Variational Gaussian Process Timeseries Inference

Carl Edward Rasmussen

January 8th 2016

Abstract

Variational inference in nonlinear dynamical models.

The model is specified by

$$
\begin{aligned}
f_e(\tilde{x}) &\sim \mathcal{GP}\big(0, k_e(\cdot,\cdot)\big), && \text{where } \tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{u}) \text{ and } e = 1, \ldots, E, && (1)\\
\mathbf{x}_t|\mathbf{f}_t &\sim \mathcal{N}(\mathbf{x}_t|\mathbf{f}_t, Q), && Q \text{ diagonal}, && (2)\\
\mathbf{y}_t|\mathbf{x}_t &\sim \mathcal{N}(\mathbf{y}_t|C\mathbf{x}_t, R), && && (3)
\end{aligned}
$$

and $\mathbf{u}_t, \mathbf{y}_t, t = 1, \ldots, T$ are the control inputs and measurements (both observed), and $\mathbf{f}_t, t = 2, \ldots, T$ and $\mathbf{x}_t, t = 1, \ldots, T$ 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 $\mathbf{f}_{2:t-1}, \mathbf{x}_{1:t-2}$

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

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

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

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)

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

with the following definitions

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

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

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

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

$$\Psi_1 \;=\; \sum_{t=2}^{T}\langle k(\mathbf{z},\mathbf{x}_{t-1})Q^{-1}\mathbf{x}_t\rangle_{q(\mathbf{x}_{t-1:t})}, \;\text{ and }\; \Psi_2 \;=\; \sum_{t=2}^{T}\langle k(\mathbf{z},\mathbf{x}_{t-1})Q^{-1}k(\mathbf{x}_{t-1},\mathbf{z})\rangle_{q(\mathbf{x}_{t-1})}, \tag{5}$$

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

$$\begin{aligned}\mathcal{L}(\mathbf{y}|q(\mathbf{x}),\theta) \;=&\; -\mathrm{KL}(q^*(\mathbf{v})\|p(\mathbf{v})) + H(q(\mathbf{x})) + \sum_{t=1}^{T}\langle\log p(\mathbf{y}_t|\mathbf{x}_t)\rangle_{q(\mathbf{x}_t)}\\ &+\sum_{t=2}^{T} -\tfrac{1}{2}\langle\mathrm{tr}\big(Q^{-1}(B_{t-1}+A_{t-1}\Sigma A_{t-1})\big)\rangle_{q(\mathbf{x}_{t-1})} + \langle\log\mathcal{N}(\mathbf{x}_t|A_{t-1}\boldsymbol{\mu},Q)\rangle_{q(\mathbf{x}_{t-1:t})}.\end{aligned} \tag{6}$$

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. (6) in detail

$$-\mathrm{KL}(q^*(\mathbf{v})\|p(\mathbf{v})) \;=\; -\tfrac{1}{2}\sum_{e=1}^{E}\mathrm{tr}(K_e+\Psi_{2e})^{-1}K_e + \boldsymbol{\mu}_e^\top K_e^{-1}\boldsymbol{\mu}_e - M - \log|(K_e+\Psi_{2e})^{-1}K_e|, \tag{7}$$

and

$$\begin{aligned}-\tfrac{1}{2}\sum_{t=2}^{T}\mathrm{tr}Q^{-1}\langle B_{t-1}\rangle_{q(x_{t-1})} \;=&\; -\tfrac{T-1}{2}\mathrm{tr}Q^{-1} + \tfrac{1}{2}\sum_{t=2}^{T}\mathrm{tr}K^{-1}\langle k(\mathbf{z},\mathbf{x}_{t-1})Q^{-1}k(\mathbf{x}_{t-1},\mathbf{z})\rangle_{q(x_{t-1})}\\ =&\; \tfrac{1}{2}\sum_{e=1}^{E}\mathrm{tr}K_e^{-1}\Psi_{2e} - \tfrac{T-1}{2}\mathrm{tr}Q^{-1},\end{aligned} \tag{8}$$

