3rd Year Project Final Report

SleepAppnea

George Cochrane, Sachin Mylavarapu, Tuan Anh Le, Sophie Louth

March 28, 2014

Abstract

The abstract text goes here.

Introduction

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Medical Information

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Rationale

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

The app

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Design Process

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.1 Questionnaire selection

5.2 Signal Analysis - simple method

5.3 Machine Learning – Theory

Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data [9]. The supervised classification problem in machine learning concerns finding the unknown target function that classifies certain input data into classes based on some set of training examples containing labelled input data.

In our case, given a sampled sound signal of a sleeper, we want to identify apnoea periods. In particular, we represent our input (sampled sound signal) as $\{s_1, s_2, \ldots, s_T\} \equiv \{s_i\}_{i=1}^{T}$, and we want to output the classifiers for every K samples, $\{y_i\}_{i=1}^{T/K}$, where the classifier $y_i \in \{0, 1\}$ corresponds to samples $\{s_j\}_{j=(i-1)K+1}^{iK}$ of the signal. We assume that T is divisible by K, however if this is not the case, we discard an appropriate number of signal samples to make it so.

We have researched three models for our problem which we will discuss in this section. Firstly, we will discuss Support Vector Machines which are one of the most widely used algorithms in Machine Learning today, then we will discuss the State-Space and Hidden Markov Models, which turn out to be more well-suited to our problem due to their temporal nature.

5.3.1 Support Vector Machines

Here, we present the theory for Support Vector Machines (SVMs) based on the lecture notes from Prof. Andrew Ng [3]. SVMs are one of the most widely used and many argue among the best "off-the-shelf" supervised learning algorithms. This is mainly due to the sound theoretical framework, efficiency and good generalisation guarantees even for high-dimensional and linearly non-separable data.

Notation

Having m training examples, where

- $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is the d-dimensional i-th training data.
- $y^{(i)} \in \{-1, 1\}$ is the i-th training label.

we want to find the parameters $\mathbf{w} \in \mathbb{R}^d$ which describe the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that separates our two classes. Thus, we can define our classifier as $h_{\mathbf{w},b}(\mathbf{x}) = g\left(\mathbf{w}^T\mathbf{x} + b\right)$, such that $g(z) = 1$ if $z \geq 0$ and

$g(z) = -1$ otherwise. Note that this is a non-probabilistic learning model as we are not considering the probability of each class or the data.

Objectives

The essence of SVMs lies in finding the decision boundary (hyperplane) which maximises the gap between the closest training points to it. For the i-th training point $x^{(i)}$, we call the "gap", geometric margin $\gamma^{(i)}$, which is the distance to the decision hyperplane and can be found by considering a point $\mathbf{x}$ on it, such that $\mathbf{w}^T\mathbf{x} = -b$:

$$\begin{aligned}\gamma^{(i)} &= \left| \frac{\mathbf{w}^T}{\|\mathbf{w}\|} \left(\mathbf{x}^{(i)} - \mathbf{x}\right) \right| \\ &= \frac{1}{\|\mathbf{w}\|} y^{(i)} \left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\end{aligned}$$

However, we only consider the closest point $\mathbf{x}^{(i^*)} : i^* = \arg\min_i \gamma^{(i)}$ with the geometric margin of $\gamma = \min_i \gamma^{(i)}$. Since scaling $\mathbf{w}$ and b does not change the output of the classifier, nor the geometric margin $\gamma^{(i)}$, for convenience, we decide to scale $\mathbf{w}$ and b such that $\left|\mathbf{w}^T\mathbf{x}^{(i^*)} + b\right| = y^{(i^*)}\left(\mathbf{w}^T\mathbf{x}^{(i^*)} + b\right) = 1$. Thus, maximising the geometric margin of the closest point becomes

$$\begin{aligned}\max_{\mathbf{w},b} \quad & \frac{1}{\|\mathbf{w}\|} \\ \text{s.t.} \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad i = 1, \ldots, m.\end{aligned}$$

which is equivalent to

$$\begin{aligned}\min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad i = 1, \ldots, m.\end{aligned} \tag{5.1}$$

The solution to the optimisation problem (5.1), which can be found using off-the-shelf Quadratic Programming (QP) packages, gives us the parameters required to do classification.

