# 02457 Non-Linear Signal Processing, Exercise 12: Dynamic linear models

This exercise is based on Christopher Bishop: *Machine Learning and Pattern Recognition*, sections 3.3, 13.1, 13.3, and on the "Note for week 12", uploaded with the slides for lecture 12 (in the Campusnet filesharing).

The aim of the exercise is to illustrate dynamically updated linear learning and issues that arise in realtime application of learned models.

## Bayesian linear models

Let $\mathcal{D} = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), ..., (\mathbf{x}_N, t_N)\}$ be a data set of $N$ samples with $\mathbf{x} \in \mathbb{R}^d$. We will focus on function approximation and assume the generative model is linear

$$t = \mathbf{w}^\top \mathbf{x} + \epsilon \tag{1}$$

with $\epsilon \sim \mathcal{N}(0, \beta^{-1})$, i.e., normally distributed white noise with precision $\beta$. With these assumptions the likelihood function becomes

$$p(\mathcal{D}|\mathbf{w}, \beta) = \left(\sqrt{\frac{\beta}{2\pi}}\right)^N \exp\left(-\frac{\beta}{2}\sum_{n=1}^N (t_n - \mathbf{w}^\top \mathbf{x}_n)^2\right) \tag{2}$$

As in exercise 4 we will assign a standard Gaussian prior to the weights $\mathbf{w} \sim \mathcal{N}(0, \alpha^{-1}\mathbf{I})$, where $\mathbf{I}$ is a $d$-dimensional unit matrix, leading to the posterior distribution

$$\begin{aligned} p(\mathbf{w}|\alpha, \beta, \mathcal{D}) &= \frac{p(\mathcal{D}|\mathbf{w}, \beta) p(\mathbf{w}|\alpha)}{p(\mathcal{D}|\alpha, \beta)} \\ &\propto \left(\sqrt{\frac{\beta}{2\pi}}\right)^N \exp\left(-\frac{\beta}{2}\sum_{n=1}^N (t_n - \mathbf{w}^\top \mathbf{x}_n)^2\right) \left(\sqrt{\frac{\alpha}{2\pi}}\right)^d \exp\left(-\frac{\alpha}{2}||\mathbf{w}||^2\right). \end{aligned}$$

The posterior is a product of two normal probability density functions. Combining the exponents we obtain a quadratic form in $\mathbf{w}$, hence again a normal distribution. For such a product there is a combination rule that we will need again below: The product between $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, is proportional to $\mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ with mean vector and covariance matrix given by,

$$\begin{aligned} \boldsymbol{\mu}_p &= \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1} \left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right) \\ \boldsymbol{\Sigma}_p &= \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}. \end{aligned} \tag{3}$$

In this case the prior is given by $\boldsymbol{\mu}_2 \equiv \boldsymbol{\mu}_{\text{prior}} = \mathbf{0}$ and $\boldsymbol{\Sigma}_2 \equiv \boldsymbol{\Sigma}_{\text{prior}} = \alpha^{-1}\mathbf{I}$. For the likelihood a bit of algebra leads to,

$$\begin{aligned} \boldsymbol{\mu}_1 &\equiv \left(\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top\right)^{-1} \sum_{n=1}^N \mathbf{x}_n t_n \\ \boldsymbol{\Sigma}_1 &\equiv \left(\beta \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top\right)^{-1}. \end{aligned} \tag{4}$$

Hence the posterior mean vector and covariance matrix are found as

$$\begin{aligned}
\boldsymbol{\mu}_p &\equiv \left(\alpha\mathbf{I} + \sum_{n=1}^{N} \mathbf{x}_n\mathbf{x}_n^\top\right)^{-1} \sum_{n=1}^{N} \mathbf{x}_n t_n \\
\boldsymbol{\Sigma}_p &\equiv \left(\alpha\mathbf{I} + \beta\sum_{n=1}^{N} \mathbf{x}_n\mathbf{x}_n^\top\right)^{-1} . \qquad (5)
\end{aligned}$$

The predictive density is computed as

$$p(t_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}) = \int p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w}. \qquad (6)$$

