Notes on Gaussian Processes

1 Forewords

Most of the content of these notes is extracted from *Gaussian Processes for Machine Learning*, by Rasmussen & Williams. Therefore we adopt a number of their conventions. In particular, vectors are denoted as bold lowercase mathematical symbols ($\mathbf{x}$), scalars are lowercase symbols in normal font (x), while matrices are uppercase symbols (X).

The section on sparse Gaussian processes in inspired from (in reverse publication order):

- Almosallam et al. (2016). *GPz : non-stationary sparse Gaussian processes for heteroscedastic uncertainty estimation in photometric redshifts*. MNRAS, 462, 726.
- Almosallam et al. (2016). *A sparse Gaussian process framework for photometric redshift estimation*. MNRAS, 455, 2387.
- Snelson & Ghahramani (2006). *Sparse Gaussian Processes using pseudo-inputs*. In Weiss Y., Schölkopf B., Platt J., eds, Advances in Neural Information Processing Systems 18. MIT Press, Cambridge, MA, p. 1257.
- Seeger et al. (2003). *Fast forward selection to speed up sparse Gaussian process regression*. In C. M. Bishop and B. J. Frey, editors, Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics.

2 Standard (full) GP

2.1 Predictions from a given training set and covariance function

Suppose we have input values $\mathbf{x}$ (in general, a vector of input values, but it could be a scalar too) that are linked to an output value y. We have a training set, $\{\mathbf{x_t}, y_t\}$, which is the stuff we know. We have a test set, $\{\mathbf{x}_*, y_*\}$, which is the stuff for which we want to make a prediction. We denote $X_t = \{\mathbf{x_t}\}$ the set of training inputs, and $X_* = \{\mathbf{x}_*\}$ the set of test inputs, both of which are given. We denote $\mathbf{y} = \{y\}$ the set of output training values, which are also given, and $\mathbf{y}_* = \{y_*\}$ the set of test output values, which are unknown. Our job is to make a prediction for what $\mathbf{y}_*$ should be.

In the GP approach, we use a probabilistic approach to model the *process* $y|\mathbf{x}$ that describes the training data. This is called a "process" rather than a "function" (i.e., $y = f(\mathbf{x})$) because the values y are evaluated from some unattainable function f that is itself drawn from a "distribution of functions". To do so we must therefore assume a "prior on functions" which is determined by a covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. From this covariance function, we can form several covariance matrices involving inputs from both the training and the test sets:

$$K_{tt} : K_{tt,ij} = k(\mathbf{x_t}_{,i}, \mathbf{x_t}_{,j}) \,, \tag{2.1}$$

$$K_{**} : K_{**,ij} = k(\mathbf{x}_{*,i}, \mathbf{x}_{*,j}) \,, \tag{2.2}$$

$$K_{t*} : K_{t*,ij} = k(\mathbf{x_t}_{,i}, \mathbf{x}_{*,j}) \,, \tag{2.3}$$

This covariance function serves to model our training set with the following Gaussian Process:

$$\mathbf{y_t}|X_t \sim \mathcal{N}(\bar{\mathbf{y}}_\mathbf{t}, K_{tt}) \tag{2.4}$$

where $\bar{\mathbf{y}}_\mathbf{t}$ is the mean of the prior, which is assumed to be zero in all that follows. If we want a non zero prior, it can be easily implemented by subtracting this non-zero prior from the training output values $\mathbf{y_t}$ before the training stage, and then adding it back to the predicted test output values $\mathbf{y}_*$ at the very end of the calculations. Therefore, in what follows, we assume:

$$\mathbf{y_t}|X_t \sim \mathcal{N}(0, K_{tt})\,. \tag{2.5}$$

The key prediction of the GP is that the values of $\mathbf{y}_*$, given we know the training data X_t and $\mathbf{y_t}$ as well as the new positions X_*, are drawn from a Gaussian distribution of mean $\bar{\mathbf{y}}_*$ and covariance matrix C_*:

$$\mathbf{y}_*|X_*, X_t, \mathbf{y_t} \sim \mathcal{N}(\bar{\mathbf{y}}_*, C_*)\,, \tag{2.6}$$

with:

