Lucas Versini

Practical session report
December 14, 2024

Theoretical Foundations of Deep Learning

Practical session report

Remark: For the questions specified on the website, the answers can be found in this report. For other questions which require some interpretation, the answers are in the notebooks.

Week 1 - Bayesian Models and Uncertainty

Question 1.4

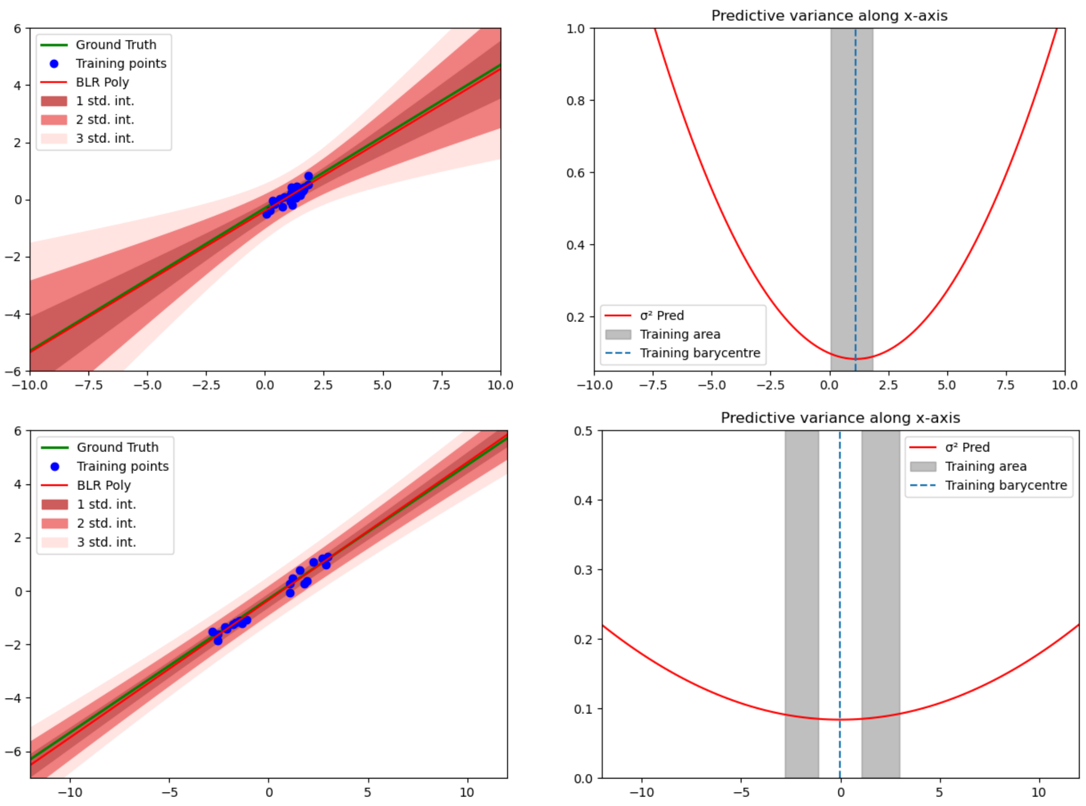

Figure 1: Predictive distribution on the synthetic datasets

In the first case, we see that the predictive variance increases far from the data. This is very intuitive, since far away from the data, we do not know how the data actually behaves.

However, in the second case, we notice that the variance is minimal somewhere where there is no data point. This is not intuitive: like before, we would expect the uncertainty to be large where there is no data (and therefore, where we do not know how the data behaves).

In fact, we will see in the next question that (at least for $\alpha = 0, \beta = 1$) the variance is minimal at the barycenter of the data points.

Question 1.5

Intuitively, if x is close to the training distribution, then it should be easy to predict the behavior of f at x, based on what we know about the training distribution. But when x is far from the training distribution, there should be more uncertainty.

Let us assume $\alpha = 0$ (no prior, *i.e.* uniform one) and $\beta = 1$.

Then $\mathbf{\Sigma}^{-1} = \alpha \mathbf{I_2} + \beta \mathbf{\Phi^T \Phi} = \mathbf{\Phi^T \Phi} = \begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_N \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} = \begin{pmatrix} N & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 \end{pmatrix}$.

So $\mathbf{\Sigma} = \begin{pmatrix} N & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 \end{pmatrix}^{-1} = C \begin{pmatrix} \sum_{i=1}^N x_i^2 & -\sum_{i=1}^N x_i \\ -\sum_{i=1}^N x_i & N \end{pmatrix}$, where $C = \dfrac{1}{n \sum_{i=1}^N x_i^2 - \left(\sum_{i=1}^N x_i^2 \right)} > 0$ (Cauchy-Schwarz).
And we know that the variance at x is:

$$\sigma^2_{\text{pred}}(x) = \underbrace{\frac{1}{\beta}}_{=1} + \mathbf{\Phi}(x)^T \mathbf{\Sigma \Phi}(x) = \Sigma_{2,2} x^2 + 2\Sigma_{1,2} x + (\Sigma_{1,1} + 1).$$

Now, notice that $\Sigma_{2,2} = CN > 0$.
So $\sigma^2_{\text{pred}}(x)$ goes to $+\infty$ as x approaches $\pm\infty$, *i.e.*, $\boxed{\text{the variance increases far from training distribution}}$.

