Pattern Recognition Assignment - Deadline 6/12

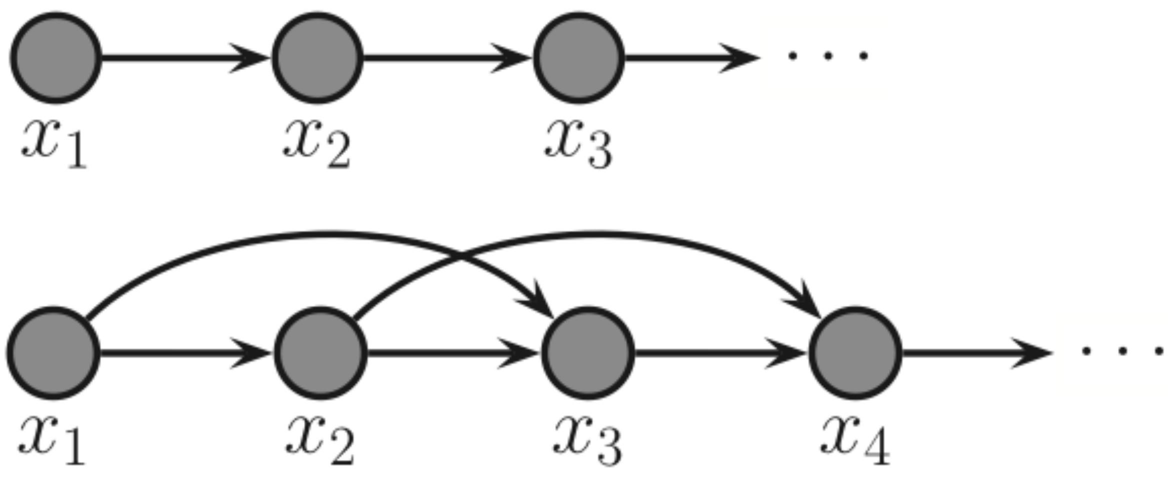
Name: Morten D. Laursen

Student id: a0197363

1. Discuss the similarity and difference between Kalman filter and hidden Markov model (forward algorithm).

Hidden Markov Model

A Markov Model is a stochastic model to depict randomly changing systems eg. transitions from one state to another. A model where the current state is directly observable is called a Markov Chain. In a Markov Chain we assume that the future/the new state only depends on the current state, and not on past events. Note that this doesn't mean, that we are "just" looking at *the* previous state, but we can look at for instance the two previous states. This is known as order. A markov model that decides the future state based on the two previous states, is known as a second order markov model. The picture below shows a first order (top) and a second order markov chain.



The difference between a markov chain and a hidden markov model (later denoted HMM) is the observability of the data. In a HMM the current state is not directly observable (thus is *hidden*), but the output of that depends on the current state is.

Example: As of writing this assignment, I am currently in Japan. In Japan, unlike in Denmark, umbrellas are used both when it's raining or the sun is shining. I want to predict the weather, but I want to do it, without looking out the window and instead I will watch wether my neighbour comes home with or without an umbrella. Let's first assume, that we know the probability of using an umbrella given the weather:

