Extending the Bayesian Deep Learning Method MultiSWAG

Scott Brownlie

Master of Science
School of Informatics
University of Edinburgh
2021

Abstract

This skeleton demonstrates how to use the `infthesis` style for MSc dissertations in Artificial Intelligence, Cognitive Science, Computer Science, Data Science, and Informatics. It also emphasises the page limit, and that you must not deviate from the required style. The file `skeleton.tex` generates this document and can be used as a starting point for your thesis. The abstract should summarise your report and fit in the space on the first page.

Acknowledgements

Any acknowledgements go here.

Table of Contents

Chapter 1

Introduction

The preliminary material of your report should contain:

- The title page.
- An abstract page.
- Optionally an acknowledgements page.
- The table of contents.

As in this example `skeleton.tex`, the above material should be included between:

```
\begin{preliminary}
    ...
\end{preliminary}
```

This style file uses roman numeral page numbers for the preliminary material.

The main content of the dissertation, starting with the first chapter, starts with page 1. ***The main content must not go beyond page 40.***

The report then contains a bibliography and any appendices, which may go beyond page 40. The appendices are only for any supporting material that's important to go on record. However, you cannot assume markers of dissertations will read them.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Over length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

Chapter 2

Factor Analysis

2.1 Background

FA is a latent variable model which generates observations $\theta \in \mathbb{R}^d$ as follows. First, a latent vector $\mathbf{h} \in \mathbb{R}^K$, for some $K < d$, is sampled from $p(\mathbf{h}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Next, $\mathbf{h}$ is transformed onto a K-dimensional linear subspace of $\mathbb{R}^d$ by left-multiplying it by a *factor loading* matrix $\mathbf{F} \in \mathbb{R}^{d \times K}$. The origin of this subspace is then shifted by adding a bias term $\mathbf{c} \in \mathbb{R}^d$. Finally, the data is perturbed by adding some zero mean Gaussian noise $\varepsilon \in \mathbb{R}^d$ sampled from $\mathcal{N}(\mathbf{0}, \Psi)$, where Ψ is a $d \times d$ diagonal matrix [1]. Putting all this together, an observation $\theta \in \mathbb{R}^d$ is generated according to

$$\theta = \mathbf{F}\mathbf{h} + \mathbf{c} + \varepsilon. \tag{2.1}$$

In this context, an observation θ is the parameter vector of a neural network.

It follows that, given $\mathbf{h}$, the observations θ are Gaussian distributed with mean $\mathbf{F}\mathbf{h} + \mathbf{c}$ and covariance Ψ [1]. Formally,

$$p(\theta|\mathbf{h}) = \mathcal{N}\Big(\mathbf{F}\mathbf{h} + \mathbf{c}, \Psi\Big) = \frac{1}{\sqrt{(2\pi)^d |\Psi|}} \exp\Big(-\frac{1}{2}(\theta - \mathbf{F}\mathbf{h} - \mathbf{c})^\intercal \Psi^{-1} (\theta - \mathbf{F}\mathbf{h} - \mathbf{c})\Big), \tag{2.2}$$

where $|\Psi|$ is the *determinant* of Ψ. From [1], integrating $p(\theta|\mathbf{h})$ over $\mathbf{h}$ gives the marginal distribution

$$p(\theta) = \mathcal{N}\big(\mathbf{c}, \mathbf{F}\mathbf{F}^\intercal + \Psi\big). \tag{2.3}$$

The parameters of the model are $\mathbf{c}, \mathbf{F}$ and Ψ. The value of $\mathbf{c}$ which maximises the likelihood of the observed data is the empirical mean of the observations [1], which in this case is θ_{SWA}. Having set the bias term, an expectation-maximisation (EM) or singular value decomposition (SVD) algorithm can find the maximum likelihood

estimates of $\mathbf{F}$ and Ψ [1]. However, both methods require storing all the observations in memory, making them impractical for high-dimensional data, such as the parameter vectors of deep neural networks. Two alternative online algorithms are presented in Sections 2.2 and 2.3.

2.2 Online Stochastic Gradient Ascent

In situations where learning latent variable models with the classic EM algorithm is slow, [1] suggests optimising the log-likelihood of the model parameters via gradient methods. Since FA is a latent variable model, this approach can be applied here.

In this case, the log-likelihood of the parameters $\mathbf{F}$ and Ψ given the observed data, $\theta_1, \ldots, \theta_T$, is

$$L(\mathbf{F}, \Psi) = \frac{1}{T} \sum_{t=1}^{T} \log p(\theta_t | \mathbf{F}, \Psi). \tag{2.4}$$

The partial derivatives of the log-likelihood with respect to $\mathbf{F}$ and Ψ are therefore

$$\nabla_{\mathbf{F},\Psi} L(\mathbf{F}, \Psi) = \frac{1}{T} \sum_{t=1}^{T} \nabla_{\mathbf{F},\Psi} \log p(\theta_t | \mathbf{F}, \Psi). \tag{2.5}$$

Computing $\nabla_{\mathbf{F},\Psi} L(\mathbf{F}, \Psi)$ in full would require a pass over all observations, $\theta_1, \ldots, \theta_T$. However, a stochastic gradient algorithm can be used instead, as long as the expectation of the sample derivatives is proportional to $\nabla_{\mathbf{F},\Psi} L(\mathbf{F}, \Psi)$. By using $\nabla_{\mathbf{F},\Psi} \log p(\theta_t | \mathbf{F}, \Psi)$, $t = 1, \ldots, T$, as the sample derivatives, this condition clearly holds. Hence, as long as the partial derivatives $\nabla_{\mathbf{F},\Psi} \log p(\theta_t | \mathbf{F}, \Psi)$ can be computed efficiently, they can be used in conjunction with SGD or any of its variants to optimise $\mathbf{F}$ and Ψ online.

By adapting an argument for general latent variable models in [1] to FA, the re-

quired sample derivatives can be written as

$$\begin{aligned}
\nabla_{\mathbf{F},\Psi} \log p(\theta_t|\mathbf{F},\Psi) &= \frac{1}{p(\theta_t|\mathbf{F},\Psi)} \nabla_{\mathbf{F},\Psi} p(\theta_t|\mathbf{F},\Psi) \\
&= \frac{1}{p(\theta_t|\mathbf{F},\Psi)} \nabla_{\mathbf{F},\Psi} \int_{\mathbf{h}_t} p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \\
&= \frac{1}{p(\theta_t|\mathbf{F},\Psi)} \int_{\mathbf{h}_t} \nabla_{\mathbf{F},\Psi} p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \\
&= \frac{1}{p(\theta_t|\mathbf{F},\Psi)} \int_{\mathbf{h}_t} p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \nabla_{\mathbf{F},\Psi} \log p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \\
&= \int_{\mathbf{h}_t} \frac{p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi)}{p(\theta_t|\mathbf{F},\Psi)} \nabla_{\mathbf{F},\Psi} \log p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \\
&= \int_{\mathbf{h}_t} p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi) \nabla_{\mathbf{F},\Psi} \log p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \\
&= \mathbb{E}_{p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi)} \left[\nabla_{\mathbf{F},\Psi} \log p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) \right].
\end{aligned} \tag{2.6}$$

This is as far as the derivation in [1] goes. However, given the form of the FA model, it is possible to manipulate the sample derivatives further. In particular, using the fact that $\mathbf{h}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ is independent from $\mathbf{F}$ and Ψ,

$$\begin{aligned}
\nabla_{\mathbf{F},\Psi} \log p(\theta_t,\mathbf{h}_t|\mathbf{F},\Psi) &= \nabla_{\mathbf{F},\Psi} \log \left(p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi) p(\mathbf{h}_t|\mathbf{F},\Psi) \right) \\
&= \nabla_{\mathbf{F},\Psi} \log \left(p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi) p(\mathbf{h}_t) \right) \\
&= \nabla_{\mathbf{F},\Psi} \left(\log p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi) + \log p(\mathbf{h}_t) \right) \\
&= \nabla_{\mathbf{F},\Psi} \log p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi).
\end{aligned} \tag{2.7}$$

Substituting Equation (2.7) into Equation (2.6),

$$\nabla_{\mathbf{F},\Psi} \log p(\theta_t|\mathbf{F},\Psi) = \mathbb{E}_{p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi)} \left[\nabla_{\mathbf{F},\Psi} \log p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi) \right]. \tag{2.8}$$