$$\bar{\mathbf{y}}_* = {K_{t*}}^T\, {K_{tt}}^{-1}\, \mathbf{y_t} \tag{2.7}$$

$$C_* = K_{**} - {K_{t*}}^T\, {K_{tt}}^{-1}\, K_{t*}\,. \tag{2.8}$$

Given a choice of a covariance function k, we can use the above two equations to make predictions for $\mathbf{y}_*$; all we need to do is build the K matrices by evaluating the function k, invert K_{tt}, and do simple matrix/vector multiplications and additions above. Once $\bar{\mathbf{y}}_*$ and C_* are computed, we can use a random number generator to build a set of random variables that follow the joint distribution $\mathcal{N}(\bar{\mathbf{y}}_*, C_*)$. For N test output values, this is done by computing the Cholesky decomposition $C_* = L_*\, L_*^T$, generate N Gaussian random numbers u_i with variance unity and mean zero, namely $\mathbf{u} \sim \mathcal{N}(0, 1)$, and we then obtain one realization of our prediction using $\mathbf{y}_* = \bar{\mathbf{y}}_* + L_*\, \mathbf{u}$.

2.2 Training the covariance function

The problem is then to decide how to choose the covariance function. In general this covariance function will depend on several "hyperparameters" $\boldsymbol{\theta} = \{\theta_i\}$, which define its amplitude and shape. In the calculation above, these hyperparameters are fixed, and used to make a prediction. But some choices of hyperparameters may provide better predictions than others. Ideally, in a fully probabilistic approach, we would marginalize over these hyperparameters to make our predictions, however in practice this is simply too hard. A much simpler way is to optimize these hyperparameters by maximizing the marginal likelihood on the training set, namely $p(\mathbf{y_t}|X_t)$. Given Eq. 2.5, this likelihood is simply given by our assumed prior $\mathbf{y_t}|X_t \sim \mathcal{N}(0, K_{tt})$, which translates to:

$$p(\mathbf{y_t}|X_t) = \frac{\exp\left(-\frac{1}{2}\, {\mathbf{y_t}}^T\, {K_{tt}}^{-1}\, \mathbf{y_t}\right)}{(2\pi)^{n_t/2}\, \sqrt{|K_{tt}|}}\,, \tag{2.9}$$

where $|K_{tt}|$ is the determinant of the covariance matrix, and n_t is the number of objects in the training set. By optimizing this quantity, we will pick the best hyperparameters as a trade off of between, on the one hand, providing the best fit to the training data

(maximizing the exponential term), and on the other hand, using the simplest possible model (minimizing the determinant).

In practice it is easier to minimize:

$$\mathcal{L} \equiv -2 \log p(\mathbf{y_t}|X_t) = \mathbf{y_t}^T K_{tt}^{-1} \mathbf{y_t} + \log |K_{tt}| + n_t \log(2\pi) \,. \tag{2.10}$$

This can be efficiently evaluated by computing the Cholesky decomposition $K_{tt} = L_t L_t^T$, use it to solve for $K_{tt}\, \tilde{\mathbf{y}}_\mathbf{t} = \mathbf{y_t}$, and compute $\log |K_{tt}|$ using the trace of $\log L_t$:

$$\mathcal{L} = \mathbf{y_t}^T \tilde{\mathbf{y}}_\mathbf{t} + 2 \operatorname{Tr} (\log L_t) + n_t \log(2\pi) \,. \tag{2.11}$$

Using the trace of the log is more numerically stable than computing the log of the product of the diagonal elements of L_t, which can easily underflow.

To perform the minimization using standard gradient descent algorithms, we also need to compute the derivative of this quantity with respect to each hyperparameter:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \operatorname{Tr} \left(K_{tt}^{-1} \frac{\partial K_{tt}}{\partial \theta_i} \right) - \mathbf{y_t}^T K_{tt}^{-1} \frac{\partial K_{tt}}{\partial \theta_i} K_{tt}^{-1} \mathbf{y_t} \,, \tag{2.12}$$

where $\frac{\partial K_{tt}}{\partial \theta_i}$ is constructed as:

$$\frac{\partial K_{tt,ij}}{\partial \theta_i} = \frac{\partial k(\mathbf{x_t}_{,i}, \mathbf{x_t}_{,j})}{\partial \theta_i} \,, \tag{2.13}$$

and the trace term can be optimized by only computing the diagonal elements of the matrix product.

