Engineering Tripos Part IIB FOURTH YEAR

Module 4F10: DEEP LEARNING AND STRUCTURED DATA

Solutions to Examples Paper 2

1. (a) The derivative for the first term is simple, the identity matrix $\boldsymbol{I}$. For the second term can use the expansion

$$\frac{\partial \boldsymbol{y}^{(l)}}{\partial \boldsymbol{x}^{(l)}} = \boldsymbol{I} + \frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{x}^{(l)}}\frac{\partial \boldsymbol{y}^{(l)}}{\partial \boldsymbol{z}^{(l)}}$$

These parts can then be treated separately. Only need to consider diagonal elements for the first term

$$\frac{\partial y_i^{(l)}}{\partial z_i^{(l)}} = \phi(z_i)(1-\phi(z_i))$$

from the first examples paper. The other term is simply a linear relationship

$$\frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{x}^{(l)}} = \mathbf{W}^{\mathsf{T}}$$

Thus the overall expression is

$$\frac{\partial \boldsymbol{y}^{(l)}}{\partial \boldsymbol{x}^{(l)}} = \boldsymbol{I} + \mathbf{W}^{\mathsf{T}}\begin{bmatrix} \phi(z_1)(1-\phi(z_1)) & 0 & \dots & 0 \\ 0 & \phi(z_2)(1-\phi(z_2)) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \phi(z_{N_l})(1-\phi(z_{N_l})) \end{bmatrix}$$

(b) When the logistic (sigmoid) function operates in the saturated region

$$\frac{\partial \boldsymbol{y}^{(l)}}{\partial \boldsymbol{z}^{(l)}} = \mathbf{0}$$

This means that when the derivatives are back-propagates through the network all derivatives from earlier layers are zero, the network doesn't change. By adding the residual connection there is always a non-zero derivative

(c) For a highway connection (ignoring bias)

$$\boldsymbol{y}^{(l)} = (\mathbf{1}-\boldsymbol{i}^{(l)})\odot\boldsymbol{x}^{(l)} + \boldsymbol{i}^{(l)}\odot\boldsymbol{\phi}\left(\mathbf{W}\boldsymbol{x}^{(l)}\right); \quad \boldsymbol{i}^{(l)} = \boldsymbol{\phi}(\mathbf{W}_{\mathsf{g}}^{(l)}\boldsymbol{x}^{(l)})$$

This now requires the gating function to not be $\mathbf{1}$ for a non-zero derivative to be propagated to earlier layers. Thus it is not guaranteed, but both sigmoid must now be in saturation.

2. Performing Eigenvector/Eigenvalue decomposition on $\boldsymbol{w}$ yields

$$\boldsymbol{h}_t = \mathbf{W}\boldsymbol{h}_{t-1} = \mathbf{W}^t\boldsymbol{h}_0 = \left(\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\text{-1}}\right)^t \boldsymbol{h}_0 = \boldsymbol{Q}\boldsymbol{\Lambda}^t\boldsymbol{Q}^{\text{-1}}\boldsymbol{h}_0$$

As t gets large $\boldsymbol{\Gamma}^t$ is dominated by the largest eigenvalue, λ. The expression then becomes

$$\boldsymbol{h}_t \approx \lambda^t \boldsymbol{q}\boldsymbol{v}^{\mathsf{T}}\boldsymbol{h}_0$$

where $\boldsymbol{q}$ is the eigenvector of the largest eigenvalue, and $\boldsymbol{v}$ is the assocoiated vector of the inverse matrix.

When training with long sequences this can result in values exploding when $\lambda > 1$ and vanishing when $\lambda < 1$.

3. (a) A sigmoid activation function is usually used for gating as it runs between 0 and 1

$$\boldsymbol{\sigma}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = \mathbf{1}./(1 + \exp(-(\mathbf{W}^{\mathtt{f}}\boldsymbol{x}_t + \mathbf{W}^{\mathtt{r}}\boldsymbol{h}_{t-1} + \mathbf{b})))$$

where ./ indicates elemntwise division, and $\mathbf{1}$ is a vector of ones.

