learning is unsupervised learning, where only data input is given. In many ways unsupervised learning tries to find a natural grouping or clustering of data, such that input data that

lies next to each other is grouped similarly. This can be useful to e.g. separate colours from images or to find patterns in given input for further post-processing. While it maybe hard to find real-world data that groups in obvious ways, and even harder making it fit some desired output; Unsupervised learning has one advantage, that generalization is free. This is because unsupervised relies solely on input-processing and does not change any output metrics and as such the models that can be used to not have to conform to an output specification. Examples of unsupervised learning methods are k-means clustering[2], nearest neighbours[5] and PCA[6](see Section 3.2 - Principal Component Analysis for further explanation).

Selected Types of Learning To reduce the input space in our project, we resort to the use of PCA, which is as previously mentioned an unsupervised learning technique and thus have no generalization cost. This allows us both to visualize how the variation/data is grouped, and allows us to use supervised learning techniques to do the actual separation of data with much less noise and more performance than otherwise. The choice of supervised learning is due to the fact that it offers the best trade-off between usefulness of output and generalization cost.

3.1.2 Models for Classification

To be able to classify the resulting data from our experiments, we have chosen some simple supervised machine learning models to apply. In the following sections we will thus present an overview of these models.

Linear Classification The simplest type of classification is called linear classification. The idea is simple: given an input dataset $\mathbf{X}$ consisting of data-vectors $\mathbf{x}$ with input parameters $x_1, \ldots, x_n \in \mathbf{x}$ and the corresponding output parameter $y \in \{-1; +1\}$ (see Figure 1), find a linear formula which separates the outputs given the input parameters. The classes of output are usually denoted as either being negative (the output being -1) or positive (the output being $+1$). This makes it easier to run algorithms and is usually not a problem, because it is often possible to do a mapping from any given output format.

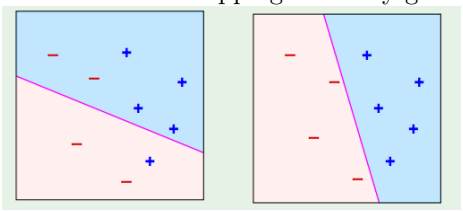

Figure 1: Linear classification algorithm running on two dimensional linear separable data

To separate the two classes of output, a given linear learning algorithm must assign weight coefficients to each input parameter. To get a positive or negative classification out of the real-numbered output, the sign of the output is taken yielding two output classes. The equation to be calculated therefore looks as follows:

$$h(x) = sign(w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n)$$

In order to allow separation of threshold-values that are non-zero an additional learning parameter w_0 is added.

Logistic Regression

Support Vector Machines

Multiclass Classification

3.1.3 Generalization

3.1.4 Error and Noise

3.1.5 Validation and Testing

3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique that can be used to identify patterns in high dimensional data, for example images. It aims to identify principal components and their effect on the overall learning data, and the most important principal components can be separated from the less important ones. It is a linear model in the sense that it expects the data to be linearly separable, which can be a limitation in some cases. An advantage it has, is the fact that it is a very visual model. PCA can generate images based on a means-image from the learning data and a principal component. This allows one to generate images with various weights on a principal component and identify what it actually does. Examples in this section are inspired by the PCA tutorial by Smith [6].

3.2.1 How It Works

Input data formatting. First of all, the data should be prepared for PCA. In the case of images, the image should be flattened to a single vector. This means an image of size M by N with Z dimensions (colour values) should turn into a vector of length $M \times N \times Z$. Turning the images into grayscale is often a good idea as it reduces the complexity by a third, and the added colour information may not be relevant. A mean vector of all the input vectors should then be calculated and subtracted from the input vectors, producing a new input set with a mean of 0. For 2 grayscale images of size 2×2 pixels, the process could look like table 1. All of the adjusted input vectors should then be placed in a matrix where every row is an adjusted image vector.