and

$$\begin{aligned}-\tfrac{1}{2}\sum_{t=2}^{T}\mathrm{tr}Q^{-1}\langle A_{t-1}\Sigma A_{t-1}\rangle_{q(x_{t-1})} \;=&\; -\tfrac{1}{2}\sum_{t=2}^{T}\mathrm{tr}\big(\Sigma K^{-1}\langle k(x_{t-1},\mathbf{z})Q^{-1}k(\mathbf{z},x_{t-1})\rangle_{q(x_{t-1})}K^{-1}\big)\\ =&\; -\tfrac{1}{2}\sum_{e=1}^{E}\mathrm{tr}(K_e+\Psi_{2e})^{-1}\Psi_{2e},\end{aligned} \tag{9}$$

and

$$\begin{aligned}&\sum_{t=2}^{T}\langle\log\mathcal{N}(\mathbf{x}_t|A_{t-1}\boldsymbol{\mu},Q)\rangle_{q(x_{t-1:t})}\\ &\quad= -\tfrac{(T-1)E}{2}\log(2\pi) - \tfrac{T-1}{2}\log|Q| - \tfrac{1}{2}\sum_{t=2}^{T}\langle(\mathbf{x}_t-A_{t-1}\boldsymbol{\mu})^\top Q^{-1}(\mathbf{x}_t-A_{t-1}\boldsymbol{\mu})\rangle_{q(x_{t-1:t})}\\ &\quad= -\tfrac{(T-1)E}{2}\log(2\pi) - \tfrac{T-1}{2}\log|Q| - \tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}\langle\mathbf{x}_t^\top\mathbf{x}_t\rangle_{q(x_t)} + \boldsymbol{\mu}^\top\langle\mathbf{x}_tQ^{-1}A_{t-1}\rangle_{q(x_{t-1:t})}\\ &\qquad\qquad - \tfrac{1}{2}\boldsymbol{\mu}^\top K^{-1}\langle k(\mathbf{x}_{t-1},\mathbf{z})Q^{-1}k(\mathbf{z},\mathbf{x}_{t-1})\rangle_{q(x_{t-1})}K^{-1}\boldsymbol{\mu}\\ &\quad= -\tfrac{(T-1)E}{2}\log(2\pi) - \tfrac{T-1}{2}\log|Q| - \tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}\langle\mathbf{x}_t^\top\mathbf{x}_t\rangle_{q(x_t)} - \tfrac{1}{2}\boldsymbol{\mu}^\top K^{-1}\Psi_2K^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}^\top K^{-1}\Psi_1.\end{aligned} \tag{10}$$

Pulling together all terms from eq. (7-10) we get the following contribution

$$\begin{aligned}&\tfrac{1}{2}\sum_{e=1}^{E}\log|(K_e+\Psi_{2e})^{-1}K_e|+\mathrm{tr}K_e^{-1}\Psi_{2e}+\Psi_{1e}^\top(K_e+\Psi_{2e})^{-1}\Psi_{1e}\\ &-\tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}(I+\mu_t^\top\mu_t+\Sigma_{t,t})-\tfrac{T-1}{2}\log|Q|-\tfrac{(T-1)E}{2}\log(2\pi).\end{aligned}\tag{11}$$

Entropy

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

$$\mathcal{H}(q(\mathbf{x})) \;=\; \tfrac{TE}{2}(1+\log(2\pi))+\tfrac{1}{2}\sum_{t=2}^{T}\log|\Sigma_{t-1:t,t-1:t}|-\tfrac{1}{2}\sum_{t=2}^{T-1}\log|\Sigma_t| \tag{12}$$

3a ⟨*entropy* 3a⟩≡ (4d)

```
function [L dLd dLo] = gaussMarkovEntropy(d, o);
[E, E, T] = size(d); dd = zeros(T,1); dp = zeros(T-1,1);
for t = 1:T, dd(t) = det(d(:,:,t)); end                        % det of diagonals
for t = 1:T-1, dp(t) = dd(t)*det(d(:,:,t+1)-o(:,:,t)'/d(:,:,t)*o(:,:,t)); end;
L = E*T*(1+log(2*pi))/2 + sum(log(dp))/2 - sum(log(dd(2:T-1)))/2;      % entropy
if nargout > 1                                                 % want derivatives?
  dLd = zeros(E,E,T); dLo = zeros(E,E,T-1);
  for t = 1:T-1
    dLd(:,:,t) = dLd(:,:,t) + inv(d(:,:,t)-o(:,:,t)/d(:,:,t+1)*o(:,:,t)')/2;
    dLd(:,:,t+1) = inv(d(:,:,t+1)-o(:,:,t)'/d(:,:,t)*o(:,:,t))/2;
    dLo(:,:,t) = -d(:,:,t)\o(:,:,t)/(d(:,:,t+1)-o(:,:,t)'/d(:,:,t)*o(:,:,t));
  end
  for t = 2:T-1, dLd(:,:,t) = dLd(:,:,t) - inv(d(:,:,t))/2; end
end
```

