Master's Thesis

Human Activity Recognition and Behavioral Prediction using Wearable Sensors and Deep Learning

Victor Bergelin

LiTH-MAT-EX–2017/04–SE

Human Activity Recognition and Behavioral Prediction using Wearable Sensors and Deep Learning

Mathematical Statistics, MAI, Linköpings Universitet

Victor Bergelin

LiTH-MAT-EX–2017/04–SE

This thesis was made in collaboration with the Psychiatric Research Center, Geisel School of Medicine and Dartmouth College.

Master Thesis: **30 hp**

Level: **D**

Supervisors: **G. McHugo**,
Dartmouth Psychiatric Research Center, Geisel School of Medicine
M. Singull,
Mathematical Statistics, MAI, Linköpings Universitet

Examiner: **M. Singull**, Linköpings Universitet

Linköping: **March 2017**

GEISEL
SCHOOL OF
MEDICINE
AT DARTMOUTH

Abstract

While currently moving into a more connected and automated world, a more intelligent behavioral recognition tool is suggested. With an increased availability in wearable sensors we explore a better understanding of human needs. The study, at the Psychiatric Research Center, has examined the viability of detecting and further on predicting human behavior and complex tasks. The field of smoking detection was challenged by using the Q-sensor by Affectiva as a prototype. Further more, this study implemented a framework for future research on the basis for developing a low cost, connected, device with Thayer Engineering School at Dartmouth College. With 3 days of data from 10 subjects smoking sessions, events were detected with just under 90% accuracy using the Conditional Random Field algorithm. However, predicting smoking with Electrodermal Momentary Assessment (EMA) remains an unanswered question.

Keywords: Wearable sensors, Machine learning, Deep learning, Long Short-Term Memory, Conditional Random Fields, Computational psychiatry.

URL for electronic version:

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-138064

Acknowledgements

My first and dearest thanks goes to Gregory McHugo for having me as an intern twice at the Dartmouth Psychiatric Research Center, Dartmouth College. Thank you for guiding me through the process of medical work and our study. Working and learning from you meant a great deal to me. I wish you the best of luck in your research and a well deserved retirement.

Martin Singull has made the formalities with Linköping University as easy as they could have been. By acting as supervisor and examiner, the communication and administrative work was kept at a minimum. Thank you for your help and support.

Victor Bergelin, 2017

Nomenclature

Most of the reoccurring symbols and abbreviations are described here.

Symbols

Format	**Example**
Vectors use bold lower case	$\mathbf{y}$
Matrices use bold upper case	$\mathbf{X}$
Subscripts indicate weight and input time steps and indices	w_i and i_i
True label	y
Predicted label	$\hat{y}$
Sigmoid function	$sig()$
Network weights	V, W and U
Node/cell input	x_i, i_i
Label on data point d	t_d
Network input	net_j
Current output on data point d	o_d
Classifier output	Y
Node function	s_i
Node and edge features	$\theta(\mathbf{y}, \mathbf{x})$
Output of memory cell	y^{out_j}

Abbreviations

Abrevation	Explanation
BFGS	Broyden Fletcher Goldfarb Shanno
BP	Back Propagation
BPTT	Back Propagation Through Time
CEC	Constant Error Carousel
CPD	Conditional Probability Distribution
CRF	Conditional Random Fields
DGM	Directed graphical model
EDA	Electrodermal activity
EMA	Electrodermal Momentary Assessment
ETL	Extract Transform Load
FFT	Fast Fourier Transform
HAR	Human Activity Recognition
HMM	Hidden Markov Model
LBFGS	Limited Memory BFGS
LSTM	Long Short Term Memory
MAG	Accelerometer total magnitude of input
MEMM	Maximum Entropy Markov Models
NN	Neural Network
RNN	Recurrent Neural Network
RTRL	Real-Time Recurrent Learning
SC	Skin conductance
SLP	Single-Layer Perceptron
ST	Skin temperature

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Problem Specification and Desired Outcome

Recognizing human activities is a noble task moving into an era of connected sensors and commonly available wearable computing, also known as Internet of Things (IoT). It is at the core of assistive technologies to provide knowledge of what activity is performed by users when trying to understand their behavior. With further knowledge of activity classification using large amounts of unlabeled data, researchers and further on, users, can benefit from more intelligent and understanding machines around them.

This thesis assesses how a complex task, specifically cigarette smoking sessions, can be detected using a cheap, commercially available wrist worn accelerometer. The task is to detect and classify complex activities and differentiate it from other normal life activities. An overview is shown in 1.1.

The group, led by Gregory McHugo and Sarah Lord at Gaisel Medical School at Dartmouth College, would with this measurement and predictive capability ultimately look for what is known as the 'just-in-time intervention' of addictive

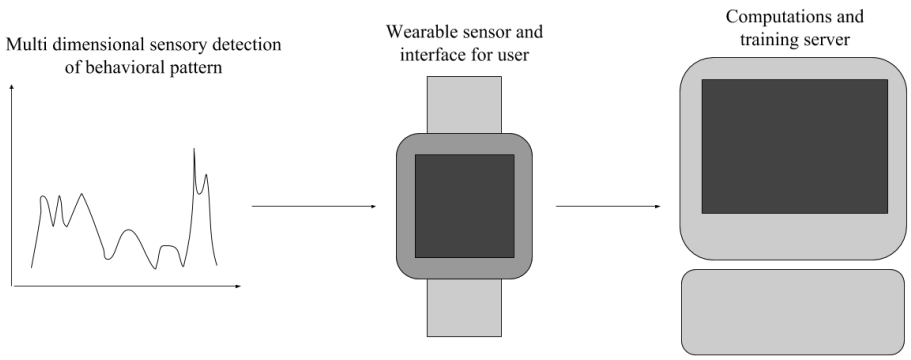

Figure 1.1: System overview

behavior. This has long been a holy grail in behavioral science and psychiatry research, here introduced to some state of the art computational methods in sequential prediction.

1.2 Hypothesis and Approach

By introducing Machine- and Deep Learning in classifying the hypothesis suggest some improvements in the predictability of a complex human activity. For example the hand movement of smoking a cigarette. Based on previous results, the focus will be to investigate and compare several computational and knowledge based approaches and assess differences in recognition performance compared to Deep Learning.

Whilst the master thesis hypothesis is stated above the much bigger Q-sensor study had a broader set of goals in mind (stated in Table 2.1).

1.3 Evaluation of Hypothesis

This thesis project aims to examine the difference in Human Activity Recognition (HAR) performance using machine learning time series and recurrent deep learning networks. The goal is then to examine collected data, feature extract using these two methods and compare efficiency. This study would like to map feasibility on behavior recognition, craving and predicting smoking.

1.4 Limitations

This thesis considers a set of activities under specific circumstances. Although this methodology and approach is applicable in many fields the goal is not to build an algorithm suited for traditional HAR (walking, sitting, lying, running, etc.) but rather increase the performance with more complex activity recognition for behavioral prediction.