Calculate the covariance matrix. The covariance between two dimensions can be defined as follows, assuming the mean has already been subtracted from X and Y.

$$cov(X, Y) = \frac{\sum_{i=1}^{n} X_i Y_i}{n - 1}$$

The covariance should be calculated for all dimensions. For example, for a 3 dimensional data set with dimensions (x, y, z) the covariance can be calculated for

Vector	(X_1, Y_1)	(X_2, Y_1)	(X_1, Y_2)	(X_2, Y_2)
$Image_1$	150	220	123	136
$Image_2$	20	110	240	11
Mean	85	165	181.5	73.5
$AdjustedImg_1$	65	55	-58.5	62.5
$AdjustedImg_2$	-65	-55	58.5	-62.5
AdjustedMean	0	0	0	0

Table 1: Input data formatting example

(x, y), (x, z) and (y, z). The covariance matrix for a data set with n dimensions is defined as follows.

$$C^{n\times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j))$$

The covariance matrix for the previous imaginary data set with dimensions (x, y, z) is the following.

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

The matrix is a square $n \times n$ matrix, and is symmetrical around the main diagonal, as $cov(a, b) = cov(b, a)$. It is also worth noting that down the main diagonal the value is the covariance between a dimension and itself.

Calculate eigenvectors and eigenvalues of the covariance matrix. Eigenvectors are vectors that when multiplied with another matrix work like a constant. For example:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Some properties of eigenvectors: eigenvectors of a matrix can only be found for square matrices, and not every square matrix has eigenvectors. Given an $n \times n$ matrix that does have eigenvectors, there are n of them. The eigenvectors should be unit vectors, that being their length is exactly one. There is no easy way to calculate eigenvectors, but most programming languages have libraries with support for calculating them.

Eigenvalues are closely related to eigenvector and there was one in the previous example, namely 4. Eigenvalues comes in pairs with eigenvectors.

Choosing components. The eigenvector with the highest eigenvalue is the principle component of the data set. Sorting the eigenvectors by eigenvalue allows us to see what components are most important, and allows us to ignore insignificant components with low eigenvalues if necessary. After eliminating insignificant principal components, we can then form the feature vector which is a matrix of the eigenvectors we want to keep.

Deriving the new data set. The final step of Principal Component Analysis is to apply our feature vector to the adjusted input data where the mean has been subtracted. The input data is transposed to get the data items in the columns and the dimensions in the rows.

$$FinalData = FeatureVector \times AdjustedInputData^{\top}$$

This gives us the original data expressed in terms of the patterns identified.

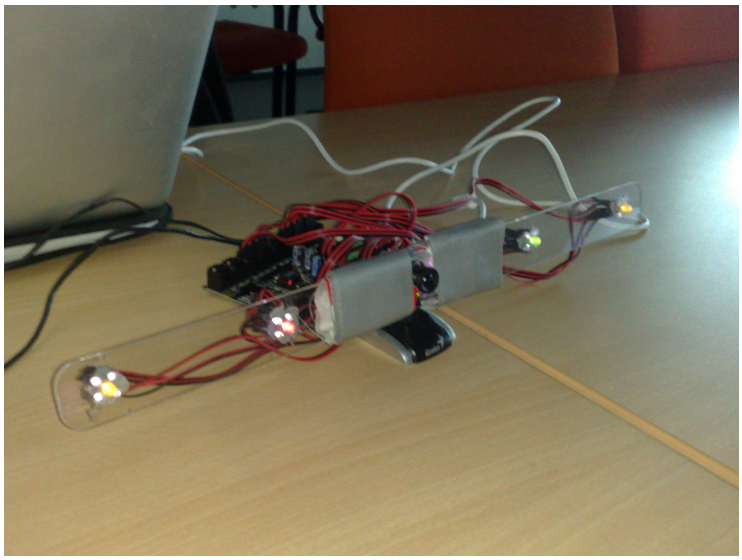

Figure 2: Web-cam with mounted LED and infra-red lights.

4 Technical Solution

4.1 Set-up

The web-cam we have used to gather our data with is a Genius iSlim 321R that can take both regular and infra-red images. Mounted to the web-cam is a plastic frame with 4 LED lights and 4 infra-red lights that are placed just besides the LED lights. The LED and infra-red lights are controlled by a Phidget single board computer. The LED lights have colours and are placed in the following pattern: Yellow-Red-Camera-Green-Yellow. The web-cam set up can be seen on figure 2.