Re-using $\tilde{\mathbf{y}}_\mathbf{t}$ as computed above, this can be evaluated more efficiently as

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \operatorname{Tr} \left(K_{tt}^{-1} \frac{\partial K_{tt}}{\partial \theta_i} \right) - \tilde{\mathbf{y}}_\mathbf{t}^T \frac{\partial K_{tt}}{\partial \theta_i} \tilde{\mathbf{y}}_\mathbf{t} \,. \tag{2.14}$$

This is the fastest way to compute the marginal likelihood exactly, however the Cholesky decomposition is still an $O({n_t}^3)$ operation, therefore it does not scale well for large inputs; this is where a Sparse Gaussian Process becomes useful.

3 The squared exponential covariance function

A general form for the commonly used "squared exponential" covariance is:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_v^2 \exp \left(-\frac{1}{2} (\mathbf{x}_i - \mathbf{x}_j)^T Q \, (\mathbf{x}_i - \mathbf{x}_j) \right) + \delta_{ij} \, \sigma_n^2 \,, \tag{3.1}$$

in which Q is a "direction matrix" which controls the directions and scale lengths of the covariances. A simpler form, where the covariance scale-length is the same in all dimensions of the input space is:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_v^2 \exp \left(-\frac{1}{2} \frac{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}{\ell^2} \right) + \delta_{ij} \, \sigma_n^2 \,, \tag{3.2}$$

which, for the one-dimensional case, translates to:

$$k(x_i, x_j) = \sigma_v^2 \exp \left(-\frac{1}{2} \frac{(x_i - x_j)^2}{\ell^2} \right) + \delta_{ij} \, \sigma_n^2 \,. \tag{3.3}$$

- σ_v controls how far away from zero the "true" smooth function values are allowed to go (i.e., the amplitude of the function).
- ℓ (or the elements of the matrix Q) control the distance (and, for Q, the directions) over which neighboring points should have similar values (i.e., how smooth the function is).
- σ_n controls the amount of uncorrelated noise on top of that smooth function (i.e., how much random scatter there is on top of the smooth function).

For σ_n, the δ_{ij} factor means that this noise term should only be added for two identical input points $\mathbf{x}_i$, namely, not just identical in value, but identical as entities (e.g., a point in the input *training* set with the same value as a point in the input *test* set are not the same entities, therefore they should not have a noise term even though their values coincide). This serves to model the noise in the observed values for the *training* set (hence σ_n is usually known and given), and in most cases we do not want to propagate this noise into the predictions. In practice, this means this noise term should be added to the training covariance matrix K_{tt}, but not to K_{**} or K_{*t}.

However, it is also possible that the model should have some extra uncorrelated noise, which can be useful to model an uncertainty in the prediction that is unrelated to measurement noise and instead introduce an inherent variance in how well y can be estimated from x even when we know the mean prediction very precisely. In this case one would replace $\delta_{ij}\,\sigma_n^2$ by $\delta_{ij}\,(\sigma_n^2 + \sigma_m^2)$, where σ_m^2 is a "model" variance. These two noise terms would be added to the training covariance matrix K_{tt}, while the test covariance matrix K_{**} would only get the σ_m term, and the mixed matrix K_{*t} would get neither. Note, however, that this model can only work if σ_n is known and given as input (i.e., because we know the actual noise on our measurements). If σ_n is not known, then there is mathematically no way to disentangle measurement noise (σ_n) from model variance (σ_m).

4 Sparse GP (SPGP)

4.1 Formula for GP predictions

As described above, evaluating the likelihood of a GP for large data sets can become too expensive because of the need to inverse an $n_t \times n_t$ matrix (or doing its Cholesky decomposition, which is faster but has the same algorithmic complexity). To cope with this, several approximations are possible, and one of them are Sparse GPs (Snelson & Ghahramani 2006), or SPGPs. In this approximation, we introduce n_p "pseudo inputs" locations $X_p = \{\mathbf{x}_p\}$ that we can freely adjust, and use these to solve the GP. The covariance matrix we need to invert is now K_{pp}, which is evaluated at the pseudo inputs, and is of size $n_p \times n_p$. If we choose $n_p \ll n_t$, then the computations become much faster; $O({n_p}^2\, n_t)$ instead of $O({n_t}^3)$, and the results will become an approximation that still preserves a number of properties of full GPs. In practice it can be an excellent approximation if the pseudo inputs are chosen carefully, and in fact they can be optimized like any other hyperparameter.