Note that $p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi)$ is just the Gaussian distribution in Equation (2.2), given $\mathbf{F}$ and Ψ. Hence, substituting $\mathbf{c} = \theta_{\text{SWA}}$ into Equation (2.2) and applying the logarithm,

$$\begin{aligned}
\log p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi) &= -\frac{1}{2}(\theta_t - \mathbf{F}\mathbf{h}_t - \theta_{\text{SWA}})^\intercal \Psi^{-1} (\theta_t - \mathbf{F}\mathbf{h}_t - \theta_{\text{SWA}}) - \frac{1}{2}\log|\Psi| - \frac{d}{2}\log 2\pi \\
&= -\frac{1}{2}(\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^\intercal \Psi^{-1} (\mathbf{d}_t - \mathbf{F}\mathbf{h}_t) - \frac{1}{2}\log|\Psi| - \frac{d}{2}\log 2\pi,
\end{aligned} \tag{2.9}$$

where $\mathbf{d}_t = \theta_t - \theta_{\text{SWA}}$. This is convenient, since Equation (2.9) can be differentiated with respect to both $\mathbf{F}$ and Ψ. Of course, this requires access to θ_{SWA}, which is not available during training. As a compromise - and following the approach of the SWAG covariance approximation - θ_{SWA} can be replaced by the running average of the neural network's parameter vectors.

2.2.1 Partial derivatives with respect to $\mathbf{F}$

From [4], for any symmetric matrix $\mathbf{W}$,

$$\nabla_{\mathbf{A}}(\mathbf{x}-\mathbf{A}\mathbf{s})^{\intercal}\mathbf{W}(\mathbf{x}-\mathbf{A}\mathbf{s}) = -2\mathbf{W}(\mathbf{x}-\mathbf{A}\mathbf{s})\mathbf{s}^{\intercal}. \tag{2.10}$$

Hence, differentiating Equation (2.9) with respect to $\mathbf{F}$ gives

$$\nabla_{\mathbf{F}} \log p(\theta_t|\mathbf{h}_t,\mathbf{F},\Psi) = \Psi^{-1}(\mathbf{d}_t-\mathbf{F}\mathbf{h}_t)\mathbf{h}_t^{\intercal}. \tag{2.11}$$

It then follows from Equation (2.8) that $\nabla_{\mathbf{F}} \log p(\theta_t|\mathbf{F},\Psi)$ is the expected value of $\Psi^{-1}(\mathbf{d}_t-\mathbf{F}\mathbf{h}_t)\mathbf{h}_t^{\intercal}$ over the distribution $p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi)$. Letting $\mathbb{E}[\cdot]$ denote $\mathbb{E}_{p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi)}[\cdot]$ to simplify the notation,

$$\begin{aligned}\nabla_{\mathbf{F}} \log p(\theta_t|\mathbf{F},\Psi) &= \mathbb{E}\left[\Psi^{-1}(\mathbf{d}_t-\mathbf{F}\mathbf{h}_t)\mathbf{h}_t^{\intercal}\right] \\ &= \Psi^{-1}\left(\mathbb{E}\left[\mathbf{d}_t\mathbf{h}_t^{\intercal}\right]-\mathbb{E}\left[\mathbf{F}\mathbf{h}_t\mathbf{h}_t^{\intercal}\right]\right) \\ &= \Psi^{-1}\left(\mathbf{d}_t\mathbb{E}\left[\mathbf{h}_t^{\intercal}\right]-\mathbf{F}\mathbb{E}\left[\mathbf{h}_t\mathbf{h}_t^{\intercal}\right]\right).\end{aligned} \tag{2.12}$$

From the E-step of the EM algorithm for FA in [1], $p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi) \propto \mathcal{N}(\mathbf{m}_t,\Sigma)$, where

$$\Sigma = (\mathbf{I}+\mathbf{F}^{\intercal}\Psi^{-1}\mathbf{F})^{-1} \quad \text{and} \quad \mathbf{m}_t = \Sigma\mathbf{F}^{\intercal}\Psi^{-1}\mathbf{d}_t. \tag{2.13}$$

Hence, using identities from [4],

$$\mathbb{E}\left[\mathbf{h}_t^{\intercal}\right] = \mathbf{m}_t^{\intercal} \quad \text{and} \quad \mathbb{E}\left[\mathbf{h}_t\mathbf{h}_t^{\intercal}\right] = \Sigma+\mathbf{m}_t\mathbf{m}_t^{\intercal}. \tag{2.14}$$

Finally, substituting Equation (2.14) into Equation (2.12),

$$\nabla_{\mathbf{F}} \log p(\theta_t|\mathbf{F},\Psi) = \Psi^{-1}\left(\mathbf{d}_t\mathbf{m}_t^{\intercal}-\mathbf{F}(\Sigma+\mathbf{m}_t\mathbf{m}_t^{\intercal})\right). \tag{2.15}$$

2.2.2 Partial derivatives with respect to Ψ

In order to differentiate Equation (2.9) with respect to Ψ, it helps to use the fact that Ψ is diagonal. First consider $\mathbf{X}^{-1} = \text{diag}(\frac{1}{x_1},\ldots,\frac{1}{x_d})$ and $\mathbf{a} = (a_1,\ldots,a_d)^{\intercal}$. Then

$$\mathbf{a}^{\intercal}\mathbf{X}^{-1}\mathbf{a} = \sum_{i=1}^{d}\frac{a_i^2}{x_i}, \tag{2.16}$$

and so

$$\frac{\partial}{\partial x_i}\mathbf{a}^{\intercal}\mathbf{X}^{-1}\mathbf{a} = \frac{-a_i^2}{x_i^2} \tag{2.17}$$

for $i = 1, \ldots, d$. Since the partial derivatives of Equation (2.16) with respect to the off-diagonal entries of $\mathbf{X}$ are zero,

$$\begin{aligned} \nabla_{\mathbf{X}}(\mathbf{a}^\intercal \mathbf{X}^{-1}\mathbf{a}) &= \operatorname{diag}\Big(\frac{-a_1^2}{x_1^2}, \ldots, \frac{-a_d^2}{x_d^2}\Big) \\ &= -\operatorname{diag}\big(\operatorname{diag}(\mathbf{X}^{-2}) \odot \mathbf{a}^2\big), \end{aligned} \tag{2.18}$$

where $\odot$ denotes the element-wise matrix product (with broadcasting, if applicable) and the square of a vector is applied element-wise. Also, when applied to a d-length vector, $\operatorname{diag}(\cdot)$ represents the $d \times d$ diagonal matrix with the vector on its diagonal, and when applied to a $d \times d$ matrix, $\operatorname{diag}(\cdot)$ represents the d-length vector consisting of the diagonal entries of the matrix. Substituting $\mathbf{X} = \Psi$ and $\mathbf{a} = \mathbf{d}_t - \mathbf{F}\mathbf{h}_t$, into Equation (2.18),

$$\nabla_{\Psi}(\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^\intercal \Psi^{-1}(\mathbf{d}_t - \mathbf{F}\mathbf{h}_t) = -\operatorname{diag}\big(\operatorname{diag}(\Psi^{-2}) \odot (\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^2\big). \tag{2.19}$$

Also, using the identity $\nabla_{\mathbf{X}} \log|\mathbf{X}| = \mathbf{X}^{-\intercal}$ from [4] and the fact that $\Psi^{-\intercal} = \Psi^{-1}$,

$$\nabla_{\Psi} \log|\Psi| = \Psi^{-1}. \tag{2.20}$$

Hence, the partial derivatives of Equation (2.9) with respect to Ψ are

$$\nabla_{\Psi} \log p(\theta_t|\mathbf{h}_t, \mathbf{F}, \Psi) = \frac{1}{2}\operatorname{diag}\big(\operatorname{diag}(\Psi^{-2}) \odot (\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^2\big) - \frac{1}{2}\Psi^{-1}. \tag{2.21}$$