In the suggested approach some assumptions used as priors will be made from domain knowledge, and performance differences with and without assumptions will be evaluated. One example is performance increase using prior knowledge of the recurrence in the monitored activity, for example in cigarette smoking. Also, the implementation does not consider on-device computations as all calculations and modeling is made offline.

1.5 Methodology

Ideas of complex relationships within psychiatric measurements and new wearable sensors suggest that stress related predictors could increase performance in activity detection with sequential and deep classification algorithms. This

study assesses the hypothesis by building a framework for automatic sensor fusion and classification, with semi-labeled training data. The approach both verifies itself with enclosed code, but also provide a tool for continued research and applications. Results are compared with sub sampled truth labels in training dataset.

1.5.1 Implementation

The implementation of this study consists of a classification and analysis framework, written in Python code as a package. It has dependencies to some standard packages such as time, csv, numpy and more specific packages: sklearn-crfsuite, Keras, Theano and TensorFlow. The implementation was done in python and bash.

1.5.2 Literature Study

Publications was chosen in three different areas, 1. Psychiatric research and feature selection, 2. Sequential classification algorithms and 3. Deep learning algorithms.

For step 1, previous work on wearable sensors for stress measures was assessed, primarily Santosh Kumar's work with the AutoSense suite of sensors. Further on in step 2 was Conditional Random Fields (CRF) chosen for sequential baseline classifier. This was done with help from professor Quiang Liu at the Computer Science department at Dartmouth, and with reference to 'Machine Learning - A probabilistic perspective' by Kevin P. Murphy[1]. Step 3 follows: implementing a deep learning method. Best resource for deep learning proved to be original papers on recurrent neural networks[2] and LSTM[3] and for implementation reference manuals for Keras[4].

1.6 Related Work

Inspiration for this study lies in the work performed by Santosh Kumar with the AutoSense suite[5].

Currently, Kumars group get 95% accuracy in training with multiple accelerometers, and prediction with just one sensor. Our study collected more quantitative data, resulting in more data for model building[5].

1.7 Thesis Outline

Short outline of thesis content

- Background: HAR, the Q-sensor and the data.
- Theory: Feature extraction, data modeling and deep learning.
- Implementation: Data collection and analysis.
- Results: A presentation of results and findings.
- Discussion: Quantitative and qualitative measures discussed.

Chapter 2

Background

2.1 The Q-sensor Study

This thesis project was conducted together with the Gaisel School of Medicine at Dartmouth College. The research group, which is funded by NIDA (National Institute for Drug Abuse), wanted to answer the following questions:

1. Feasibility
   (a) Will participants wear a wrist sensor for extended periods of time? - Keep it charged, keep it dry, not damage it?
   (b) Will participants press the button on the sensor as requested? - When meating, when smoking, when craving?
   (c) Will participants respond to Electrodermal Momentary Assessment (EMA) prompts six times per day? - Use smartphone appropriately, charge phone?
   (d) Will participants adhere to a multi-week, complex protocol? - Come to the lab and complete tasks (smoke, stress, self-report)?
2. Can we detect smoking with the sensors in the Q-sensor (accelerometer, skin conductance, skin temperature)?
   (a) Can we predict smoking based on marked events (supervised learning)?
      i. What is the overall accuracy rate of detecting smoking?
      ii. What is the distribution of smoking detection rates across individuals?
      iii. What are the key features used for classification of smoking?
   (b) Can we predict smoking without marking (unsupervised learning)?
      i. What is the overall rate of detecting smoking?
      ii. What is the distribution of detection rates across individuals?

iii. What are the key features used for classification of smoking?

3. Can we detect common features during the minutes prior to marked smoking? (prediction)
   (a) What is the overall accuracy rate of predicting smoking?
   (b) What is the distribution of smoking rates across individuals?
   (c) What are the key features used for classification of pending smoking?

This thesis contributed to data exploration and modeling in these questions. Much work has also been put in to collect the data, perform the interviews and manage the setup needed to make all this work.

2.2 Complex Human Activity Recognition (CHAR)

Thorough research has been put into the field of Human Activity Recognition (HAR), the study to differentiate common activities in daily life (walking, running, sitting, standing, etc). Complex Human Activity Recognition is then suggested to account for a broader variety of tasks and activities with an algorithmically more complex structure. In this study cigarette smoking patterns was analyzed, which happen in irregular intervals and potentially while performing other activities.

2.2.1 Feature Extraction

As shown by Stephen J. Preece et al[6] in the study 'A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data', a mixture of time and frequency features is most effective. Their feature extraction paper shows FFT magnitude being the best classifier in time/frequency domain for classical activity recognition, followed by 'mean low' and 'mean rectified' high pass filtered signals', utilizing low pass filtering to separate the AC and DC component.

2.3 The Q-sensor

The sensor chosen for the task is a well performing research sensor made by Affectiva (has now stopped production). The reason for this is to utilize the rare skin conductance- and temperature sensor in conjunction with accelerometer data. In psychiatric research, connections between stress and Electrodermal activity (EDA), or skin conductance, has been found[7].

2.3.1 Specifications

The Q-Sensor has the following technical specifications:

- Electrodermal activity (EDA), by conductance in micro Siemens
- Skin temperature in Celsius or Fahrenheit
- 3-Axis accelerometer measured in G's
- 36h of battery life

The accelerometer has sampling rates at 2,4,8,16,32 Hz. 4 Hz was chosen, a compromise on data quality and battery life.

2.4 Measuring Performance

When evaluating algorithms and classification, an index and tool for measuring success accurately is necessary. Some of the most common in the machine learning community is explained below.

When defining terms such as recall, precision, F_1-score and support within information retrieval we need intermediate stepping stones in constructing the terms.

Term	**Definition**
True positive (TP)	Correctly classified event, equivalent with a hit
True negative (TN)	Correctly rejected event
False positive (FP)	Type I error, equivalent with a false alarm
False negative (FN)	Type II error, equivalent with a miss

Table 2.1: Definitions of terms when predicting classification performance

2.4.1 Precision

Precision can be seen as the positive predicted value rate. Of all samples classified as class i, how many were actually i? Precision is given by

$$Precision = \frac{TP}{TP + FP}.$$

The statistic tells us how well we differentiate class i from other classes.

2.4.2 Recall

Measuring recall is sometimes called the true positive rate, resembled by

$$Recall = \frac{TP}{TP + FN}.$$

This measure determine the rate of how many instances of class i where correctly classified as i, and how many was missed - showing how well we detect class i.

2.4.3 F_1-score

The F_1 is a try to represent accuracy with both Recall and Precision, using the average of the two. F_1 of the different F_β-scores represents the harmonic mean

$$F_1 = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}.$$

2.4.4 Support

The support is the number of true occurrences of each class; the number of training samples per label class. As implemented in scikit learn can be found on their website[8].

2.5 Data Collection

Data from this study was collected from 9 different subjects, all wearing the Q-sensor described in earlier chapters. The data contains 4 days of labeled data and 6 additional days with unlabeled data. The patients were first educated and then equipped with sensors to capture the patterns of the smoking sessions in 'the wild'.

When collecting data, a general issue in real world samples is labeling. Attempts have been made to correct noisy, binary, classification problems with some success[9]. With imprecise labeling, learning is polluted with false labels that would be affecting the performance of the classification.