We also notice that $\Sigma_{2,2} x^2 + 2\Sigma_{1,2} x + (\Sigma_{1,1} + 1)$ is minimal for $x = \dfrac{-2\Sigma_{1,2}}{2\Sigma_{2,2}} = \dfrac{1}{N} \sum_{i=1}^N x_i$, i.e., **the variance is minimal at the barycenter**. Now, if we had $\alpha \neq 0$, then there is a (non-uniform) prior, which would change the x minimizing $\sigma^2_{\text{pred}}(x)$.

Questions 2.4 / 2.5

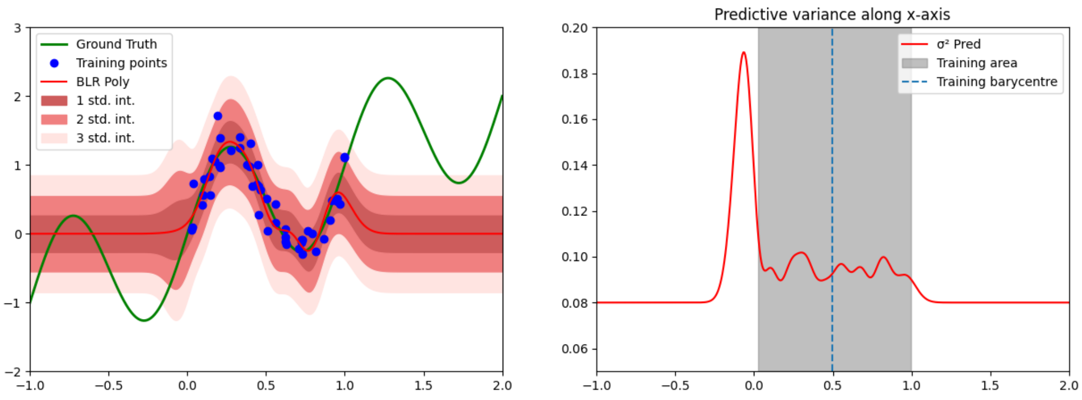

Figure 2: Gaussian basis feature maps

We have $\sigma^2_{\text{pred}}(x) = \frac{1}{\beta} + \mathbf{\Phi}(x)^T \mathbf{\Sigma \Phi}(x)$,, where $\mathbf{\Phi}(x) = \left(\exp\left(-\dfrac{(x-\mu_0)^2}{2s^2} \right) \cdots \exp\left(-\dfrac{(x-\mu_M)^2}{2s^2} \right) \right)^T$.

So we see that $\mathbf{\Phi}(x) \xrightarrow[x \to \pm\infty]{} \mathbf{0}$, so $\mathbf{\Phi}(x)^T \mathbf{\Sigma \Phi}(x) \xrightarrow[x \to \pm\infty]{} 0$ and $\boxed{\sigma^2_{\text{pred}}(x) \xrightarrow[x \to \pm\infty]{} \frac{1}{\beta}}$.

In the notebook, $\beta = 12.5$, so $\dfrac{1}{\beta} = 0.08$, which is indeed the limit of the predictive variance far from training distribution that we see in Fig 2.

Week 2 - Bayesian Neural Networks

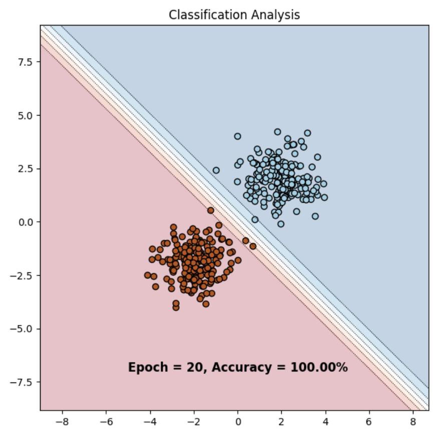

Figure 3: MAP estimate

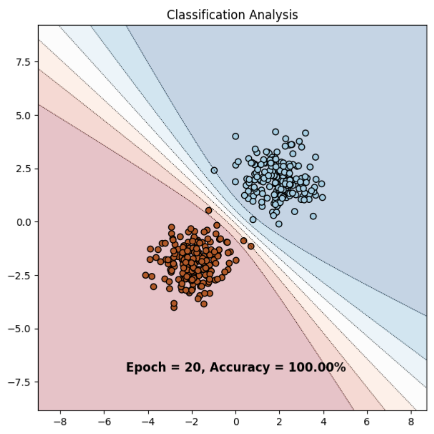

Figure 4: Laplace approximation

Question 1.2

We see in Fig. 3 that when we use the MAP estimate, $p(\mathbf{y} = 1 \mid \boldsymbol{x}, \boldsymbol{w}_{\text{MAP}})$ remains constant far from the training data, close to 0 or 1 depending on the considered region (in fact, the uncertainty only depends on the distance to the prediction hyperplane). Even though this works well in the above dataset, in practice we would prefer having a probability close to 0.5, which would show that the model is uncertain far from the training data.