Again, letting $\mathbb{E}[\cdot]$ denote $\mathbb{E}_{p(\mathbf{h}_t|\theta_t,\mathbf{F},\Psi)}[\cdot]$, it follows from Equation (2.8) that

$$\begin{aligned} 2 \cdot \nabla_{\Psi} \log p(\theta_t|\mathbf{F}, \Psi) &= \mathbb{E}\big[\operatorname{diag}\big(\operatorname{diag}(\Psi^{-2}) \odot (\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^2\big) - \Psi^{-1}\big] \\ &= \operatorname{diag}\Big(\mathbb{E}\big[\operatorname{diag}(\Psi^{-2}) \odot (\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^2\big]\Big) - \mathbb{E}\big[\Psi^{-1}\big] \\ &= \operatorname{diag}\Big(\operatorname{diag}(\Psi^{-2}) \odot \mathbb{E}\big[(\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^2\big]\Big) - \Psi^{-1}. \end{aligned} \tag{2.22}$$

Now, expanding the expectation inside Equation (2.22) and substituting in the expressions from Equation (2.14),

$$\begin{aligned} \mathbb{E}\big[(\mathbf{d}_t - \mathbf{F}\mathbf{h}_t)^2\big] &= \mathbb{E}\big[\mathbf{d}_t \odot \mathbf{d}_t\big] - 2\mathbb{E}\big[\mathbf{d}_t \odot (\mathbf{F}\mathbf{h}_t)\big] + \mathbb{E}\big[(\mathbf{F}\mathbf{h}_t) \odot (\mathbf{F}\mathbf{h}_t)\big] \\ &= \mathbf{d}_t \odot \mathbf{d}_t - 2\mathbf{d}_t \odot \big(\mathbf{F}\mathbb{E}\big[\mathbf{h}_t\big]\big) + \mathbb{E}\big[\operatorname{diag}(\mathbf{F}\mathbf{h}_t\mathbf{h}_t^\intercal\mathbf{F}^\intercal)\big] \\ &= \mathbf{d}_t \odot \mathbf{d}_t - 2\mathbf{d}_t \odot (\mathbf{F}\mathbf{m}_t) + \operatorname{diag}\big(\mathbb{E}\big[\mathbf{F}\mathbf{h}_t\mathbf{h}_t^\intercal\mathbf{F}^\intercal\big]\big) \\ &= \mathbf{d}_t \odot \mathbf{d}_t - 2\mathbf{d}_t \odot (\mathbf{F}\mathbf{m}_t) + \operatorname{diag}\big(\mathbf{F}\mathbb{E}\big[\mathbf{h}_t\mathbf{h}_t^\intercal\big]\mathbf{F}^\intercal\big) \\ &= \mathbf{d}_t \odot \mathbf{d}_t - 2\mathbf{d}_t \odot (\mathbf{F}\mathbf{m}_t) + \operatorname{diag}\big(\mathbf{F}(\Sigma + \mathbf{m}_t\mathbf{m}_t^\intercal)\mathbf{F}^\intercal\big). \end{aligned} \tag{2.23}$$

Finally, substituting Equation (2.23) into Equation (2.22) and rearranging,

$$\nabla_{\Psi} \log p(\theta_t|\mathbf{F},\Psi) = \operatorname{diag}\left(\frac{1}{2}\operatorname{diag}(\Psi^{-2}) \odot \Big(\mathbf{d}_t \odot \mathbf{d}_t - 2\mathbf{d}_t \odot (\mathbf{F}\mathbf{m}_t) + \operatorname{diag}\big(\mathbf{F}(\Sigma + \mathbf{m}_t\mathbf{m}_t^{\intercal})\mathbf{F}^{\intercal}\big)\Big)\right) - \frac{1}{2}\Psi^{-1}. \tag{2.24}$$

2.2.3 Practical implementation

In practice, storing the full $d \times d$ covariance matrix Ψ (or its inverse) would be infeasible for high-dimensional $\theta_t \in \mathbb{R}^d$. However, since Ψ is diagonal and the partial derivatives of the log-likelihood with respect to the off-diagonal are always zero, it suffices to work with the diagonal entries only. All occurrences of $d \times d$ matrices can then be removed from the gradient calculations.

First, note that the partial derivatives with respect to both $\mathbf{F}$ and Ψ depend on $\mathbf{m}_t$ and Σ from Equation (2.13), which themselves depend on Ψ^{-1}. Let $\psi = \operatorname{diag}(\Psi)$ and $\psi^{-n} = \operatorname{diag}(\Psi^{-n})$ for $n \in \mathbb{N}^+$. Then

$$\mathbf{F}^{\intercal}\Psi^{-1} = (\mathbf{F} \odot \psi^{-1})^{\intercal}. \tag{2.25}$$

Now, setting $\mathbf{C} = (\mathbf{F} \odot \psi^{-1})^{\intercal}$ and substituting into Equation (2.13), it follows that

$$\Sigma = (\mathbf{I} + \mathbf{C}\mathbf{F})^{-1} \quad \text{and} \quad \mathbf{m}_t = \Sigma\mathbf{C}\mathbf{d}_t. \tag{2.26}$$

These more efficient values can then be used in Equation (2.15), which itself can be simplified to

$$\nabla_{\mathbf{F}} \log p(\theta_t|\mathbf{F},\Psi) = \psi^{-1} \odot \big(\mathbf{d}_t\mathbf{m}_t^{\intercal} - \mathbf{F}(\Sigma + \mathbf{m}_t\mathbf{m}_t^{\intercal})\big). \tag{2.27}$$

Also, the partial derivatives with respect to Ψ in Equation (2.24) can be re-written with respect to ψ, as

$$\nabla_{\psi} \log p(\theta_t|\mathbf{F},\Psi) = \frac{1}{2}\psi^{-2} \odot \Big(\mathbf{d}_t \odot \mathbf{d}_t - 2\mathbf{d}_t \odot (\mathbf{F}\mathbf{m}_t) + \operatorname{sum}\big((\mathbf{F}(\Sigma + \mathbf{m}_t\mathbf{m}_t^{\intercal})) \odot \mathbf{F},\ \dim = 1\big)\Big) - \frac{1}{2}\psi^{-1}, \tag{2.28}$$

where $\operatorname{sum}(\cdot,\ \dim = 1)$ denotes the operation of summing along the rows of a matrix.

One final point is that the elements of ψ must be positive, since they represent variances. One way to achieve this is by using the re-parameterisation $\psi = \exp\beta$ for

some unconstrained parameter $\beta \in \mathbb{R}^d$. Then the gradient updates can be performed on β instead of ψ. Using the chain rule from calculus,

$$\begin{aligned}\nabla_\beta \log p(\theta_t|\mathbf{F},\Psi) &= \nabla_\psi \log p(\theta_t|\mathbf{F},\Psi) \odot \nabla_\beta \exp\beta \\ &= \nabla_\psi \log p(\theta_t|\mathbf{F},\Psi) \odot \exp\beta \\ &= \nabla_\psi \log p(\theta_t|\mathbf{F},\Psi) \odot \psi,\end{aligned} \tag{2.29}$$

where $\nabla_\psi \log p(\theta_t|\mathbf{F},\Psi)$ is given in Equation (2.28).

Pseudo code for online stochastic gradient ascent (SGA) for FA is given in Algorithm 1.