This approximation can be understood as follows. In the full GP, the prediction for the mean posterior function can reformulated as a sum over the training set of the covariance function, evaluated with the test input used as one $\mathbf{x}$:

$$\bar{y}_*(\mathbf{x}_*) = \sum_i^{n_t} c_i\, k(\mathbf{x}_{\mathbf{t},i}, \mathbf{x}_*)\,, \tag{4.1}$$

with the weighting coefficients c_i learned to fit the training output $\mathbf{y}$. The SPGP approximation simplifies this model by summing instead over the (fewer) n_p chosen pseudo inputs:

$$\bar{y}_*(\mathbf{x}_*) = \sum_i^{n_p} d_i \, k(\mathbf{x_p}_{,i}, \mathbf{x}_*) \,, \tag{4.2}$$

and the new weighting coefficients d_i are still learned to fit the training output $\mathbf{y}$. In fact, this illustrates one advantage of the SPGP over the full GP: the pseudo inputs can in principle lie outside of the training set, and this flexibility can offer a better description of the data.

Since the pseudo output values $\mathbf{y_p}$ (obtained at the location of the pseudo inputs X_p) are unknown, we must marginalized over them. Assuming they have the same prior for the pseudo outputs as for the real data, the GP prediction becomes:

$$\mathbf{y}_*|X_*, X_t, \mathbf{y_t}, X_p \sim \mathcal{N}(\bar{\mathbf{y}}_*, C_*) \,, \tag{4.3}$$

with:

$$\bar{\mathbf{y}}_* = {K_{p*}}^T \, Q^{-1} \, {K_{tp}}^T \, \Lambda^{-1} \, \mathbf{y_t} \tag{4.4}$$

$$C_* = K_{**} - {K_{p*}}^T \, ({K_{pp}}^{-1} - Q^{-1}) \, K_{p*} \,, \tag{4.5}$$

and where:

$$Q = K_{pp} + {K_{tp}}^T \, \Lambda^{-1} \, K_{tp} \tag{4.6}$$

$$\Lambda_{ii} = K_{tt,ii} - \mathbf{k_{tp}}_{,i} \, {K_{pp}}^{-1} \, {\mathbf{k_{tp}}_{,i}}^T \,, \tag{4.7}$$

using the notation $\mathbf{k_{tp}}_{,i}$ to mean the ith row of the K_{tp} matrix, so the second term in that last equation is an inner product over the n_p pseudo inputs. The matrix Q is $n_p \times n_p$, while Λ is diagonal and is thus cheap to invert.

4.2 Optimization of hyperparameters

The marginal likelihood, transformed again as $\mathcal{L} = -2 \log p(\mathbf{y_t}|X_t)$, becomes:

$$\mathcal{L} = {\mathbf{y_t}}^T \, (K_{tp} \, {K_{pp}}^{-1} \, {K_{tp}}^T + \Lambda)^{-1} \, \mathbf{y_t} + \log \left| K_{tp} \, {K_{pp}}^{-1} \, {K_{tp}}^T + \Lambda \right| + n_t \, \log(2\pi) \,. \tag{4.8}$$

This is not good however, because it still implies $n_t \times n_t$ matrices. Using some algebra (see appendix on matrix identities in Rasmussen & Williams), the first and second terms can be reformulated as:

$${\mathbf{y_t}}^T \, (K_{tp} \, {K_{pp}}^{-1} \, {K_{tp}}^T + \Lambda)^{-1} \, \mathbf{y_t} = {\mathbf{y_t}}^T \, \Lambda^{-1} \, \mathbf{y_t} - {\mathbf{y_t}}^T \, \Lambda^{-1} \, K_{tp} \, Q^{-1} \, {K_{tp}}^T \, \Lambda^{-1} \, \mathbf{y_t} \tag{4.9}$$

$$\log \left| K_{tp} \, {K_{pp}}^{-1} \, {K_{tp}}^T + \Lambda \right| = \mathrm{Tr} \, (\log \Lambda) + \log \left[\left| K_{pp}^{-1} \right| \, \left| K_{pp} + {K_{tp}}^T \, \Lambda^{-1} \, K_{tp} \right| \right] \tag{4.10}$$

which both involve inverting $n_p \times n_p$ matrices and Λ, which is much cheaper to evaluate.