(b) This is a standard GRU, hence relationships

$$\begin{array}{rcl}
\boldsymbol{i}_{\mathtt{f}} & = & \boldsymbol{\sigma}(\mathbf{W}^{\mathtt{f}}_{\mathtt{f}}\boldsymbol{x}_t + \mathbf{W}^{\mathtt{r}}_{\mathtt{f}}\boldsymbol{h}_{t-1} + \mathbf{b}_{\mathtt{f}}) \\
\boldsymbol{i}_{\mathtt{o}} & = & \boldsymbol{\sigma}(\mathbf{W}^{\mathtt{f}}_{\mathtt{o}}\boldsymbol{x}_t + \mathbf{W}^{\mathtt{r}}_{\mathtt{o}}\boldsymbol{h}_{t-1} + \mathbf{b}_{\mathtt{o}}) \\
\tilde{\boldsymbol{h}}_t & = & \mathbf{f}(\mathbf{W}^{\mathtt{f}}_{\mathtt{h}}\boldsymbol{x}_t + \mathbf{W}^{\mathtt{r}}_{\mathtt{h}}(\boldsymbol{i}_{\mathtt{f}} \odot \boldsymbol{h}_{t-1}) + \mathbf{b}_{\mathtt{h}}) \\
\boldsymbol{h}_t & = & \boldsymbol{i}_{\mathtt{o}} \odot \boldsymbol{h}_{t-1} + (\mathbf{1} - \boldsymbol{i}_{\mathtt{o}}) \odot \tilde{\boldsymbol{h}}_t \\
\boldsymbol{y}_t & = & \mathbf{g}(\mathbf{W}_{\mathtt{y}}\boldsymbol{h}_t + \mathbf{b}_{\mathtt{y}})
\end{array}$$

The gating activation functions have been given in (a). $\mathbf{f}()$ can be any form of activation function (e.g. tanh/ReLU) and $\mathbf{g}()$ is a softmax activation function as this is a classification task.

(c) The overall functions have the form

$$\begin{array}{rcl}
\boldsymbol{h}^{(1)}_t & = & \text{GRU}(\boldsymbol{x}_t, \boldsymbol{h}^{(1)}_{t-1}) \\
\boldsymbol{h}^{(2)}_t & = & \text{GRU}(\boldsymbol{h}^{(1)}_t, \boldsymbol{h}^{(2)}_{t-1}) \\
\boldsymbol{y}_t & = & \mathbf{g}(\mathbf{W}_{\mathtt{y}}\boldsymbol{h}^{(2)}_t + \mathbf{b}_{\mathtt{y}})
\end{array}$$

where GRU() includes the first 4 expressions in (b).

4. (a) Both normalisations apply the same form of transformation, a whitening of the data to yield zero mean and unit variance. The difference is how the means and varianvces are computed: batch norm operates on a batch of data with

a separate term for each elementy of the feature vector;l layer norm acts on a single input computing the mean and variance over the complete layer.

The difference in calculation of the normalisation terms impacts gthe complexity of applying the normalisation at run time. For batch norm an expected mean and variance must be computed, as at inference time there is no batch to operate over. Layer norm can just operate directly as in only requires a single input.

(optional) It is possible to generalise both forms of normalisation to include trainable parameters, this is the form used in the transformer networks. Here

$$\tilde{\boldsymbol{y}}_p^{(k)} = \boldsymbol{w}^{(k)} \odot \left(\frac{1}{\tilde{\sigma}_p^{(k)}} (\boldsymbol{y}_p^{(k)} - \tilde{\mu}_p^{(k)} \mathbf{1}) \right) + \mathbf{b}^{(k)}$$

where $\odot$ is elementwise multiplication, and $\boldsymbol{w}^{(k)}$ and $\boldsymbol{b}^{(k)}$ are trainable vectors of size n_k.

(b) Rewrite expression in the same form as batch normalisation

$$\frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{y}^{(k)}} = \frac{\partial \tilde{\boldsymbol{y}}^{(k)}}{\partial \boldsymbol{y}^{(k)}} \frac{\partial E(\boldsymbol{\theta})}{\partial \tilde{\boldsymbol{y}}^{(k)}}$$

The second term of the RHS is the same as the form when no layer-normalisation is applied. For the first term

$$\frac{\partial \tilde{\boldsymbol{y}}^{(k)}}{\partial \boldsymbol{y}^{(k)}} = \begin{bmatrix} 1 - \frac{1}{n_k} & -\frac{1}{n_k} & \dots & -\frac{1}{n_k} \\ -\frac{1}{n_k} & 1 - \frac{1}{n_k} & \dots & -\frac{1}{n_k} \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{1}{n_k} & -\frac{1}{n_k} & \dots & 1 - \frac{1}{n_k} \end{bmatrix}$$