$$p(umbrella|rain) = 0.8\ ,\ p(umbrella|sun) = 0.2 \tag{1}$$

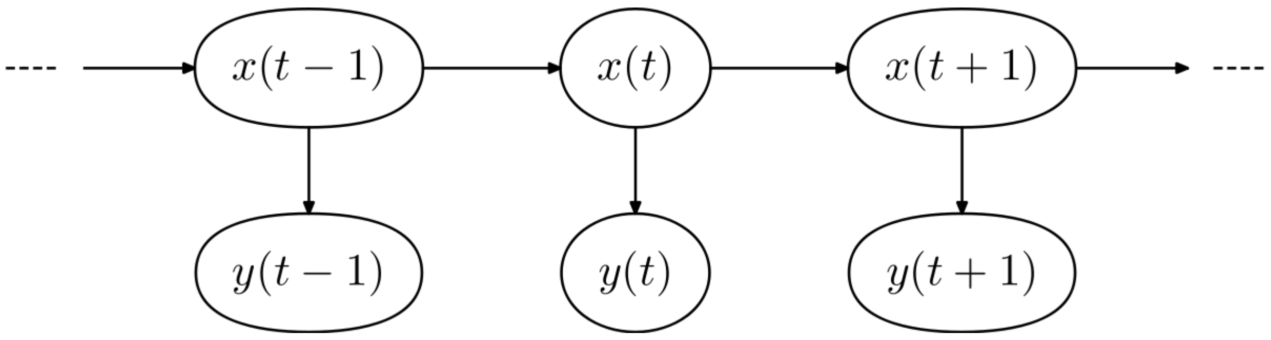
Let us assume, that we also know the probability of the weather changing or the weather staying the same (a markov chain). Let's use the following properties

$$p(sun|sun) = 0.8\ ,\ p(sun|rain) = 0.4\ ,\ p(rain|sun) = 0.2\ ,\ p(rain|rain) = 0.6 \tag{2}$$

By combining (1) and (2) we now have our HMM, which can be illustrated as the following:

[create picture]

The forward algorithm: The forward algorithm is used to calculate the probability of a state at a certain point given previous observations. The joint probability $p(x_t, y_{1:t})$, where x_t denotes the (hidden) state at point t, and $y_{1:t}$ denotes the previous observations up till that point. To do this we would normally marginalize all possible state sequences, which would require t^2 computations. The forward algorithm avoids this by using recursion.



To compute the joint probability, we start by assuming that we have our transition probabilities (as seen in (2)). To introduce these, we will marginalize over previous states:

$$p(x_1, y_{1:t}) = \sum_{x_{t-1}} p(x_t, x_{t-1}, y_{1:t}) \tag{3}$$

By using the general product rule we can factor the last part of (3), so we are able to use some of the things, we already know eg. transition probabilities and conditional probabilities.

$$p(x_t, y_{1:t}) = \sum_{x_{t-1}} (p(y_t|x_t, x_{t-1}, y_{1:t}) * p(x_t|x_{t-1}, y_{1:t-1}) * p(x_{t-1}, y_{1:t-1})) \tag{4}$$

due to y_t being conditionally independant we can rewrite the joint probability as

$$p(x_t, y_{1:t}) = \sum_{x_{t-1}} (p(y_t|x_t) * p(x_t|x_{t-1}) * p(x_{t-1}, y_{1:t-1}) \tag{5}$$

thus note that $p(y_t|x_t)$ is our conditional probability given in (1), $p(x_t|x_{t-1})$ is a transition probability given in (2) and the last part $p(x_{t-1}, y_{1:t-1})$ is our recursion step. Thus the forward algorithm can calculate the probability of a certain state by far less computation than marginalizing over previous state sequences.

Kalman filter

The Kalman Filter is another way of determining a state based on a proxy observations but unlike the HMM the system is continuously changing. Also The Kalman Filter doesn't need to keep track of the history, but uses only the current state. The way it works is by combining our current information of how a system changes with some observation regarding the state. An example could be to determine the next position of a flight. If we have a flight that moves with a certain velocity at a certain position, we can *guess* where it's going to be in the next state, but this is affected by uncertainty such at wind resistance, air flows, sudden turbulence etc. This is known as a dynamic model as is the state probability is denoted $p(x_t|x_{t-1})$. Let's assume that we also can observe the flight's position through the GPS system. The GPS system is also characterised by inaccuracy and is known as an aberration model and is denoted $p(y_t|x_t)$. In the Kalman Filter we combine both models to get a more precise estimate.

