Probabilistic Sequential Matrix Factorization

Ö. Deniz Akyildiz

The Alan Turing Institute
University of Cambridge

Joint work with Gerrit J.J. van den Burg (Amazon Alexa), Theodoros Damoulas (Warwick), Mark F. J. Steel (Warwick).

Background

The Probabilistic Model

Inference (Optimal and Approximate)

Parameter Estimation

Experimental results

Conclusions

Problem definition

Matrix factorization

We are interested in the problem factorizing a data matrix $Y \in \mathbb{R}^{d \times n}$

$$Y \approx CX$$

with $C \in \mathbb{R}^{d \times r}$, *the dictionary*, and $X \in \mathbb{R}^{r \times n}$ the *coefficients*.

Problem definition

Matrix factorization

We are interested in the problem factorizing a data matrix $Y \in \mathbb{R}^{d \times n}$

$$Y \approx CX$$

with $C \in \mathbb{R}^{d \times r}$, *the dictionary*, and $X \in \mathbb{R}^{r \times n}$ the *coefficients*.

Why is this useful?

Problem definition

Matrix factorization

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{X}$$

- Dimensionality reduction of Y learning the dictionary C and low-dimensional encodings X.

Problem definition

Matrix factorization

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{X}$$

- Dimensionality reduction of Y learning the dictionary C and low-dimensional encodings X.
  - Clustering, representation, interpretability.

Problem definition

Matrix factorization

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{X}$$

- Dimensionality reduction of Y learning the dictionary C and low-dimensional encodings X.
  - Clustering, representation, interpretability.
- Missing data is easy to handle, reconstruction CX is a natural imputation strategy.

Problem definition

Matrix factorization

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{X}$$

- Dimensionality reduction of Y learning the dictionary C and low-dimensional encodings X.
  - Clustering, representation, interpretability.
- Missing data is easy to handle, reconstruction CX is a natural imputation strategy.

Some use cases

- Image clustering, video sequence embedding and clustering
- Recommendation systems
- Genome data analysis
- Audio signal processing, separation, denoising, restoration

Problem definition

Matrix factorization

The classical approach (also the most famous) is nonnegative matrix factorization (Lee et al. 1999).

Problem definition

Matrix factorization

The classical approach (also the most famous) is nonnegative matrix factorization (Lee et al. 1999).

- This is an approach to find nonnegative C and X (for better interpretability) by minimizing $\|Y - CX\|_F^2$.

Problem definition

Matrix factorization

The classical approach (also the most famous) is nonnegative matrix factorization (Lee et al. 1999).

- This is an approach to find nonnegative C and X (for better interpretability) by minimizing $\|Y - CX\|_F^2$.
- A multiplicative gradient descent approach gives the update rules (element-wise):

$$C \leftarrow C \frac{(YX^T)}{(CXX^T)} \tag{1}$$

$$X \leftarrow X \frac{(C^T Y)}{(C^T C X)}. \tag{2}$$

Problem definition

Matrix factorization

The classical approach (also the most famous) is nonnegative matrix factorization (Lee et al. 1999).

- This is an approach to find nonnegative C and X (for better interpretability) by minimizing $\|Y - CX\|_F^2$.
- A multiplicative gradient descent approach gives the update rules (element-wise):

$$C \leftarrow C \frac{(YX^T)}{(CXX^T)} \tag{1}$$

$$X \leftarrow X \frac{(C^TY)}{(C^TCX)}. \tag{2}$$

- The paper has 13649 citations as it currently stands...

Problem definition

Matrix factorization

What do we want to do?

We would like to develop an algorithm that is

- Probabilistic: We want to obtain approximate probability measures over C and X

Problem definition

Matrix factorization

What do we want to do?

We would like to develop an algorithm that is

- Probabilistic: We want to obtain approximate probability measures over C and X
- Dynamic: We are interested in the case where Y is a Markovian process (e.g. induced by a time-series dataset Y).

Problem definition

Matrix factorization

What do we want to do?