Likelihood

The linear Gaussian log likelihood is

$$\sum_{t=1}^{T}\langle\log p(\mathbf{y}_t|\mathbf{x}_t)\rangle_{q(\mathbf{x}_t)} \;=\; -\tfrac{DT}{2}\log(2\pi)-\tfrac{T}{2}\log|R|-\tfrac{1}{2}\mathrm{tr}R^{-1}\sum_{t=1}^{T}\big((\mathbf{y}-C\mu_t)(\mathbf{y}-C\mu_t)^\top+C\Sigma_tC^\top\big). \tag{13}$$

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

$$R^* \;=\; \tfrac{1}{T}\Big[\sum_{t=1}^{T}y_ty_t^\top-C^*\sum_{t=1}^{T}\mu_t y^\top\Big],\ \text{ and }\ C^* \;=\; \sum_{t=1}^{T}y_t\mu_t^\top\Big[\sum_{t=1}^{T}\mu_t\mu_t^\top+\Sigma_{t,t}\Big]^{-1}, \tag{14}$$

and the maximum attained is

$$\mathcal{L}^* \;=\; -\tfrac{DT}{2}(1+\log(2\pi))-\tfrac{T}{2}\log|R^*|, \tag{15}$$

with derivatives

$$\frac{\mathcal{L}^*}{\partial\mu_t} \;=\; C^\top R^{-1}(y_t-C\mu_t),\ \text{ and }\ \frac{\mathcal{L}^*}{\partial\Sigma_{t,t}} \;=\; -\tfrac{1}{2}C^\top R^{-1}C,\ \text{ evaluated at }\ C=C^*,\ \text{ and }\ R=R^*. \tag{16}$$

3b ⟨*likelihood* 3b⟩≡ (4d)

```
function [L, R, C, dm, dS] = likelihood(y, qx)
D = size(y{1},2); E = size(qx(1).m,1); T = sum(arrayfun(@(x)size(x.m,2),qx));
N = size(y,2); yy = zeros(D); ym = zeros(D, E+1); mm = zeros(E+1);
for n = 1:N
  m = [qx(n).m' ones(size(qx(n).m,2),1)]; mm = mm + m'*m; ym = ym + y{n}'*m;
  yy = yy + y{n}'*y{n}; mm(1:E,1:E) = mm(1:E,1:E) + sum(qx(n).Sd,3);
end
C = ym/mm; R = (yy - C*ym')/T;
L = -D*T*(1+log(2*pi))/2 - T*sum(log(diag(chol(R))));             % log likelihood
```

```
if nargout > 3                                     % do we want derivatives?
  dm = cell(N,1);
  for n = 1:N,
    dm{n} = C(:,1:E)'/R*(y{n}'-C*[qx(n).m; ones(1,size(qx(n).m,2))]);
  end
  dS = -C(:,1:E)'/R*C(:,1:E)/2;             % all dS identical, return once only
end
```

The lower bound

Pulling together all terms

$$
\begin{aligned}
\mathcal{L}(\mathbf{y}|q(\mathbf{x}),\theta) \;=\;& \tfrac{1}{2}\sum_{e=1}^{E} \log|(K_e+\Psi_{2e})^{-1}K_e| + \mathrm{tr}K_e^{-1}\Psi_{2e} + \Psi_{1e}^{\top}(K_e+\Psi_{2e})^{-1}\Psi_{1e} + \tfrac{1}{2}\sum_{t=2}^{T}\log|\Sigma_{t-1:t,t-1:t}| \\
&- \tfrac{1}{2}\sum_{t=2}^{T-1}\log|\Sigma_t| - \tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}(I+\mu_t^{\top}\mu_t+\Sigma_{t,t}) - \tfrac{T-1}{2}\log|Q| - \tfrac{T}{2}\log|R^*| - \tfrac{(D-E)T}{2} - \tfrac{TD-E}{2}\log(2\pi).
\end{aligned} \tag{17}
$$