5. (a) When $q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1) = p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\lambda})$

$$\begin{aligned} \sum_{i=1}^{n} \left\langle \log \left(\frac{p(\boldsymbol{x}_i, \boldsymbol{z}; \boldsymbol{\lambda})}{q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1)} \right) \right\rangle_{q_2(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_2)} &= \sum_{i=1}^{n} \left\langle \log \left(\frac{p(\boldsymbol{x}_i, \boldsymbol{z}; \boldsymbol{\lambda})}{p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\lambda})} \right) \right\rangle_{q_2(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_2)} \\ &= \sum_{i=1}^{n} \langle \log (p(\boldsymbol{x}_i; \boldsymbol{\lambda})) \rangle_{q_2(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_2)} \\ &= \sum_{i=1}^{n} \log (p(\boldsymbol{x}_i; \boldsymbol{\lambda})) \end{aligned}$$

provided $q_2(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_2)$ is a valid PDF.

(b) When $q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1) = q_2(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_2)$

$$\sum_{i=1}^{n} \left\langle \log \left(\frac{p(\boldsymbol{x}_i, \boldsymbol{z}; \boldsymbol{\lambda})}{q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1)} \right) \right\rangle_{q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1)} = \sum_{i=1}^{n} \left\langle \log \left(\frac{p(\boldsymbol{z}|\boldsymbol{x}_i; \boldsymbol{\lambda})}{q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1)} \right) + \log \left(\frac{p(\boldsymbol{x}_i, \boldsymbol{z}; \boldsymbol{\lambda})}{p(\boldsymbol{z}|\boldsymbol{x}_i; \boldsymbol{\lambda})} \right) \right\rangle_{q_1(\boldsymbol{z}; \tilde{\boldsymbol{\lambda}}_1)}$$

But using the attributes of the KL-divergence between two distributions

$$\left\langle \log\left(\frac{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\lambda})}{q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1)}\right)\right\rangle_{q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1)} = -\left\langle \log\left(\frac{q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1)}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\lambda})}\right)\right\rangle_{q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1)} \leq 0$$

with equality only when $q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\lambda})$ for all values of $\boldsymbol{x}$ and $\boldsymbol{z}$. Hence

$$\sum_{i=1}^{n}\left\langle \log\left(\frac{p(\boldsymbol{x}_i,\boldsymbol{z};\boldsymbol{\lambda})}{q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1)}\right)\right\rangle_{q_1(\boldsymbol{z};\tilde{\boldsymbol{\lambda}}_1)} \leq \sum_{i=1}^{n}\log\left(p(\boldsymbol{x}_i;\boldsymbol{\lambda})\right)$$

using the result of part (a).

(c) For some models, for example the Variational Autoencoder, it is not possible to compute the (log-)likelihood exactly. This means that even gradient-based approaches cannot be used. Rather than optimising the log-likelihood, a lower bound defined by the variational approximation given can be used.

Additionally for a wide-range of models it is not possible to compute $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\lambda})$ which is used in EM. Again this form of variational (lower-bound) approximation can be applied to yield variational EM.

6. As the name implies linear classifiers only generate decision boundaries of the form $\mathbf{w}^{\mathrm{T}}\boldsymbol{x} + b = 0$. Non linear mappings of the feature can increase the effective dimensionality. A linear decision boundary in this mapped space will be non-linear in the original space. Note there is an increase in the number of model parameters that need to be trained for the decision boundary.

   A mapping will exist if the points have distinct labels (i.e. no point has multiple class labels associated with it.)

7. Recall that the training set is

$$\begin{aligned}
\boldsymbol{x}_1 &= (1,1)\,, & t_1 &= 1\,,\\
\boldsymbol{x}_2 &= (-1,-1)\,, & t_2 &= 1\,,\\
\boldsymbol{x}_3 &= (-1,1)\,, & t_3 &= -1\,,\\
\boldsymbol{x}_4 &= (1,-1)\,, & t_4 &= -1\,.
\end{aligned}$$

Given the kernel $k(\boldsymbol{x}_n,\boldsymbol{x}_m) = (1+\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_m)^2$, the Gram matrix is

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1\\ 1 & 9 & 1 & 1\\ 1 & 1 & 9 & 1\\ 1 & 1 & 1 & 9 \end{bmatrix}, \tag{1}$$

We find the bias b by noting that the classifier's output at the support vectors should be ± 1. In particular, $y(\mathbf{x}_1)$ should be 1 since $t_1 = 1$:

$$y(\boldsymbol{x}_1) = b + \sum_{n=1}^{4} a_n t_n k(\boldsymbol{x}_n, \boldsymbol{x}_1) = b + \frac{1}{8} \sum_{n=1}^{4} t_n k_{n,1} = b + 1 = 1 \Rightarrow b = 0\,.$$