We would like to develop an algorithm that is

- Probabilistic: We want to obtain approximate probability measures over C and X
- Dynamic: We are interested in the case where Y is a Markovian process (e.g. induced by a time-series dataset Y).
- Sequential: We want to process the columns of Y sequentially in time in a scalable way.

$$\underbrace{\begin{bmatrix} \underbrace{\begin{matrix}\times \\ \times \\ \times\end{matrix}}_{y_1} & \begin{matrix}\times \\ \times \\ \times\end{matrix} & \begin{matrix}\times \\ \times \\ \times\end{matrix} & \begin{matrix}\times \\ \times \\ \times\end{matrix} & \begin{matrix}\times \\ \times \\ \times\end{matrix} \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{C_1} \underbrace{\begin{bmatrix} \underbrace{\begin{matrix}\times \\ \times\end{matrix}}_{x_1} & \begin{matrix}\times \\ \times\end{matrix} & \begin{matrix}\times \\ \times\end{matrix} & \begin{matrix}\times \\ \times\end{matrix} & \begin{matrix}\times \\ \times\end{matrix} \end{bmatrix}}_{X}$$

Problem definition

Matrix factorization

What do we want to do?

We would like to develop an algorithm that is

- **Probabilistic**: We want to obtain approximate probability measures over C and X
- **Dynamic**: We are interested in the case where Y is a Markovian process (e.g. induced by a time-series dataset Y).
- **Sequential**: We want to process the columns of Y sequentially in time in a scalable way.

$$\underbrace{\begin{bmatrix} \times & \boldsymbol{\times} & \times & \times & \times \\ \times & \boldsymbol{\times} & \times & \times & \times \\ \times & \underbrace{\boldsymbol{\times}}_{y_2} & \times & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} \\ \boldsymbol{\times} & \boldsymbol{\times} \\ \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}_{C_2} \underbrace{\begin{bmatrix} \times & \boldsymbol{\times} & \times & \times & \times \\ \times & \underbrace{\boldsymbol{\times}}_{x_2} & \times & \times & \times \end{bmatrix}}_{X}$$

Problem definition

Matrix factorization

What do we want to do?

We would like to develop an algorithm that is

- Probabilistic: We want to obtain approximate probability measures over C and X
- Dynamic: We are interested in the case where Y is a Markovian process (e.g. induced by a time-series dataset Y).
- Sequential: We want to process the columns of Y sequentially in time in a scalable way.

$$\underbrace{\begin{bmatrix} \times & \times & \boldsymbol{\times} & \times & \times \\ \times & \times & \boldsymbol{\times} & \times & \times \\ \times & \times & \underbrace{\boldsymbol{\times}}_{y_3} & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} \\ \boldsymbol{\times} & \boldsymbol{\times} \\ \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}_{C_3} \underbrace{\begin{bmatrix} \times & \times & \boldsymbol{\times} & \times & \times \\ \times & \times & \underbrace{\boldsymbol{\times}}_{x_3} & \times & \times \end{bmatrix}}_{X}$$

Problem definition

Matrix factorization

What do we want to do?

We would like to develop an algorithm that is

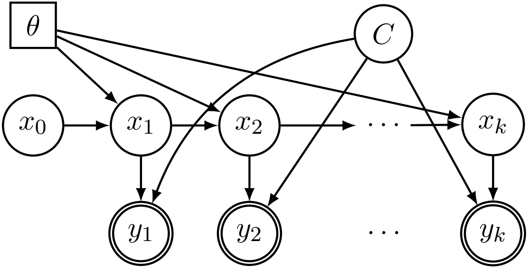
- Probabilistic: We want to obtain approximate probability measures over C and X
- Dynamic: We are interested in the case where Y is a Markovian process (e.g. induced by a time-series dataset Y).
- Sequential: We want to process the columns of Y sequentially in time in a scalable way.

$$\underbrace{\begin{bmatrix} \times & \times & \times & \boldsymbol{\times} & \times \\ \times & \times & \times & \boldsymbol{\times} & \times \\ \times & \times & \times & \underbrace{\boldsymbol{\times}}_{y_4} & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} \\ \boldsymbol{\times} & \boldsymbol{\times} \\ \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}_{C_4} \underbrace{\begin{bmatrix} \times & \times & \times & \boldsymbol{\times} & \times \\ \times & \times & \times & \underbrace{\boldsymbol{\times}}_{x_4} & \times \end{bmatrix}}_{X}$$

The Probabilistic Model

A state-space formulation

The model:

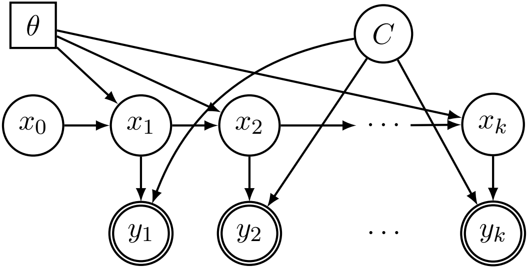


$$
\begin{aligned}
p(C) &= \mathcal{MN}(C; C_0, I_d, V_0), \\
p(x_0) &= \mathcal{N}(x_0; \mu_0, P_0) \\
p_\theta(x_t|x_{t-1}) &= \mathcal{N}(x_t; f_\theta(x_{t-1}), Q_t) \\
p(y_t|x_t, C) &= \mathcal{N}(y_t; Cx_t, R_t),
\end{aligned}
$$

The Probabilistic Model

A state-space formulation

The model:



$$
\begin{aligned}
p(C) &= \mathcal{MN}(C; C_0, I_d, V_0), \\
p(x_0) &= \mathcal{N}(x_0; \mu_0, P_0) \\
p_\theta(x_t | x_{t-1}) &= \mathcal{N}(x_t; f_\theta(x_{t-1}), Q_t) \\
p(y_t | x_t, C) &= \mathcal{N}(y_t; C x_t, R_t),
\end{aligned}
$$

(i) Ensures $y_t \approx C x_t$ (which implies $Y \approx CX$), (ii) encoding via f_θ.

Inference – Optimal and Approximate

Given a probabilistic model of the form:

$$
\begin{aligned}
c &\sim p(c), \\
x_0 &\sim p(x_0), \\
x_k | x_{k-1} &\sim p(x_k | x_{k-1}), \\
y_k | x_k, c &\sim p(y_k | x_k, c),
\end{aligned}
$$

how do we perform optimal inference?

To derive one step of the method, assume that $p(c|y_{1:k-1})$ and $p(x_{k-1}|y_{1:k-1})$ are known[1].

[1] For $k = 1$, they are just priors, so this defines the full recursion if we describe the one-step update.

Inference – Optimal and Approximate

prediction

Optimal: Given $p(x_{k-1}|y_{1:k-1})$, the first step of the algorithm performs prediction:

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})\mathrm{d}x_{k-1}.$$

Note that this step is independent of the dictionary (given that past marginal is known).

Inference – Optimal and Approximate

prediction

Optimal: Given $p(x_{k-1}|y_{1:k-1})$, the first step of the algorithm performs prediction:

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})\mathrm{d}x_{k-1}.$$

Note that this step is independent of the dictionary (given that past marginal is known).

Approximate: We perform extended Kalman prediction given a Gaussian approximation: $\tilde{p}(x_{k-1}|y_{1:k-1}) = \mathcal{N}(\mu_{k-1}, P_{k-1})$.

Inference – Optimal and Approximate

update of c

Optimal: In order to compute *updates*, we define the incremental marginal likelihood:

$$p(y_k|y_{1:k-1}) = \int\int p(y_k|c, x_k)p(x_k|y_{1:k-1})p(c|y_{1:k-1})\mathrm{d}x_k\mathrm{d}c.$$

Inference – Optimal and Approximate

update of c

Optimal: In order to compute *updates*, we define the incremental marginal likelihood:

$$p(y_k|y_{1:k-1}) = \int\int p(y_k|c, x_k)p(x_k|y_{1:k-1})p(c|y_{1:k-1})\mathrm{d}x_k\mathrm{d}c.$$

Based on this, *the dictionary update* is given by

$$p(c|y_{1:k}) = p(c|y_{1:k-1})\frac{p(y_k|c, y_{1:k-1})}{p(y_k|y_{1:k-1})},$$

where

$$p(y_k|c, y_{1:k-1}) = \int p(y_k|c, x_k)p(x_k|y_{1:k-1})\mathrm{d}x_k.$$

Inference – Optimal and Approximate

update of c: Scalable and efficient inference with matrix updates

Approximate:

Proposition 1

Given $\tilde{p}(c|y_{1:k-1}) = \mathcal{N}(c; c_{k-1}, V_{k-1} \otimes I_d)$ *and the likelihood* $\tilde{p}(y_k|c, y_{1:k-1}) = \mathcal{N}(y_k; C\bar{\mu}_k, \eta_k \otimes I_d)$ *the approximate posterior is* $\tilde{p}(c|y_{1:k}) = \mathcal{N}(c; c_k, V_k \otimes I_d)$*, where* $c_k = \mathrm{vec}(C_k)$ *and the posterior column-covariance matrix* V_k *is given by*

$$V_k = V_{k-1} - \frac{V_{k-1}\bar{\mu}_k\bar{\mu}_k^\top V_{k-1}}{\bar{\mu}_k^\top V_{k-1}\bar{\mu}_k + \eta_k} \qquad \text{for} \quad k \geq 1, \tag{3}$$

and the posterior mean C_k *of the dictionary* C *can be obtained in matrix-form as*

$$C_k = C_{k-1} + \frac{(y_k - C_{k-1}\bar{\mu}_k)\bar{\mu}_k^\top V_{k-1}^\top}{\bar{\mu}_k^\top V_{k-1}\bar{\mu}_k + \eta_k} \quad \text{for } k \geq 1. \tag{4}$$

Inference – Optimal and Approximate

update of x_k

Optimal: The coefficients update is given by

$$p(x_k|y_{1:k}) = p(x_k|y_{1:k-1})\frac{p(y_k|x_k, y_{1:k-1})}{p(y_k|y_{1:k-1})},$$

where

$$p(y_k|x_k, y_{1:k-1}) = \int p(y_k|x_k, c)p(c|y_{1:k-1})\mathrm{d}c.$$

Inference – Optimal and Approximate

update of x_k

Optimal: The coefficients update is given by

$$p(x_k|y_{1:k}) = p(x_k|y_{1:k-1})\frac{p(y_k|x_k, y_{1:k-1})}{p(y_k|y_{1:k-1})},$$

where

$$p(y_k|x_k, y_{1:k-1}) = \int p(y_k|x_k, c)p(c|y_{1:k-1})\mathrm{d}c.$$

Approximate: After some (Gaussian) approximations for $p(y_k|x_k, y_{1:k-1})$, the update is nothing but the standard extended Kalman update (see the paper).

Parameter estimation

Iterative and recursive

To estimate the parameters of f_θ, we need to solve

$$\theta^\star \in \operatorname*{argmax}_{\theta \in \Theta} \log p_\theta(y_{1:n}), \tag{5}$$

using gradient-based schemes (Kantas et al. 2015).

We consider two schemes:

Parameter estimation

Iterative and recursive

To estimate the parameters of f_θ, we need to solve

$$\theta^\star \in \underset{\theta \in \Theta}{\operatorname{argmax}} \log p_\theta(y_{1:n}), \tag{5}$$

using gradient-based schemes (Kantas et al. 2015).

We consider two schemes:

- Iterative estimation: For relatively short sequences,

$$\theta_{i+1} = \theta_i + \gamma \nabla \log p_\theta(y_{1:n})\big|_{\theta=\theta_i}$$

Parameter estimation

Iterative and recursive

To estimate the parameters of f_θ, we need to solve

$$\theta^\star \in \operatorname*{argmax}_{\theta \in \Theta} \log p_\theta(y_{1:n}), \tag{5}$$

using gradient-based schemes (Kantas et al. 2015).

We consider two schemes:

- Iterative estimation: For relatively short sequences,

$$\theta_{i+1} = \theta_i + \gamma \nabla \log p_\theta(y_{1:n})\big|_{\theta=\theta_i}$$

- Recursive estimation: Purely online estimation procedure for long sequences.

$$\theta_{k+1} = \theta_k + \gamma \nabla \log \tilde{p}_\theta(y_k | y_{1:k-1})\big|_{\theta=\theta_k}.$$

Parameter estimation

But what is the marginal log-likelihood?