Lagrange duality

While solving the optimisation problem (5.1) using QP solves the problem, we move on to derive the dual form of the problem using the principle of Lagrange duality, which will help us to solve the problem using a more efficient algorithm. More importantly, it will also allow us to efficiently transform the data points to high-dimensional spaces using the "kernel trick", capturing the non-linear nature of the decision boundary

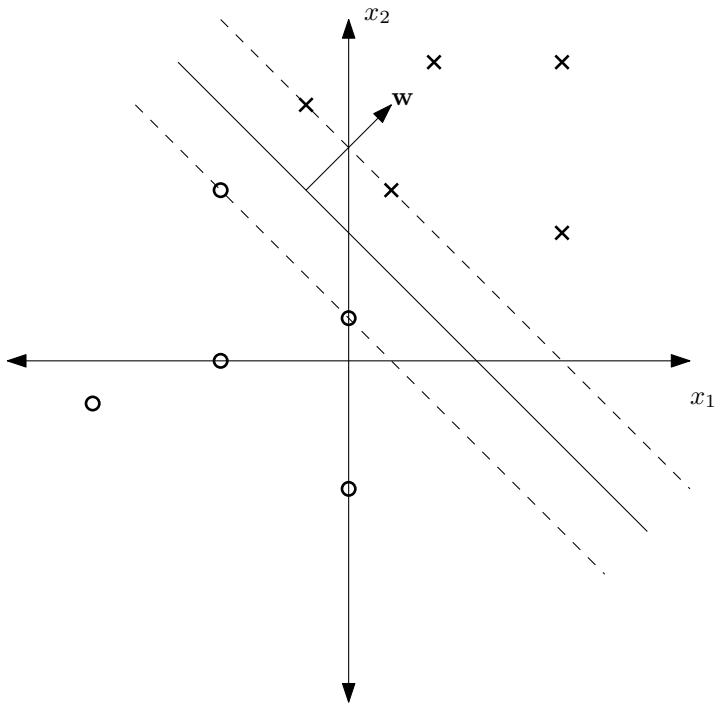

Figure 5.1: Illustration of the optimal margin classifier.

without losing the generalisation guarantees. The Lagrangian of our primal minimisation problem (5.1) (with parameters $\boldsymbol{\alpha} \in \mathbb{R}^m$) can be formed as

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)} \left(\mathbf{w}^T \mathbf{x}^{(i)} + b \right) - 1 \right] \tag{5.2}$$

To find the dual form of the problem, $W(\boldsymbol{\alpha}) = \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha})$, we find the gradients of (5.2) with respect to $\mathbf{w}$ and b and set them to zero to get

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)} \tag{5.3}$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \tag{5.4}$$

Substituting (5.3) and (5.4) back to (5.2) gives us the dual form, $W(\boldsymbol{\alpha})$

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \right\rangle \tag{5.5}$$

Thus, under certain conditions which are in this case fulfilled, our original optimisation problem (5.1) becomes equivalent to the dual optimisation problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) \\ \text{s.t.} \quad & \alpha_i \geq 0, \qquad\qquad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \quad i = 1, \ldots, m \end{aligned} \tag{5.6}$$

This problem can be solved using a QP algorithm, however a more efficient Sequential Minimal Optimization (SMO) algorithm can be used. Once the optimal value $\boldsymbol{\alpha}^*$ is obtained, we can find the corresponding parameters of the SVM $\mathbf{w}^*$ using (5.3) and b^* using

$$b^* = -\frac{\max_{i:y^{(i)}=-1} \mathbf{w}^{*T}\mathbf{x}^{(i)} + \min_{i:y^{(i)}=1} \mathbf{w}^{*T}\mathbf{x}^{(i)}}{2} \tag{5.7}$$

$$= -\frac{\max_{i:y^{(i)}=-1} \sum_{j=1}^{m} \alpha_i^* y^{(i)} \left\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \right\rangle + \min_{i:y^{(i)}=1} \sum_{j=1}^{m} \alpha_i^* y^{(i)} \left\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \right\rangle}{2} \tag{5.8}$$

Thus the classification of a test point $\mathbf{x}$ can be done by evaluating the argument of $g(\mathbf{w}^{*T}\mathbf{x} + b^*)$

$$\mathbf{w}^{*T}\mathbf{x} + b^* = \sum_{i=1}^{m} \alpha_i^* y^{(i)} \left\langle \mathbf{x}^{(i)}, \mathbf{x} \right\rangle + b^* \tag{5.9}$$

We note that due to certain conditions (Karush-Kuhn-Tucker conditions) the α_i^*'s that are non-zero correspond to the points $\mathbf{x}^{(i)}$ that lie on the margin. We call these points the Support Vectors (SVs).