Let $\boldsymbol{t} = (t_1, t_2, t_3, t_4)^{\mathrm{T}}$. Then

$$(y(\boldsymbol{x}_1), y(\boldsymbol{x}_2), y(\boldsymbol{x}_3), y(\boldsymbol{x}_4))^{\mathrm{T}} = \frac{1}{8}\mathbf{K}\boldsymbol{t} = (1, 1, -1, -1)^{\mathrm{T}}\,.$$

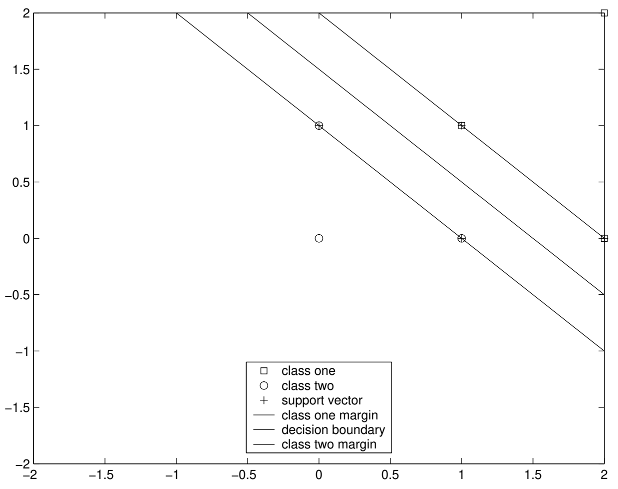
It is then easy to see that $t_n y(\boldsymbol{x}_n) \geq 1$ for $n = 1, \ldots, 4$.

Let $\boldsymbol{x} = (x_1, x_2)^{\mathrm{T}}$ be an arbitrary input to the classifier and let us define $\boldsymbol{k}(\boldsymbol{x}) = (k(\boldsymbol{x}_1, \boldsymbol{x}), k(\boldsymbol{x}_2, \boldsymbol{x}), k(\boldsymbol{x}_3, \boldsymbol{x}), k(\boldsymbol{x}_4, \boldsymbol{x}))^{\mathrm{T}}$. The classifier's output at $\boldsymbol{x}$ is defined as

$$y(\mathbf{x}) = \frac{1}{8}\boldsymbol{t}^{\mathrm{T}}\boldsymbol{k}(\boldsymbol{x}) = \frac{1}{8}\left[(1 + \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x})^2 + (1 + \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x})^2 - (1 + \boldsymbol{x}_3^{\mathrm{T}}\boldsymbol{x})^2 - (1 + \boldsymbol{x}_4^{\mathrm{T}}\boldsymbol{x})^2\right] = x_1 x_2\,.$$

Therefore, $y(\mathbf{x}) = 0 \Leftrightarrow x_1 x_2 = 0$.

8. (a) The decision boundary and margins are shown below.



(b) There are four support vectors (points on the margin):

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \boldsymbol{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \boldsymbol{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \boldsymbol{x}_3 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

We note that $y(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\mathbf{w} + b$ and that the vector $\mathbf{w}$, which is orthogonal to the decision boundary, is a scalar times the unit vector $(1,1)^{\mathrm{T}}$, that is, $\mathbf{w} = w \cdot (1,1)^{\mathrm{T}}$. We also note that $y(\boldsymbol{x}_1) = 1$, $y(\boldsymbol{x}_5) = -1$ and $y(\boldsymbol{x}_6) = -1$. Given these equations, we can solve for b and the scalar w to obtain $b = -3$ and $w = 2$. Therefore, $\mathbf{w} = (2,2)^{\mathrm{T}}$.

(c) There are multiple solutions for the Lagrange multipliers $\boldsymbol{a}$ (though a unique decision boundary) as it is an under-specified problem. We will assume that the Lagrange multiplier for the fourth support vector is actually zero. The classifier can be expressed in terms of the Lagrange multipliers as

$$\begin{aligned} y(\boldsymbol{x}) &= a_1\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x} - a_5\boldsymbol{x}_5^{\mathrm{T}}\boldsymbol{x} - a_6\boldsymbol{x}_6^{\mathrm{T}}\boldsymbol{x} + b \\ &= a_1\begin{bmatrix}1\\1\end{bmatrix}^{\mathrm{T}}\boldsymbol{x} - a_5\begin{bmatrix}1\\0\end{bmatrix}^{\mathrm{T}}\boldsymbol{x} - a_6\begin{bmatrix}0\\1\end{bmatrix}^{\mathrm{T}}\boldsymbol{x} + b\,. \end{aligned}$$