Since both of the sources are characterized by some degree of uncertainty, we assume that they are Gaussian distributed. This means that each source of information has a mean μ, which is the most likely next state, and a variance σ^2, which is the uncertainty. Another assumption is, that our sources are correlated. In the flight example this makes sense because the velocity of the flight would influence the future position. To model the problem as a Gaussian, we denote our best estimate μ, as $\hat{x}_t$, and the covariance matrix as P_t

$$\hat{x}_t = \begin{bmatrix} x \\ y \end{bmatrix} \tag{6}$$

$$P_t = \begin{bmatrix} var(X) & cov(X,Y) \\ cov(X,Y) & var(Y) \end{bmatrix} \tag{7}$$

to predict the state, we use a prediction function denoted by F_t:

$$\hat{x}_t = F_t \hat{x}_{t-1} \tag{8}$$

And since $Cov(Kx) = KCov(x)K^T = KP_tK^T$ the new covariance will be:

$$P_t = F_t P_{t-1} F_t^T \tag{9}$$

And because we know the our model is a Gaussian, we can rewrite x_t as:

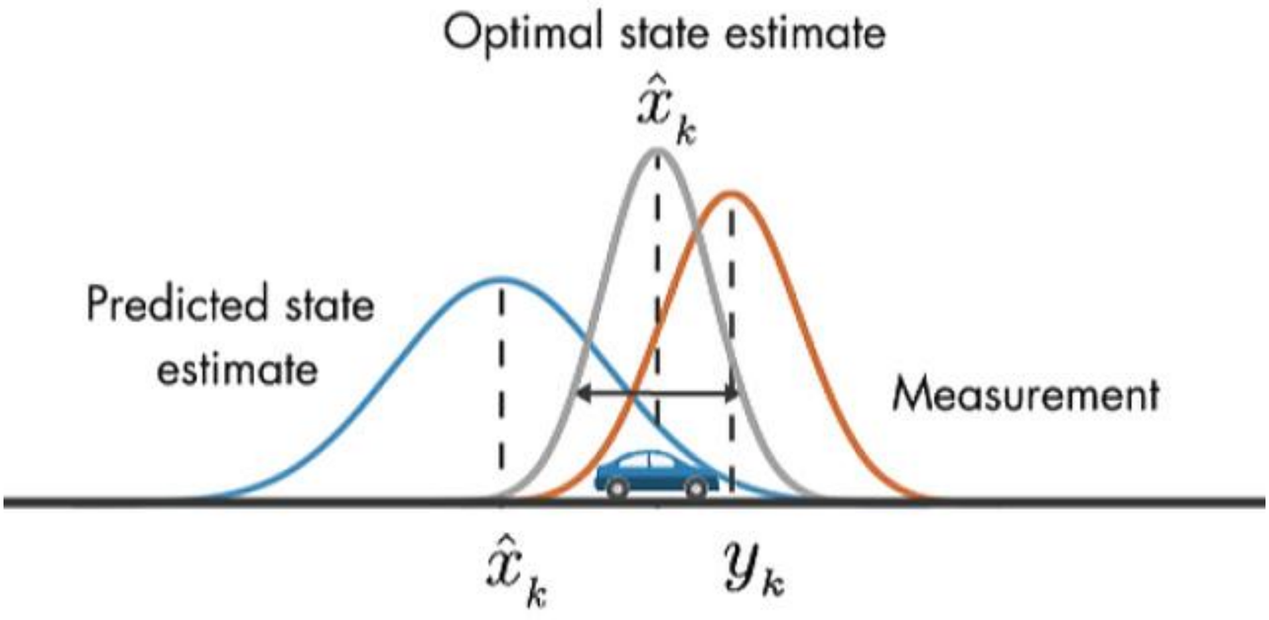
$$x_t = p(x_t|x_{t-1}) = N(Fx_{t-1}, Q_t) \tag{10}$$

Note that Q_t denotes random noise, external uncertainty and is a covariance, which again affects where the data is positioned. We can do the same as above in the observation model, but here we will model the observation with a matrix, H_t. Thus we get:

$$y_t = p(y_t|x_t) = N(Hx_t, R_t) \tag{11}$$

where R_t denotes external uncertainty.

By combining the prediction with the observation, we will first use the prediction step, and then correct it by the observation step afterwards. A very important property for this to work is, that two Gaussians multiplied the product of the Gaussians is another Gaussian. As shown in the picture below, the Kalman Filter combines the predicted state estimate, with a measurement to predict most likely state.