Kernel trick

Note that according to (5.9) (and (5.8)), we only need to evaluate the inner products of $\mathbf{x}$ and the support vectors in order to classify the point $\mathbf{x}$. This fact is used in the "kernel trick", in which a kernel function $K(\mathbf{x}, \mathbf{z})$ is used as a proxy for the inner product $\langle \mathbf{x}, \mathbf{z} \rangle$. This effectively simulates transforming the data points to another, possibly unknown, space via a transformation function $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \to \mathcal{Z}$ as long as there exists such $\mathcal{Z}$, i.e. $\boldsymbol{\phi}(\cdot)$ such that $K(\mathbf{x}, \mathbf{z}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{z})$. It is shown that a kernel $K(\cdot, \cdot)$ is valid if and only if it satisfies Mercer's condtitions. This allows us to transform our input domain into much higher dimensional domains without actually doing so which turns out to improve the time complexity

significantly.

$$K_{\text{poly}}(\mathbf{x}, \mathbf{z}) = \left(a\mathbf{x}^T\mathbf{z} + c\right)^k \tag{5.10}$$

$$K_{\text{rbf}}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right) \tag{5.11}$$

In our application, we are going to use the above kernels. The first one, (5.10), is the polynomial kernel which effectively transforms the input domain space into a feature space whose features are products all possible permmutations of the input features of degrees up to k. The second one, (5.11), is a radial basis kernel (RBF), whose corresponding transformation function transforms the input domain space to an infinite dimensional one. This kernel gives a large inner product to two points if they are close to each other.

Soft margin SVMs

Soft margin SVMs is a slight modification of the original problem that allows small violations of the margin. The purpose is to guarantee that the algorithm doesn't fail because of the non-separability of the input data even after using the kernel trick. In practice, these two techniques are used simultaneously. It turns out that only minor changes arise and the SMO algorithm can still be used.

Our problem

In our problem, we are going to use the `SVM` package in `MATLAB`® in order to train our model and classify sleep apnoea. The package includes implementations of both the QP and SMO algorithms, as well as various kernels, including the polynomial kernel and the RBF kernel. Recall that in our problem, we have the signal $\{s_i\}_{i=1}^T$ and the classifiers for every K samples $\{y_i\}_{i=1}^{T/K}$, where $y_i \in \{0, 1\}$ classifies the time period represented by $\{s_j\}_{j=(i-1)K+1}^{iK}$. When using SVMs, the input vectors and the classifiers become

$$\mathbf{x}^{(i)} = \left[s_{(i-1)K+1}, \ldots, s_{iK}\right]^T$$

$$y^{(i)} = \begin{cases} 1 & \text{if } y_i = 1 \\ -1 & \text{otherwise.} \end{cases}$$

5.3.2 State-Space Models

Here, we present the State-Space Models (SSMs) (also known as Linear Dynamical Systems) based on Kevin Murphy's book [2, Chapter 18] and Prof. Zoubin Ghahramani's paper [1]. SSMs allow us to model systems that are dynamic. Unlike the SVMs, SSMs will model dependencies of the current state on the previous ones.

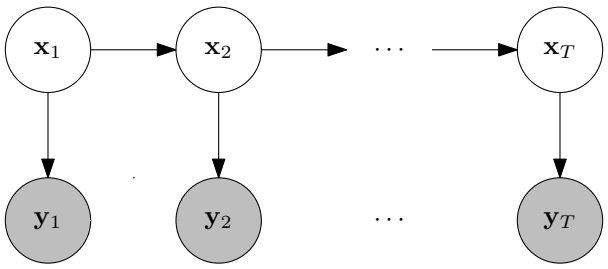

Figure 5.2: Probabilistic Graphical Model of a SSM and a HMM

We have two set of random variables, the hidden states $\left\{\mathbf{x}_t, \mathbf{x}_t \in \mathbb{R}^d\right\}_{t=1}^T$ and the observed vectors $\left\{\mathbf{y}_t, \mathbf{y}_t \in \mathbb{R}^k\right\}_{t=1}^T$. We can represent SSMs graphically in Figure 5.2, and write out the joint probability of the model as

$$p\left(\{\mathbf{x}_t, \mathbf{y}_t\}_{t=1}^T\right) = p(\mathbf{x}_1)p(\mathbf{y}_1|\mathbf{x}_1)\prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{y}_t|\mathbf{x}_t) \tag{5.12}$$

In our case, we consider these particular transition and observation models with zero-mean Gaussian noises:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_t \tag{5.13}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t \tag{5.14}$$

$$\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{5.15}$$

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{5.16}$$

Since we assume that $\pi = p(\mathbf{x}_1)$ is also Gaussian, all our conditional probability distributions will be Gaussian due to the linearity of the transition and observation models. Once the parameters are obtained, the problem of inference and state estimation consists of

1. **Filtering.** We want to find $p(\mathbf{x}_t|\{\mathbf{y}_t\}_1^t)$. This is solved by the Kalman filtering algorithm.

2. **Smoothing.** We want to find $p(\mathbf{x}_t|\{\mathbf{y}_T\}_1^t)$. This is solved by the Kalman smoothing algorithm.

3. **Prediction.** We want to find $p(\mathbf{x}_{t+\tau}|\{\mathbf{y}_t\}_1^t)$. This is done by solving to 1. and then simulating using the state transition function.

We are mainly interested in filtering and smoothing.