This is again a normal distribution with mean and variance,

$$\begin{aligned}
\boldsymbol{\mu}_{t_{N+1}} &= \boldsymbol{\mu}_p^\top \mathbf{x}_{N+1}, \\
\sigma^2_{t_{N+1}} &= \beta^{-1} + \mathbf{x}_{N+1}^\top \boldsymbol{\Sigma}_p \mathbf{x}_{N+1}. \qquad (7)
\end{aligned}$$

## Dynamic Bayesian models

The basic assumption in the previous derivation is that the parameter vector is stationary. In a dynamic setting we relax this assumption and assume that $\mathbf{w}_n$ is changing as data arrives. A natural prior is then the Markovian random walk $\mathbf{w}_n = \mathbf{w}_{n-1} + \boldsymbol{\nu}_n$ with $\boldsymbol{\nu}_n \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$,

$$p(\mathbf{w}_n|\mathbf{w}_{n-1}, \alpha) = \left(\sqrt{\frac{\alpha}{2\pi}}\right)^d \exp\left(-\frac{\alpha}{2}||\mathbf{w}_n - \mathbf{w}_{n-1}||^2\right),$$

where a high value of precision parameter $\alpha$ means small changes in $\mathbf{w}_n$ as time progresses.

To simplify the notation, let us define $\mathbf{z}_n = (t_n, \mathbf{x}_n)$ and let us denote the set of all data observed until $n$ by $\mathbf{z}_{1:n}$. We are interested in the 'dynamic posterior' $p(\mathbf{w}_n|\mathbf{z}_{1:n})$ and it turns out, it can be computed in a recursive manner. For the proportional quantity, the joint density $p(\mathbf{w}_n, \mathbf{z}_{1:n})$, a forward recursion can be derived,

$$\begin{aligned}
p(\mathbf{w}_n, \mathbf{z}_{1:n}) &= \int p(\mathbf{w}_n, \mathbf{w}_{n-1}, \mathbf{z}_{1:n})d\mathbf{w}_{n-1} \qquad (8) \\
&= p(\mathbf{z}_n|\mathbf{w}_n)\int p(\mathbf{w}_n|\mathbf{w}_{n-1})p(\mathbf{w}_{n-1}, \mathbf{z}_{1:(n-1)})d\mathbf{w}_{n-1}. \qquad (9)
\end{aligned}$$

Here $p(\mathbf{z}_n|\mathbf{w}_n) = p(\mathbf{z}_n|\mathbf{w}_n, \beta)$ is the observation likelihood, while $p(\mathbf{w}_n|\mathbf{w}_{n-1}) = p(\mathbf{w}_n|\mathbf{w}_{n-1}, \alpha)$ is Markov prior. As $p(\mathbf{w}_{n-1}, \mathbf{z}_{1:(n-1)})$ is the sought joint distribution evaluated at the previous time step $n-1$, we see that by performing a single $d$-dimensional integral (wrt. $\mathbf{w}_{n-1}$) and subsequent multiplication by $p(\mathbf{z}_n|\mathbf{w}_n)$ we arrive at the 'updated' joint distribution. The posterior distribution of $\mathbf{w}_n$, in turn, can be obtained by normalization,

$$p(\mathbf{w}_n|\mathbf{z}_{1:n}) = \frac{p(\mathbf{w}_n, \mathbf{z}_{1:n})}{\int p(\mathbf{w}_n, \mathbf{z}_{1:n})d\mathbf{w}_n}. \qquad (10)$$

Here we recognize the normalization constant is the model likelihood $p(\mathbf{z}_{1:n}|\text{Model}) = \int p(\mathbf{w}_n, \mathbf{z}_{1:n})d\mathbf{w}_n$, hence it is a by product of the 'forward recursion'.