Algorithm 1 Online Stochastic Gradient Ascent for Factor Analysis

Input: Observation dimension d, latent dimension K, learning rate $\alpha > 0$

Initialise $\mathbf{F} \in \mathbb{R}^{d\times K}$, $\psi > \mathbf{0} \in \mathbb{R}^d$, $\overline{\theta}_0 = \mathbf{0}^d$
$\beta \leftarrow \log\psi$
for $t = 1,\ldots,T$ **do**
    Sample observation $\theta_t \in \mathbb{R}^d$
    $\overline{\theta}_t \leftarrow \overline{\theta}_{t-1} + \frac{1}{t}\left(\overline{\theta}_t - \overline{\theta}_{t-1}\right)$
    $\mathbf{d}_t \leftarrow \theta_t - \overline{\theta}_t$
    $\mathbf{C} \leftarrow (\mathbf{F} \odot \psi^{-1})^\intercal$ (with broadcasting)
    $\Sigma \leftarrow (\mathbf{I} + \mathbf{CF})^{-1}$
    $\mathbf{m}_t \leftarrow \Sigma\mathbf{C}\mathbf{d}_t$
    Compute $\nabla_\mathbf{F} \log p(\theta_t|\mathbf{F},\Psi)$ according to Equation (2.27)
    Compute $\nabla_\beta \log p(\theta_t|\mathbf{F},\Psi)$ according to Equation (2.29)
    $\mathbf{F} \leftarrow \mathbf{F} + \alpha\nabla_\mathbf{F} \log p(\theta_t|\mathbf{F},\Psi)$
    $\beta \leftarrow \beta + \alpha\nabla_\beta \log p(\theta_t|\mathbf{F},\Psi)$
    $\psi \leftarrow \exp\beta$
end for
$\theta_{\text{SWA}} \leftarrow \overline{\theta}_T$
return $\theta_{\text{SWA}}, \mathbf{F}, \psi$

2.3 Online Expectation-Maximisation

The classic EM algorithm for FA iteratively optimises the log-likelihood of $\mathbf{F}$ and Ψ by alternating "E" and "M" steps until convergence. Using properties of the Kullback-

Leibler divergence, it can be shown that

$$L(\mathbf{F}, \Psi) \geq -\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{h}_t|\theta_t)}\big[\log q(\mathbf{h}_t|\theta_t)\big] + \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{h}_t|\theta_t)}\big[\log p(\mathbf{h}_t, \theta_t|\mathbf{F}, \Psi)\big], \tag{2.30}$$

where the first and second terms on the right-hand side are called the *entropy* and *energy*, respectively, and $q(\mathbf{h}_t|\theta_t)$, $t = 1, \ldots, T$, are known as the *variational* distributions [1]. The EM algorithm optimises this lower bound on the log-likelihood with respect to $\mathbf{F}$, Ψ and also $q(\mathbf{h}_t|\theta_t)$, hence the name "variational distributions". The idea is that, by pushing up the lower bound, the log-likelihood $L(\mathbf{F}, \Psi)$ will hopefully increase as well. In fact, it is guaranteed that each iteration of EM does not decrease $L(\mathbf{F}, \Psi)$, which again follows from the properties of the Kullback-Leibler divergence [1].

2.3.1 E-step

In the batch E-step, $\mathbf{F}$ and Ψ are fixed and Equation (2.30) is maximised with respect to $q(\mathbf{h}_t|\theta_t)$, $t = 1, \ldots, T$. From [1], the optimal variational distributions are

$$q(\mathbf{h}_t|\theta_t) = p(\mathbf{h}_t|\theta_t, \mathbf{F}, \Psi) \propto \mathcal{N}(\mathbf{m}_t, \Sigma), \tag{2.31}$$

where $\mathbf{m}_t$ and Σ are given in Equation (2.26). Note that this optimisation can be performed separately for each θ_t as it is sampled, using the estimates of $\mathbf{F}$ and Ψ on iteration t. However, in batch EM all $q(\mathbf{h}_t|\theta_t)$ are updated on every iteration. This is clearly not possible in an online algorithm which discards θ_t before sampling θ_{t+1}. Therefore, as a compromise, in the online version each $q(\mathbf{h}_t|\theta_t)$ will be computed once only, on iteration t, and held fixed thereafter. The only other detail is that the batch algorithm uses θ_{SWA} to compute $\mathbf{m}_t$. As θ_{SWA} is not available during training, it will be replaced by the running average of the neural network's parameter vectors, as in the online SGA algorithm from Section 2.2.

2.3.2 M-step

In the batch M-step, $q(\mathbf{h}_t|\theta_t)$, $t = 1, \ldots, T$, are fixed and Equation (2.30) is maximised with respect to $\mathbf{F}$ and Ψ. From [1], the optimal values are

$$\mathbf{F} = \mathbf{A}\mathbf{H}^{-1}, \tag{2.32}$$

where

$$\mathbf{A} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{d}_t \mathbf{m}_t^\intercal \quad \text{and} \quad \mathbf{H} = \Sigma + \frac{1}{T}\sum_{t=1}^{T} \mathbf{m}_t \mathbf{m}_t^\intercal, \tag{2.33}$$

and

$$\Psi = \operatorname{diag}\left(\operatorname{diag}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{d}_t\mathbf{d}_t^\intercal - 2\mathbf{F}\mathbf{A}^\intercal + \mathbf{F}\mathbf{H}\mathbf{F}^\intercal\right)\right). \tag{2.34}$$

Note that this optimisation involves summing over $t = 1,\ldots,T$. Moreover, on each iteration all components of the sums in Equation (2.33) and Equation (2.34) are updated. As in the E-step, updating all components is not possible in an online algorithm. Therefore, in the online version these sums will be updated incrementally on each iteration.

2.3.3 Practical implementation

Let $\overline{\mathbf{A}}_t$ and $\overline{\mathbf{H}}_t$ be the estimates of $\mathbf{A}$ and $\mathbf{H}$ from Equation (2.33), respectively, after iteration t of online EM. That is,

$$\overline{\mathbf{A}}_t = \frac{1}{t}\sum_{i=1}^{t}\mathbf{d}_i\mathbf{m}_i^\intercal \quad \text{and} \quad \overline{\mathbf{H}}_t = \Sigma + \frac{1}{t}\sum_{i=1}^{t}\mathbf{m}_i\mathbf{m}_i^\intercal. \tag{2.35}$$

Then the estimates of $\mathbf{F}$ and Ψ on iteration t become

$$\mathbf{F} = \overline{\mathbf{A}}_t\overline{\mathbf{H}}_t^{-1} \tag{2.36}$$

and

$$\Psi = \operatorname{diag}\left(\operatorname{diag}\left(\frac{1}{t}\sum_{i=1}^{t}\mathbf{d}_i\mathbf{d}_i^\intercal - 2\mathbf{F}\overline{\mathbf{A}}_t^\intercal + \mathbf{F}\overline{\mathbf{H}}_t\mathbf{F}^\intercal\right)\right). \tag{2.37}$$

As in the online SGA algorithm from Section 2.2, it suffices to work with $\psi = \operatorname{diag}(\Psi)$ instead of Ψ. The diagonal entries of Equation (2.37) can be computed more efficiently as

$$\begin{aligned}\psi &= \frac{1}{t}\sum_{i=1}^{t}\mathbf{d}_i^2 - 2\cdot\operatorname{sum}\big(\mathbf{F}\odot\overline{\mathbf{A}}_t,\ \dim = 1\big) + \operatorname{sum}\big((\mathbf{F}\overline{\mathbf{H}}_t)\odot\mathbf{F},\ \dim = 1\big)\\ &= \frac{1}{t}\sum_{i=1}^{t}\mathbf{d}_i^2 + \operatorname{sum}\big((\mathbf{F}\overline{\mathbf{H}}_t)\odot\mathbf{F} - 2\mathbf{F}\odot\overline{\mathbf{A}}_t,\ \dim = 1\big).\end{aligned} \tag{2.38}$$

(TODO: check whether this can be negative). All that remains to complete the online algorithm is to define the incremental update rules for $\overline{\mathbf{A}}_t$ and $\overline{\mathbf{H}}_t$. Both are similar, and

for $\overline{\mathbf{A}}_t$ the derivation is

$$\begin{aligned}
\overline{\mathbf{A}}_t &= \frac{1}{t}\sum_{i=1}^{t}\mathbf{d}_i\mathbf{m}_i^\intercal \\
&= \frac{1}{t}\left(\sum_{i=1}^{t-1}\mathbf{d}_i\mathbf{m}_i^\intercal + \mathbf{d}_t\mathbf{m}_t^\intercal\right) \\
&= \frac{1}{t}\left(\frac{t-1}{t-1}\sum_{i=1}^{t-1}\mathbf{d}_i\mathbf{m}_i^\intercal + \mathbf{d}_t\mathbf{m}_t^\intercal\right) \\
&= \frac{1}{t}\left(t\cdot\frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{d}_i\mathbf{m}_i^\intercal - \frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{d}_i\mathbf{m}_i^\intercal + \mathbf{d}_t\mathbf{m}_t^\intercal\right) \\
&= \frac{1}{t}\left(t\overline{\mathbf{A}}_{t-1} - \overline{\mathbf{A}}_{t-1} + \mathbf{d}_t\mathbf{m}_t^\intercal\right) \\
&= \overline{\mathbf{A}}_{t-1} + \frac{1}{t}\left(\mathbf{d}_t\mathbf{m}_t^\intercal - \overline{\mathbf{A}}_{t-1}\right).
\end{aligned} \tag{2.39}$$

Pseudo code for online EM for FA is given in Algorithm 2.