Our problem

Although SSMs are well-suited for time-series data, they are not very well suited for our problem because the hidden state $\mathbf{x}_t$ in this model is continuous, whereas in our problem, the sleeper's state is binary. While there are techniques to map from a continuous domain to a discrete one, we move on to discuss a more promising model.

5.3.3 Hidden Markov Models

Here, we present the Hidden Markov Models (HMMs), one of the most popular statistical models in machine learning with applications in many fields including but not limited to Crytanalysis, Speech recognition and Bioinformatics [8]. The material is based on [4], [5] and [2]. We will first present HMMs with discrete observations, then we extend this to include models with continuous observations.

Discrete observations

Similarly to SSMs, we have two sets of random variables. Observed variables $\{y_t\}_{t=1}^T$ which are drawn from the observation alphabet $V = \{v_1, \ldots, v_M\}$, and hidden states $\{x_t\}_{t=1}^T$ which are drawn from the hidden state alphabet $S = \{s_1, \ldots, s_N\}$. The probabilistic graphical model of the HMM is identical to the one in Figure 5.2. The hidden states obey Markov assumptions, i.e. $P(x_t|x_{t-1}, \ldots, x_1) = P(x_t|x_{t-1}) = \text{const.}, t = 2, \ldots, T$. Moreover, the observed variables are only dependent on the corresponding hidden state, i.e. $P(y_t|x_t, \ldots, x_1, y_{t-1}, \ldots, y_1) = P(y_t|x_t), t = 1, \ldots, T$.

We parameterise a HMM using the transition matrix $\mathbf{A}$, emission matrix $\mathbf{B}$, and the initial state distribution $\boldsymbol{\pi}$. The transition matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ describes the transitions between hidden states, $A_{ij} = P(x_{t+1} = s_j|x_t = s_i)$. The emission matrix $\mathbf{B} \in \mathbb{R}^{N\times M}$ describes the probability of an observation conditioned on a hidden state, $B_{jk} = B_j(v_k) = P(y_t = v_k|x_t = s_j)$. The initial state distribution $\boldsymbol{\pi} \in [0, 1]^N$ simply describes the initial probabilities of the hidden state, $\pi_i = P(x_1 = s_i)$. The model is fully described if we know these parameters, which we group into what is called a parameter set of the model $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$.

The three main questions of a HMM are

1. Find the probability of observations given the model, $P\left(\{y_t\}_1^T; \lambda\right)$.
2. Find the most likely series of hidden states $\{x_t\}_1^T$ to have generated the observations $\{y_t\}_1^T$, $\{x_t^*\}_1^T = \arg\max_{\{x_t\}_1^T} P\left(\{y_t\}_1^T | \{x_t\}_1^T; \lambda\right)$.
3. Find the parameters λ to maximise $P\left(\{y_t\}_1^T; \lambda\right)$.

We will discuss algorithmic solution to these three problems in turn.

Solution to the first problem. To find the probability of an observed sequence, we use the dynamic programming algorithm, called the FORWARD PROCEDURE (outlined in Algorithm 1) which calculates the forward variable $\alpha_t(i) = P(\{y_t\}_1^T, x_t = s_i; \lambda)$. As we can see, the algorithm has a time complexity of

Algorithm 1 FORWARD PROCEDURE for computing $\alpha_t(i)$.

1. **Initialisation.**
$$\alpha_1(i) = \pi_i B_i(y_1), 1 \leq i \leq N$$
2. **Induction.**
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) A_{ij}\right] B_j(y_{t+1}), \quad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix}$$
3. **Termination.** (solution to the first problem)
$$P\left(\{y_t\}_1^T; \lambda\right) = \sum_{i=1}^{N} \alpha_T(i)$$

$O(TN)$.

Solution to the second problem. To solve the problem of finding the most likely sequence of hidden states, we use the VITERBI ALGORITHM proposed by Andrew Viterbi in 1967. The most likely sequence of hidden states is also called the Viterbi path. Firstly, we define the quantity $\delta_t(i)$, which stores the highest probability along a single path ending at the state $x_t = s_i$:

$$\delta_t(i) = \max_{x_1,\ldots,x_{t-1}} P\left(x_1, \ldots, x_{t-1}, x_t = s_i, y_1, \ldots, y_t; \lambda\right) \tag{5.17}$$

By induction, we have

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) A_{ij}\right] B_j(y_{t+1}) \tag{5.18}$$

We also keep track of the index of the hidden state that maximises this quantity in

$$\psi_{t+1}(j) = \arg\max_i \delta_t(i) A_{ij} \tag{5.19}$$

Having defined these quantities, we present the VITERBI ALGORITHM in Algorithm 2. As we can see, the

Algorithm 2 VITERBI ALGORITHM for computing the most likely sequence of hidden states.