Our project is written in python 2.7 using the following libraries:

- Scitkit-learn for machine learning algorithms.
- OpenCV for image analysis and manipulation.
- Numpy for matrix manipulation, also required by some of the other libraries.
- Matplotlib for plotting results.
- Phidgets python API to control the LED- and infra-red lights.

To control the mounted lights, we modified a phidgets script by Adam Stelmack. See LED-simple.py for details.

4.2 Experiment

To perform the actual experiment, first of all data must be gathered. The initial scenarios we wanted to test were combinations of the head being fixated and looking at a point, the head moving while looking at a point, and the

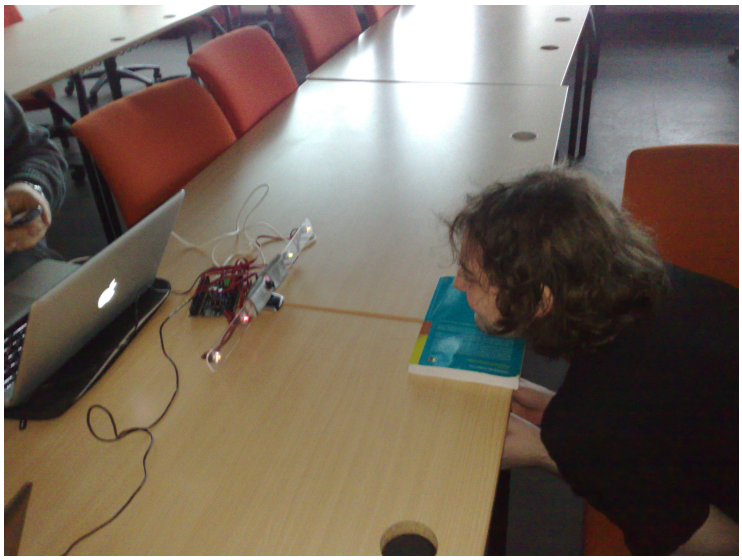

Figure 3: Data gathering on test-subject.

infra-red lights being turned on or off. This leaves us with the 4 scenarios with-light-head-move (WLHM), no-lights-head-move (NLHM), with-lights-head-still (WLHS) and no-lights-head-still (NLHS). Test-subjects were then asked to place their head at a set point and look at the different points. For images of our test-subjects, see figures 3, 4, 5, 6 and 7. We then recorded video sequences of about 5 seconds in length at 30 frames per second of the various scenarios and saved the videos with appropriate names according to the scenarios.

Images were then extracted from the videos every 10 frames using OpenCV and named in such a way it was easy to identify the scenario and the frame it was from. See Image-grabber.py for details. Bad images, such as frames were the eyes were obscured or closed, were then manually filtered out and added to an ignore-file.

The corners of the eye and the centre of the pupil were then marked manually using OpenCV and the three points were saved to meta-data files. See EyeMarker.py for details. This data can be used for feature based learning, though more information such as glints may be desirable.

For high-dimensional learning, we also needed to separate the eye images. To do this, eye images were extracted from the larger images using the previously marked points. To extract the image, the images were loaded and converted to grayscale. They were then histogram-equalized to broaden the contrast in the image. The eye was then sliced out of the larger image and rescaled to a desired size, such as 20×20, 40×40 or 60×60, and then saved. For more information see TransformEyeImages.py.

PCA was then run on the eye images and the results were plotted. See run_pca.py and pca.py for more information.

After plotting the results, we identified which principal components made the different lights separable and in which cases. We then tried generating images with different weights on the principal component that made the different points separable to see what effect it had on the image.

5 Evaluation

Evaluation goes here.

6 Discussion

Write later. Main points that could be improved/be interesting: Test other machine learning algorithms and compare results. Test on more people. More points/points not on a straight line.