For the linear model we analyzed above, we get specifically,

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{w}_n,\beta) &= p(t_n|\mathbf{w}_n,\mathbf{x}_n,\beta)p(\mathbf{x}_n) = \sqrt{\frac{\beta}{2\pi}}\exp\left(-\frac{\beta}{2}(t_n-\mathbf{w}^\top\mathbf{x}_n)^2\right)p(\mathbf{x}_n) \\ p(\mathbf{w}_n|\mathbf{w}_{n-1},\alpha) &= \left(\sqrt{\frac{\alpha}{2\pi}}\right)^d \exp\left(-\frac{\alpha}{2}||\mathbf{w}_n-\mathbf{w}_{n-1}||^2\right). \end{aligned}$$

The recursion starts as $p(\mathbf{w}_1,\mathbf{z}_1) \propto p(\mathbf{z}_1|\mathbf{w}_1)p(t_1|\mathbf{x}_1,\mathbf{w}_1)p(\mathbf{x}_1)$, hence the quantity of interest starts out being proportional to a normal density function in terms of $\mathbf{w}_1$. We also see that in order to compute the update for $p(\mathbf{w}_2,\mathbf{z}_{1:2})$ we perform an integral over the product of two normal distributions

$$p(\mathbf{w}_2,\mathbf{z}_{1:2}) = p(\mathbf{z}_2|\mathbf{w}_2,\beta)\int p(\mathbf{w}_2|\mathbf{w}_1,\alpha)p(\mathbf{w}_1,\mathbf{z}_1)d\mathbf{w}_1. \tag{11}$$

The result of this integral is a again a normal distribution and as this is followed by multiplication by the local likelihood, also an un-normalized normal density, we obtain a $p(\mathbf{w}_2,\mathbf{z}_{1:2})$ which itself is proportional to a normal distribution. It then follows by induction that all the following terms $p(\mathbf{w}_n,\mathbf{z}_{1:n})$ are (un-normalized) normal density functions, in case of the linear model.

Tracking the means and covariances of these un-normalized density functions, which involves the product rule (Eq. 3) two times, we get a message passing scheme for mean and covariance of the un-normalized posterior $p(\mathbf{w}_n,\mathbf{z}_{1:n})$

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{w},n} &= \left(\left(\boldsymbol{\Sigma}^{-1}_{\mathbf{w},n-1}+\alpha^{-1}\right)^{-1}+\beta\mathbf{x}_n\mathbf{x}_n^\top\right)^{-1}\left(\boldsymbol{\Sigma}^{-1}_{\mathbf{w},n-1}\boldsymbol{\mu}_{\mathbf{w},n-1}+\beta t_n\mathbf{x}_n\right) \\ \boldsymbol{\Sigma}_{\mathbf{w},n} &= \left(\left(\boldsymbol{\Sigma}^{-1}_{\mathbf{w},n-1}+\alpha^{-1}\right)^{-1}+\beta\mathbf{x}_n\mathbf{x}_n^\top\right)^{-1} \end{aligned} \tag{12}$$

At any given time we can use the predictive means in Eq. 7 to find the estimator and the associated uncertainty.

## Checkpoint 12.1

Use the matlab script main12a.m to create a sequence of weights for a dynamic linear model, using Eq. (8)with a specified 'teacher' $\alpha_0$. Inspect the sequence of weights, why is the magnitude increasing?. Generate a similar test sequence also based on $\alpha_0$.

Based on the two sequences of weights we generate sequences of training and test observations $(t_n,\mathbf{x}_n)$. We simulate on random i.i.d. normal input and additive noise with precision $\beta$. Using the dynamic updates we will make predictions on train and test sets for ranges of $\alpha$ and $\beta$. Exaplin how we can use training set predictions to estimate the optimal combination of $(\alpha,\beta)$.

## Checkpoint 12.2

Use the matlab script main12b.m to investigate the dynamic linear model on the sun spot data for fixed $(\alpha,\beta)$. Inspect the deviations of the dynamic linear model from the global linear model. Is the Markov random walk hypothesis, i.e., a non-stationary model with 'growing weights', useful in the sunspot case?.

## Checkpoint 12.3

We finally use a linear model to classify sounds in a realtime setting. Use the matlab script `main12c.m` to acquire sound in blocks of length $T$. We will use linear model to classify sounds in two classes labeled by targets $t = +/-1$.

The basic features are derived from the frequency content (using Fourier a window transform and principal component based dimensional reduction).

We first pre-train the classifier by creating two sound sequences, e.g., an 'impact sound' from hitting the table with a pen and a clapping sound.

Following the training, start executing new sounds for 60 seconds and inspect the classifications in the Matlab window.

Lars Kai Hansen, November 2016.