Alternatively, following Seeger et al. (2003) and using similar algebraic tricks, if we do the Cholesky decomposition $K_{pp} = L_p \, {L_p}^T$, and define $V = {L_p}^{-1} \, {K_{tp}}^T$ and $M = 1 + V \, \Lambda^{-1} \, V^T$, then further decompose $M = L_M \, {L_M}^T$ to define $\boldsymbol{\beta} = {L_M}^{-1} \, V \, \Lambda^{-1} \, \mathbf{y_t}$, we get the simpler formula:

$$\mathcal{L} = {\mathbf{y_t}}^T \, \Lambda^{-1} \, \mathbf{y_t} - \boldsymbol{\beta}^T \boldsymbol{\beta} + \mathrm{Tr} \, (\log \Lambda) + 2 \, \mathrm{Tr} \, (\log L_M) + n_t \, \log(2\pi) \,. \tag{4.11}$$

The details of the calculations can be found in Appendix A.1.

A Detail of calculations

A.1 Marginal likelihood for SPGP

$$\begin{aligned}
\mathbf{y_t}^T \left(K_{tp} {K_{pp}}^{-1} {K_{tp}}^T + \Lambda\right)^{-1} \mathbf{y_t} &= \mathbf{y_t}^T \left(K_{tp} {L_p}^{-1,T} L_p^{-1} {K_{tp}}^T + \Lambda\right)^{-1} \mathbf{y_t} && \text{(A.1)}\\
&= \mathbf{y_t}^T \left(V^T V + \Lambda\right)^{-1} \mathbf{y_t} && \text{(A.2)}\\
&= \mathbf{y_t}^T \left(\Lambda^{-1} - \Lambda^{-1} V^T \left(1 + V \Lambda^{-1} V^T\right)^{-1} V \Lambda^{-1}\right) \mathbf{y_t} && \text{(A.3)}\\
&= \mathbf{y_t}^T \left(\Lambda^{-1} - \Lambda^{-1} V^T M^{-1} V \Lambda^{-1}\right) \mathbf{y_t} && \text{(A.4)}\\
&= \mathbf{y_t}^T \left(\Lambda^{-1} - \Lambda^{-1} V^T {L_M}^{-1,T} L_M^{-1} V \Lambda^{-1}\right) \mathbf{y_t} && \text{(A.5)}\\
&= \mathbf{y_t}^T \Lambda^{-1} \mathbf{y_t} - \boldsymbol{\beta}^T \boldsymbol{\beta} && \text{(A.6)}
\end{aligned}$$

$$\begin{aligned}
\log \left| K_{tp} {K_{pp}}^{-1} {K_{tp}}^T + \Lambda \right| &= \log \left| K_{tp} {L_p}^{-1,T} L_p^{-1} {K_{tp}}^T + \Lambda \right| && \text{(A.7)}\\
&= \log \left| V^T V + \Lambda \right| && \text{(A.8)}\\
&= \log |\Lambda| + \log \left| 1 + V \Lambda^{-1} V \right| && \text{(A.9)}\\
&= \log |\Lambda| + \log |M| && \text{(A.10)}\\
&= \log |\Lambda| + \log |L_M|^2 && \text{(A.11)}\\
&= \operatorname{Tr}(\log \Lambda) + 2 \operatorname{Tr}(\log L_M) && \text{(A.12)}
\end{aligned}$$

A.2 Derivatives of marginal likelihood for SPGP

$$\begin{aligned}
\frac{\partial}{\partial \theta_\ell} \left(\mathbf{y_t}^T \Lambda^{-1} \mathbf{y_t} \right) &= \mathbf{y_t}^T \frac{\partial}{\partial \theta_\ell} \left(\Lambda^{-1} \right) \mathbf{y_t} && \text{(A.13)}\\
&= \mathbf{y_t}^T \Lambda^{-1} \frac{\partial \Lambda}{\partial \theta_\ell} \Lambda^{-1} \mathbf{y_t} && \text{(A.14)}
\end{aligned}$$