4a ⟨*lower bound* 4a⟩≡ (4d)

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

4b ⟨*bound derivatives* 4b⟩≡ (4d)

```
for e = 1:E
  dnlml.hyp(e).pn = dnlml.hyp(e).pn - exp(-2*hyp(e).pn)*(L2(e)+T-N)+T-N;
end
iQ = diag(exp(-2*[hyp(:).pn]));
for n = 1:N
  dnlml.qx(n).m(:,2:end) = dnlml.qx(n).m(:,2:end) + iQ*qx(n).m(:,2:end);
  dnlml.qx(n).m = dnlml.qx(n).m - dm{n};
  dnlml.qx(n).Sd(:,:,2:end) = bsxfun(@plus, dnlml.qx(n).Sd(:,:,2:end), iQ/2);
  dnlml.qx(n).Sd = dnlml.qx(n).Sd - bsxfun(@plus, dS, dLd{n});
  dnlml.qx(n).So = dnlml.qx(n).So - dLo{n};
end

out1 = nlml; out2 = dnlml; out3 = struct('C', C, 'R', R);     % rename outputs
```

4c ⟨*predictions* 4c⟩≡ (4d)

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

4d ⟨*vgpt.m* 4d⟩≡

```
function [out1, out2, out3] = vgpt(p, data, x);
⟨usage 5a⟩

N = length(p.qx); z = p.z; [M, F, E] = size(z); D = size(data(1).y,2); hyp = p.hyp;
u = arrayfun(@(x)(x.u),data,'UniformOutput',false); [qx(1:N).m] = deal(p.qx(:).m);
y = arrayfun(@(x)(x.y),data,'UniformOutput',false);
for n = 1:N, [qx(n).Sd qx(n).So] = convert(p.qx(n).s); end % convert covariance

if nargin == 2
  ⟨lower bound 4a⟩
  ⟨bound derivatives 4b⟩
  ⟨revert covariances 8b⟩
```

```
else
  ⟨predictions 4c⟩
end

⟨Psi 5b⟩
⟨entropy 3a⟩
⟨likelihood 3b⟩
⟨convert 8a⟩
⟨revert 9a⟩
⟨maha 9b⟩
```

5a ⟨*usage* 5a⟩≡ (4d)

```
% Variational GP Timeseries inference. Compute the nlml lower bound and its
% derivative wrt hyp hyperparameters, qx distribution and z inducing inputs.
%
% p                    parameter struct
%   hyp      1 x E     GP hyperparameter struct
%     l      F x 1     log length scale
%     pn     1 x 1     log process noise std dev
%   qx       1 x N     struct array for Gaussian q(x) distribution
%     m      E x T_n   mean
%     s   2ExE x T_n   representation of covariance
%   z      M x F x E   inducing inputs
% data       1 x N     data struct
%   y      T_n x D     cell array of observations
%   u      T_n x U     cell array of control inputs
%
% 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:

$$\psi \;=\; \tfrac{1}{2}\sum_{e=1}^{E} \log|K_e| - \log|K_e+\Psi_{2e}| - \mathrm{tr}K_e^{-1}\Psi_{2e} - \Psi_{1e}^\top(K_e+\Psi_{2e})^{-1}\Psi_{1e}. \tag{18}$$

5b ⟨*Psi* 5b⟩≡ (4d)