Using $y(\boldsymbol{x}_1) = 1$, $y(\boldsymbol{x}_5) = -1$, $y(\boldsymbol{x}_6) = -1$, $b = -3$, and taking into account the training constraint $\sum_{n=1}^{6} t_n a_t = 0$, we can show that $a_1 = 4$, $a_5 = 2$, $a_6 = 2$, while $a_2, a_3, a_4 = 0$ (because $\boldsymbol{x}_2$ and $\boldsymbol{x}_4$ are not support vectors and a_3 is assumed to be zero). The KKT conditions are:

$$\begin{aligned} a_n \geq 0 \quad &\text{for } n = 1,\ldots,6\,, \\ t_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + b) - 1 \geq 0 \quad &\text{for } n = 1,\ldots,6\,, \\ a_n\left[t_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + b) - 1\right] = 0 \quad &\text{for } n = 1,\ldots,6\,, \\ \nabla\boldsymbol{w}\left\{\frac{1}{2}||\boldsymbol{w}||^2 - \sum_{n=1}^{6} a_n\left[t_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + b) - 1\right]\right\} = \boldsymbol{w} - \sum_{n=1}^{6} a_n t_n \boldsymbol{x}_n = \mathbf{0}\,, \\ \nabla_b\left\{\frac{1}{2}||\boldsymbol{w}||^2 - \sum_{n=1}^{6} a_n\left[t_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + b) - 1\right]\right\} = -\sum_{n=1}^{6} a_n t_n = 0\,. \end{aligned}$$

By replacing $\boldsymbol{w}$, b and $\boldsymbol{a}$ with their values, we can verify that all these constraints are satisfied.

9. (a) The feature-space maps the variable length sequences into a fixed dimensionality. SVMs (empirically) generalise well for large dimensional feature spaces which will occur when M gets large. For the case given the dimensionality is $M + (M(M+1))/2$ [noting the symmetry in the second derivative (the function is twice differentiable and continuous].

(b) We need

$$\frac{\partial}{\partial\mu_i}\log\left(\prod_{t=1}^{T}\sum_{m=1}^{M} c_m\mathcal{N}(x_t;\mu_m,\sigma_m^2)\right) = \sum_{t=1}^{T}\frac{\partial}{\partial\mu_i}\log\left(\sum_{m=1}^{M} c_m\mathcal{N}(x_t;\mu_m,\sigma_m^2)\right)$$

This can be simply written as

$$\begin{aligned} \frac{\partial}{\partial\mu_i}\log(P(\mathbf{X}_{1:T})) &= \sum_{t=1}^{T}\frac{1}{p(x_t)}c_i\frac{\partial}{\partial\mu_i}\mathcal{N}(x_t;\mu_i,\sigma_i^2) \\ &= \sum_{t=1}^{T} P(i|x_t)\frac{1}{\sigma_i^2}(x_t - \mu_i) \end{aligned}$$

(c) From part (b) (note it assumed that $i \neq j$)

$$\frac{\partial^2}{\partial\mu_j\partial\mu_i}\log(p(\mathbf{X}_{1:T})) \quad = \quad \sum_{t=1}^{T}\frac{\partial}{\partial\mu_j}\left(P(i|x_t)\frac{1}{\sigma_i^2}(x_t-\mu_i)\right)$$

Only the posterior is a function of the mean of component j. This can be calculated

$$\begin{aligned}\frac{\partial}{\partial\mu_j}P(i|x_t) &= -\frac{c_i\mathcal{N}(x_t;\mu_i,\sigma_i^2)}{(p(x_t))^2}c_j\frac{\partial}{\partial\mu_j}\mathcal{N}(x_t;\mu_j,\sigma_j^2) \\ &= -P(i|x_t)P(j|x_t)\frac{1}{\sigma_j^2}(x_t-\mu_j)\end{aligned}$$

It is simple to see that the form in the question is simply obtained.

$$\frac{\partial^2}{\partial\mu_j\partial\mu_i}\log(p(\mathbf{X}_{1:T})) = -\sum_{t=1}^{T}P(i|x_t)P(j|x_t)\frac{(x_t-\mu_j)(x_t-\mu_i)}{\sigma_i^2\sigma_j^2}$$

These second order statistics have the potential for additional information as they are not a linear transform of the first order statistics. Furthermore it is not possible to obtain this form from a standard kernel operation on the first order statistics due to the summation over time.

M.J.F. Gales & J. M. Hernandez-Lobato
Nov 2017,2018,2020