2.6 Machine Learning

Machine learning is a subfield of the broader group of computer science[10]. The study of Machine learning is data analysis that automates analytical model building, and in some sense computer science answer to induction. The field has many similarities with data mining, although now the results are not for human interpretation. In machine learning the outcome is (usually) to continuously update program actions accordingly.

As the name 'Machine Learning' suggests, it is about computers learning from data, the field is divided into two subgroups of learning: supervised and unsupervised. This thesis primarily explores supervised learning, to learn from

labeled training data and apply it to unseen data for predictions. Unsupervised algorithms can draw inferences from datasets with no labels to perform clustering, feature selection, etc. [1]

2.6.1 Deep Learning

When looking at machine learning algorithms, a simple one layer architecture is utilized on the form of $x \rightarrow y$ for supervised models. A recent trend in the field has adopted the model of processing used by the brain; using many levels of learning features at increasing levels of abstraction. The standard model of the visual cortex suggests that (roughly speaking) the brain extracts features in multiple layers. First edges, then patches, then surfaces and lastly objects. This method has inspired the method of machine learning known as deep learning which attempts to replicate this architecture in algorithms used in computers[11].

Chapter 3

Theory

When performing activity recognition and prediction one first needed to understand the algorithms in machine- and deep learning used to classify and discriminate different activities. This chapter explains some prerequisites and the two algorithms that are being compared.

3.1 Softmax Function

In probability theory, the softmax function is a generalized logistic function that normalize a m-dimensional vector $\mathbf{z}$ of arbitrary real values to a vector $\sigma(\mathbf{z})$ of real values between 0 and 1, also in dimension m, such that the sum of elements is 1:

$$\forall \mathbf{z} \in \mathbb{R}^m : \qquad \sigma(\mathbf{z}) \in [0,1] \text{ and } dim(\sigma(\mathbf{z})) = m.$$

The function is given by

$$\sigma(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}.$$

3.2 Feature Extraction

For features, X,Y and Z dimensions was first merged to also include total acceleration magnitude (Mag). The total set of inputs now include: X, Y, Z, Mag, Skin conductance (SC) and Skin temperature (ST). For every one of the sensors 6 dimension, the features was extracted and compared in their importance. To evaluate the significance of the features, a leave-one-out approach was used in predicting accuracy for the full dataset. Also, a package specific implementation for highlighting the static importance of each feature was assessed.

Test with leave-one-out was performed in groups of the following features left out:

- Frequency-based approach for all features
- Square of sums for percentiles (below 25 and over 75) in magnitude, for all features
- SC and ST individually held out

For this, statistical features was used in both time- and frequency domain data of labeled sequences. Features include:

- Mean amplitude of window
- Variance of window
- Sum of difference between consecutive measurements
- Square of sum, below 25th percentile in magnitude of feature
- Square of sum, above 75th percentile in magnitude of feature

All features above are made from the time- and frequency domain signal of each sensor dimension.

3.3 Conditional Random Fields

In this project we used linear chain Conditional Random Fields (CRF) (La erty et al. 2001). A model using a posterior over sequences of feature data to make classifications, based on labeled training data. The algorithm is therefore a discriminative, undirected probabilistic graphical model. Similar to a Markov Random Field (MRF) but with clique potentials conditioned on input features:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \prod_c \psi_c(\mathbf{y}_c|\mathbf{x}, \mathbf{w}).$$

CRFs can be thought of as an extension of the logistic regression with structured output. In this application the structure is a linear chain in time. As for the association with logistic regression, a log-linear representation of the potentials is assumed:

$$\psi_c(\mathbf{y}_c|\mathbf{x}, \mathbf{w}) = \exp(\mathbf{w}_c^T \phi(\mathbf{x}, \mathbf{y}_c)),$$

where $\phi(\mathbf{x}, \mathbf{y}_c)$ is a feature vector derived from inputs $\mathbf{x}$ and labels $\mathbf{y}$.
Next, a brief look is spent at generative directed Hidden Markov Models (HMM):

$$p(\mathbf{x}, \mathbf{y}|\mathbf{w}) = \prod_{t=1}^{T} p(y_t|y_{t-1}, \mathbf{w}) p(x_t|y_t, \mathbf{w}).$$

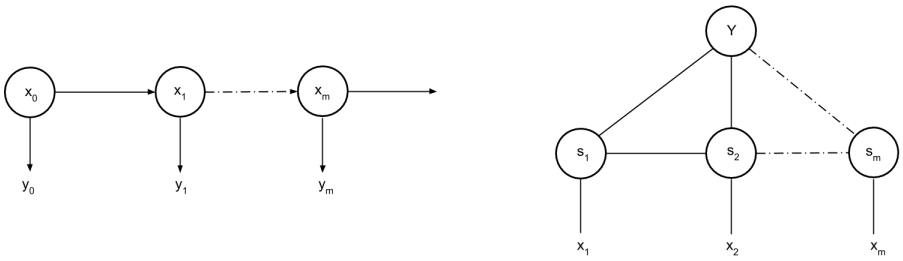

Figure 3.1: HMM and CRF

Traditionally it would be an opponent to CRFs, but it has some distinct differences that will soon be more apparent. In Figure 3.1 follows a graphical representation of HMM and CRFs to highlight important differences between the two related sequential models.

By reversing the arrows of the HMM, pointing from input x to the output, a directed discriminative model is defined. This model is usually called a Maximum Entropy Markov Model (MEMM) (McCallum et al. 2000; Kakade et al. 2002) and it represents a Markov chain in which the state transition probabilities are conditioned on the input features. Formally it is also a special case of an input-output HMM. It has the following form:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \prod_t p(y_t|y_{y-1}, \mathbf{x}, \mathbf{w}),$$

where $x = (x_{1:T}, x_g)$: x_g are global features, and x_t are features specific to node t. This model has one significant problem called the label bias problem (Lafferty et al. 2001). It basically means that local features at time t do not influence states prior to time t. Looking at the graphical representation in the Figure above we see the arrows going from a previous state progressing in time forward with no influence of previous states. This has consequences as we try to predict and generalise over long sequences of sensor data, as a model which could utilize context better would have higher predictive capability. Now consider CRFs from the Figure above. It has the following form:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \prod_{t=1}^{T} \psi(y_t|\mathbf{x}, \mathbf{w}) \prod_{t=1}^{T-1} \psi(y_t, y_{t+1}|\mathbf{x}, \mathbf{w}).$$

Here, the state y_t does not block the flow of information from x_t (as seen in the Figure) and the label bias problem do not exist. The label bias problem in MEMMs occurs because the directed models are locally normalized while the undirected models, such as MRFs and CRFs, are globally normalized. This means that the CPD for MEMMs both sum to 1 locally, while this is not the strict case for undirected models. Consequently CRFs are not as useful as Directed Graphical Models (DGMs) for online or real-time inference and since Z depend on all nodes and parameters it makes training of CRFs slow.