1. **Initialisation.**

$$\begin{aligned} \delta_1(i) &= \pi_i B_i(y_1), && 1 \le i \le N \\ \psi_1(i) &= 0, && 1 \le i \le N \end{aligned}$$

2. **Recursion.**

$$\begin{aligned} \delta_t(j) &= \max_{1 \le i \le N} \left[\delta_{t-1}(i) A_{ij}\right] B_j(y_t), && \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \\ \psi_t(j) &= \arg\max_{1 \le i \le N} \left[\delta_{t-1}(i) A_{ij}\right], && \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \end{aligned}$$

3. **Termination.**

$$\begin{aligned} P^* &= \max_{1 \le i \le N} \delta_T(i) \\ x_T^* &= \arg\max_{1 \le i \le N} \delta_T(i) \end{aligned}$$

4. **Path backtracking.**

$$x_t^* = \psi_{t+1}\left(x_{t+1}^*\right), T - 1 \ge t \ge 1$$

5. **Return** $\{x_t^*\}_1^T$**.**

algorithm has a time complexity of $O(TN^2)$.

Solution to the third problem. To learn the parameters of the model, $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, we use the FORWARD-BACKWARD ALGORITHM by Baum-Welch. We will need to introduce few quantities. Firstly, we introduce the backward variable $\beta_t(i) = P\left(y_{t+1}, \ldots, y_T | x_t = s_i; \lambda\right)$ which can be computed using a dynamic programming algorithm in Algorithm 3. We also define the variable $\gamma_t(i) = P\left(x_t = s_i | \{y_t\}_1^T; \lambda\right)$

Algorithm 3 BACKWARD ALGORITHM for computing $\beta_t(i)$.

1. **Initialisation.**
$$\beta_T(i) = 1, 1 \leq i \leq N$$
2. **Induction.**
$$\beta_t(i) = \sum_{j=1}^{N} A_{ij} B_j(y_{t+1}) \beta_{t+1}(j), \quad \begin{matrix} T-1 \leq t \leq 1 \\ 1 \leq j \leq N \end{matrix}$$

which can be expressed as

$$\begin{aligned} \gamma_t(i) &= P\left(x_t = s_i | \{y_t\}_1^T; \lambda\right) \\ &= \frac{P\left(x_t = s_i, \{y_\tau\}_1^t; \lambda\right) P\left(\{y_\tau\}_{t+1}^T | x_t = s_i; \lambda\right)}{P\left(\{y_t\}_1^T; \lambda\right)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)} \end{aligned} \tag{5.20}$$

We also define the quantity $\xi_t(i,j)$ as

$$\begin{aligned} \xi_t(i,j) &= P\left(x_t = s_i, x_{t+1} = s_j | \{y_t\}_1^T; \lambda\right) \\ &= \frac{\alpha_t(i) A_{ij} B_j(y_{t+1}) \beta_{t+1}(j)}{\sum_{i,j=1}^N \alpha_t(i) A_{ij} B_j(y_{t+1}) \beta_{t+1}(j)} \end{aligned} \tag{5.21}$$

Now, we are in a position to present the FORWARD-BACKWARD ALGORITHM by Baum-Welch in Algorithm 4. The algorithm belongs to a family of Expectation-maximisation (EM) algorithms for finding maximum likelihood (ML) or maximum a posteriori (MAP) estimates of parameters in statistical models [7].

Algorithm 4 FORWARD-BACKWARD ALGORITHM (BAUM-WELCH) for estimating HMM parameters λ.

1. **Initialisation.** Set $\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\pi}$ to be random valid probability matrices/vectors.
2. **Repeat until convergence:**
   - **E-step.** Run FORWARD and BACKWARD PROCEDURES to get $\alpha_t(i)$ and $\beta_t(i)$. Evaluate $\gamma_t(i)$ using (5.20).
   - **M-step.** Re-estimate parameters using
$$\begin{aligned} \pi_i &= \gamma_1(i) \\ A_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \\ B_j(v_k) &= \frac{\sum_{t=1, \text{s.t.} y_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned}$$

Parameters estimation with known hidden states. In case the hidden states $\{x_t\}_1^T$ are given to us, in addition to the observed variables $\{y_t\}_1^T$, the problem of parameters estimation simply reduces to counting transitions, i.e.

$$A_{ij} = \frac{\sum_{t=1}^{T-1} 1\{x_t = s_i \wedge x_{t+1} = s_j\}}{\sum_{t=1}^{T} 1\{x_t = s_i\}} \tag{5.22}$$

$$B_j(k) = \frac{\sum_{t=1}^{T} 1\{x_t = s_j \wedge y_t = v_k\}}{\sum_{t=1}^{T} 1\{x_t = s_j\}} \tag{5.23}$$

where $1(\cdot)$ is an indicator function (1 if the boolean argument is true, 0 otherwise).