```
function [lml, dnlml] = Psi(hyp, qx, z, u, test);
% hyp       1 x E    GP hyperparameter struct
%   l       F x 1    log length scale
%   pn      1 x 1    log process noise std dev
% qx        1 x N    Gaussian q(x) distribution
%   m     E x T_n    mean
%   Sd  ExE x T_n    diagonal elements of covariance matrix
%   So  ExE x T_n-1  immediately off-diagonal elements of covariance matrix
% z       M x F x E  inducing inputs
% u       T_n x U    cell array of control inputs
% lml       1 x 1    contribution to the log marginal likelihood
% dnlml              derivatives

persistent K Psi1 Psi2;                     % keep these around if necessary
[M, F, E] = size(z);                                             % get sizes
⟨expectations 6⟩
if nargout > 0
  ⟨expectation derivatives 7a⟩
end
```

The Ψ_1 and Ψ_2 expectations

The expectations from eq. (5) 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

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

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

$$k_e(\mathbf{x}_{t-1}, \mathbf{z}_{ie}) = \exp\Big(-\tfrac{1}{2}\begin{bmatrix}\mathbf{x}_{t-1}-\mathbf{z}_{ie}\\ \mathbf{x}_t\end{bmatrix}^\top \begin{bmatrix}\Lambda_e^{-1} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}_{t-1}-\mathbf{z}_{ie}\\ \mathbf{x}_t\end{bmatrix}\Big),$$

yielding

$$\begin{aligned}\int \mathbf{x}_t k_e(\mathbf{x}_{t-1}, \mathbf{z}_{ie}) q(\mathbf{x}_{t-1:t}) d\mathbf{x}_{t-1} d\mathbf{x}_t &= \big(\boldsymbol{\mu}_t + \Sigma_{t,t-1}[\Lambda_e + \Sigma_{t-1,t-1}]^{-1}(\mathbf{z}_{ie} - \boldsymbol{\mu}_{t-1})\big) \\ &\quad \times |I + \Lambda_e^{-1}\Sigma_{t-1,t-1}|^{-1/2} \exp\big(-\tfrac{1}{2}(\boldsymbol{\mu}_{t-1} - \mathbf{z}_{ie})[\Lambda_e + \Sigma_{t-1,t-1}]^{-1}(\boldsymbol{\mu}_{t-1} - \mathbf{z}_{ie})\big).\end{aligned} \tag{19}$$

For Ψ_2 we have from eq. (5)

$$\begin{aligned}\int k_e(\mathbf{z}_{ie}, \mathbf{x}_{t-1}) k_e(\mathbf{x}_{t-1}, \mathbf{z}_{je}) q(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} &= \exp(-(\mathbf{z}_{ie} - \mathbf{z}_{je})\Lambda_e^{-1}(\mathbf{z}_{ie} - \mathbf{z}_{je})/4) \\ &\quad \times |I + 2\Lambda_e^{-1}\Sigma_{t-1,t-1}|^{-1/2} \exp(-(\tfrac{\mathbf{z}_{ie}+\mathbf{z}_{je}}{2} - \boldsymbol{\mu}_{t-1})[\Lambda_e/2 + \Sigma_{t-1,t-1}]^{-1}(\tfrac{\mathbf{z}_{ie}+\mathbf{z}_{je}}{2} - \boldsymbol{\mu}_{t-1})/2).\end{aligned} \tag{20}$$

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

6 ⟨*expectations* 6⟩≡ (5b)

