

Variational Gaussian Process Timeseries Inference

Carl Edward Rasmussen

May 7th 2017

Abstract

Variational inference in nonlinear dynamical models.

The model is specified by

$$f_e(\tilde{\mathbf{x}}) \sim \mathcal{GP}(0, k_e(\cdot, \cdot)), \quad \text{where } \tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{u}) \text{ and } e = 1, \dots, E, \quad (1)$$

$$\mathbf{x}_t | \mathbf{f}_t \sim \mathcal{N}(\mathbf{x}_t | \mathbf{f}_t, \mathbf{Q}), \quad \mathbf{Q} \text{ diagonal}, \quad (2)$$

$$\mathbf{y}_t | \mathbf{x}_t \sim \mathcal{N}(\mathbf{y}_t | \mathbf{C}\mathbf{x}_t, \mathbf{R}), \quad (3)$$

and $\mathbf{u}_t, \mathbf{y}_t, t = 1, \dots, T$ are the control inputs and measurements (both observed), and $\mathbf{f}_t, t = 2, \dots, T$ and $\mathbf{x}_t, t = 1, \dots, T$ are unobserved, latent variables. The GPs implement the non-linear transition from one time point to the next conditioned on the state \mathbf{x}_{t-1} and all the previous transition pairs $\mathbf{f}_{2:t-1}, \mathbf{x}_{1:t-1}$

$$\mathbf{f}_t(\mathbf{x}_{t-1}) = p(\mathbf{f}_t | \mathbf{f}_{2:t-1}, \mathbf{x}_{1:t-1}), \text{ where } t = 2, \dots, T.$$

The joint probability of all the variables is given by the product of T observation probabilities and $T - 1$ transition probabilities

$$p(\mathbf{y}, \mathbf{x}, \mathbf{f}) = \prod_{t=1}^T p(\mathbf{y}_t | \mathbf{x}_t) \prod_{t=2}^T p(\mathbf{x}_t | \mathbf{f}_t) p(\mathbf{f}_t | \mathbf{f}_{1:t-1}, \mathbf{x}_{1:t-1}).$$

Each GP is augmented with a set of M inducing inputs \mathbf{z} and corresponding targets \mathbf{v} such that $\mathbf{v}_e = f_e(\mathbf{z}_e)$. The augmented joint is

$$p(\mathbf{y}, \mathbf{x}, \mathbf{f}, \mathbf{v}) = p(\mathbf{y} | \mathbf{x}) p(\mathbf{x}, \mathbf{f} | \mathbf{v}) p(\mathbf{v}).$$

Exact inference in the model is intractable, instead we fit the model by optimizing a variational lower bound based on an approximating distribution q , which we chose to have the following form

$$q(\mathbf{x}, \mathbf{f}, \mathbf{v}) = q(\mathbf{v}) q(\mathbf{x}) \prod_{t=2}^T p(\mathbf{f}_t | \mathbf{f}_{1:t-1}, \mathbf{x}_{1:t-1}, \mathbf{v}), \text{ where } q(\mathbf{x}) = \mathcal{N}(\mu_{\mathbf{x}}, \Sigma_{\mathbf{x}}),$$

the assumptions being that 1) the joint on \mathbf{v} and \mathbf{x} factorizes, 2) that $q(\mathbf{x})$ is Gaussian and 3) that the conditional $q(\mathbf{f} | \mathbf{x}, \mathbf{v})$ is chosen to be equal to the conditional *prior*. Generally, we would expect the variational bound to be tight if the approximating distribution is close to the *posterior*, but for tractability we are forced to set the conditional $q(\mathbf{f} | \mathbf{x}, \mathbf{v})$ to be equal to the conditional prior. This may still be a good approximation, since we are conditioning on the inducing targets \mathbf{v} . If the inducing targets are able to capture the properties of the posterior, then the bound may still be good.

The variational log marginal likelihood lower bound is a single time series (contributions for multiple time series are simply added together)

$$\begin{aligned} \mathcal{L}(\mathbf{y} | q(\mathbf{v}), q(\mathbf{x}), \theta) = & -\text{KL}(q(\mathbf{v}) || p(\mathbf{v})) + H(q(\mathbf{x})) + \sum_{t=1}^T \langle \log p(\mathbf{y}_t | \mathbf{x}_t) \rangle_{q(\mathbf{x}_t)} \\ & + \sum_{t=2}^T -\frac{1}{2} \text{tr}(\mathbf{Q}^{-1} \langle \mathbf{B}_{t-1} \rangle_{q(\mathbf{x}_{t-1})}) + \langle \log \mathcal{N}(\mathbf{x}_t | \mathbf{A}_{t-1} \mathbf{v}, \mathbf{Q}) \rangle_{q(\mathbf{v}), q(\mathbf{x}_{t-1:t})}, \end{aligned} \quad (4)$$

with the following definitions

$$\mathbf{A}_{t-1} = \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{z}) \mathbf{K}^{-1}, \text{ and } \mathbf{B}_{t-1} = \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) - \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{z}) \mathbf{K}^{-1} \mathbf{k}(\mathbf{z}, \mathbf{x}_{t-1}).$$

A free form optimization of this bound wrt $q(\mathbf{v})$ yields independent Gaussians for each GP

$$q^*(\mathbf{v}_e) = \mathcal{N}(\boldsymbol{\mu}_e = \mathbf{K}_e(\mathbf{K}_e + \Psi_{2e})^{-1}\Psi_{1e}, \Sigma_e = \mathbf{K}_e(\mathbf{K}_e + \Psi_{2e})^{-1}\mathbf{K}_e), \quad (5)$$

where $\mathbf{K}_e = \mathbf{k}_e(\mathbf{z}_e, \mathbf{z}_e)$ and we have defined the expectations

$$\Psi_1 = \sum_{t=2}^T \langle \mathbf{k}(\mathbf{z}, \mathbf{x}_{t-1}) \mathbf{Q}^{-1} \mathbf{x}_t \rangle_{q(\mathbf{x}_{t-1:t})}, \text{ and } \Psi_2 = \sum_{t=2}^T \langle \mathbf{k}(\mathbf{z}, \mathbf{x}_{t-1}) \mathbf{Q}^{-1} \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{z}) \rangle_{q(\mathbf{x}_{t-1})}, \quad (6)$$

of size $M \times E$ and $M \times M \times E$ respectively.