Algorithm 2 Online Expectation-Maximisation for Factor Analysis

Input: Observation dimension d, latent dimension K

Initialise $\mathbf{F} \in \mathbb{R}^{d\times K}$, $\boldsymbol{\psi} > \mathbf{0} \in \mathbb{R}^d$
Initialise $\overline{\boldsymbol{\theta}}_0 = \mathbf{0}^d, \overline{\mathbf{A}}_0 = \mathbf{0}^{d\times K}, \overline{\mathbf{B}}_0 = \mathbf{0}^{K\times K}, \overline{\mathbf{d}^2}_0 = \mathbf{0}^d$
for $t = 1,\ldots,T$ **do**
    Sample observation $\boldsymbol{\theta}_t \in \mathbb{R}^d$
    $\overline{\boldsymbol{\theta}}_t \leftarrow \overline{\boldsymbol{\theta}}_{t-1} + \frac{1}{t}\left(\overline{\boldsymbol{\theta}}_t - \overline{\boldsymbol{\theta}}_{t-1}\right)$
    $\mathbf{d}_t \leftarrow \boldsymbol{\theta}_t - \overline{\boldsymbol{\theta}}_t$
    $\mathbf{C} \leftarrow (\mathbf{F}\odot\boldsymbol{\psi}^{-1})^\intercal$ (with broadcasting)
    $\Sigma \leftarrow (\mathbf{I} + \mathbf{C}\mathbf{F})^{-1}$
    $\mathbf{m}_t \leftarrow \Sigma\mathbf{C}\mathbf{d}_t$
    $\overline{\mathbf{B}}_t \leftarrow \overline{\mathbf{B}}_{t-1} + \frac{1}{t}(\mathbf{m}_t\mathbf{m}_t^\intercal - \overline{\mathbf{B}}_{t-1})$
    $\overline{\mathbf{H}}_t \leftarrow \Sigma + \overline{\mathbf{B}}_t$
    $\overline{\mathbf{A}}_t \leftarrow \overline{\mathbf{A}}_{t-1} + \frac{1}{t}(\mathbf{d}_t\mathbf{m}_t^\intercal - \overline{\mathbf{A}}_{t-1})$
    $\mathbf{F} \leftarrow \overline{\mathbf{A}}_t\overline{\mathbf{H}}_t^{-1}$
    $\overline{\mathbf{d}^2}_t \leftarrow \overline{\mathbf{d}^2}_{t-1} + \frac{1}{t}(\mathbf{d}_t^2 - \overline{\mathbf{d}^2}_{t-1})$
    $\boldsymbol{\psi} \leftarrow \overline{\mathbf{d}^2}_t + \mathrm{sum}\left((\mathbf{F}\overline{\mathbf{H}}_t)\odot\mathbf{F} - 2\mathbf{F}\odot\overline{\mathbf{A}}_t,\ \mathrm{dim} = 1\right)$
end for
$\boldsymbol{\theta}_{\mathrm{SWA}} \leftarrow \overline{\boldsymbol{\theta}}_T$
return $\boldsymbol{\theta}_{\mathrm{SWA}}, \mathbf{F}, \boldsymbol{\psi}$

2.4 Factors and Noise Initialisation

One detail missing from Algorithms 1 and 2 is how $\mathbf{F}$ and ψ are initialised. Suppose $\mathbf{F}$ is initialised with ones on its diagonal and zeros everywhere else. That is,

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{d \times K}, \tag{2.40}$$

where $\mathbf{I} \in \mathbb{R}^{K \times K}$ and $\mathbf{0} \in \mathbb{R}^{(d-K) \times K}$. Then for $\mathbf{h} \in \mathbb{R}^K$, the product $\mathbf{Fh} \in \mathbb{R}^d$ is just the vector formed by appending $d-K$ zeros onto the end of $\mathbf{h}$. In order to increase the initial diversity of $\mathbf{Fh}$, noise is added to the off-diagonal entries of $\mathbf{F}$. That is, $\mathbf{F}$ is initialised as

$$\mathbf{F} = \begin{bmatrix} 1 & \nu_{1,2} & \cdots & \nu_{1,K} \\ \nu_{2,1} & 1 & & \nu_{2,K} \\ \vdots & & \ddots & \vdots \\ \nu_{K,1} & \nu_{K,2} & \cdots & 1 \\ \nu_{K+1,1} & \nu_{K+1,2} & \cdots & \nu_{K+1,K} \\ \vdots & \vdots & \vdots & \vdots \\ \nu_{d,1} & \nu_{d,2} & \cdots & \nu_{d,K} \end{bmatrix}, \tag{2.41}$$

where $\nu_{i,j} \sim \mathcal{N}(0, \sigma^2)$ for some small σ, which is a hyperparameter.

From Equation (2.3), the full covariance matrix of a FA model is $\mathbf{F}\mathbf{F}^\intercal + \Psi$, where $\Psi = \operatorname{diag}(\psi)$. To ensure that neither $\mathbf{F}\mathbf{F}^\intercal$ nor Ψ dominates in the initial covariance matrix, the initial entries of ψ are set equal to the maximum diagonal value of $\mathbf{F}\mathbf{F}^\intercal$. That is,

$$\psi = (s, s, \ldots, s)^\intercal \in \mathbb{R}^d, \quad s = \max\big(\operatorname{diag}(\mathbf{F})\big). \tag{2.42}$$

2.5 Experiments

2.5.1 Methodology

The aim of these experiments is to test how well the online FA algorithms from Section 2 are able to fit observations sampled from actual FA models. The form of a FA model is given in Equation (2.3). In particular, it has parameters $\mathbf{c}, \mathbf{F}$ and Ψ. The maximum likelihood estimate of $\mathbf{c}$, given the data, is just the empirical mean of the sampled observations [1]. This is computed exactly in both online FA algorithms (Algorithms 1 and 2) by maintaining a running average of the observations. The other parameters, $\mathbf{F}$ and Ψ, appear in the FA model as part of the full covariance matrix, $\mathbf{F}\mathbf{F}^\intercal + \Psi$. Note

that right-multiplying $\mathbf{F}$ by any orthogonal matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$ will result in the exact same covariance [1], since

$$(\mathbf{FA})(\mathbf{FA})^\intercal + \Psi = \mathbf{FAA}^\intercal\mathbf{F}^\intercal + \Psi = \mathbf{FF}^\intercal + \Psi. \tag{2.43}$$

Therefore, the $\mathbf{F}$ estimated by an online FA algorithm cannot be directly compared to the true $\mathbf{F}$. The comparison must be made between the full covariance matrices.

Since the covariance matrices have shape $d \times d$, memory requirements dictate that d, the observation dimension, cannot be too large. Therefore, values of $d = 100$ and $d = 1000$ were used in all experiments in this section. In all cases the latent dimension was set to $K = 10$, meaning that two different observation to latent ratios were tested: $\frac{d}{K} = 10, 100$ for $d = 100, 1000$, respectively. Since computing the maximum likelihood estimate of $\mathbf{c} \in \mathbb{R}^d$ is trivial, in all experiments the entries of $\mathbf{c}$ were simply sampled independently from a standard normal distribution. Each factor loading matrix $\mathbf{F}$ was generated in such a way that its columns spanned the K-dimensional latent space and the conditioning number of the resultant covariance matrix could be controlled. This was achieved by finding the first K eigenvectors of a symmetric positive semi-definite matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, and then setting the columns of $\mathbf{F}$ equal to these eigenvectors multiplied by a vector $\mathbf{s} > \mathbf{0} \in \mathbb{R}^d$ (element-wise). By construction, the K eigenvectors are linearly independent and therefore span the latent space, and scaling them by $\mathbf{s}$ affects the conditioning number of $\mathbf{FF}^\intercal$. The conditioning number of a matrix is equal to

$$\max_{i,j} \frac{|\lambda_i|}{|\lambda_j|}, \tag{2.44}$$

where the λ_i and λ_j are eigenvalues of the matrix [2]. This ratio can be controlled by specifying the range of $\mathbf{s}$, or alternatively, the range of $\mathbf{s}^2$, which is called the *spectrum*. The larger the ratio of the upper to lower bound of the spectrum, the larger the ratio of the eigenvalues. In each experiment the spectrum was sampled from a uniform distribution with one of the following ranges: $[1, 10]$, $[1, 100]$ or $[1, 10, 000]$. Finally, the diagonal entries of Ψ were sampled from a uniform distribution with upper bound equal to the maximum value of $\mathbf{s}^2$. This is consistent with the FA assumption that an observation, $\theta = \mathbf{Fh} + \mathbf{c} + \varepsilon$, is generated by corrupting the signal $\mathbf{Fh} + \mathbf{c}$ with some random noise $\varepsilon \sim \mathcal{N}(\mathbf{0}, \Psi)$. The full details of how FA models were generated are given in Algorithm 3.