Recall our approximation

$$\tilde{p}(y_k|y_{1:k-1}, c) = \mathcal{N}(y_k; C f_\theta(\mu_{k-1}), \eta_k \otimes I_d),$$

and the most recent dictionary posterior

$$p(c|y_{1:k-1}) = \mathcal{N}(c; c_{k-1}, V_{k-1} \otimes I_d).$$

Based on this, we can approximate the marginal by integrating out c, which results in

$$\begin{aligned} -\log \tilde{p}_\theta(y_k|y_{1:k-1}) \overset{c}{=} & \frac{d}{2} \log \left(\|f_\theta(\mu_{k-1})\|^2_{V_{k-1}} + \eta_k \right) \\ & + \frac{1}{2} \frac{\|y_k - C_{k-1} f_\theta(\mu_{k-1})\|^2}{\eta_k + \|f_\theta(\mu_{k-1})\|^2_{V_{k-1}}} \end{aligned} \tag{6}$$

Simply, this is a "loss" arises from the model itself, which we optimise w.r.t. θ using automatic differentiation.

Experimental results

A synthetic nonlinear periodic subspace

We consider the coefficient dynamics

$$x_k = f_\theta(x_{k-1}) = \cos(2\pi\theta k + x_{k-1})$$

, where $\theta \in \mathbb{R}_+^r$ and $Q_k = 0$ for all $k \geq 1$.

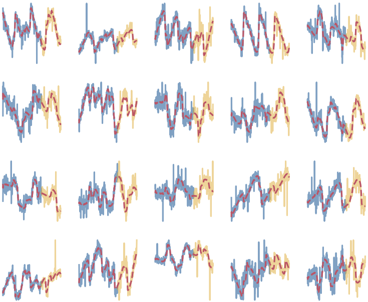
- This defines a deterministic subspace with highly periodic structure. We choose $d = 20$ and $r = 6$ and generate the data from the model with $\theta^\star = 10^{-3} \cdot [1, 2, 3, 4, 5, 6]$.
- We furthermore use iterative parameter estimation using the Adam optimizer with standard parameterization.
- We generate Y using a random C and run the PSMF to infer
  - C
  - $(x_k)_{k=1}^n$,
  - The parameters θ

jointly.

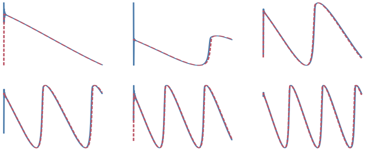
Experimental results

A synthetic nonlinear periodic subspace

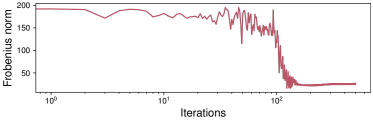
When the subspace model is well-calibrated, we can perform high-dimensional time-series prediction.



(a) Observed time series (blue) with unobserved future data (yellow) and the reconstruction (red).



(b) True (blue) and learned (red) subspace.



(c) Reconstruction error

Experimental results

Periodic modelling of air quality data (Beijing)

We have used the following (similar) model for real-world data.

- We have $n = 439$ observations and $d = 3$ variables (dew point, temperature, and atmospheric pressure).
- We compare PSMF using a random walk subspace model,

$$x_k = f(x_{k-1}) = x_{k-1},$$

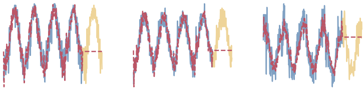
  against a periodic subspace model

$$x_k = f_\theta(x_{k-1}) = \theta_1 \sin(2\pi\theta_2 k + \theta_3 x_{k-1}) + \theta_4 \cos(2\pi\theta_5 k + \theta_6 x_{k-1}).$$

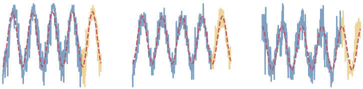
- In both settings we use $r = 1$, run iterative PSMF with 100 iterations and fit C, $(x_k)_{k=1}^n$, θ.

Experimental results

Periodic modelling of air quality data (Beijing)



(a) Random walk subspace model.



(b) Periodic subspace model.

Figure: Comparison of random walk and periodic subspace models on a time series of weather measurements in Beijing. This shows that with the appropriate subspace model, PSMF correctly identifies the nonlinear dynamics of the data and accurately extrapolates into the future. Observed time series (blue) with unobserved future data (yellow) and the reconstruction (red).