3.3.1 Training CRFs

Using a similar, gradient based approach, as for MRFs we can implement this method for CRFs. The scaled log-likelihood here becomes

$$\ell(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w}) = \frac{1}{N}\sum_i \left[\sum_c \mathbf{w}_c^T \phi_c(\mathbf{y}_i,\mathbf{x}_i) - \log Z(\mathbf{w},\mathbf{x}_i)\right]$$

and the gradient simplifies to

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{w}_c} &= \frac{1}{N}\sum_i \left[\phi_c(\mathbf{y}_i,\mathbf{x}_i) - \frac{\partial}{\partial \mathbf{w}_c}\log Z(\mathbf{w},\mathbf{x}_i)\right] \\ &= \frac{1}{N}\sum_i [\phi_c(\mathbf{y}_i,\mathbf{x}_i) - \mathbb{E}[\phi_c(\mathbf{y},\mathbf{x}_i)]].\end{aligned}$$

Looking at the pairwise case of x and y we can write the model as follows

$$p(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{1}{Z(\mathbf{w},\mathbf{x})}\exp(\mathbf{w}^T\phi(\mathbf{y},\mathbf{x}))$$

where $\mathbf{w} = [w_n, w_e]$ are the node (w_n) and edge (w_e) parameters, and

$$\phi(\mathbf{y},\mathbf{x}) \triangleq \left[\sum_t \phi_t(y_t,\mathbf{x}), \sum_{s\ t} \phi_{st}(y_s,y_t,\mathbf{x})\right]$$

are the summed node and edge features (the sufficient statistics). The gradient can then be modified to handle this case.

3.4 Long Short-Term Memory RNNs

The Long Short-Term Memory Recurrent Neural Networks (LSTM RNN) has proven highly efficient in utilizing memory cells in recurrent networks, under certain conditions. Before introducing LSTM, a brief introduction to neural networks and recurrent neural networks follows.

3.4.1 Feed-Forward Neural Networks

As introduced previously, around deep learning, the brain uses a 'deep' structure of connections. The (feed-forward) Neural network (NN) represent this architecture. It is the artificial creation of connections between units that does not form a cycle, as such it is different from the recurrent neural network (feed-forward for only feeding information forward). An illustration of the structure

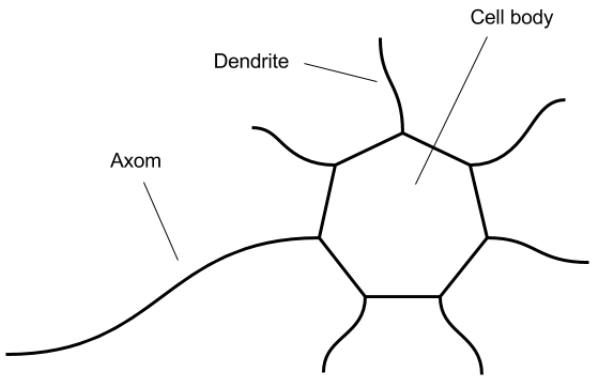

Figure 3.2: Neuron of a brain

of connections in the brain is shown in Figure 3.2

The perceptron. The simplest model of the Feed-forward Neural Network is the Single-Layer Perceptron. This network has a single layer of output nodes and the input is fed directly to the outputs via a series of weights (see Figure 3.3).

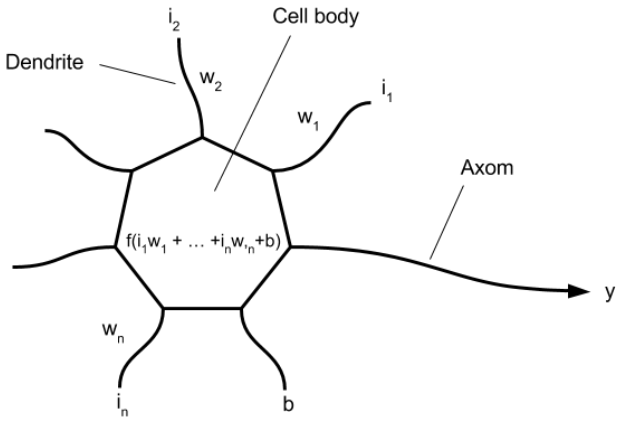

Figure 3.3: Single Layer Perceptron

For this neuron, the output y is simply the weighted sum of the inputs plus a bias

$$y = f(i_1 w_1 + \cdots + i_n w_n + b),$$

where i_i is the inputs, w_i the weights and b the bias. For any NN the designer has a choice of activation function f, but it is commonly chosen as the logistic function (also known as the sigmoid function)

$$f(x) = \frac{1}{1 + \exp(-x)}.$$

When choosing the logistic function for activation one has now created the logistic regression model, which is common in statistical modeling.

The logistic function has has a continuous derivative, which allows it to be used in back propagation (explained below) for learning. The derivative is especially suitable since it can easily be calculated as:

$$f' = f(1 - f)$$

The Multi-Layer Perceptron. When adding depth to the Neural Network we reach the state of Multi-layer Perceptron, seen in Figure 3.4.

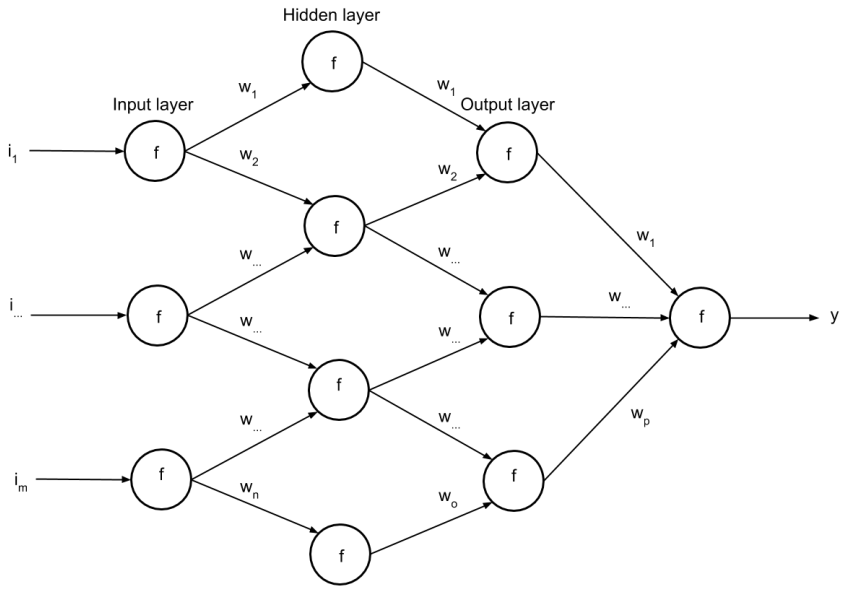

Figure 3.4: Multi layer perceptron

Back propagation. From: 'Back propagation of errors', this method is highly utilized in training networks and finding suitable weights. Combined with a proven search method, the back propagation always converges - which is desired. During one epoch of back propagation, the algorithm iterates over all training examples once and updates the weights at learning rate α. The iteration continues until a stopping criterion is satisfied. Before training the network a structure (number of hidden layers, hidden nodes and connectivity) is set, a differential transfer (activation) function is chosen. Then a (differentiable) error function is defined, here squared error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2,$$

where d is one point in dataset D, t_d is the label of d:th example, o_d current output on d:th example.