Having generated a FA model, observations were sampled according to Equation (2.1). Using the data, the parameters of the model were then estimated by three separate FA algorithms: online SGA (Algorithm 1), online EM (Algorithm 2) and the

Algorithm 3 Generate a Factor Analysis Model

Input: Observation dimension d, latent dimension K, spectrum range $[a, b]$

1: Generate $\mathbf{c} \in \mathbb{R}^d$ by sampling entries independently from $\mathcal{N}(0,1)$
2: Generate $\mathbf{A} \in \mathbb{R}^{d \times d}$ by sampling entries independently from $\mathcal{N}(0,1)$
3: $\mathbf{M} \leftarrow \mathbf{A}\mathbf{A}^\intercal$
4: Compute the K eigenvectors, $\mathbf{v}_1, \ldots, \mathbf{v}_K \in \mathbb{R}^d$, corresponding to the K largest eigenvalues of $\mathbf{M}$
5: Construct the matrix $\mathbf{V}_K \in \mathbb{R}^{d \times K}$ with columns $\mathbf{v}_1, \ldots, \mathbf{v}_K$
6: Generate $\mathbf{s}^2 \in \mathbb{R}^d$ by sampling entries independently from $\mathcal{U}(a, b)$
7: $\mathbf{s} \leftarrow \sqrt{\mathbf{s}^2}$ (square root is applied element-wise)
8: $\mathbf{F} \leftarrow \mathbf{V}_K \odot \mathbf{s}$ (with broadcasting)
9: $s_{max} \leftarrow \max(\mathbf{s}^2)$ (largest element of $\mathbf{s}^2$)
10: Generate $\psi \in \mathbb{R}^d$ by sampling entries independently from $\mathcal{U}(0, s_{max})$
11: $\Psi \leftarrow \mathrm{diag}(\psi)$
12: **return** $\mathbf{c}, \mathbf{F}, \Psi$

scikit-learn `FactorAnalysis` estimator [3]. The scikit-learn implementation is based on the batch SVD algorithm from [1]. The *randomised* version of SVD was used, which for most applications is "sufficiently precise while providing significant speed gains" [3]. Each algorithm was tested on samples of varying size, from 100 up to 100,000 observations. The latent dimension of each approximate model was set to the true latent dimension K. All other hyperparameters of the scikit-learn algorithm were set to their default values. For both online algorithms, $\mathbf{F}$ and Ψ were initialised according to Equations (2.41) and (2.42) with $\sigma = 0.1$, and online SGA ran with a learning rate of 0.001. Each experiment was repeated ten times. In each trial a different random seed was used for generating the true FA model and also initialising the parameters of the online algorithms.

2.5.2 Results

Figure 2.1 shows the relative distance between the true covariance matrix of each FA model and the corresponding estimated covariance matrix of each learning algorithm, as a function of the number of samples used to fit the approximate models. The distance between two matrices is measured by the Frobenius norm (also sometimes called the Euclidean norm) of the difference between the two matrices. The distance is then

divided by the Frobenius norm of the true covariance matrix to get the relative distance.

In the first row of the figure are the results corresponding to FA models whose factor loading matrices were generated with a spectrum range of $[1, 10]$. When $d = 100$ (top-left), online EM approximates the true covariance matrix better than online SGA when the number of samples, T, is less than or equal to 5000. For $T > 5000$, online SGA performs much better, as the distance continues to decrease at a much faster rate. In particular, for $T \geq 50,000$, online SGA actually gets closer to the true covariance matrix than batch SVD. When $d = 1000$ (top-right), online SGA approximates the true covariance matrix better than online EM for all sample sizes, but in this case is not able to beat batch SVD for large T.

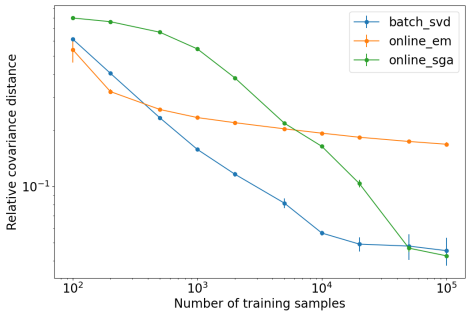
In the second row of the figure are the results corresponding to a spectrum range of $[1, 100]$. When $d = 100$ (middle-left), online EM achieves smaller distances than online SGA for all samples sizes except $T = 100,000$. In this case, neither online algorithm is able to match batch SVD when $T \geq 1000$. When $d = 1000$ (middle right), online EM is slightly better than online SGA for $T \geq 2000$ and reduces the distance to a level on par with batch SVD for the largest sample sizes.

In the third row of the figure are the results corresponding to a spectrum range of $[1, 10,000]$. In this case, the results for online SGA were extremely poor, with the relative distances several orders of magnitude greater than the corresponding results achieved by online EM and batch SVD. To avoid skewing the axes, the online SGA results are omitted from these plots. When $d = 100$ (bottom-left), online EM does not perform as well as batch SVD for $T \geq 1000$, although it is still able to reduce the relative distance to almost 0.1 for the largest sample sizes. When $d = 1000$ (bottom-right), the error bars on the online EM results are extremely large for $T \leq 500$. However, for larger sample sizes the online algorithm reduces the covariance distance to a level very close to that of batch SVD.

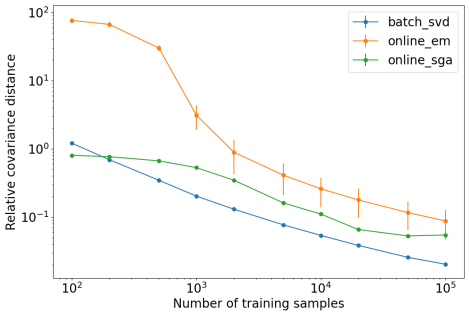
Figure 2.2 shows the 2-Wasserstein distance (TODO: add citation) between the Gaussian distribution defined by each estimated FA model and the Gaussian distribution defined by the true FA model, for the same combinations of observation dimension, latent dimension, spectrum range and sample size. Since all algorithms are able to compute the mean of the true Gaussian distribution exactly, the Wasserstein distance is dominated by the difference between the true and estimated covariance matrices. Therefore, the shape of the plots in Figure 2.2 is very similar to the shape of the plots in Figure 2.1, albeit the y-axis scales are different.

These results show that there is no clear winner between online SGA and online

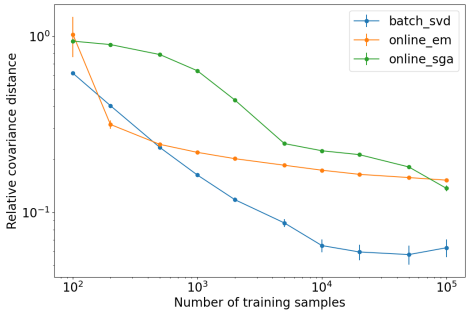
EM over the full grid of parameter combinations tested. However, when enough samples are available, online EM seems to be more robust in the more difficult scenarios. That is, when the observation dimension and the spectrum range are larger. One interesting difference between online SGA and online EM is the step size used in each iteration of the two algorithms. In these experiments, the learning rate of online SGA was set to a constant of 0.001. Online EM does not have an explicit learning rate parameter, however, it does have an implicit step size which decreases on each iteration. In Algorithm 2, the running averages $\overline{\mathbf{A}}_t, \overline{\mathbf{B}}_t$ and $\overline{\mathbf{d}^2}_t$ are moved a fraction $\frac{1}{t}$ of the way towards the latest data point sampled on iteration t. When t is small the step size will be relatively large, but as t increases the step size will become increasingly small. In particular, for $t = 100,000$ the step size is only 0.00001, two orders of magnitude less than the constant learning rate in online SGA. This could explain why, when $d = 100$, online EM initially uses the samples more efficiently than online SGA but eventually flattens off. The same behaviour is not observed when $d = 1000$. Perhaps the large step sizes in the initial iterations of online EM are too aggressive when the observation dimension is large, leading to large errors for small sample sizes. However, with enough data, online EM is comparable to batch SVD.