```
K = zeros(M,M,E); Psi1 = zeros(M,E); Psi2 = zeros(M,M,E); Sd = zeros(F,F);
lml = 0;
for e = 1:E                                                   % for each GP
  K(:,:,e) = exp(-maha(z(:,:,e),[],diag(exp(-2*hyp(e).l)))/2) + 1e-6*eye(M);
  iL = diag(exp(-hyp(e).l)); L2 = diag(exp(2*hyp(e).l));
  b1 = zeros(M,1); b2 = zeros(M,M);
  for n = 1:length(qx)                                 % for each time series
    for t = 2:size(qx(n).m, 2)                           % for each time step
      Sd(1:E,1:E) = qx(n).Sd(:,:,t-1);          % covariance in top left corner
      r1 = prod(diag(chol(eye(F)+iL*Sd*iL)));                       % sqrt det
      r2 = prod(diag(chol(eye(F)+2*iL*Sd*iL)));                     % sqrt det
      s = bsxfun(@minus, z(:,:,e), [qx(n).m(:,t-1)' u{n}(t-1,:)]);
      a = s/(L2+Sd);
      b1 = b1 + (qx(n).m(e,t) + a(:,1:E)*qx(n).So(:,e,t-1)) ...
                                                .*exp(-sum(a.*s,2)/2)/r1;
      b2 = b2 + exp(-maha(s,-s,inv(L2+2*Sd))/4) / r2;
    end
  end
  if test
    w = (K(:,:,e)+Psi2(:,:,e))\Psi1(:,e);
    W = -K(:,:,e)\Psi2(:,:,e)/(Psi2(:,:,e)+K(:,:,e)) + w*w';
    lml = lml + exp(-2*hyp(e).pn)*(b1'*Psi1(:,e) ...
                   - sum(sum(b2.*exp(-maha(z(:,:,e),[],inv(L2))/4).*W))/2);
  else
    Psi1(:,e) = b1 * exp(-2*hyp(e).pn);
    Psi2(:,:,e) = b2 * exp(-2*hyp(e).pn) .* exp(-maha(z(:,:,e),[],inv(L2))/4);
    lml = lml - sum(log(diag(chol(K(:,:,e)+Psi2(:,:,e))))) + ...
          sum(log(diag(chol(K(:,:,e))))) + trace(K(:,:,e)\Psi2(:,:,e))/2 + ...
                                  Psi1(:,e)'/(K(:,:,e)+Psi2(:,:,e))*Psi1(:,e)/2;
  end
end
```

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

`Psi` derivatives

We need to compute the derivatives of Ψ wrt the parameters of the $q(x)$ distribution, wrt the u inducing inputs and wrt the hyperparameters. First, from eq. (18) we note

$$
\begin{aligned}
\frac{\partial\psi}{\partial\Psi_{1e}} &= -(K_e+K_{2e})^{-1}\Psi_{1e} \;=\; -\mathbf{w}_e,\\
\frac{\partial\psi}{\partial\Psi_{2e}} &= -\tfrac{1}{2}K_e^{-1}\Psi_{2e}(K_e+\Psi_{2e})^{-1}+\tfrac{1}{2}\mathbf{w}_e\mathbf{w}_e^\top \;=\; -\tfrac{1}{2}R_e(K_e+\Psi_{2e})^{-1}+\tfrac{1}{2}\mathbf{w}_e\mathbf{w}_e^\top \;=\; W_e,\\
\frac{\partial\psi}{\partial K_e} &= -\tfrac{1}{2}R_e(K_e+\Psi_{2e})^{-1}R_e^\top+\tfrac{1}{2}\mathbf{w}_e\mathbf{w}_e^\top \;=\; V_e,
\end{aligned}
$$

where we have defined $\mathbf{w}_e = (K_e+K_{2e})^{-1}\Psi_{1e}$ and $R_e = K_e^{-1}\Psi_{2e}$. These can be used together with the derivatives of Ψ_1e, Ψ_{2e} and K_e and the chain rule to get the desired derivatives.

7a ⟨*expectation derivatives* 7a⟩≡ (5b)

```
dnlml.z = zeros(M,F,E);
for n = 1:size(qx,2), dnlml.qx(n).m = 0*qx(n).m; dnlml.qx(n).So = 0*qx(n).So;
dnlml.qx(n).Sd = -0*qx(n).Sd; end;
for e = 1:E
  w = (K(:,:,e)+Psi2(:,:,e))\Psi1(:,e);
  R = K(:,:,e)\Psi2(:,:,e);
  W = -R/(Psi2(:,:,e)+K(:,:,e)) + w*w';
  ⟨hyp derivatives 7b⟩
  ⟨Psi derivatives 7c⟩
end
```

7b ⟨*hyp derivatives* 7b⟩≡ (7a)

```
dnlml.hyp(e).pn = 2*sum(w.*Psi1(:,e)) - sum(sum(W.*Psi2(:,:,e)));
```

7c ⟨*Psi derivatives* 7c⟩≡ (7a)