$$\frac{\partial \Lambda_{ii}}{\partial \theta_\ell} = \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \frac{\partial}{\partial \theta_\ell}\left(\mathbf{k_{tp}}_{,i}\, {K_{pp}}^{-1}\, {\mathbf{k_{tp}}_{,i}}^T\right) \tag{A.15}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \frac{\partial}{\partial \theta_\ell}\left(\mathbf{k_{tp}}_{,i}\, {L_p}^{-1,T}\, {L_p}^{-1}\, {\mathbf{k_{tp}}_{,i}}^T\right) \tag{A.16}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \frac{\partial}{\partial \theta_\ell}\left(V^T V\right)_{ii} \tag{A.17}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \frac{\partial}{\partial \theta_\ell}\left(\sum_j {V_{ji}}^2\right) \tag{A.18}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - 2\left(\sum_j V_{ji}\frac{\partial V_{ji}}{\partial \theta_\ell}\right) \tag{A.19}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - 2\left(V^T \frac{\partial V}{\partial \theta_\ell}\right)_{ii} \tag{A.20}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - 2\left(V^T \frac{\partial}{\partial \theta_\ell}[{L_p}^{-1}\, {K_{tp}}^T]\right)_{ii} \tag{A.21}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - 2\left(V^T \left[{L_p}^{-1}\frac{\partial {K_{tp}}^T}{\partial \theta_\ell} + \frac{\partial}{\partial \theta_\ell}{L_p}^{-1}\, {K_{tp}}^T\right]\right)_{ii} \tag{A.22}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - 2\left(V^T \left[{L_p}^{-1}\frac{\partial {K_{tp}}^T}{\partial \theta_\ell} - {L_p}^{-1}\frac{\partial L_p}{\partial \theta_\ell}\, {L_p}^{-1}\, {K_{tp}}^T\right]\right)_{ii} \tag{A.23}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - 2\left(V^T {L_p}^{-1}\left[\frac{\partial {K_{tp}}^T}{\partial \theta_\ell} - \frac{\partial L_p}{\partial \theta_\ell} V\right]\right)_{ii} \tag{A.24}$$

$$\tag{A.25}$$

$$\frac{\partial \Lambda_{ii}}{\partial \theta_\ell} = \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \frac{\partial}{\partial \theta_\ell}\left(\mathbf{k_{tp}}_{,i}\, {K_{pp}}^{-1}\, {\mathbf{k_{tp}}_{,i}}^T\right) \tag{A.26}$$

$$\tag{A.27}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \frac{\partial}{\partial \theta_\ell}\left(K_{tp}\, {K_{pp}}^{-1}\, K_{tp}^T\right)_{ii} \tag{A.28}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \left(\frac{\partial K_{tp}}{\partial \theta_\ell}\, {K_{pp}}^{-1}\, K_{tp}^T + K_{tp}\frac{\partial {K_{pp}}^{-1}}{\partial \theta_\ell}\, K_{tp}^T + K_{tp}\, {K_{pp}}^{-1}\frac{\partial K_{tp}^T}{\partial \theta_\ell}\right)_{ii} \tag{A.29}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \left(2\frac{\partial K_{tp}}{\partial \theta_\ell}\, {K_{pp}}^{-1}\, K_{tp}^T + K_{tp}\frac{\partial {K_{pp}}^{-1}}{\partial \theta_\ell}\, K_{tp}^T\right)_{ii} \tag{A.30}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \left(2\frac{\partial K_{tp}}{\partial \theta_\ell}\, {K_{pp}}^{-1}\, K_{tp}^T + K_{tp}\, {K_{pp}}^{-1}\frac{\partial K_{pp}}{\partial \theta_\ell}\, {K_{pp}}^{-1}\, K_{tp}^T\right)_{ii} \tag{A.31}$$

$$= \frac{\partial K_{tt,ii}}{\partial \theta_\ell} - \left(2\frac{\partial K_{tp}}{\partial \theta_\ell}\, {L_p}^{-1,T}\, V + V^T\, {L_p}^{-1}\frac{\partial K_{pp}}{\partial \theta_\ell}\, {L_p}^{-1,T}\, V\right)_{ii} \tag{A.32}$$

$$\tag{A.33}$$

$$\frac{\partial}{\partial \theta_\ell}\left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right) = \frac{\partial}{\partial \theta_\ell}\left({\mathbf{y_t}}^T\, \Lambda^{-1}\, K_{tp}\, Q^{-1}\, {K_{tp}}^T\, \Lambda^{-1}\, \mathbf{y_t}\right) \tag{A.34}$$

$$= {\mathbf{y_t}}^T\frac{\partial}{\partial \theta_\ell}\left(\Lambda^{-1}\, K_{tp}\, Q^{-1}\, {K_{tp}}^T\, \Lambda^{-1}\right)\mathbf{y_t} \tag{A.35}$$