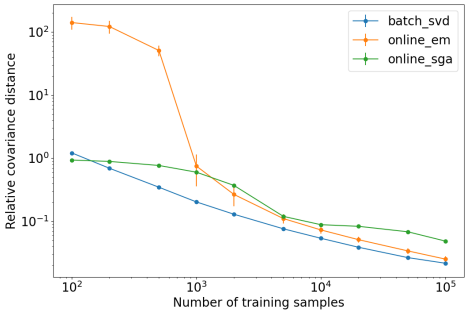
(a) $d = 100$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10)$



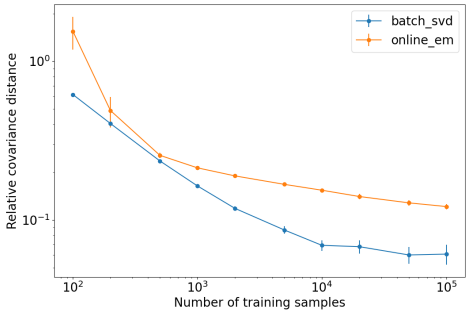
(b) $d = 1000$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10)$



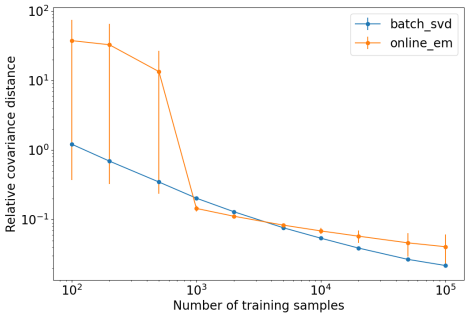
(c) $d = 100$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 100)$



(d) $d = 1000$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 100)$

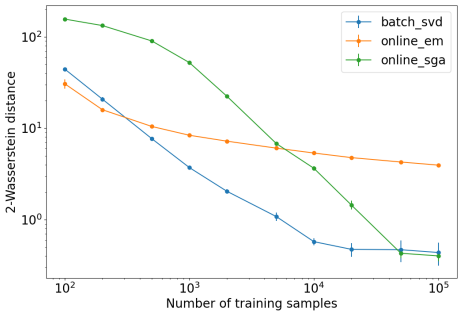


(e) $d = 100$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10000)$

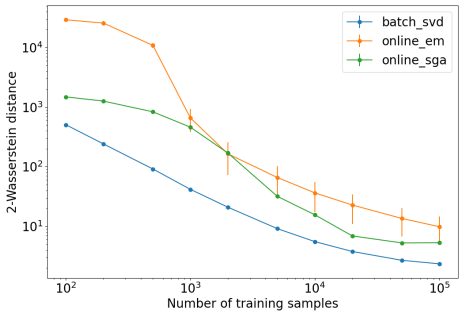


(f) $d = 1000$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10000)$

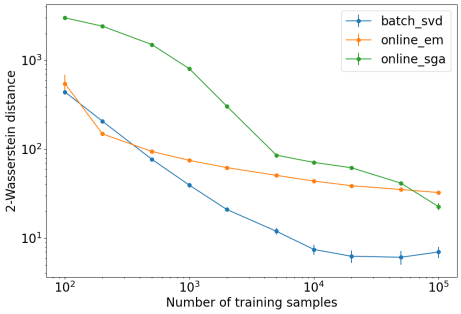
Figure 2.1: The relative distance of the estimated FA covariance matrices from the true covariance matrix as a function of the number of samples used to learn the models. The blue, orange and green lines show, for batch SVD, online EM and online SGA, respectively, the Frobenius norm of the difference between the true covariance matrix and the estimated covariance matrix divided by the Frobenius norm of the true covariance matrix. Each data point shows the mean value over ten trials with different random seeds, and standard error bars are also plotted. The different plots correspond to different combinations of the observation dimension d, the latent dimension K and the range of the spectrum $\mathbf{s}^2$. The plots on the bottom row omit the results for online SGA as the distances are several orders of magnitude greater than the other results.



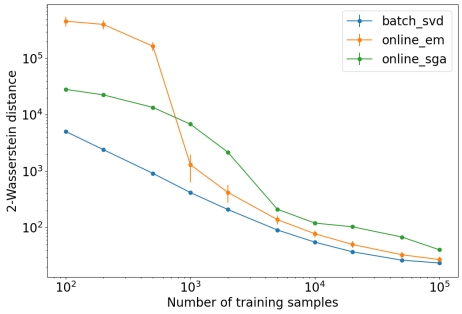
(a) $d = 100$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10)$



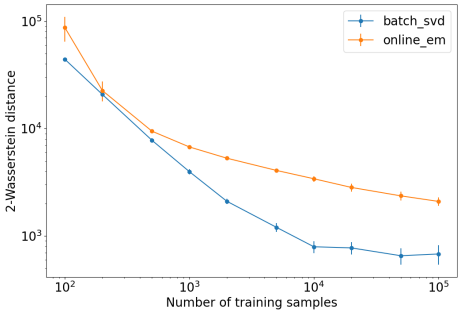
(b) $d = 1000$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10)$



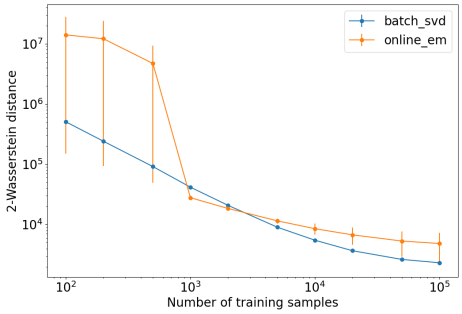
(c) $d = 100$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 100)$



(d) $d = 1000$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 100)$



(e) $d = 100$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10000)$



(f) $d = 1000$, $K = 10$, $\mathbf{s}^2 \sim \mathcal{U}(1, 10000)$

Figure 2.2: The 2-Wasserstein distance between the Gaussian distribution defined by the estimated FA model and the Gaussian distribution defined by the true FA model. The blue, orange and green lines show the 2-Wasserstein distance corresponding to batch SVD, online EM and online SGA, respectively. Each data point shows the mean value over ten trials with different random seeds, and standard error bars are also plotted. The different plots correspond to different combinations of the observation dimension d, the latent dimension K and the range of the spectrum $\mathbf{s}^2$. The plots on the bottom row omit the results for online SGA as the distances are several orders of magnitude greater than the other results.

Chapter 3

Bayesian Linear Regression

3.1 Background

A linear regression model is a mapping from vector inputs $\mathbf{x} \in \mathbb{R}^d$ to scalar outputs $y \in \mathbb{R}$ via an affine transformation. Given a set of observed input-output pairs, $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, it is assumed that each output is generated according to

$$y_n = \theta^\intercal \mathbf{x}_n + \varepsilon \tag{3.1}$$

for some unknown $\theta \in \mathbb{R}^d$. The underlying signal, $\theta^\intercal \mathbf{x}_n$, is corrupted by additive noise,

$$\varepsilon \sim \mathcal{N}(0, \sigma^2), \tag{3.2}$$

for some $\sigma > 0$ [1]. The model is often written with an explicit bias term, but this can be absorbed into θ by adding a constant of one to the input, leading to the expression in Equation (3.1).