Plugging the optimal $q^*(\mathbf{v})$ back into the bound eq. (4), we get

$$\begin{aligned} \mathcal{L}(y|q(\mathbf{x}), \theta) &= -\text{KL}(q^*(\mathbf{v})||p(\mathbf{v})) + H(q(\mathbf{x})) + \sum_{t=1}^T \langle \log p(y_t|\mathbf{x}_t) \rangle_{q(\mathbf{x}_t)} \\ &\quad + \sum_{t=2}^T -\frac{1}{2} \langle \text{tr}(\mathbf{Q}^{-1}(\mathbf{B}_{t-1} + \mathbf{A}_{t-1}\Sigma\mathbf{A}_{t-1})) \rangle_{q(\mathbf{x}_{t-1})} + \langle \log \mathcal{N}(\mathbf{x}_t|\mathbf{A}_{t-1}\boldsymbol{\mu}, \mathbf{Q}) \rangle_{q(\mathbf{x}_{t-1:t})}. \end{aligned} \quad (7)$$

Note that except for the entropy $H(q(\mathbf{x}))$, the bound only depends on $q(\mathbf{x})$ through its pair-wise marginals. This means that the model will be identical for all $q(\mathbf{x})$ which have the same pair-wise marginals, except for an offset in the bound which depends on the entropy. We will chose $q(\mathbf{x})$ to be Markovian, ie the precision Σ_x^{-1} is block tri-diagonal.

Transition model

Writing out each term from the transition model from eq. (7) in detail

$$-\text{KL}(q^*(\mathbf{v})||p(\mathbf{v})) = -\frac{1}{2} \sum_{e=1}^E \text{tr}(\mathbf{K}_e + \Psi_{2e})^{-1}\mathbf{K}_e + \boldsymbol{\mu}_e^\top \mathbf{K}_e^{-1} \boldsymbol{\mu}_e - M - \log |(\mathbf{K}_e + \Psi_{2e})^{-1}\mathbf{K}_e|, \quad (8)$$

and

$$\begin{aligned} -\frac{1}{2} \sum_{t=2}^T \text{tr} \mathbf{Q}^{-1} \langle \mathbf{B}_{t-1} \rangle_{q(\mathbf{x}_{t-1})} &= -\frac{T-1}{2} \text{tr} \mathbf{Q}^{-1} + \frac{1}{2} \sum_{t=2}^T \text{tr} \mathbf{K}^{-1} \langle \mathbf{k}(\mathbf{z}, \mathbf{x}_{t-1}) \mathbf{Q}^{-1} \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{z}) \rangle_{q(\mathbf{x}_{t-1})} \\ &= \frac{1}{2} \sum_{e=1}^E \text{tr} \mathbf{K}_e^{-1} \Psi_{2e} - \frac{T-1}{2} \text{tr} \mathbf{Q}^{-1}, \end{aligned} \quad (9)$$

and

$$\begin{aligned} -\frac{1}{2} \sum_{t=2}^T \text{tr} \mathbf{Q}^{-1} \langle \mathbf{A}_{t-1} \Sigma \mathbf{A}_{t-1} \rangle_{q(\mathbf{x}_{t-1})} &= -\frac{1}{2} \sum_{t=2}^T \text{tr}(\Sigma \mathbf{K}^{-1} \langle \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{z}) \mathbf{Q}^{-1} \mathbf{k}(\mathbf{z}, \mathbf{x}_{t-1}) \rangle_{q(\mathbf{x}_{t-1})} \mathbf{K}^{-1}) \\ &= -\frac{1}{2} \sum_{e=1}^E \text{tr}(\mathbf{K}_e + \Psi_{2e})^{-1} \Psi_{2e}, \end{aligned} \quad (10)$$

and

$$\begin{aligned} \sum_{t=2}^T \langle \log \mathcal{N}(\mathbf{x}_t|\mathbf{A}_{t-1}\boldsymbol{\mu}, \mathbf{Q}) \rangle_{q(\mathbf{x}_{t-1:t})} &= -\frac{(T-1)E}{2} \log(2\pi) - \frac{T-1}{2} \log |\mathbf{Q}| - \frac{1}{2} \sum_{t=2}^T \langle (\mathbf{x}_t - \mathbf{A}_{t-1}\boldsymbol{\mu})^\top \mathbf{Q}^{-1} (\mathbf{x}_t - \mathbf{A}_{t-1}\boldsymbol{\mu}) \rangle_{q(\mathbf{x}_{t-1:t})} \\ &= -\frac{(T-1)E}{2} \log(2\pi) - \frac{T-1}{2} \log |\mathbf{Q}| - \frac{1}{2} \text{tr} \mathbf{Q}^{-1} \sum_{t=2}^T \langle \mathbf{x}_t^\top \mathbf{x}_t \rangle_{q(\mathbf{x}_t)} + \boldsymbol{\mu}^\top \langle \mathbf{x}_t \mathbf{Q}^{-1} \mathbf{A}_{t-1} \rangle_{q(\mathbf{x}_{t-1:t})} \\ &\quad - \frac{1}{2} \boldsymbol{\mu}^\top \mathbf{K}^{-1} \langle \mathbf{k}(\mathbf{x}_{t-1}, \mathbf{z}) \mathbf{Q}^{-1} \mathbf{k}(\mathbf{z}, \mathbf{x}_{t-1}) \rangle_{q(\mathbf{x}_{t-1})} \mathbf{K}^{-1} \boldsymbol{\mu} \\ &= -\frac{(T-1)E}{2} \log(2\pi) - \frac{T-1}{2} \log |\mathbf{Q}| - \frac{1}{2} \text{tr} \mathbf{Q}^{-1} \sum_{t=2}^T \langle \mathbf{x}_t^\top \mathbf{x}_t \rangle_{q(\mathbf{x}_t)} - \frac{1}{2} \boldsymbol{\mu}^\top \mathbf{K}^{-1} \Psi_2 \mathbf{K}^{-1} \boldsymbol{\mu} + \boldsymbol{\mu}^\top \mathbf{K}^{-1} \Psi_1. \end{aligned} \quad (11)$$