```
iL = diag(exp(-hyp(e).l)); L2 = diag(exp(2*hyp(e).l));
W1 = W .* exp(-maha(z(:,:,e),[],inv(L2))/4);
D = zeros(M,F); H = zeros(F,1);
for n = 1:length(qx)
  T = size(qx(n).m,2);
  A = zeros(E,T); B = zeros(E,E,T); C = zeros(E,E,T-1);
  for t = 2:T
    Sd(1:E,1:E) = qx(n).Sd(:,:,t-1);          % covariance in top left corner
    r2 = prod(diag(chol(eye(F)+2*iL*Sd*iL)));                       % sqrt det
    s = bsxfun(@minus, z(:,:,e), [qx(n).m(:,t-1)' u{n}(t-1,:)]);
    a = s/(L2+Sd);
    a2 = s/(2*L2+4*Sd);
    SiS = (L2(1:E,1:E)+Sd(1:E,1:E))\qx(n).So(:,e,t-1);
    r = exp(-sum(a.*s,2)/2) / prod(diag(chol(eye(F)+iL*Sd*iL)));
    g = (qx(n).m(e,t) + a(:,1:E)*qx(n).So(:,e,t-1)).*w.*r;
    W2 = W1.*exp(-maha(s,-s,inv(L2+2*Sd))/4);
    X = bsxfun(@plus,permute(a2,[1 3 2]),permute(a2,[3 1 2]));
    A(:,t-1) = A(:,t-1) + SiS*(w'*r) - a(:,1:E)'*g + ...
                   squeeze(sum(sum(bsxfun(@times,W2,X(:,:,1:E)),2),1))/r2;
    A(e,t) = -w'*r;
    B(:,:,t-1) = squeeze(sum(sum(bsxfun(@times, ...
       bsxfun(@times,W2,X(:,:,1:E)),permute(X(:,:,1:E),[1 2 4 3])),2),1))/r2;
    B(:,:,t-1) = B(:,:,t-1) + SiS*((w.*r)'*a(:,1:E)) ...
                             + inv(L2(1:E,1:E)+Sd(1:E,1:E))*sum(g)/2 ...
                             - a(:,1:E)'*bsxfun(@times,g,a(:,1:E))/2 ...
                     - inv(L2(1:E,1:E)+2*Sd(1:E,1:E))*sum(sum(W2))/r2/2;
    C(:,e,t-1) = -bsxfun(@times,a(:,1:E),r)'*w;
    if ~test
      D(:,1:E) = D(:,1:E) - bsxfun(@times,w,r)*SiS';
      D = D + bsxfun(@times,g,a) - W2*a2/r2 - bsxfun(@times,sum(W2,2),a2)/r2;
      H = H + diag(Sd/(L2+2*Sd))*sum(sum(W2))/r2 ...
         + exp(2*hyp(e).l).*squeeze(sum(sum(bsxfun(@times,W2,X.^2),2),1))/r2;
      H(1:E) = H(1:E) ...
```

```
              + 2*exp(2*hyp(e).l(1:E)).*diag(SiS*bsxfun(@times,w,r)'*a(:,1:E));
        H = H - diag(Sd/(L2+Sd))*sum(g);
        H = H - exp(2*hyp(e).l).*diag(a'*bsxfun(@times,g,a));
      end
    end
    dnlml.qx(n).m = dnlml.qx(n).m + A * exp(-2*hyp(e).pn);
    dnlml.qx(n).Sd = dnlml.qx(n).Sd + B * exp(-2*hyp(e).pn);
    dnlml.qx(n).So = dnlml.qx(n).So + C * exp(-2*hyp(e).pn);
  end
  if ~test
  G = W.*Psi2(:,:,e);
  a = z(:,:,e)*diag(exp(-2*hyp(e).l)/2);
  dnlml.z(:,:,e) = D*exp(-2*hyp(e).pn) + G*a - bsxfun(@times,sum(G,2),a);
  B = bsxfun(@minus,permute(a,[1 3 2]),permute(a,[3 1 2]));
  dnlml.hyp(e).l = H * exp(-2*hyp(e).pn)  ...
            + exp(2*hyp(e).l).*squeeze(sum(sum(bsxfun(@times,G,B.^2),1),2));
  G = (R/(K(:,:,e)+Psi2(:,:,e))*R' + w*w').*K(:,:,e);
  a = z(:,:,e)*diag(exp(-2*hyp(e).l));
  B = bsxfun(@minus,permute(z(:,:,e),[1 3 2]),permute(z(:,:,e),[3 1 2]));
  dnlml.hyp(e).l = dnlml.hyp(e).l ...
        + exp(-2*hyp(e).l).*squeeze(sum(sum(bsxfun(@times,B.^2,G),1),2))/2;
  dnlml.z(:,:,e) = dnlml.z(:,:,e) + G*a - bsxfun(@times,sum(G,2),a);
  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