The algorithm searches for weights that minimize the error function through gradient descent (or other optimization method). When using gradient descent, a minimum exist where the gradient of the error function is zero

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 \\ &= \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2 \\ &= \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\ &= \sum_{d\in D}\frac{\partial}{\partial w_i}(-o_d).\end{aligned}$$

For the special case of the logistic (sigmoid) function as transfer function, one can show that the gradient simplifies to

$$\nabla E(\mathbf{w}) = -\sum_{d\in D}(t_d - o_d)sig(\mathbf{w}\cdot\mathbf{x}_d)(1 - sig(\mathbf{w}\cdot\mathbf{x}_d))\cdot\mathbf{x}_d.$$

In stochastic gradient descent (in comparison to 'batch gradient decent' for example) each weight update goes through all training instances one at a time. The weight update step can thus be expressed as

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\nabla E(\mathbf{w}),$$

with learning rate α.

3.4.2 Recurrent Neural Networks

A RNN is an artificial neural network with connections in cycles, where data in a feed-forward NN is only flowing in one direction over the layers. Information in an RNNs travels in loops from layer to layer so that influence is taken from previous steps. This group of networks add to the ability to algorithmically preserve a temporal dynamic state or memory. It is the natural architecture of neural networks for sequential data[12] [2].

In Figure 3.5, s_t is the hidden state at time t. It is calculated based on previous hidden states and the input x_t of the current step. Here, again, using the sigmoid

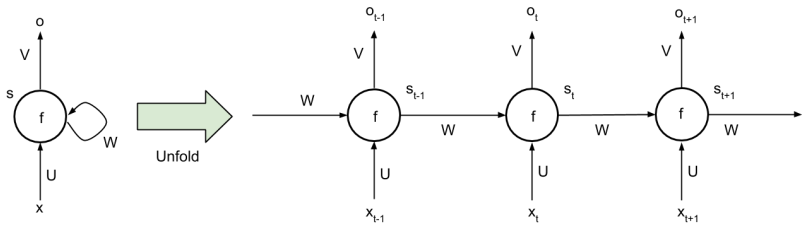

Figure 3.5: RNN sequence

($sig()$) function as activation we estimete the prediction $\hat{y}_t$. The weights V, W and U will need to be trained as:

$$s_t = sig(Wx_t + Us_{t-1}) \tag{3.1}$$
$$\hat{y}_t = softmax(Vs_t). \tag{3.2}$$

Here the softmax function is used to assign probabilities to the output classes in y_t at time step t.

It is worth to note that, unlike traditional deep neural networks, that uses different parameters at each layer, a RNN share the same parameters across all steps. The same task is performed at each step, only with different inputs. This greatly reduces the total number of parameters needed to learn.

Training RNNs using BP through time (BPTT). Just as for a traditional Neural Networks the RNN uses the back propagation algorithm, but with some modification. Parameters are shared over all steps in time and thus the gradient at each output not only depend on the calculation of the current time step, but also previous steps. When summing gradients back in time we perform Back Propagation Through Time (BPTT).

We have our RNN equations from Section 3.4.2 and will now define the loss function E as the cross entropy, given by

$$\begin{aligned} E_t(y_t, \hat{y}_t) &= -y_t \log \hat{y}_t \\ E(y, \hat{y}) &= \sum_t E_t(y_t \hat{y}_t) \\ &= -\sum_t y_t \log \hat{y}_t, \end{aligned}$$

where $\hat{y}_t$ is the prediction at time t, and y_t is the label with the true value.

The gradient is calculated gradient of the error from the weight parameters V, W and U using Stochastic Gradient Descent. Comparing with equation 3.4.1 we now sum the gradients at each time step for each training example

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \tag{3.3}$$

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \tag{3.4}$$

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}. \tag{3.5}$$

To perform the updates just

$$\begin{aligned} \mathbf{U} &\leftarrow \mathbf{U} - \alpha \nabla E(\mathbf{U}) \\ \mathbf{V} &\leftarrow \mathbf{V} - \alpha \nabla E(\mathbf{V}) \\ \mathbf{W} &\leftarrow \mathbf{W} - \alpha \nabla E(\mathbf{W}). \end{aligned} \tag{3.6}$$

A solution to the three partial derivatives, in equations (3.3), (3.4) and (3.5), is needed. When calculating the error E at time t, the algorithm propagates back in time. First with weight V for the output layer, since it has the least dependencies:

$$\begin{aligned} \frac{\partial E_i}{\partial V} &= \frac{E_i}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial V} \\ &= \frac{E_i}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \frac{\partial z_i}{\partial V} \\ &= \quad \dots \\ &= (\hat{y}_i - y_i) \otimes s_i. \end{aligned} \tag{3.7}$$

The calculations of the derivatives has been simplified to show that $\frac{\partial E_i}{\partial V}$ at time step i (or at any other point in time) only depends on values in the current time step: $\hat{y}_i$, y_i and s_i. So by using (3.7) it is a matter of matrix multiplication to calculate the gradient for V.

Though, for $\frac{\partial E_i}{\partial U}$ and $\frac{\partial E_i}{\partial W}$, a different approach is needed. In the following equations $\mathbf{Z}$ represents both U and W, and k is the step in time currently calculating (starting from 0)

$$\begin{aligned} \frac{\partial E_i}{\partial \mathbf{Z}} &= \frac{E_i}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \frac{\partial z_i}{\partial \mathbf{Z}} \\ &= \sum_{k=0}^{i} \frac{E_i}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \frac{\partial z_i}{\partial z_k} \frac{\partial z_k}{\partial \mathbf{Z}}. \end{aligned} \tag{3.8}$$

With Equation (3.7) and (3.8) we can expand the equation in time as seen in Figure 3.6.

By using equation (3.6) we can now update the weights of the recurrent network at learning rate α[12].

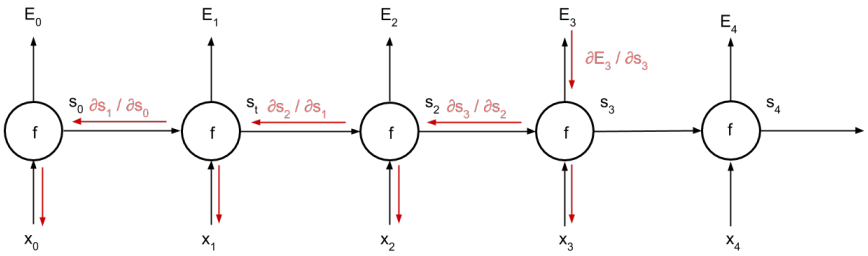

Figure 3.6: Information propagating back through time

3.4.3 Error Back-flow

Before LSTM, approaches such as Back-Propagation Through Time (BPTT, e.g. Williams and Zipser 1992, Werbos 1988) or Real-Time Recurrent Learning (RTRL, e.g. Robinson and Fallside 1987) were used indisputably. These approaches both uses error signals that 'flow backwards in time', whom suffers from challenges of errors' vanishing or exploding (becoming to large). This causes oscillating weights and issues with detecting in long time lags. This results in long training periods for long time lag data, or the algorithm failing (Hochreiter, 1991).