Collecting terms from eq. (8-11) which depend on Ψ_1 and Ψ_2 , two possibilities arise, either training or testing, in both cases we introduce μ and Σ from eq. (5) for the *training* cases, giving rise to

$$\begin{aligned}\Psi &= \frac{1}{2} \sum_{e=1}^E \log |K_e| - \log |K_e + \Psi_{2e}| + \text{tr} K_e^{-1} \Psi_{2e} + \Psi_{1e}^\top (K_e + \Psi_{2e})^{-1} \Psi_{1e}, \\ \Psi^* &= \frac{1}{2} \sum_{e=1}^E \text{tr} K_e^{-1} \Psi_{2e} (K_e + \Psi_{2e})^{-1} \Psi_{2e}^* - \text{tr} (K_e + \Psi_{2e})^{-1} \Psi_{1e} \Psi_{1e}^\top (K_e + \Psi_{2e})^{-1} \Psi_{2e}^* + 2 \Psi_{1e}^\top (K_e + \Psi_{2e})^{-1} \Psi_{1e}^*\end{aligned}\quad (12)$$

for training and test respectively.

Pulling together all terms from eq. (8-11) we get the following contribution

$$\begin{aligned}\frac{1}{2} \sum_{e=1}^E \log |(K_e + \Psi_{2e})^{-1} K_e| + \text{tr} K_e^{-1} \Psi_{2e} + \Psi_{1e}^\top (K_e + \Psi_{2e})^{-1} \Psi_{1e} \\ - \frac{1}{2} \text{tr} Q^{-1} \sum_{t=2}^T (I + \mu_t^\top \mu_t + \Sigma_{t,t}) - \frac{T-1}{2} \log |Q| - \frac{(T-1)E}{2} \log(2\pi).\end{aligned}\quad (13)$$

Entropy

The entropy of Markovian Gaussian with specified E dimensional marginals and $2E$ dimensional consecutive pair-wise marginals and marginals is given by

$$\mathcal{H}(q(\mathbf{x})) = \frac{TE}{2} (1 + \log(2\pi)) + \frac{1}{2} \sum_{t=2}^T \log |\Sigma_{t-1:t, t-1:t}| - \frac{1}{2} \sum_{t=2}^{T-1} \log |\Sigma_t| \quad (14)$$

3a $\langle \text{entropy } 3a \rangle \equiv$ (4d)

```

1 function [L dLd dLo] = gaussMarkovEntropy(d, o);
2 [E, E, T] = size(d); dd = zeros(T,1); dp = zeros(T-1,1);
3 for t = 1:T, dd(t) = det(d(:, :, t)); end % det of diagonals
4 for t = 1:T-1, dp(t) = dd(t)*det(d(:, :, t+1)-o(:, :, t)'/d(:, :, t)*o(:, :, t))/2; end;
5 L = E*T*(1+log(2*pi))/2 + sum(log(dp))/2 - sum(log(dd(2:T-1)))/2; % entropy
6 if nargin > 1 % want derivatives?
7   dLd = zeros(E,E,T); dLo = zeros(E,E,T-1);
8   for t = 1:T-1
9     dLd(:, :, t) = dLd(:, :, t) + inv(d(:, :, t)-o(:, :, t)/d(:, :, t+1)*o(:, :, t)')/2;
10    dLd(:, :, t+1) = inv(d(:, :, t+1)-o(:, :, t)'/d(:, :, t)*o(:, :, t))/2;
11    dLo(:, :, t) = -d(:, :, t)\o(:, :, t)/(d(:, :, t+1)-o(:, :, t)'/d(:, :, t)*o(:, :, t));
12  end
13  for t = 2:T-1, dLd(:, :, t) = dLd(:, :, t) - inv(d(:, :, t))/2; end
14 end

```

Likelihood

The linear Gaussian log likelihood is

$$\sum_{t=1}^T \langle \log p(\mathbf{y}_t | \mathbf{x}_t) \rangle_{q(\mathbf{x}_t)} = -\frac{DT}{2} \log(2\pi) - \frac{T}{2} \log |R| - \frac{1}{2} \text{tr} R^{-1} \sum_{t=1}^T ((\mathbf{y} - C\mu_t)(\mathbf{y} - C\mu_t)^\top + C\Sigma_t C^\top). \quad (15)$$