$$\begin{aligned}&\sum_{e=1}^{E}\tfrac{1}{2}\mathrm{tr}K_e^{-1}\Psi_{2e}(K_e+\Psi_{2e})^{-1}\Psi^*_{2e}-\tfrac{1}{2}\mathrm{tr}(K_e+\Psi_{2e})^{-1}\Psi_{1e}\Psi_{1e}^\top(K_e+\Psi_{2e})^{-1}\Psi^*_{2e}+\Psi_{1e}^\top(K_e+\Psi_{2e})^{-1}\Psi^*_{1e}\\&+\tfrac{1}{2}\sum_{t=2}^{T}\log|\Sigma_{t-1:t,t-1:t}|-\tfrac{1}{2}\sum_{t=2}^{T-1}\log|\Sigma_t|-\tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}(I+\mu_t^\top\mu_t+\Sigma_{t,t})-\tfrac{T-1}{2}\log|Q|-\tfrac{(T-1)E}{2}\log(2\pi).\end{aligned}\tag{21}$$

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

The $q(\mathbf{x})$ distribution is parameterised through its mean qx.m and the marginal and pairwise covariances. Conceptually, we wish to parameterize the E by E covariance matrices (for the marginal distibutions) which we call qx.Sd (for diagonal) and the E by E covariances between consequtive time points (for the pairwise marginals) which we call qx.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 qx.s such that

$$S_{d,t} \;=\; s_t^\top s_t, \;\text{ and }\; S_{o,t-1} \;=\; s_{t-1}^\top s_t. \tag{22}$$

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.

8a ⟨*convert* 8a⟩≡ (4d)

```
function [Sd, So] = convert(s)
[t, E, T]= size(s); Sd = zeros(E,E,T); So = zeros(E,E,T-1);
for t = 1:T, Sd(:,:,t) = s(:,:,t)'*s(:,:,t); end                 % diagonal terms
for t = 2:T, So(:,:,t-1) = s(:,:,t-1)'*s(:,:,t); end               % off-diagonal
```

8b ⟨*revert covariances* 8b⟩≡ (4d)

```
out2.qx = rmfield(out2.qx,{'Sd','So'});        % change to qx.s representation
[out2.qx.s] = deal([]);                                 % create the "s" field
%for n = 1:N
%  Sd = sum(dnlml.qx(n).Sd,3) / size(dnlml.qx(n).Sd,3);
```

```
%   for t = 1:size(dnlml.qx(n).Sd,3), dnlml.qx(n).Sd(:,:,t) = Sd; end
%end
for n = 1:N
  out2.qx(n).s = revert(p.qx(n).s, dnlml.qx(n).Sd, dnlml.qx(n).So);
end
```

The derivatives are

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial s_t} &= \frac{\partial \mathcal{L}}{\partial S_{d,t}}\frac{\partial S_{d,t}}{\partial s_t} + \frac{\partial \mathcal{L}}{\partial S_{o,t-1}}\frac{\partial S_{o,t-1}}{\partial s_t} + \frac{\partial \mathcal{L}}{\partial S_{o,t}}\frac{\partial S_{o,t}}{\partial s_t} = \frac{\partial}{\partial s_t}\mathrm{tr}\Big(\frac{\partial \mathcal{L}}{\partial S_{d,t}} s_t^\top s_t\Big) \\ &+ s_{t-1}\frac{\partial \mathcal{L}}{\partial S_{o,t-1}} + s_t\Big[\frac{\partial \mathcal{L}}{\partial S_{o,t}}\Big]^\top = s_t\Big(\frac{\partial \mathcal{L}}{\partial S_{d,t}} + \Big[\frac{\partial \mathcal{L}}{\partial S_{d,t}}\Big]^\top\Big) + s_{t-1}\frac{\partial \mathcal{L}}{\partial S_{o,t-1}} + s_t\Big[\frac{\partial \mathcal{L}}{\partial S_{o,t}}\Big]^\top.\end{aligned} \tag{23}$$

9a ⟨*revert* 9a⟩≡ (4d)

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

9b ⟨*maha* 9b⟩≡ (4d)

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