3.4.4 LSTM

Long short-term memory networks is a recurrent network architecture that overcomes these error back-flow problems. It learns to detect time intervals separated by thousands of steps without loss of short time lag capabilities. It operates over noisy and incompressible input sequences to predict sequences over an appropriate gradient-based learning algorithm. Results are achieved using a constant error flow (neither vanishing nor exploding) through internal states of special units within the algorithm.

Naive approach to Constant Error Flow. Looking at a single unit connected to itself, we asses how to avoid vanishing error signals. At time t, j's local error back flow is $\vartheta_j(t) = f'_j(net_j(t))\vartheta_j(t+1)w_{jj}$. The requirement to enforce constant error flow through j is

$$f'_j(net_j(t))w_{jj} = 1.0 \tag{3.9}$$

Constant Error Carousel (CEC). By integrating the differential equation (3.9), we obtain $f_j(net_j(t)) = \frac{net_j(t)}{w_{jj}}$ for an arbitrary $net_j(t)$. This implies that: f_j has to be linear, and unit j's activation has to be constant

$$y_j(t+1) = f_j(net_j(t+1)) = f_j(W_{jj}y^j(t).$$

This is ensured by utilizing the identity function $f_j : f_j(x) = x, \forall x$ and by setting weight $w_{jj} = 1.0$. The procedure and this setting is called the constant error carousel (CEC) by Sepp Hochreiter and Jürgen Schmidhuber, the authors. This is the central feature of LSTM. Two complications are worth mentioning, also common in other gradient-base approaches:

1. Input weight conflict. With the simplest approach, we observe a single weight update w_{ji}. Further we assume that the total error can be reduced by turning on unit j in respons to a particular input and keeping it active until ready: when it's helpful to compute a desired output. By having non-zero input weight i, we see that the same unit j has to be used for both storing and ignoring various inputs (recall that j is linear). This results in w_{ji} receiving conflicting weight update signals; (1) storing the input by turning j on, (2) protecting the input by preventing j from being switched off by irrelevant later inputs. The resulting update signals make learning difficult and calls for a more intelligent, context-sensitive mechanism for controlling write operations on input weights.

2. Output weight conflict. As for the Input weight conflict, the output weight has a conflict in it's update. The same principle apply when weight w_{kj} receive different signals for (1) accessing information stored in j and at different times (2) protecting unit k from being perturbated by j. Again, the conflict makes learning difficult and require an active, context-sensitive, mechanism for the read operations of output weights.

Memory cells and gate units. The key is constant error flow. This is achieved through special, self-connected units extended from the CEC of self-connected, linear unit by introducing additional features. A multiplicative input and output gate is introduced to protect the unit from incoming and outgoing perturbation caused by irrelevant input at the input gate and output from the memory cell at the output gate. The more complex unit is called a memory cell (see Figure 3.7). The memory cell is denoded c_j. Memory cells are built around central linear units with fixed self-connection (the CEC). More than net_{c_j}, c_j get input from a multiplicative unit out_j (the "output gate"), and another multiplicative unit in_j (the "input gate"). in_j's activation at time t is denoted by $y^{in_j}(t)$, out_j's by $y^{out_j}(t)$. Then we state

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t))y^{in_j}(t) = f_{in_j}(t)),$$

where

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1)$$

and

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1).$$

We also have that

$$net_{c_j}(t) = \sum_u w_{c_j u} y^u(t-1).$$

All summation indices are general for input units, gate units or memory cells, all determined by the topology of the network, defined by the user. Gates can even be recurrent self-connected (w_{c_j,c_j}). At time t, c_j's output $y^{c_j}(t)$ is computed as

$$y^{out_j}(t)h(s_{c_j}(t)),$$

where the "internal state" $s_{c_j}(t)$ is

$$s_{c_j}(0) = 0,\ s_{c_j}(t) = s_{c_j}(t-1) + y^{in_j}(t)g(net_{c_j}(t)) \text{ for } t > 0.$$

The function g compresses net_{c_j} to a single signal and it is differentiable. h is also differentiable and scales the output from the memory cell, derived from the internal state s_{c_j}.

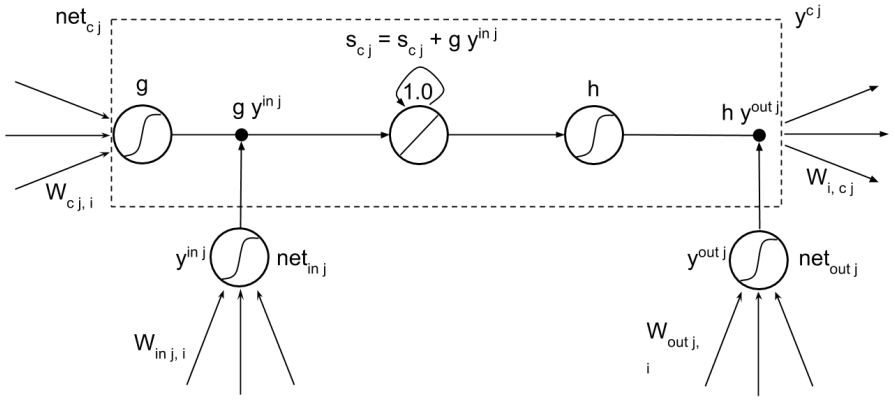

Figure 3.7: LSTM internal memory state

The network uses in_j to decide when to keep or override information in memory cell c_j. In the same way is out_j used to decide when to access memory cell c_j and when to prevent access to other units (usualy to prevent disturbance of the system from c_j). In more detail: The gate unit in_j controls input weight conflicts by managing the error flow to memory call c_j's input connections w_{c_j}. out_j is used to avoid wight conflicts in outputs from c_j by controlling the error flow from unit j's output connections.

Chapter 4

Implementation

4.1 Extract, Transform and Load

The data was imported as CSV (Comma Separated Values) for offline processing using the Q software installed on a Mac, connected to the Q-sensor by micro usb. Through a Python script, the data was imported for manipulation, learning and classification.

4.2 Feature Extraction Module

A fix seed was used for sampling the data when assessing the feature performance. This give a more accurate comparison over multiple runs.

Moreover, the feature extraction was implemented in python. Features was calculated using Numpy and Scipy packages [13].

4.3 Machine Learning Module

A Python implementation called CRFsuite was used for CRF implementation. More specifically: sklearn_crfsuite, which utilizes the popular and standardized SciKit learn interface for machine learning. This implementation of CRF uses the convex optimization routine lbfgs for training (limited memory BFGS)[14].

4.4 Deep Learning Module

For the LSTM RNNs, the Keras framework was chosen for model building. A variety of layers and setups was tested as shown in Table 4.1 below while also adjusting hyper parameters.