Maximizing the log likelihood wrt observation noise covariance R and the parameters C yields:

$$R^* = \frac{1}{T} \left[\sum_{t=1}^T \mathbf{y}_t \mathbf{y}_t^\top - C^* \sum_{t=1}^T \mu_t \mathbf{y}_t^\top \right], \text{ and } C^* = \sum_{t=1}^T \mathbf{y}_t \mu_t^\top \left[\sum_{t=1}^T \mu_t \mu_t^\top + \Sigma_{t,t} \right]^{-1}, \quad (16)$$

and the maximum attained is

$$\mathcal{L}^* = -\frac{DT}{2} (1 + \log(2\pi)) - \frac{T}{2} \log |R^*|, \quad (17)$$

with derivatives

$$\frac{\mathcal{L}^*}{\partial \mu_t} = C^\top R^{-1} (\mathbf{y}_t - C\mu_t), \text{ and } \frac{\mathcal{L}^*}{\partial \Sigma_{t,t}} = -\frac{1}{2} C^\top R^{-1} C, \text{ evaluated at } C = C^*, \text{ and } R = R^*. \quad (18)$$

3b $\langle \text{likelihood } 3b \rangle \equiv$ (4d)

```

1 function [L, R, C, dm, dS] = likelihood(y, qx)
2 D = size(y{1},2); E = size(qx(1).m,1); T = sum(arrayfun(@(x) size(x.m,2), qx));
3 N = size(y,2); yy = zeros(D); ym = zeros(D, E+1); mm = zeros(E+1);
4 for n = 1:N
5     m = [qx(n).m' ones(size(qx(n).m,2),1)]; mm = mm + m'*m; ym = ym + y{n}'*m;
6     yy = yy + y{n}'*y{n}; mm(1:E,1:E) = mm(1:E,1:E) + sum(qx(n).Sd,3);
7 end
8 C = ym/mm; R = (yy - C*ym')/T;
9 L = -D*T*(1+log(2*pi))/2 - T*sum(log(diag(chol(R)))); % log likelihood
10 if narginout > 3 % do we want derivatives?
11     dm = cell(N,1);
12     for n = 1:N,
13         dm{n} = C(:,1:E)'/R*(y{n}'-C*[qx(n).m; ones(1,size(qx(n).m,2))]);
14     end
15     dS = -C(:,1:E)'/R*C(:,1:E)/2; % all dS identical, return once only
16 end

```

The lower bound

Pulling together all terms

$$\begin{aligned}
 \mathcal{L}(y|q(x), \theta) = & \frac{1}{2} \sum_{e=1}^E \log |(\mathbf{K}_e + \Psi_{2e})^{-1} \mathbf{K}_e| + \text{tr} \mathbf{K}_e^{-1} \Psi_{2e} + \Psi_{1e}^T (\mathbf{K}_e + \Psi_{2e})^{-1} \Psi_{1e} + \frac{1}{2} \sum_{t=2}^T \log |\Sigma_{t-1:t, t-1:t}| \\
 & - \frac{1}{2} \sum_{t=2}^{T-1} \log |\Sigma_t| - \frac{1}{2} \text{tr} \mathbf{Q}^{-1} \sum_{t=2}^T (\mathbf{I} + \boldsymbol{\mu}_t^T \boldsymbol{\mu}_t + \Sigma_{t,t}) - \frac{T-1}{2} \log |\mathbf{Q}| - \frac{T}{2} \log |\mathbf{R}^*| - \frac{(D-E)T}{2} - \frac{TD-E}{2} \log(2\pi).
 \end{aligned} \tag{19}$$

4a $\langle \text{lower bound 4a} \rangle \equiv$ (4d)

```

1 [L1, dnlm1] = Psi(hyp, qx, z, u);
2 T = sum(arrayfun(@(x) size(x.m,2), qx)); L2 = 0; L3 = 0;
3 dLd = cell(1,N); dLo = cell(1,N);
4 for n = 1:N
5     L2 = L2 + sum(qx(n).m(:,2:end).^2,2) + diag(sum(qx(n).Sd(:, :, 2:end), 3));
6     [L dLd{n} dLo{n}] = gaussMarkovEntropy(qx(n).Sd, qx(n).So); L3 = L3 + L;
7 end
8 L5 = -exp(-2*[hyp(:).pn]) * (L2+T-N) / 2;
9 L4 = -(T-N)*sum([hyp(:).pn]) - (T-N)*E*log(2*pi)/2;
10 [L6, R, C, dm, dS] = likelihood(y, qx);
11 nlm1 = -L1-L5-L3-L4-L6;
12 %keyboard

```

4b $\langle \text{bound derivatives 4b} \rangle \equiv$ (4d)

```

1 for e = 1:E
2     dnlm1.hyp(e).pn = dnlm1.hyp(e).pn - exp(-2*hyp(e).pn)*(L2(e)+T-N)+T-N;
3 end
4 iQ = diag(exp(-2*[hyp(:).pn]));
5 for n = 1:N
6     dnlm1.qx(n).m(:,2:end) = dnlm1.qx(n).m(:,2:end) + iQ*qx(n).m(:,2:end);
7     dnlm1.qx(n).m = dnlm1.qx(n).m - dm{n};
8     dnlm1.qx(n).Sd(:, :, 2:end) = bsxfun(@plus, dnlm1.qx(n).Sd(:, :, 2:end), iQ/2);
9     dnlm1.qx(n).Sd = dnlm1.qx(n).Sd - bsxfun(@plus, dS, dLd{n});
10    dnlm1.qx(n).So = dnlm1.qx(n).So - dLo{n};
11 end
12
13 out1 = nlm1; out2 = dnlm1; out3 = struct('C', C, 'R', R); % rename outputs

```

4c $\langle \text{predictions 4c} \rangle \equiv$ (4d)

```

1 [Psi1, Psi2] = Psi(hyp, qx, z, u);

```

4d $\langle \text{vgpt.m 4d} \rangle \equiv$

```

1 function [out1, out2, out3] = vgpt(p, data, x);
2 %usage 5a
3