Continuous observations

We now discuss the case in which the observations $\{y_t\}_1^T$ are not drawn from a finite set V, but are real-valued vectors $\{\mathbf{y}_t\}_1^T$, drawn from $\mathbb{R}^d$, i.e. $\mathbf{y}_t \in \mathbb{R}^d, 1 \leq t \leq T$ (Note: hidden states $\{x_t\}_1^T$ are still drawn from a finite set S). In this case, we cannot describe the observations using an emission matrix anymore. Since the observation random variables are now continuous, they must be drawn from some probability distribution function. This function can be arbitrary, however in order to learn anything useful, we parameterise it. We assume a simple case and let it be a Gaussian, parameterised by its mean, and covariance. In particular, $B_j(\cdot)$ becomes a probability distribution function:

$$\begin{aligned} B_j(\mathbf{y}_t) &= P\left(\mathbf{y}_t | x_t = s_j\right) \\ &= \mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_j, \mathbf{U}_j), \qquad \begin{matrix} 1 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \end{aligned} \tag{5.24}$$

This means that instead of the original emission matrix $\mathbf{B}$, we are now parameterising the emissions using the means $\boldsymbol{\mu} = \{\boldsymbol{\mu}_j \in \mathbb{R}^d\}_1^N$ and the covariances $\mathbf{U} = \{\mathbf{U}_j \in \mathbb{R}^{d \times d}\}_1^N$. Thus, our parameters set becomes $\lambda = \{\mathbf{A}, \boldsymbol{\mu}, \mathbf{U}, \boldsymbol{\pi}\}$.

Changes to the algorithms. While the three main questions for the HMM remain, we must change the algorithms used in the discrete observations case. It turns out that in the Forward Procedure (Algorithm 1), Backward Procedure (Algorithm 3), and the Viterbi Algorithm (Algorithm 2), we don't need to change anything except the interpretation of $B_j(\cdot)$. Whereas before, we had a single value for this quantity, now we must evaluate it using the parameters of the pdf (in this case mean and covariance) in (5.24). The problem of estimating parameters, given the observations becomes slightly

more different and will be left untouched in our report. However, we discuss parameters estimation, given both the observations and the hidden states, which simply becomes fitting a Gaussian, i.e.

$$\boldsymbol{\mu}_j = \frac{\sum_{t=1}^{T} 1\{x_t = s_j\}\mathbf{y}_t}{\sum_{t=1}^{T} 1\{x_t = s_j\}} \tag{5.25}$$

$$\mathbf{U}_j = \frac{\sum_{t=1}^{T} 1\{x_t = s_j\}(\mathbf{y}_t - \boldsymbol{\mu}_j)(\mathbf{y}_t - \boldsymbol{\mu}_j)^T}{\sum_{t=1}^{T} 1\{x_t = s_j\}} \tag{5.26}$$

Different probability density functions. We have assumed that the conditional distribution of the observations is Gaussian. One disadvantage of this method is that it may be too simple to capture the real pdf the observations are drawn from. One of the alternatives is the Gaussian Mixture Model (GMM), however fine-tuning the number of modes can be difficult. It turns out that finding the optimal MLE for a GMM is intractable [2].

Our problem

In our problem, we model the apnoeatic states as the hidden states $\{x_t\}_1^T$ drawn from a binary set $\{0, 1\}$. Although the observed signal is a sampled one-dimensional signal, since we only have annotations every K samples, we stack all K samples to a vector and treat it as a K-dimensional, real-valued observed variable $\mathbf{y}_t \in \mathbb{R}^d$, $(d = K)$. Since we assume that we have the annotated signal, we can do the training offline. Thus we are interested in the third problem (with known hidden states), for which the solution is just fitting the parameters; and the second problem, finding the most likely sequence of the apnoeatic diagnoses, given the model and the observations, using the VITERBI ALGORITHM. We will train the model using the annotated data and once trained, we will use the trained model to diagnose sleep apnoea from new data. Implementations of the algorithms for our applications are available for `MATLAB`® in the packages `pmtk3` (`https://github.com/probml/pmtk3`) and `HMM Toolbox` (`http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`) from Kevin Murphy.

5.3.4 Conditioning input data

Working in the time domain, the data is usually transformed to the frequency domain to analyse where the pattern is found more easily. Also, whichever model we choose to use, we will work in high dimensions which means we need a lot of input data in order to find the pattern. In the case of SVMs, it will be high-dimensional training points $\{\mathbf{x}^{(i)}\}_1^m$, in case of SSMs and HMMs, it will be the high-dimensional observed variables $\{\mathbf{y}_t\}_1^T$. Thus, we will present two ways to condition the input data, before

analysing it using the Machine Learning algorithms above. In practice, we use Principal Components Analysis to reduce the dimensionality of the spectrograms.