Experimental results

Missing data imputation, air quality data for London

	NO_2	PM10	PM25	S&P500	Gas
PSMF	0.76	0.76	**0.92**	0.83	**0.89**
rPSMF	**0.85**	**0.89**	0.87	**0.83**	0.86
MLE-SMF	0.43	0.56	0.80	0.48	0.56

Average coverage proportion of the missing data by the 2σ uncertainty bars of the posterior predictive estimates, averaged over 100 repetitions.

To sum up, we relied on

To sum up, we relied on

- the assumed Kronecker structure on the covariance of the prior on C which results in **tractable** matrix updates,

To sum up, we relied on

- the assumed Kronecker structure on the covariance of the prior on C which results in **tractable** matrix updates,
- the extended Kalman updates and automatic differentiation to obtain the Jacobian of the coefficient dynamics f_θ

To sum up, we relied on

- the assumed Kronecker structure on the covariance of the prior on C which results in **tractable** matrix updates,
- the extended Kalman updates and automatic differentiation to obtain the Jacobian of the coefficient dynamics f_θ
- gradient descent on the approximate (and tractable) marginal likelihood $\tilde{p}_\theta(y_t|y_{1:t-1})$ to optimise the parameters of f_θ

To sum up, we relied on

- the assumed Kronecker structure on the covariance of the prior on C which results in **tractable** matrix updates,
- the extended Kalman updates and automatic differentiation to obtain the Jacobian of the coefficient dynamics f_θ
- gradient descent on the approximate (and tractable) marginal likelihood $\tilde{p}_\theta(y_t|y_{1:t-1})$ to optimise the parameters of f_θ
  - leverage (again) automatic differentiation
  - take advantage of modern non-convex optimisers, such as Adam.

To sum up, we relied on

- the assumed Kronecker structure on the covariance of the prior on C which results in **tractable** matrix updates,
- the extended Kalman updates and automatic differentiation to obtain the Jacobian of the coefficient dynamics f_θ
- gradient descent on the approximate (and tractable) marginal likelihood $\tilde{p}_\theta(y_t|y_{1:t-1})$ to optimise the parameters of f_θ
    - leverage (again) automatic differentiation
    - take advantage of modern non-convex optimisers, such as Adam.

the algorithm performed well on all tasks and we also developed a robust version handling t-likelihoods.

Our reliance on automatic differentiation makes the future applications easy and fruitful. Some future directions

Our reliance on automatic differentiation makes the future applications easy and fruitful. Some future directions

- the use of neural networks for coefficient dynamics f_θ

Our reliance on automatic differentiation makes the future applications easy and fruitful. Some future directions

- the use of neural networks for coefficient dynamics f_θ
- the use of ODE/PDE solvers as f_θ (physics informed PSMF).

Our reliance on automatic differentiation makes the future applications easy and fruitful. Some future directions

- the use of neural networks for coefficient dynamics f_θ
- the use of ODE/PDE solvers as f_θ (physics informed PSMF).
- the use of switching Markov processes to model $(x_k)_{k\geq 1}$.

Thanks!

The citation:

Akyildiz, O. D., van den Burg, G., Damoulas, T., & Steel, M. (2021, March). Probabilistic sequential matrix factorization. *In International Conference on Artificial Intelligence and Statistics (AISTATS 2021)* (pp. 3484-3492). PMLR.

References I

Lee, Daniel D. and H. Sebastian Seung (Oct. 1999). "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755, pp. 788–791.

Mairal, Julien, Francis Bach, Jean Ponce, and Guillermo Sapiro (2010). "Online learning for matrix factorization and sparse coding". In: *The Journal of Machine Learning Research* 11, pp. 19–60.

Mnih, Andriy and Ruslan R Salakhutdinov (2008). "Probabilistic matrix factorization". In: *Advances in neural information processing systems*, pp. 1257–1264.

Salakhutdinov, Ruslan and Andriy Mnih (2008). "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 880–887.

References II

Cemgil, Ali Taylan (Jan. 2009). "Bayesian Inference for Nonnegative Matrix Factorisation Models". In: *Computational Intelligence and Neuroscience*, 4:1–4:17. ISSN: 1687-5265.