```

```

4 N = length(p.qx); z = p.z; [M, F, E] = size(z); D = size(data(1).y,2); hyp = p.hyp;
5 u = arrayfun(@(x)(x.u),data,'UniformOutput',false); [qx(1:N).m] = deal(p.qx(:).m);
6 y = arrayfun(@(x)(x.y),data,'UniformOutput',false);
7 for n = 1:N, [qx(n).Sd qx(n).So] = convert(p.qx(n).s); end % convert covariance
8
9 if nargin == 2
10     <lower bound 4a>
11     <bound derivatives 4b>
12     <revert covariances 9a>
13 else
14     <predictions 4c>
15 end
16
17 <Psi 5b>
18 <entropy 3a>
19 <likelihood 3b>
20 <convert 8>
21 <revert 9b>
22 <maha 9c>

```

5a *<usage 5a>*≡ (4d)

```

1 % Variational GP Timeseries inference. Compute the nlml lower bound and its
2 % derivative wrt hyp hyperparameters, qx distribution and z inducing inputs.
3 %
4 % p                parameter struct
5 %   hyp            1 x E    GP hyperparameter struct
6 %   l              F x 1    log length scale
7 %   pn            1 x 1    log process noise std dev
8 %   qx            1 x N    struct array for Gaussian q(x) distribution
9 %   m              E x T_n  mean
10 %   s             2xE x T_n representation of covariance
11 %   z             M x F x E inducing inputs
12 % data            1 x N    data struct
13 %   y             T_n x D   cell array of observations
14 %   u             T_n x U   cell array of control inputs
15 %
16 % Copyright (C) 2016 by Carl Edward Rasmussen, 20160530.

```

The Ψ function

In the implementation, a function `Psi` handles the part of the (negative) log marginal likelihood which depends on the quantities Ψ_1 and Ψ_2 :

$$\psi = \frac{1}{2} \sum_{e=1}^E \log |K_e| - \log |K_e + \Psi_{2e}| - \text{tr} K_e^{-1} \Psi_{2e} - \Psi_{1e}^T (K_e + \Psi_{2e})^{-1} \Psi_{1e}. \quad (20)$$

5b *<Psi 5b>*≡ (4d)

```

1 function [lml, dnlml] = Psi(hyp, qx, z, u, test);
2 % hyp            1 x E    GP hyperparameter struct
3 %   l            F x 1    log length scale
4 %   pn          1 x 1    log process noise std dev
5 %   qx          1 x N    Gaussian q(x) distribution
6 %   m            E x T_n  mean
7 %   Sd          ExE x T_n diagonal elements of covariance matrix
8 %   So          ExE x T_n-1 immediately off-diagonal elements of covariance matrix
9 %   z            M x F x E inducing inputs
10 % u            T_n x U   cell array of control inputs
11 % lml           1 x 1    contribution to the log marginal likelihood
12 % dnlml         derivatives
13
14 persistent K Psi1 Psi2; % keep these around if necessary
15 [M, F, E] = size(z); % get sizes

```

```

16 test = 0;
17 <expectations 6>
18 if nargout > 0
19     <expectation derivatives 7a>
20 end

```

The Ψ_1 and Ψ_2 expectations

The expectations from eq. (6) and derivatives wrt hyperparameters, the parameters of the $q(\mathbf{x})$ distribution and the pseudo-inputs \mathbf{z} are calculated by the Psi function. To compute these expectations, the pairwise joint

$$q(\mathbf{x}_{t-1:t}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{t-1} \\ \boldsymbol{\mu}_t \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{t-1,t-1} & \boldsymbol{\Sigma}_{t-1,t} \\ \boldsymbol{\Sigma}_{t,t-1} & \boldsymbol{\Sigma}_{t,t} \end{bmatrix}\right),$$

is multiplied with the covariance function, which can be written as an un-normalized joint Gaussian

$$k_e(\mathbf{x}_{t-1}, \mathbf{z}_{ie}) = \exp\left(-\frac{1}{2} \begin{bmatrix} \mathbf{x}_{t-1} - \mathbf{z}_{ie} \\ \mathbf{x}_t \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Lambda}_e^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{t-1} - \mathbf{z}_{ie} \\ \mathbf{x}_t \end{bmatrix}\right),$$

yielding

$$\int \mathbf{x}_t k_e(\mathbf{x}_{t-1}, \mathbf{z}_{ie}) q(\mathbf{x}_{t-1:t}) d\mathbf{x}_{t-1} d\mathbf{x}_t = (\boldsymbol{\mu}_t + \boldsymbol{\Sigma}_{t,t-1} [\boldsymbol{\Lambda}_e + \boldsymbol{\Sigma}_{t-1,t-1}]^{-1} (\mathbf{z}_{ie} - \boldsymbol{\mu}_{t-1})) \times |I + \boldsymbol{\Lambda}_e^{-1} \boldsymbol{\Sigma}_{t-1,t-1}|^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{\mu}_{t-1} - \mathbf{z}_{ie}) [\boldsymbol{\Lambda}_e + \boldsymbol{\Sigma}_{t-1,t-1}]^{-1} (\boldsymbol{\mu}_{t-1} - \mathbf{z}_{ie})\right). \quad (21)$$