Spectrograms for Frequency-domain analysis

Discrete Fourier Transform Discrete Fourier Transform (DFT) is used to transform a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids [6]. It can be said to convert the sampled function from its original domain to the frequency domain [6]. The DFT, $\mathcal{F}$ can be defined to transform $\mathbf{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ to $\mathbf{X} = \{X_0, X_1, \ldots, X_{N-1}\}$ as

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-i2\pi kn/N\right), 0 \leq k \leq N-1 \tag{5.27}$$

and can be performed using the Fast Fourier Transform (FFT) algorithm.

Spectrogram To create a spectrogram of a signal $\mathbf{x} = \{x_0, x_1, \ldots, x_T\}$ (we are overloading the notation), we must choose a window size T_w and an offset T_o. Then, we keep shifting the window by T_o and each time, we transform and store the window's content (in time domain) to the frequency domain using the DFT. This way, we will have a frequency domain data of the size T_w every T_o (roughly), which we will call $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_\tau$. We usually represent this frequency domain data using colour intensities and plot them as τ columns of colour intensities over time on the horizontal axis and frequency on the vertical axis. Figure 5.3 illustrates the creation of a spectrogram.

Principal Components Analysis

Here we present the theory for Principal Components Analysis (PCA) based on Andrew Ng's lecture notes [3] and Kevin Murphy's book [2]. The objective is to take m n-dimensional input data points $\{\mathbf{x}^{(i)} \in \mathbb{R}^n\}_1^m$ and transform them into m k-dimensional data points ($k < n$) $\{\mathbf{y}^{(i)} \in \mathbb{R}^m\}_1^m$, where $\mathbf{y}^{(i)}$'s are projections of $\mathbf{x}^{(i)}$'s onto k orthonormal basis vectors $\{\mathbf{u}_i\}_1^k$ while "preserving the most variance". We assume that over all m points, mean $\left[\{x_j^{(i)}\}_{i=1}^m\right] = 0$ and var $\left[\{x_j^{(i)}\}_{i=1}^m\right] = 1$ for all features of the input points $1 \leq j \leq n$. We normalise the data first if this is not the case. For $n = 2$, $k = 1$, Figure 5.4 shows the projections to the new axis.