Note that in the above examples, we have used only two dimensional data for easy understanding, but a Kalman Filter can work with any number of dimensions.

Summary: Hidden Markov Model vs. Kalman Filter

In both the HMM and the Kalman Filter we are trying to predict an unobservable state that changes over time. In the HMM the state can only be a few distinct classes and the probability of the movement between these states are known before hand (transition probabilities). Compare this to the Kalman Filter where we are trying to predict a continuous state, were we are assuming the model is moving somewhat linear and is Gaussian distribituted. If not a Particle Filter can be used. It terms of computational expense a naive HMM runs in exponential time, but by implementing the forward algorithm this can be reduced to linear. The Kalman Filter on the other hand get it's speed from not having a history (thus not following a chain like HMM) and all of the computations is done by relatively fast matrix operations.

Conclusion: Both models uses a very similar idea, but their applications and how they solve the problem are very different.

2. Explain how to estimate the coefficients a_1 and a_2 of the auto-regressive model (AR model)

The AR-model is defined as follows:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \varepsilon_t \tag{12}$$

where y is observed data and $y_1, y_2, \ldots, y_t \in \mathbb{R}$, the a_l's are the parameters of the model and l is from 0 to the order of the model, in this case 2, and ε is noise. As you can see the output of the model depends on its own previous values and a stochastic variable. To find the coefficients of the model we will use the Yule-Walker equation in terms of the autocovariance function. If given a time series y of t, we can estimate the covariance of the time series, and since we have a relationsship between the covariance and the auto regressive model parameters trough the Yule-Walker equations, we can solve these to find the unknown parameters a.

We begin by rewriting equation (13) to the following form:

$$\sum_{i=0}^{m} a_i y_{t-i} = \varepsilon_t \tag{13}$$

where m is the order of the AR-model. Note that we assume $a_0 = 1$. Thus the model's output is a response to a white noise input. Next we multiply each side of the equation by x_{m-i} and take the expectation:

$$\sum_{i=0}^{m} a_i \mathrm{E}[y_{t-i} y_{t-l}] = \mathrm{E}[\varepsilon_t \varepsilon_{t-l}] \tag{14}$$

The first expectation is the definition of the autocovariance function. If the look at the other expectation, the white noise is independant from time sample to time sample this expectation goes to 0 when $l > 0$ and it goes to the variance σ^2 when $l = 0$. Thus we have the following relationsship:

$$\sum_{i=0}^{m} a_i r_{yy}[l-k] = \begin{cases} 0, & \text{if } l > 0 \\ \sigma^2, & \text{if } l = 0 \end{cases} \tag{15}$$

Next we look at $l > 0$ which can be rewritten as

$$\sum_{i=0}^{m} a_i r_{yy}[l-k] = -r_{xx}[l] \tag{16}$$

Which can be written in a vector form:

$$\begin{array}{c} l=1 \\ l=2 \\ \\ l=m \end{array} \begin{bmatrix} r_{yy}[0] & r_{yy}[-1] & \ldots & r_{yy}[1-m] \\ r_{yy}[1] & r_{yy}[0] & \ldots & r_{yy}[2-m] \\ \vdots & \vdots & \vdots & \vdots \\ r_{yy}[m-1] & r_{yy}[m-2] & \ldots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix} = - \begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[m] \end{bmatrix} \tag{17}$$

Thus we see, that if we know the autocovariance of the proces, we can solve for a_l and we can obtain σ^2 when $l = 0$.

In the above we have calculated the parameters of the autoregression model through the Yule-Walker equations, but there are many ways to do this. Other ways could have been to utilize maximum likelihood estimation or take a Bayesian approach.

Conclusion

All of the above models are closely related, and they all rely on recursion or *previous data*. They are all known as *State-Space Models*, which is a common term to describe a system where we recreate state variables from the measured input-output data. All of the models however have different applications and when choosing how to analyze your data, you should pick according to the nature of both your problem and your data.