For Ψ_2 we have from eq. (6)

$$\int k_e(\mathbf{z}_{ie}, \mathbf{x}_{t-1}) k_e(\mathbf{x}_{t-1}, \mathbf{z}_{je}) q(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} = \exp(-(\mathbf{z}_{ie} - \mathbf{z}_{je}) \boldsymbol{\Lambda}_e^{-1} (\mathbf{z}_{ie} - \mathbf{z}_{je})/4) \times |I + 2\boldsymbol{\Lambda}_e^{-1} \boldsymbol{\Sigma}_{t-1,t-1}|^{-1/2} \exp(-(\frac{\mathbf{z}_{ie} + \mathbf{z}_{je}}{2} - \boldsymbol{\mu}_{t-1}) [\boldsymbol{\Lambda}_e/2 + \boldsymbol{\Sigma}_{t-1,t-1}]^{-1} (\frac{\mathbf{z}_{ie} + \mathbf{z}_{je}}{2} - \boldsymbol{\mu}_{t-1})/2). \quad (22)$$

Both Ψ_1 and Ψ_2 are computed for each GP $e = 1, \dots, E$, each inducing input \mathbf{z}_{ie} , $i = 1, \dots, M$, and added over (N time series and) $T_n - 1$ time points:

```

6 <expectations 6>= (5b)
1 K = zeros(M,M,E); Psi1 = zeros(M,E); Psi2 = zeros(M,M,E); Sd = zeros(F,F);
2 lml = 0;
3 for e = 1:E % for each GP
4     K(:,:,e) = exp(-maha(z(:,:,e), [], diag(exp(-2*hyp(e).l))))/2) + 1e-6*eye(M);
5     iL = diag(exp(-hyp(e).l)); L2 = diag(exp(2*hyp(e).l));
6     b1 = zeros(M,1); b2 = zeros(M,M);
7     for n = 1:length(qx) % for each time series
8         for t = 2:size(qx(n).m, 2) % for each time step
9             Sd(1:E,1:E) = qx(n).Sd(:,:,t-1); % covariance in top left corner
10            r1 = prod(diag(chol(eye(F)+iL*Sd*iL))); % sqrt det
11            r2 = prod(diag(chol(eye(F)+2*iL*Sd*iL))); % sqrt det
12            s = bsxfun(@minus, z(:,:,e), [qx(n).m(:,t-1)' u{n}(t-1,:)]);
13            a = s/(L2+Sd);
14            b1 = b1 + (qx(n).m(e,t) + a(:,1:E)*qx(n).So(:,e,t-1)) ...
15                .*exp(-sum(a.*s,2)/2)/r1;
16            b2 = b2 + exp(-maha(s,-s,inv(L2+2*Sd))/4) / r2;
17        end
18    end
19    if test
20        w = (K(:,:,e)+Psi2(:,:,e))\Psi1(:,e);
21        W = -K(:,:,e)\Psi2(:,:,e)/(Psi2(:,:,e)+K(:,:,e)) + w*w';
22        lml = lml + exp(-2*hyp(e).pn)*(b1'*Psi1(:,e) ...
23            - sum(sum(b2.*exp(-maha(z(:,:,e), [], inv(L2))/4).*W))/2);
24    else
25        Psi1(:,e) = b1 * exp(-2*hyp(e).pn);
26        Psi2(:,:,e) = b2 * exp(-2*hyp(e).pn) .* exp(-maha(z(:,:,e), [], inv(L2))/4);
27        lml = lml - sum(log(diag(chol(K(:,:,e)+Psi2(:,:,e)))))) + ...
28            sum(log(diag(chol(K(:,:,e))))) + trace(K(:,:,e)\Psi2(:,:,e))/2 + ...
29            Psi1(:,e)'/(K(:,:,e)+Psi2(:,:,e))*Psi1(:,e)/2;
30    end
31 end

```

Note that in the implementation the state distribution $q(\mathbf{x})$ is concatenated with the (deterministic) control inputs \mathbf{u} .

Psi derivatives

We need to compute the derivatives of Ψ wrt the parameters of the $q(\mathbf{x})$ distribution, wrt the \mathbf{u} inducing inputs and wrt the hyperparameters. First, from eq. (20) we note

$$\begin{aligned}\frac{\partial \Psi}{\partial \Psi_{1e}} &= -(\mathbf{K}_e + \Psi_{2e})^{-1} \Psi_{1e} = -\mathbf{w}_e, \\ \frac{\partial \Psi}{\partial \Psi_{2e}} &= -\frac{1}{2} \mathbf{K}_e^{-1} \Psi_{2e} (\mathbf{K}_e + \Psi_{2e})^{-1} + \frac{1}{2} \mathbf{w}_e \mathbf{w}_e^\top = -\frac{1}{2} \mathbf{R}_e (\mathbf{K}_e + \Psi_{2e})^{-1} + \frac{1}{2} \mathbf{w}_e \mathbf{w}_e^\top = \mathbf{W}_e, \\ \frac{\partial \Psi}{\partial \mathbf{K}_e} &= -\frac{1}{2} \mathbf{R}_e (\mathbf{K}_e + \Psi_{2e})^{-1} \mathbf{R}_e^\top + \frac{1}{2} \mathbf{w}_e \mathbf{w}_e^\top = \mathbf{V}_e,\end{aligned}$$

