University of Oslo
Department of Mathematics

Author: Jonas Semprini Næss

Fall, 2024
Mandatory Assignment 3

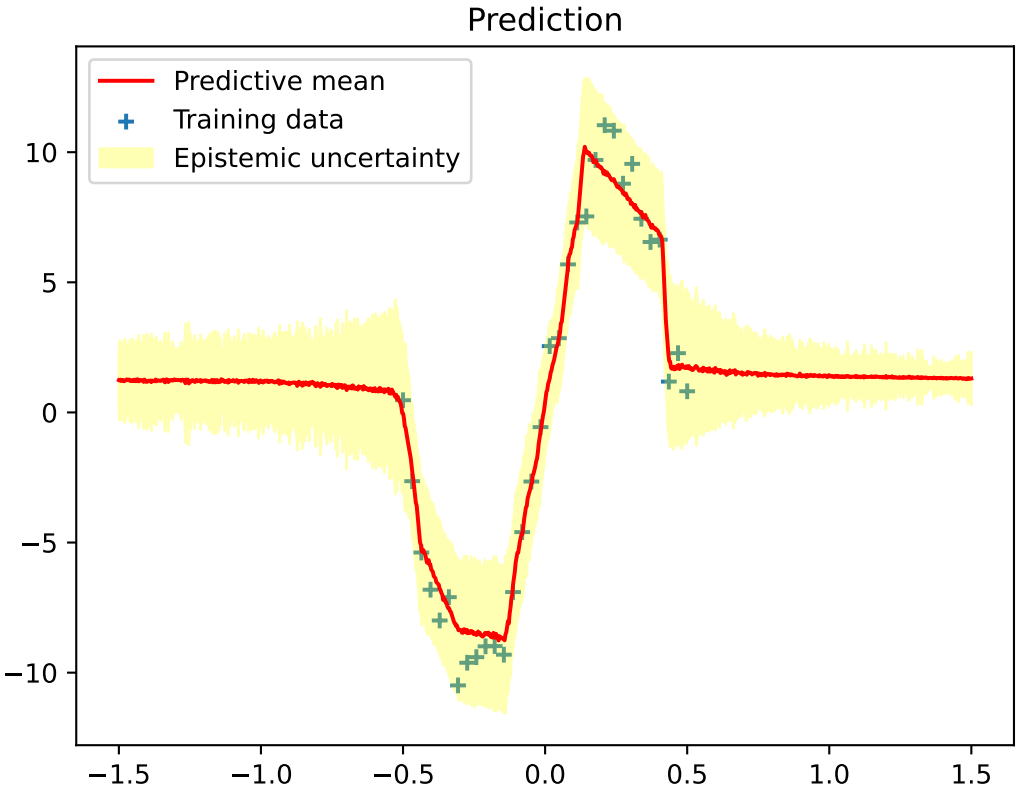
TEK5040 - Deep Learning for Autonomous Systems

Answers and Analysis

Monte Carlo Droupout Test Variance

Run `mc_train_test.py` *and comment on the variance of the output as depicted in the generated figure. Hint: Run the script several times to get a feeling of how it looks like in general. Note the difference in test and training sets, i.e. training set has 32 samples in the range of* $(-0.5, 0.5)$ *while the test set has 1500 samples in the range of* $(-1.5, 1.5)$.

Answer:



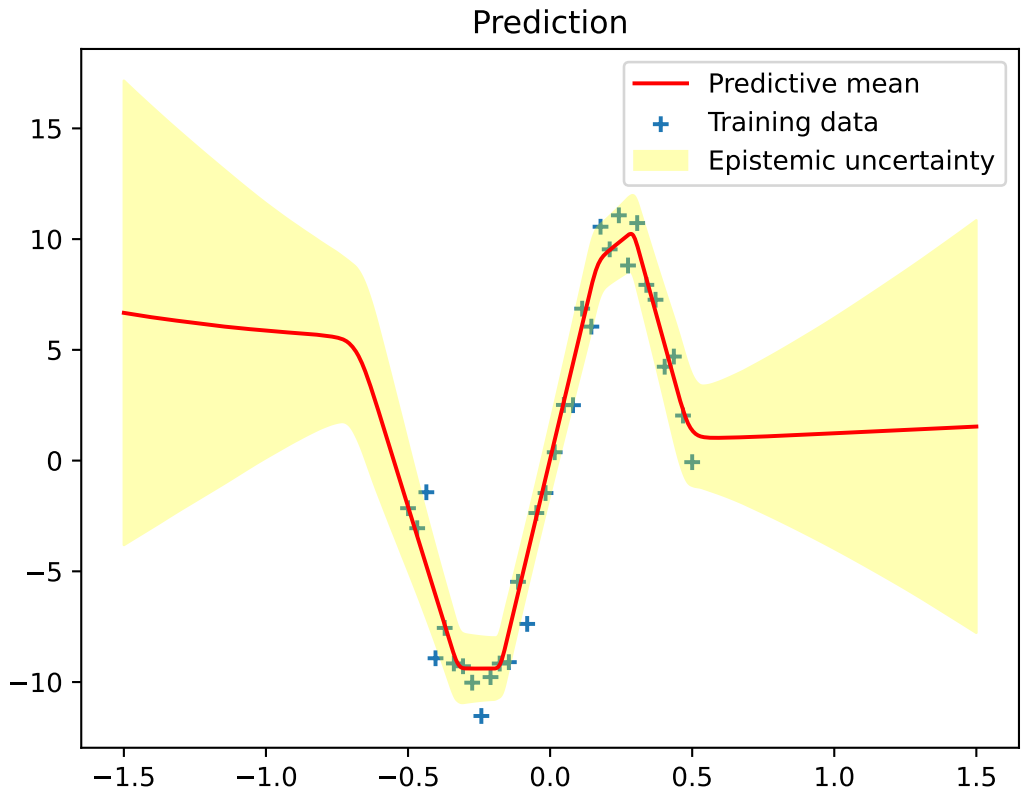
Figur 1: Monte Carlo Dropout

The figure shows that the model's prediction variance is low within the training range of -0.5 to 0.5, meaning it makes consistent predictions with minimal uncertainty. This is illustrated by a narrow yellow band around the predictive mean within this range. Outside this interval, however, the variance increases, especially in the extended test range from -1.5 to -0.5 and 0.5 to 1.5. Here, the model's predictions become more uncertain, as shown by the broader yellow shaded area. This indicates the model's confidence drops when predicting values outside its training data. The plot, therefore, demonstrates that while the model performs well within the trained range, its predictions become less reliable with inputs beyond this interval.

Variational Inference

Run vi train `test.py` *and comment on the variance of the output. Compare it with that of MC-dropout.*

Answer:



Figur 2: Variational Inference

For the output variance of the variational inference (VI) model, it tends to show lower predictive uncertainty within the range of the training data and produces more stable

predictions near the training points compared to the MC-dropout model. In VI, the uncertainty is modelled with explicit prior assumptions about weight distributions, giving more controlled and possibly narrower uncertainty bounds. Unlike MC-dropout, which uses random dropout layers to estimate uncertainty, this approach can result in slightly higher predictive variance due to the randomness in sampling.

When comparing both methods, the variance in VI is more structured and interpretable because it directly reflects the learned posterior distributions. MC-dropout focuses more on aleatoric uncertainty (randomness in the data, like noise) and often shows higher variability, especially outside the training range. Still, in our case, by observation of Figure (1) and (2) there seems to be a general pattern of considerably high variance outside the range of our training data and lesser inside.

Dense Variational (1)

Find the line numbers of the code in densevariational.py which implements the reparameterization trick $\boldsymbol{w}^s = \boldsymbol{w}(\lambda, \boldsymbol{\epsilon}^s)$

Answer:

The reparameterization trick is implemented in the `call` method of the `DenseVariational` class, specifically for kernel and bias weights. For the kernel, the trick is applied as:

- Line 59: `kernel = self.kernel_mu + kernel_sigma * tf.random.normal(self.kernel_mu.shape)`

This is mathematically represented as:

$$\boldsymbol{w}^s_{\text{kernel}} = \boldsymbol{\mu}_{\text{kernel}} + \sigma_{\text{kernel}} \cdot \boldsymbol{\epsilon}$$

where $\boldsymbol{\mu}_{\text{kernel}}$ is the mean, σ_{kernel} is the standard deviation, and $\boldsymbol{\epsilon}$ is a random noise vector. The same process applies to the bias term:

- Line 62: **bias = self.bias_mu + bias_sigma * tf.random.normal(self.bias_mu.shape)**

This is represented as:

$$\boldsymbol{w}^s_{\text{bias}} = \boldsymbol{\mu}_{\text{bias}} + \sigma_{\text{bias}} \cdot \boldsymbol{\epsilon}$$

This allows sampling with differentiable weights, crucial for gradient-based optimization.

Dense Variational (2)

Find the line numbers of the code in densevariational.py which adds the first and the third loss components in equation 3 to the computational graph.

Answer:

The first and third loss components of the Negative Evidence Lower Bound (ELBO) are added to the computational graph in lines 64, 65, and 71:

- **Lines 64-65**: The log differences between the variational distribution $q(\mathbf{w}^s)$ and the prior distribution $p(\mathbf{w}^s)$ for the weights and biases are computed. These loss components are added to the model's loss function with `self.add_loss()` for automatic inclusion in backpropagation during optimization.
- **Line 71**: The loss for both kernel weights and biases is calculated, contributing to the total ELBO loss, which is also added to the model's loss.

Dense Variational (3)

Find the line numbers of the code where the second loss component in equation 3 is added to the computation graph.

Answer:

The second loss component, regularizing the prior $p(\boldsymbol{w})$, is added implicitly in `vi_train_test.py` via the `DenseVariational` layer. This is done using the layer's `add_loss()` function, and the exact code is inside the `DenseVariational` class, not visible in `vi_train_test.py`.

- The prior regularization loss is included automatically when the `DenseVariational` layer is instantiated in `vi_train_test.py`.
- This is similar to how the data-dependent loss $\ln p(D|\boldsymbol{w})$ is added in `tf.keras.Model.compile` (lines 40-42).

Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC) is an alternative to variational inference. Give an advantage and a disadvantage of MCMC.

Answer:

Markov Chain Monte Carlo (MCMC) provides exact samples from the posterior distribution, making it a highly accurate method for Bayesian inference. However, its major disadvantage is its computational inefficiency, particularly for high-dimensional problems, where it can be slow to converge and require many iterations to produce "accurate" results. This makes MCMC more computationally expensive compared to variational inference, which can scale more easily but may sacrifice some accuracy in capturing the true posterior distribution.