$$\tag{A.36}$$

$$\frac{\partial}{\partial\theta_\ell}\left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right) = \mathbf{y_t}^T\,[\frac{\partial}{\partial\theta_\ell}\left(\Lambda^{-1}\right)\,K_{tp}\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1} \tag{A.37}$$

$$+\,\Lambda^{-1}\,\frac{\partial K_{tp}}{\partial\theta_\ell}\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1} \tag{A.38}$$

$$+\,\Lambda^{-1}\,K_{tp}\,\frac{\partial}{\partial\theta_\ell}\left(Q^{-1}\right)\,{K_{tp}}^T\,\Lambda^{-1} \tag{A.39}$$

$$+\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,\frac{\partial {K_{tp}}^T}{\partial\theta_\ell}\,\Lambda^{-1} \tag{A.40}$$

$$+\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,{K_{tp}}^T\,\frac{\partial}{\partial\theta_\ell}\left(\Lambda^{-1}\right) \tag{A.41}$$

$$]\,\mathbf{y_t} \tag{A.42}$$

$$\frac{\partial}{\partial\theta_\ell}\left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right) = \mathbf{y_t}^T\,[\Lambda^{-1}\,\frac{\partial\Lambda}{\partial\theta_\ell}\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1} \tag{A.43}$$

$$+\,\Lambda^{-1}\,\frac{\partial K_{tp}}{\partial\theta_\ell}\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1} \tag{A.44}$$

$$+\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,\frac{\partial Q}{\partial\theta_\ell}\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1} \tag{A.45}$$

$$+\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,\frac{\partial {K_{tp}}^T}{\partial\theta_\ell}\,\Lambda^{-1} \tag{A.46}$$

$$+\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1}\,\frac{\partial\Lambda}{\partial\theta_\ell}\,\Lambda^{-1} \tag{A.47}$$

$$]\,\mathbf{y_t} \tag{A.48}$$

$$\frac{\partial}{\partial\theta_\ell}\left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right) = \mathbf{y_t}^T\Lambda^{-1}\,[2\,\frac{\partial\Lambda}{\partial\theta_\ell}\,\Lambda^{-1}\,K_{tp}\,Q^{-1}\,{K_{tp}}^T \tag{A.49}$$

$$+\,2\,\frac{\partial K_{tp}}{\partial\theta_\ell}\,Q^{-1}\,{K_{tp}}^T \tag{A.50}$$

$$+\,K_{tp}\,Q^{-1}\,\frac{\partial Q}{\partial\theta_\ell}\,Q^{-1}\,{K_{tp}}^T \tag{A.51}$$

$$]\,\Lambda^{-1}\,\mathbf{y_t} \tag{A.52}$$

$$\frac{\partial}{\partial\theta_\ell}\left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right) = \mathbf{y_t}^T\Lambda^{-1}\,\left[2\,\frac{\partial\Lambda}{\partial\theta_\ell}\,\Lambda^{-1}\,K_{tp} + 2\,\frac{\partial K_{tp}}{\partial\theta_\ell} + K_{tp}\,Q^{-1}\,\frac{\partial Q}{\partial\theta_\ell}\right]\,Q^{-1}\,{K_{tp}}^T\,\Lambda^{-1}\,\mathbf{y_t} \tag{A.53}$$

$$\frac{\partial Q}{\partial\theta_\ell} = \frac{\partial}{\partial\theta_\ell}\left(K_{pp} + {K_{tp}}^T\,\Lambda^{-1}\,K_{tp}\right) \tag{A.54}$$

$$= \frac{\partial K_{pp}}{\partial\theta_\ell} + \frac{\partial}{\partial\theta_\ell}\left({K_{tp}}^T\,\Lambda^{-1}\,K_{tp}\right) \tag{A.55}$$

$$= \frac{\partial K_{pp}}{\partial\theta_\ell} + 2\,\frac{\partial {K_{tp}}^T}{\partial\theta_\ell}\,\Lambda^{-1}\,K_{tp} + {K_{tp}}^T\,\Lambda^{-1}\,\frac{\partial\Lambda}{\partial\theta_\ell}\,\Lambda^{-1}\,K_{tp} \tag{A.56}$$

Warning: the above is only true when $\partial Q/\partial\theta_\ell$ is sandwitched inside a inner product, as in A.53