6.1 Threats To Validity

6.1.1 Internal Threats

Human error during classification. The corners of the eye and the centre of the pupil were manually marked and with less than 100% accuracy. This introduces the possibility that some if not many of the images were wrongfully extracted, and may have impacted learning. It would be nice if it could be automated using eye tracking, but that also comes with some risk of misidentifying the features, and identifying errors could also be very time consuming.

Web-cam quality. The quality of the web-cam images were not very high. This makes the potential usability of our solution cheaper which is a plus, but it may impact learning results. Based on rescaling of the images, it does not seem like this is a significant factor.

Test set-up. Our test set-up required the test-subjects to sit a certain distance from the camera for the infra-red lights to be visible as glints in the eye. This had the effect that the eye were a smaller part of the input images than they could have been.

The test set-up was also somewhat loose, and heads had some movement during head-still sequences. Some noise like this is to be expected, but it may have had an effect on learning.

Resizing of learning data. Images were scaled down to 20x20 pixels to simplify the learning process. This could potentially have an impact on learning. From our results, the rescaling only had a minor effects.

6.1.2 External Threats

Generalisability. Eyes come in many shapes and forms and can vary greatly in shape, scale and location. We only tested on a few ethnicities and may have gotten very different results for others.

Given proper test set-up, it is to be expected that you could use our approach to learn what points that ethnicity is looking at, but the same training data set may not be universal.

7 Related Work

Related work goes here.

8 Conclusion

No lights head move - Not separable No lights head still - Left/right separable, PCA 1-2, 2-3, 2-4, 2-5 With lights head move - Not separable With lights head still - Separable, PCA 1-2, 2-3, (2-4), ((2-5))

References

[1] Abu-Mostafa, Y.S.: Learning from data – On-line course. `http://work.caltech.edu/telecourse.html` (2012)

[2] Abu-Mostafa, Y.S., Magdon-Ismail, M., Lin, H.T.: Learning from data. AMLBook (2012)

[3] Sutton, R.S.: Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: Advances in Neural Information Processing Systems 8, MIT Press (1996) 1038–1044

[4] Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: Safely approximating the value function. In: Advances in Neural Information Processing Systems 7, MIT Press (1995) 369–376

[5] : SciKit Learn - unsupervised nearest neighbours

[6] Smith, L.: A tutorial on principal components analysis. Cornell University, USA **51** (2002) 52

Figure 4: Data gathering on test-subject.

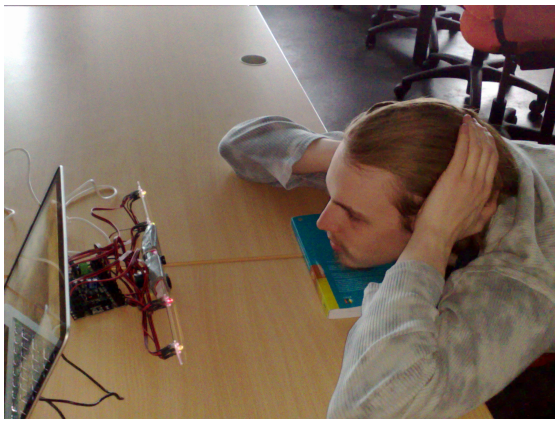

Figure 5: Data gathering on test-subject.

A Images of Test Subject During Experiments

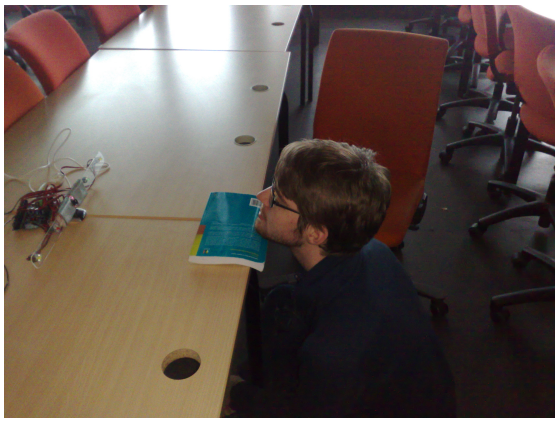

Figure 6: Data gathering on test-subject.

Figure 7: Data gathering on test-subject.