where we have defined $\mathbf{w}_e = (\mathbf{K}_e + \Psi_{2e})^{-1} \Psi_{1e}$ and $\mathbf{R}_e = \mathbf{K}_e^{-1} \Psi_{2e}$. These can be used together with the derivatives of Ψ_{1e} , Ψ_{2e} and \mathbf{K}_e and the chain rule to get the desired derivatives.

```

7a <expectation derivatives 7a>≡ (5b)
1 dnlml.z = zeros(M,F,E);
2 for n = 1:size(qx,2), dnlml.qx(n).m = 0*qx(n).m; dnlml.qx(n).So = 0*qx(n).So;
3 dnlml.qx(n).Sd = -0*qx(n).Sd; end;
4 for e = 1:E
5     w = (K(:,:,e)+Psi2(:,:,e))\Psi1(:,e);
6     R = K(:,:,e)\Psi2(:,:,e);
7     W = -R/(Psi2(:,:,e)+K(:,:,e)) + w*w';
8     <hyp derivatives 7b>
9     <Psi derivatives 7c>
10 end

7b <hyp derivatives 7b>≡ (7a)
1 dnlml.hyp(e).pn = 2*sum(w.*Psi1(:,e)) - sum(sum(W.*Psi2(:,:,e)));

7c <Psi derivatives 7c>≡ (7a)
1 iL = diag(exp(-hyp(e).l)); L2 = diag(exp(2*hyp(e).l));
2 W1 = W .* exp(-maha(z(:,:,e),[],inv(L2))/4);
3 D = zeros(M,F); H = zeros(F,1);
4 for n = 1:length(qx)
5     T = size(qx(n).m,2);
6     A = zeros(E,T); B = zeros(E,E,T); C = zeros(E,E,T-1);
7     for t = 2:T
8         Sd(1:E,1:E) = qx(n).Sd(:,:,t-1); % covariance in top left corner
9         r2 = prod(diag(chol(eye(F)+2*iL*Sd*iL))); % sqrt det
10        s = bsxfun(@minus, z(:,:,e), [qx(n).m(:,t-1)' u{n}(t-1,:)]);
11        a = s/(L2+Sd);
12        a2 = s/(2*L2+4*Sd);
13        SiS = (L2(1:E,1:E)+Sd(1:E,1:E))\qx(n).So(:,e,t-1);
14        r = exp(-sum(a.*s,2)/2) / prod(diag(chol(eye(F)+iL*Sd*iL)));
15        g = (qx(n).m(e,t) + a(:,1:E)*qx(n).So(:,e,t-1)).*w.*r;
16        W2 = W1.*exp(-maha(s,-s,inv(L2+2*Sd))/4);
17        X = bsxfun(@plus,permute(a2,[1 3 2]),permute(a2,[3 1 2]));
18        A(:,t-1) = A(:,t-1) + SiS*(w'*r) - a(:,1:E)'*g + ...
19            squeeze(sum(sum(bsxfun(@times,W2,X(:,:,1:E)),2),1))/r2;
20        A(e,t) = -w'*r;
21        B(:,:,t-1) = squeeze(sum(sum(bsxfun(@times, ...
22            bsxfun(@times,W2,X(:,:,1:E)),permute(X(:,:,1:E),[1 2 4 3])),2),1))/r2;
23        B(:,:,t-1) = B(:,:,t-1) + SiS*((w.*r)'*a(:,1:E)) ...
24            + inv(L2(1:E,1:E)+Sd(1:E,1:E))*sum(g)/2 ...
25            - a(:,1:E)'*bsxfun(@times,g,a(:,1:E))/2 ...
26            - inv(L2(1:E,1:E)+2*Sd(1:E,1:E))*sum(sum(W2))/r2/2;
27        C(:,e,t-1) = -bsxfun(@times,a(:,1:E),r)*w;
28        if ~test
29            D(:,1:E) = D(:,1:E) - bsxfun(@times,w,r)*SiS';

```

```

30     D = D + bsxfun(@times,g,a) - W2*a2/r2 - bsxfun(@times,sum(W2,2),a2)/r2;
31     H = H + diag(Sd/(L2+2*Sd))*sum(sum(W2))/r2 ...
32         + exp(2*hyp(e).1).*squeeze(sum(sum(bsxfun(@times,W2,X.^2),2),1))/r2;
33     H(1:E) = H(1:E) ...
34         + 2*exp(2*hyp(e).1(1:E)).*diag(SiS*bsxfun(@times,w,r)'+a(:,1:E));
35     H = H - diag(Sd/(L2+Sd))*sum(g);
36     H = H - exp(2*hyp(e).1).*diag(a'*bsxfun(@times,g,a));
37 end
38 end
39 dn1ml.qx(n).m = dn1ml.qx(n).m + A * exp(-2*hyp(e).pn);
40 dn1ml.qx(n).Sd = dn1ml.qx(n).Sd + B * exp(-2*hyp(e).pn);
41 dn1ml.qx(n).So = dn1ml.qx(n).So + C * exp(-2*hyp(e).pn);
42 end
43 if ~test
44 G = W.*Psi2(:, :, e);
45 a = z(:, :, e)*diag(exp(-2*hyp(e).1)/2);
46 dn1ml.z(:, :, e) = D*exp(-2*hyp(e).pn) + G*a - bsxfun(@times,sum(G,2),a);
47 B = bsxfun(@minus,permute(a,[1 3 2]),permute(a,[3 1 2]));
48 dn1ml.hyp(e).1 = H * exp(-2*hyp(e).pn) ...
49     + exp(2*hyp(e).1).*squeeze(sum(sum(bsxfun(@times,G,B.^2),1),2));
50 G = (R/(K(:, :, e)+Psi2(:, :, e))*R' + w*w').*K(:, :, e);
51 a = z(:, :, e)*diag(exp(-2*hyp(e).1));
52 B = bsxfun(@minus,permute(z(:, :, e),[1 3 2]),permute(z(:, :, e),[3 1 2]));
53 dn1ml.hyp(e).1 = dn1ml.hyp(e).1 ...
54     + exp(-2*hyp(e).1).*squeeze(sum(sum(bsxfun(@times,B.^2,G),1),2))/2;
55 dn1ml.z(:, :, e) = dn1ml.z(:, :, e) + G*a - bsxfun(@times,sum(G,2),a);
56 end