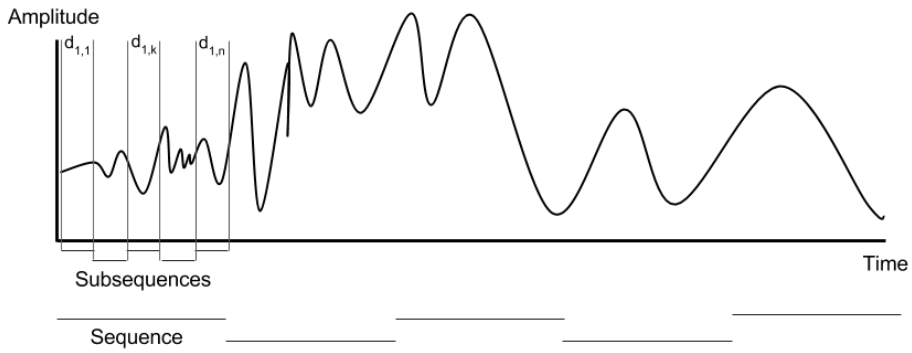

Figure 4.1: Sequences and sub sequences of an arbitrary signal

All networks had only one output as the binary classifier, smoking or not. Further more, a variation in layer configuration was assessed.

Method	**Layer configuration**
Baseline	Inputs from features, hidden: 50 → Out
Medium	Inputs from features, hidden: 50 → 50 → 50 → Out
Full	Inputs from features, hidden: 100 → 300 → 0.2 dropout → 200 → 0.2 dropout → Out

Table 4.1: Setup of LSTM layers and nodes

4.5 Hyperparameter Tuning

Tuning hyper parameters showed to be a substantial undertaking in testing the two algorithms. With many parameters in wide ranges the testing was time consuming as each run would take approximately 35 minutes with Keras and about 15 minutes for sklearn-crfsuite.

Hyperparameter	**Description**	**Range**
Sequence length	Length of full sequences	$30 - 600$ sec
Sub sequence length	Length of sub sequences	$1 - 10$ sec
Training vs testing	Proportion of data for training	$60 - 95\%$
Class weights	Balancing significance between classes	$2 - 20$ x for smoking
Label prior	Average of estimated class event length	$200 - 600$ sec (smoking)
Number of epochs	How many times iterated over data	$10 - 250$ times

Table 4.2: Hyperparameter tuning

For an explination of sequences and sub sequences see Figure 4.1.

The work included extended parameter tuning and in this process the results varied between 75% and about 90% for the crf implementation. The best results were achieved with the parameter values showed in the next section.

4.5.1 CRF

Parameter	**value**
No lable to lable size scaling	70%
Training vs testing data	80%
Sequences length	30.0 sec
Subsequences length	5.0 sec
Minimum time between labeled cigarettes (for error correction)	65 minutes
Smoking sequence label length after pressing event button	5.25 minutes (315 seconds)

Table 4.3: Parameter values in CRF implementation

4.5.2 LSTM

Parameter	**value**
No lable to lable size scaling	95% was no label (all data used)
Sequence length	864 steps, 3 minutes 36 seconds
Subsequence length	1 step, 0.25 seconds
Batch size	450 sets
Epochs	250 complete sets of batches
Class weights	0: 1, 1: 20

Table 4.4: Parameters values in LSTM implementation

Chapter 5

Results

In this chapter results are presented from methods described in previous chapters. Overall prediction accuracy on labeled data is the main performance measure of the classification and the baseline for comparing the two methods.

5.1 Data Collection

Patients collected data on all days as requested, but did miss some labels and would also miss-classify some labels by accidentally pressing the button. One subject (of 10 participants) did not finish the study but remaining 9 produced valuable data.

After the data collection phase the data shows, and interviews confirm, that the patients collected data on all days as requested. From interviews the team learned that some patients had trouble remember to mark events as instructed and some labels are therefore missing. The data also show that a type of miss-labeling occurred as some patients would click multiple times on the event label button within a short period of time. Pressing the button cannot be undone easily unfortunately so these labels remain in the dataset. One subject of the total of 10 participants starting out did not finish the study; remaining 9 all produced valuable data.

Several more important findings for the study was made from the qualitative analysis of the interviews that were held, all related to psychiatric hypothesises. Questions regarding future, larger, sensor studies in 'the wild' was also in part captured from information learned in these patient interviews.

5.2 Q-sensor Data

The data collected with the Q-sensor showed overall good quality with on-board noise cancelling. Very little noise nor technical errors in data (as seen in Figure 5.1).

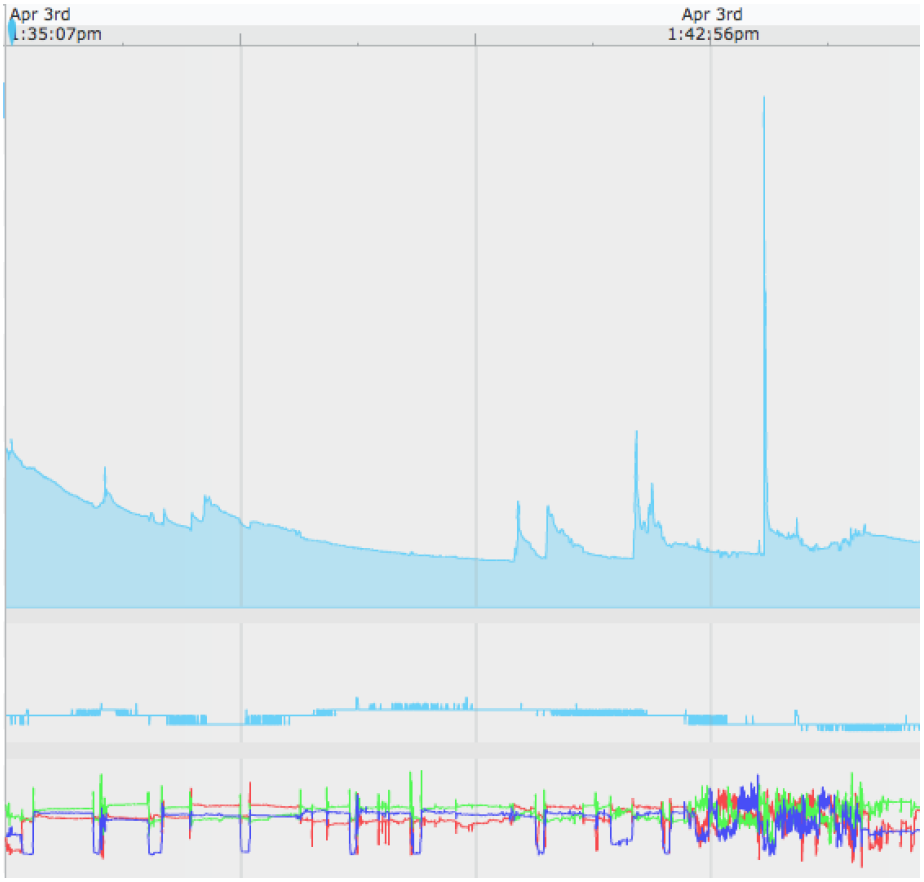

Figure 5.1: Raw sensor input after noise reduction

Figure 5.1 shows the sensory input of a patient smoking. The top curve is the skin conductance, the middle showing skin temperature and in the bottom of the graph: three dimensions of the accelerometer. The top left has a small blue marker to show the pressing of the device button, indicating he/she is lighting a cigarette. The periodic arm movement can easily be distinguished in this example. From interviews we know that some patients where also, for example, smoking while driving - which has a much different moving pattern.