3.1.1 Computing the Posterior

Due to the additive noise, each y_n is a random variable, conditioned on $\mathbf{x}_n$ and θ. Since ε is Gaussian distributed, the conditional pdf of y_n is

$$p(y_n|\theta, \mathbf{x}_n) = \mathcal{N}\big(\theta^\intercal \mathbf{x}_n, \sigma^2\big) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{1}{2\sigma^2}\big(y_n - \theta^\intercal \mathbf{x}_n\big)^2\Big). \tag{3.3}$$

Assuming that the observations in $\mathcal{D}$ are independent and identically distributed, the log-likelihood of θ having generated the data is

$$\begin{aligned}
\log p(\mathcal{D}|\theta) &= \sum_{n=1}^{N} \left[\log p(y_n|\theta, \mathbf{x}_n) + \log p(\mathbf{x}_n)\right] \\
&= \sum_{n=1}^{N} \left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(y_n - \theta^\intercal \mathbf{x}_n\right)^2\right] + \sum_{n=1}^{N} \log p(\mathbf{x}_n) \\
&= -\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y_n - \theta^\intercal \mathbf{x}_n\right)^2 - \frac{N}{2}\log\sigma^2 - \frac{N}{2}\log 2\pi + \sum_{n=1}^{N}\log p(\mathbf{x}_n) \\
&= -\frac{\beta}{2}\sum_{n=1}^{N}\left(y_n - \theta^\intercal \mathbf{x}_n\right)^2 + \frac{N}{2}\log\beta + \text{constant},
\end{aligned} \tag{3.4}$$

where $\beta = \frac{1}{\sigma^2}$ [1]. In Bayesian linear regression, a prior distribution for θ is also specified. A common choice is

$$p(\theta) = \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right) = \left(\frac{\alpha}{2\pi}\right)^{\frac{d}{2}} \exp\left(-\frac{\alpha}{2}\theta^\intercal\theta\right) \tag{3.5}$$

for some $\alpha > 0$, which is a hyperparameter known as the *precision* [1]. Applying the logarithm,

$$\begin{aligned}
\log p(\theta) &= \frac{d}{2}\log\frac{\alpha}{2\pi} - \frac{\alpha}{2}\theta^\intercal\theta \\
&= -\frac{\alpha}{2}\theta^\intercal\theta + \frac{d}{2}\log\alpha + \text{constant}.
\end{aligned} \tag{3.6}$$

Now, using Bayes' rule and Equation (3.4) and Equation (3.6), it follows that the log-posterior distribution of θ for fixed α and β is

$$\begin{aligned}
\log p(\theta|\mathcal{D}) &= \log\frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \\
&= \log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D}) \\
&= -\frac{\beta}{2}\sum_{n=1}^{N}\left(y_n - \theta^\intercal\mathbf{x}_n\right)^2 - \frac{\alpha}{2}\theta^\intercal\theta + \text{constant} \\
&= -\frac{\beta}{2}\sum_{n=1}^{N}\left(y_n^2 - 2y_n\theta^\intercal\mathbf{x}_n + (\theta^\intercal\mathbf{x}_n)^2\right) - \frac{\alpha}{2}\theta^\intercal\theta + \text{constant} \\
&= -\frac{\beta}{2}\sum_{n=1}^{N} y_n^2 + \beta\sum_{n=1}^{N} y_n\theta^\intercal\mathbf{x}_n - \frac{\beta}{2}\sum_{n=1}^{N}\theta^\intercal\mathbf{x}_n\mathbf{x}_n^\intercal\theta - \frac{\alpha}{2}\theta^\intercal\theta + \text{constant} \\
&= \left(\beta\sum_{n=1}^{N} y_n\mathbf{x}_n\right)^\intercal\theta - \frac{1}{2}\theta^\intercal\left(\alpha\mathbf{I} + \beta\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\intercal\right)\theta + \text{constant} \\
&= \mathbf{b}^\intercal\theta - \frac{1}{2}\theta^\intercal\mathbf{A}\theta + \text{constant},
\end{aligned} \tag{3.7}$$

where

$$\mathbf{A} = \alpha\mathbf{I} + \beta \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\intercal \quad \text{and} \quad \mathbf{b} = \beta \sum_{n=1}^{N} y_n \mathbf{x}_n. \tag{3.8}$$

Now, using a result from [1] to complete the square of Equation (3.7),

$$\begin{aligned} \log p(\theta|\mathcal{D}) &= -\frac{1}{2}\big(\theta - \mathbf{A}^{-1}\mathbf{b}\big)^\intercal \mathbf{A}\big(\theta - \mathbf{A}^{-1}\mathbf{b}\big) + \frac{1}{2}\mathbf{b}^\intercal \mathbf{A}^{-1}\mathbf{b} + \text{constant} \\ &= -\frac{1}{2}\big(\theta - \mathbf{A}^{-1}\mathbf{b}\big)^\intercal \mathbf{A}\big(\theta - \mathbf{A}^{-1}\mathbf{b}\big) + \text{constant} \\ &= -\frac{1}{2}\big(\theta - \mathbf{m}\big)^\intercal \mathbf{S}^{-1}\big(\theta - \mathbf{m}\big) + \text{constant}, \end{aligned} \tag{3.9}$$

where

$$\mathbf{m} = \mathbf{A}^{-1}\mathbf{b} \quad \text{and} \quad \mathbf{S} = \mathbf{A}^{-1}. \tag{3.10}$$

Hence, the posterior distribution of θ is

$$p(\theta|\mathcal{D}) = \mathcal{N}(\mathbf{m}, \mathbf{S}). \tag{3.11}$$

3.1.2 Estimating the Posterior Online

Since the posterior of the linear regression parameter vector can be computed in closed form, the ability of the online FA algorithms to learn the posterior can be evaluated in this case. For a linear regression model trained via SGD, Algorithms 1 and 2 can both be used to fit a Gaussian posterior distribution to the iterates, $\theta_1, \ldots, \theta_T$, encountered along the SGD trajectory. The mean and covariance of the learned distributions can then be compared directly to the ground truth. Recall, however, that the posterior distribution in Equation (3.11) refers to specific values of α and β. The parameter β is $\frac{1}{\sigma^2}$, where σ is the standard deviation of the outputs y_n in Equation (3.3). This can be estimated by calculating the empirical standard deviation of $\{y_n\}_{n=1}^N$.

The precision α is a hyperparameter which controls the width of the prior $p(\theta)$. In Equation (3.7), the log-posterior of θ was written as

$$\log p(\theta|\mathcal{D}) = -\frac{\beta}{2}\sum_{n=1}^{N}\big(y_n - \theta^\intercal \mathbf{x}_n\big)^2 - \frac{\alpha}{2}\theta^\intercal\theta + \text{constant}. \tag{3.12}$$

Hence, the maximum *a posteriori* (MAP) estimate of θ is that which maximises

$$-\frac{\beta}{2}\sum_{n=1}^{N}\big(y_n - \theta^\intercal \mathbf{x}_n\big)^2 - \frac{\alpha}{2}\theta^\intercal\theta. \tag{3.13}$$

Since $\beta > 0$, it is equivalent to minimise this expression scaled by $-\frac{2}{\beta}$, that is,

$$\sum_{n=1}^{N}\big(y_n - \theta^\intercal \mathbf{x}_n\big)^2 + \frac{\alpha}{\beta}\theta^\intercal\theta. \tag{3.14}$$

This objective function is analogous to the L2-regularised training loss often used in non-Bayesian linear regression [1], which is

$$\sum_{n=1}^{N} \left(y_n - \theta^\intercal \mathbf{x}_n\right)^2 + \lambda \theta^\intercal \theta \tag{3.15}$$

for some $\lambda > 0$. Setting $\lambda = \frac{\alpha}{\beta}$, the optimal θ found by L2-regularised linear regression is the same as the MAP estimate of θ in Bayesian linear regression with prior $p(\theta) = \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$. This means that, given data $\mathcal{D}$ and a particular value of α, the posterior distribution of θ can be estimated by fitting an online FA model to the SGD iterates sampled while minimising Equation (3.15) with $\lambda = \frac{\alpha}{\beta}$.

3.2 Experiments

3.2.1 Methodology

3.2.2 Results

Chapter 4

Conclusions

4.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Over length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

Bibliography

[1] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[3] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Kaare B Petersen and Michael S Pedersen. *The Matrix Cookbook*. Technical University of Denmark, 2012.