```

Test set calculation

The distinguishing factor between training and test set calculations is whether the inducing target distribution is updated (training set) or kept fixed (test set). For the test set the contribution from the transition model to the log probability is

$$\begin{aligned}
& \sum_{e=1}^E \frac{1}{2} \text{tr} K_e^{-1} \Psi_{2e} (K_e + \Psi_{2e})^{-1} \Psi_{2e}^* - \frac{1}{2} \text{tr} (K_e + \Psi_{2e})^{-1} \Psi_{1e} \Psi_{1e}^\top (K_e + \Psi_{2e})^{-1} \Psi_{2e}^* + \Psi_{1e}^\top (K_e + \Psi_{2e})^{-1} \Psi_{1e}^* \\
& + \frac{1}{2} \sum_{t=2}^T \log |\Sigma_{t-1:t, t-1:t}| - \frac{1}{2} \sum_{t=2}^{T-1} \log |\Sigma_t| - \frac{1}{2} \text{tr} Q^{-1} \sum_{t=2}^T (I + \mu_t^\top \mu_t + \Sigma_{t,t}) - \frac{T-1}{2} \log |Q| - \frac{(T-1)E}{2} \log(2\pi). \quad (23)
\end{aligned}$$

Representation of the $q(\mathbf{x})$ distribution

The $q(\mathbf{x})$ distribution is parameterised through its mean $\mathbf{q}\mathbf{x}.\mathbf{m}$ and the marginal and pairwise covariances. Conceptually, we wish to parameterize the E by E covariance matrices (for the marginal distributions) which we call $\mathbf{q}\mathbf{x}.\mathbf{Sd}$ (for diagonal) and the E by E covariances between consecutive time points (for the pairwise marginals) which we call $\mathbf{q}\mathbf{x}.\mathbf{So}$ (for off-diagonal). However it is inconvenient to parametrise these matrices directly, as it would be difficult to ensure positive definiteness of the marginal and pairwise marginal covariance matrices. Instead, we use 2E by E representation $\mathbf{q}\mathbf{x}.\mathbf{s}$ such that

$$\mathbf{S}_{d,t} = \mathbf{s}_t^\top \mathbf{s}_t, \text{ and } \mathbf{S}_{o,t-1} = \mathbf{s}_{t-1}^\top \mathbf{s}_t. \quad (24)$$

Using this representation, we can use call the optimizer with the unconstrained representation, which is the converted to the more convenient diagonal and off-diagonal representation at the beginning and the derivatives are reverted back at the end.

```

8  <convert 8>≡
1  function [Sd, So] = convert(s)                                     (4d)
2  [t, E, T] = size(s); Sd = zeros(E,E,T); So = zeros(E,E,T-1);
3  for t = 1:T, Sd(:, :, t) = s(:, :, t)'*s(:, :, t); end           % diagonal terms
4  for t = 2:T, So(:, :, t-1) = s(:, :, t-1)'*s(:, :, t); end      % off-diagonal

```


9a $\langle \text{revert covariances } 9a \rangle \equiv$ (4d)

```

1 out2.qx = rmfield(out2.qx,{'Sd','So'}); % change to qx.s representation
2 [out2.qx.s] = deal([]); % create the "s" field
3 %for n = 1:N
4 % Sd = sum(dnlml.qx(n).Sd,3) / size(dnlml.qx(n).Sd,3);
5 % for t = 1:size(dnlml.qx(n).Sd,3), dnlml.qx(n).Sd(:,t) = Sd; end
6 %end
7 for n = 1:N
8 out2.qx(n).s = revert(p.qx(n).s, dnlml.qx(n).Sd, dnlml.qx(n).So);
9 end

```

The derivatives are

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial s_t} &= \frac{\partial \mathcal{L}}{\partial S_{d,t}} \frac{\partial S_{d,t}}{\partial s_t} + \frac{\partial \mathcal{L}}{\partial S_{o,t-1}} \frac{\partial S_{o,t-1}}{\partial s_t} + \frac{\partial \mathcal{L}}{\partial S_{o,t}} \frac{\partial S_{o,t}}{\partial s_t} = \frac{\partial}{\partial s_t} \text{tr} \left(\frac{\partial \mathcal{L}}{\partial S_{d,t}} s_t^\top s_t \right) \\
&+ s_{t-1} \frac{\partial \mathcal{L}}{\partial S_{o,t-1}} + s_t \left[\frac{\partial \mathcal{L}}{\partial S_{o,t}} \right]^\top = s_t \left(\frac{\partial \mathcal{L}}{\partial S_{d,t}} + \left[\frac{\partial \mathcal{L}}{\partial S_{d,t}} \right]^\top \right) + s_{t-1} \frac{\partial \mathcal{L}}{\partial S_{o,t-1}} + s_t \left[\frac{\partial \mathcal{L}}{\partial S_{o,t}} \right]^\top.
\end{aligned} \tag{25}$$

9b $\langle \text{revert } 9b \rangle \equiv$ (4d)

```

1 function r = revert(s, dSd, dSo)
2 for t = 1:size(s,3), r(:,t) = s(:,t)*(dSd(:,t)+dSd(:,t)'); end
3 for t = 2:size(s,3)
4 r(:,t-1) = r(:,t-1) + s(:,t)*dSo(:,t-1)';
5 r(:,t) = r(:,t) + s(:,t-1)*dSo(:,t-1);
6 end

```

9c $\langle \text{maha } 9c \rangle \equiv$ (4d)

```

1 % Squared Mahalanobis distance (a-b)*Q*(a-b)'; vectors are row-vectors
2 % a, b d x n matrices containing n length d row vectors
3 % Q d x d weight matrix
4 % K n x n squared distances
5 function K = maha(a, b, Q)
6 if isempty(b), b = a; end
7 aQ = a*Q; K = bsxfun(@plus,sum(aQ.*a,2),sum(b*Q.*b,2)')-2*aQ*b';

```