1 DERIVATION OF LEAVE-ONE-OUT OBJECTIVE

In this section we derive an expression for the leave-one-out objective and show that this does not require training of \( N \) models. A similar derivation can be found in Vehtari et al. (2016). Let \( D_{\sim n} = \{ X_{\sim n}, y_{\sim n} \} \) be the dataset resulting from removing observation \( n \). Then our leave-one-out objective is given by:

\[
L_{\text{oo}}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(y_n | x_n, D_{\sim n}, \theta). \tag{1}
\]

We now that the marginal posterior can be computed as:

\[
p(f_n | D) = p(f_n | X_{\sim n}, y_{\sim n}, x_n, y_n) = \frac{p(y_n | f_n) p(f_n | x_n, D_{\sim n})}{p(y_n | x_n, D_{\sim n}, \theta)} \tag{2}
\]

and re-arranging terms

\[
\int p(f_n | x_n, D_{\sim n}, \theta) df_n = \int \frac{p(f_n | D, \theta) p(y_n | x_n, D_{\sim n}, \theta)}{p(y_n | f_n)} df_n, \tag{3}
\]

\[
p(y_n | x_n, D_{\sim n}, \theta) = 1 / \int \frac{p(f_n | D, \theta)}{p(y_n | f_n)} df_n. \tag{4}
\]

\[
\log p(y_n | x_n, D_{\sim n}; \theta) = - \log \int \frac{p(f_n | D, \theta)}{p(y_n | f_n)} df_n, \tag{5}
\]

and substituting this expression in Equation (1) we have

\[
L_{\text{oo}}(\theta) = - \frac{1}{N} \sum_{n=1}^{N} \log \int p(f_n | D, \theta) \frac{1}{p(y_n | f_n)} df_n. \tag{6}
\]

We see that the objective only requires estimation of the marginal posterior \( p(f_n | D, \theta) \), which we can approximate using variational inference, hence:

\[
L_{\text{oo}}(\theta) \approx - \frac{1}{N} \sum_{n=1}^{N} \log \int q(f_n | D, \theta) \frac{1}{p(y_n | f_n)} df_n, \tag{7}
\]

where \( q(f_n | D, \theta) \) is our approximate variational posterior.
Table 1: The datasets used in the experiments and the corresponding models used. \(N_{\text{train}}, N_{\text{test}}, D\) are the number of training points, test points and input dimensions respectively.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(N_{\text{train}})</th>
<th>(N_{\text{test}})</th>
<th>(D)</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SARCOS</td>
<td>44,484</td>
<td>4,449</td>
<td>21</td>
<td>GPRN</td>
</tr>
<tr>
<td>RECTANGLES-IMAGE</td>
<td>12,000</td>
<td>50,000</td>
<td>784</td>
<td>Binary classification</td>
</tr>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>10,000</td>
<td>784</td>
<td>Multi-class classification</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>50,000</td>
<td>10,000</td>
<td>3072</td>
<td>Multi-class classification</td>
</tr>
<tr>
<td>MNIST8M</td>
<td>8.1M</td>
<td>10,000</td>
<td>784</td>
<td>Multi-class classification</td>
</tr>
</tbody>
</table>

Figure 1: NLP for multiclass classification using a softmax likelihood model on the MNIST dataset. VAR shows the performance of AutoGP where all parameters are learned using only the variational objective \(\hat{L}_{\text{elbo}}\), while LOO represents the performance of AutoGP when hyperparameters are learned using the leave-one-out objective \(\hat{L}_{\text{loo}}\).

2 ADDITIONAL DETAILS OF EXPERIMENTS

2.1 EXPERIMENTAL SET-UP

The datasets used are described in Table 1. We trained our model stochastically using the RMSprop optimizer provided by TensorFlow (Abadi et al., 2015) with a learning rate of 0.003 and mini-batches of size 1000. We initialized inducing point locations by using the k-means clustering algorithm, and initialized the posterior mean to a zero vector, and the posterior covariances to identity matrices. When jointly optimizing \(\hat{L}_{\text{loo}}\) and \(\hat{L}_{\text{elbo}}\), we alternated between optimizing each objective for 100 epochs. Unless otherwise specified we used 100 Monte-Carlo samples to estimate the expected log likelihood term.

All timed experiments were performed on a machine with an Intel(R) Core(TM) i5-4460 CPU, 24GB of DDR3 RAM, and a GeForce GTX1070 GPU with TensorFlow 0.10rc.

2.2 ADDITIONAL RESULTS

Figure 1 shows the NLP for our evaluation of the LOO-CV-based hyperparameter learning. As with the error rates described in the main text, the NLP obtained with LOO-CV are significantly better than those obtained with a purely variational approach.

References