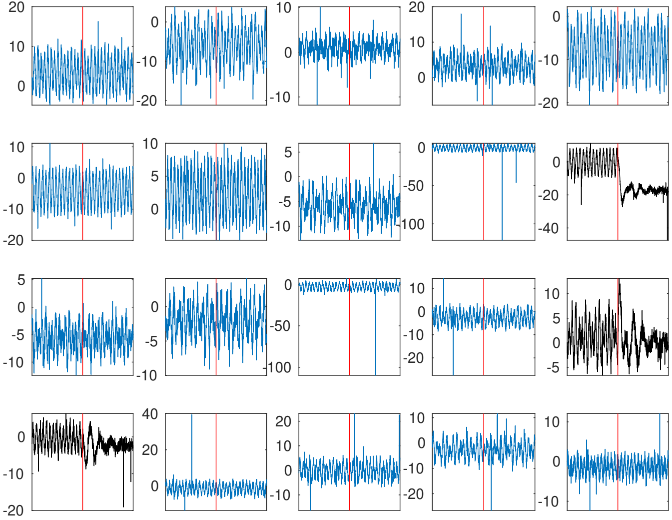
Kantas, Nikolas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, and Nicolas Chopin (2015). "On particle methods for parameter estimation in state-space models". In: *Statistical Science* 30.3, pp. 328–351.

–backup slides–

Experimental results

Changepoint detection

Consider a dataset of form



Experimental results

Changepoint detection

In order to infer the changepoint, we design a Gaussian process (GP) model using the SDE representation of Matern-$3/2$ process

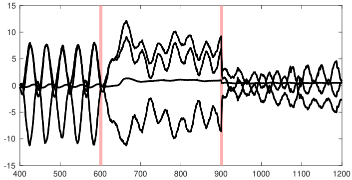
$$\mathrm{d}\mathsf{x}_i(t) = \begin{bmatrix} 0 & 1 \\ -\kappa^2 & -2\kappa \end{bmatrix} \mathsf{x}_i(t)\mathrm{d}t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{d}w_i(t) \tag{7}$$

where $\mathsf{x}_i(t) = [x_i(t), \mathrm{d}x_i(t)/\mathrm{d}t]$, $\nu = 3/2$, and $\kappa = \sqrt{2\nu}/\ell$. We choose $\sigma^2 = 0.1$ and $\ell = 0.1$ and discretize this SDE with the step-size $\gamma = 0.001$. We discretize the SDEs for $i = 1, \ldots, r$ and construct a joint state which leads to a linear dynamical system in $2r$ dimensions for which we can run PSMF.

Experimental results

Changepoint detection

What does $(x_k)_{k \geq 1}$ look like?



Comparison to classical changepoint detection methods:

	Degrees of freedom of t-contamination				
	1.5	1.6	1.7	1.8	1.9
PELT-PSMF	**85%**	**89%**	**92%**	**94%**	**95%**
PELT-Data	76%	81%	83%	85%	85%
MBOCPD	54%	58%	61%	69%	72%

Experimental results

Missing data imputation, air quality data for London

Random walk model is useful if we are just interested in imputation.

	Imputation RMSE					Runtime (s)				
	NO_2	PM10	PM25	S&P500	Gas	NO_2	PM10	PM25	S&P500	Gas
PSMF	**5.72** (0.13)	**7.44** (0.31)	3.55 (0.23)	11.56 (2.42)	**6.16** (1.07)	2.76	2.61	1.91	9.37	96.75
rPSMF	5.73 (0.22)	7.54 (0.45)	**3.50** (0.21)	**10.24** (1.67)	6.18 (1.51)	2.93	2.03	2.02	13.06	111.89
MLE-SMF	11.17 (0.58)	9.50 (0.31)	4.90 (0.36)	30.20 (0.83)	111.16 (19.95)	2.54	2.38	1.69	9.72	87.22
TMF	7.73 (0.14)	8.08 (0.22)	4.65 (0.31)	34.90 (0.79)	74.80 (8.64)	1.03	0.97	0.65	4.19	34.23
PMF*	10.51 (0.06)	10.49 (0.18)	4.05 (0.18)	40.69 (1.43)	23.77 (0.05)	1.96	1.72	0.61	2.79	28.35
BPMF*	9.22 (0.20)	8.50 (0.20)	3.68 (0.18)	27.64 (0.65)	18.31 (0.28)	2.89	2.71	1.61	3.68	91.30

Imputation error and runtime on several datasets using 30% missing values, averaged over 100 random repetitions. An asterisk marks offline methods.