5.3 Subsampling

Performance differences using our label noise cancelling implementation was unpractical to measure. Although we do not have a definitive number for how many unlabeled sessions was recorded we can say that with subsampling performed only about $30 - 50\%$ of the miss-classified events should have come through to the training phase.

5.4 CRF Results

Using CRF for sequential classification over all data gave a classification accuracy between 87%, and as low as 83%, with only variation in seed for randomized subsampling.

5.4.1 Activity Recognition

Introduction to CRF results.

Detecting smoking, as the first part of the research study, proved to work at around 85% accuracy for some different test and with different seeds as seen in Table 5.1 and 5.2.

label	precision	recall	f1-score	support
0	0.779	0.851	0.813	10467
1	0.880	0.819	0.848	13950
avg / total	0.837	0.833	0.833	24417

Table 5.1: CRF results, run 1

label	precision	recall	f1-score	support
0	0.853	0.897	0.874	14283
1	0.888	0.841	0.864	13851
avg / total	0.870	0.869	0.869	28134

Table 5.2: CRF results, run 2

Unseen Subject Labeling

In this test, generalization between subjects was measured in performance. The run is set with fix random seed and all data but the test subjects as training data. The CRF algorithm was used for this comparison.

subject	precision	recall	f1-score	support
0	0.833	0.828	0.830	41634
1	0.833	0.829	0.830	5940
2	0.824	0.822	0.823	2430
3	0.844	0.832	0.834	10440
4	0.859	0.858	0.858	5076
6	0.849	0.840	0.842	7200
7	0.839	0.836	0.837	10080
8	0.834	0.830	0.831	6993
9	0.818	0.816	0.817	9360
10	0.843	0.841	0.842	6300

Table 5.3: Average/total results for each subject prediction

As seen in Table 5.3, the result has variations of approximately 5% in precision and recall between subjects.

5.5 Predicting Behavior

5.5.1 CRF

As anticipated, predicting smoking behavior proved much harder than detecting the activities. After immense parameter tuning the test was abandoned.

label	precision	recall	f1-score	support
0	0.783	0.985	0.873	4866
1	0.520	0.056	0.100	1404
avg / total	0.724	0.777	0.700	6270

Table 5.4: Results CRF classification minutes before smoking

5.6 LSTM Results

LSTM results were overall low. With the recurrent network no ability to predict or detect smoking was obtained. This was called after thorough parameter tuning and testing.

Chapter 6

Discussion

6.1 ETL

Extract, transform and load using the Q.app interface turned out to be very time consuming as the Q-sensor uses its own logging format. Files needed to be processed in Q.app and then exported to csv to be further processed in Python. A sensor that would save the raw data right in csv would have been preferred. Data still has noise and could be filtered further for a smoother curve. If this would increase performance would be easy to examine.

6.2 Further Development

Several thoughts on implementation and algorithm development was left for future development. The most promising are explained below.

6.2.1 Clustering

With labels being slightly off in time domain, resulting in inaccurate training data, unsupervised clustering could be applied to minimize these errors.

6.2.2 Data Size

Having a total of around 24 000 sequences from 10 different subjects was deemed as enough in early planing of the project (transformed with 30 sec intervals, 22.5 hours per subject on average). The key point is not the sequence count but the true label count. This dataset was very sparse.

The result showed that the chosen deep learning approach will need many more labels to be collected as deep learning is more data intensive than Conditional Random Fields.

6.2.3 Sensor Dimensions

Similar research in preventive addiction care is being performed with other sensors. How to measure 'stress' is a topic of great divide as seen in the literature study. To perform further research, a larger test group and a more versatile sensor would be recommended.

The temperature sensor used in the Q-sensor housing was much too slow and dependent, did not show anything when analyzing the data. On the other hand, skin conductance gave promising results. Although suffering from some issues with measurement and noise, the data quality of both accelerometer and skin conductance was acceptable. To gain better performance in classification, we suggest adding a heart rate and heart rate variability sensor. Further research has been put in the sensor band developed with Gunnar Pope at Theyer Engineering School at Dartmouth College. His band shoes very promising results in durability, cost, ease of use and sensors (skin conductance, hart rate and HR variability).

6.2.4 CRF Class Weight

The chosen CRF implementation had no support for class weights which heavily reduced the used data size due to low true class cases. While showing promising results yet these limiting factors, a class weight feature could greatly improve the performance with small data sets.

6.3 LSTM

The use of Long- short term memory networks and the Keras framework was expected to capture some of the complexity in the analysis that we were performing. It came as a disappointment but not necessarily as a surprise. With only about 400 marked events of smoking totally, this is much smaller than your standard deep learning dataset.

6.4 Results

The goals was always to explore the data as Professor McHugo said of the sensor study, with the holy grail in mind (predicting smoking). Emphasis needs to be put to the exploratory value of the research he insisted. A somewhat disappointing results could be a good one too.

6.5 Ethical Considerations

If/when self-logging and wearable sensors become a part of daily life , we might look back at the pre-monitoring age in awe. Serious thought and consideration

should be put into the consequences of monitoring. Therefor we disclaim this research result to be used for anything but self-chosen addiction treatment.

Bibliography

[1] K Murphy. *Machine Learning - A probabilistic perspective*. 2012. The MIT Press.

[2] C Goller and A Kuchler. Learning task-dependent distributed representations by backpropagation through structure. *In Proc. of the ICNN-96*, pages 347–352, 1996. IEEE.

[3] S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. The MIT Press, Cambridge, MA, USA.

[4] C Francois. Keras. https://github.com/fchollet/keras, 2016. GitHub repository.

[5] S Kumar et al. E Ertin, N Stohs. Unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field. 2011. SenSys '11 The 9th ACM Conference on Embedded Network Sensor.

[6] A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data.

[7] H Critchley. Electrodermal responses: What happens in the brain. *Neuroscientist*, 8 (2):132–142, 2002. SAGE Publications.

[8] A Gramfort V Michel R Weiss V Dubourg J Vanderplas A Passos D Cournapeau M Brucher M Perrot F Pedregosa, G Varoquaux and E Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] P Ravikumar A Tewari N Natarajan, I Dhillon. Learning with noisy labels. *Advances in neural information processing systems*, 26:1196–1204, 2013. Neural Information Processing Systems Foundation, Inc.

[10] S Russell and P Norvig. *Artificial Intelligence, A Modern Approach (2nd ed.)*. 2003. Prentice Hall.

[11] D Hubel and T Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962. Wiley-Blackwell.

[12] R Williams D Rumelhart, G Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. Nature Publishing Group.

[13] Pearu Peterson et al. E Jones, T Oliphant. SciPy: Open source scientific tools for Python, 2001–.

[14] The Scipy community. Crf-suite. https://scipy.org/, 2016.