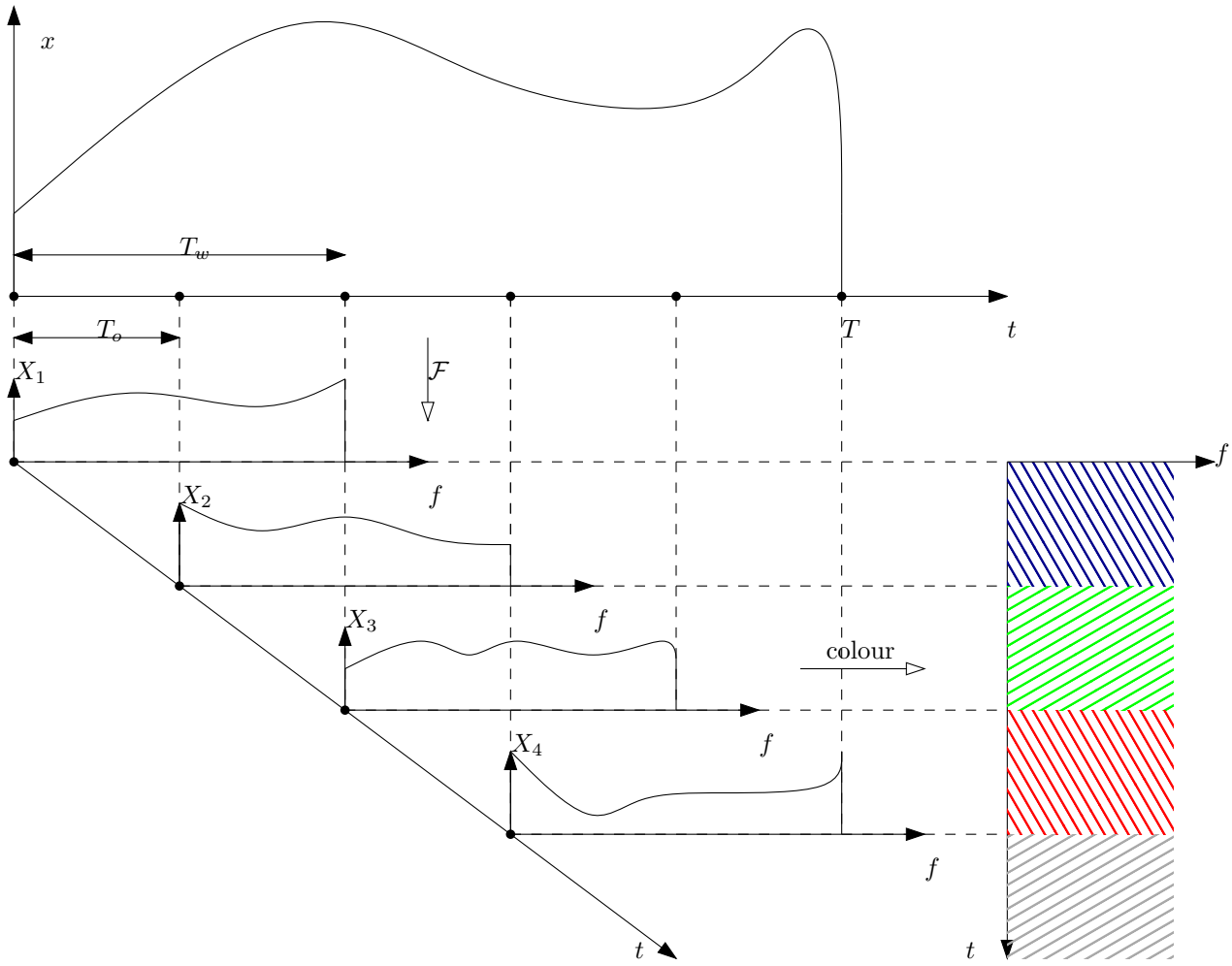

Figure 5.3: Creating a spectrogram.

We can see that the transformed points can be expressed as in (5.28).

$$\mathbf{y}^{(i)} = \begin{bmatrix} \mathbf{u}_1^T \mathbf{x}^{(i)} \\ \mathbf{u}_2^T \mathbf{x}^{(i)} \\ \vdots \\ \mathbf{u}_k^T \mathbf{x}^{(i)} \end{bmatrix}, 1 \leq i \leq m \tag{5.28}$$

It can be shown that in order to maximise $\sum_{i=1}^{m} \left\|\mathbf{y}^{(i)}\right\|^2$, conditioned on orthonormality of the bases, we must choose normalised eigenvectors corresponding to the k largest eigenvalues of the variance matrix $\mathbf{\Sigma} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}^{(i)} \mathbf{x}^{(i)^T}$ as the bases. These bases are also called the *principal components*. The eigenvalues are proportional to the actual variances of data in the new bases. As such, we can compare the relative "importance" of the bases and using this fact decide how many principal components to use.

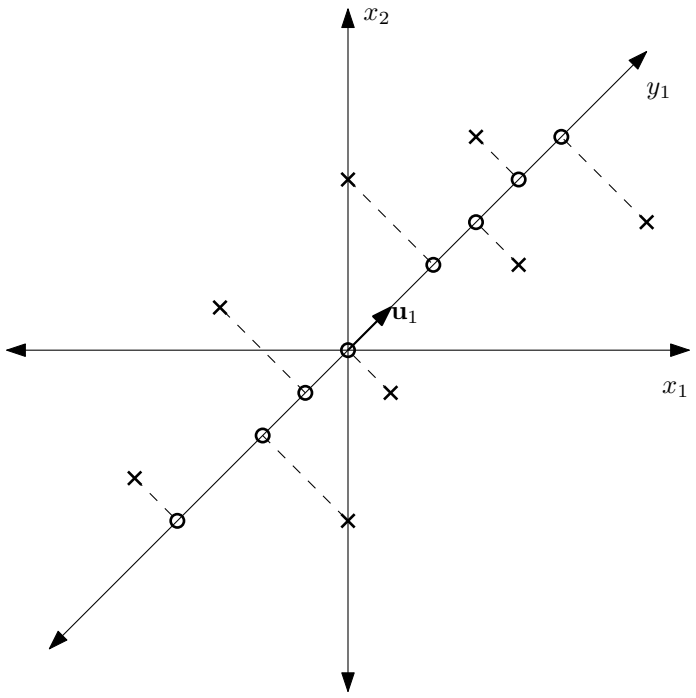

Figure 5.4: Illustration of PCA.

5.3.5 Machine Learning - Experiments

5.3.6 Android

5.4 Conclusion

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[1] Zoubin Ghahramani and Geoffrey E Hinton. Variational learning for switching state-space models. *Neural computation*, 12(4):831–864, 2000.

[2] Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

[3] Andrew Ng. Cs229 - lecture notes. Course website, 2013.

[4] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[5] Daniel Ramage. Cs229 - section notes. Course website, 2007.

[6] Wikipedia. Discrete fourier transform — Wikipedia, the free encyclopedia, 2014. [Online; accessed 28-March-2014].

[7] Wikipedia. Expectationmaximization algorithm — Wikipedia, the free encyclopedia, 2014. [Online; accessed 25-March-2014].

[8] Wikipedia. Hidden markov model — Wikipedia, the free encyclopedia, 2014. [Online; accessed 24-March-2014].

[9] Wikipedia. Machine learning — Wikipedia, the free encyclopedia, 2014. [Online; accessed 18-March-2014].