However, when using Laplace approximation, uncertainty increases far from the training data, as can be seen in Fig. 4.

We also see that both models fit the data very well (typically with an accuracy of about 100%), and are roughly similar (which is partly explained by the fact that $\mu_{lap} = w_{\text{MAP}}$).

Question 1.3

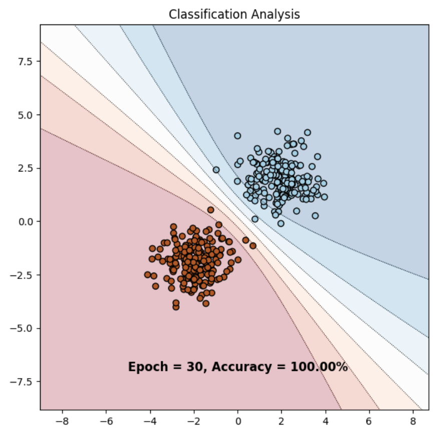

Figure 5: Variational inference

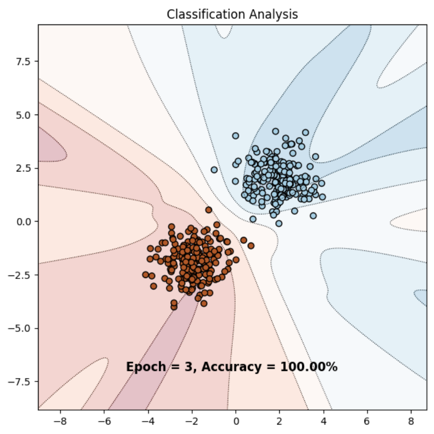

Figure 6: Variational inference (training)

We see in Fig. 5 the results obtained with Variational Inference.

The results look quite similar to the ones we obtained using Laplace approximation: the predictions are mostly the same, and uncertainty increases far from the training data.

However, during training, we see a very different evolution, see Fig. 6. So:

- The MAP estimate used a Dirac on w, which led to a poor uncertainty quantification;
- Using Laplace approximation means assuming that the posterior distribution of w is Gaussian, which led to a better quantification of the uncertainty;
- Using Variational Inference means assuming that the weights are Gaussian and independent, finding the corresponding means and covariance matrices by minimizing the KL divergence, and then predicting by sampling several w.

So the main difference between Laplace approximation and Variational Inference (VI) is that for VI, during training, the weights are random, which is not the case for Laplace approximation.

Question 2.1

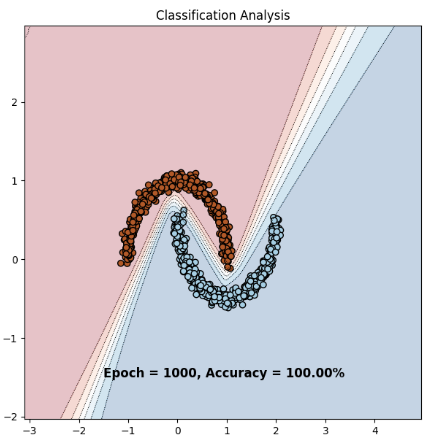

Figure 7: Bayesian VI (no dropout)

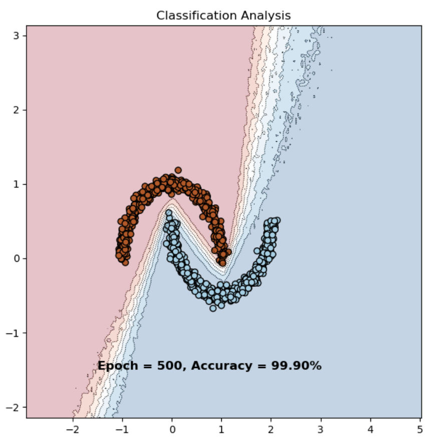

Figure 8: MC dropout

The results obtained with MC Dropout can be seen in Fig. 8.
We can see a few advantages of using MC Dropout instead of Bayesian Variational Inference:

- MC Dropout is very easy to implement.
- MC Dropout requires less parameters: there is no need for a mean and a variance for each weight.
- MC Dropout only assumes that the different layers are independent, and not that the neurons in a same layer are independent.

However, as can be seen by comparing Fig. 7 and Fig. 8, the result obtained with MC Dropout is noisy; of course, this noise can be reduced by increasing the number of times the data is fed to the model during inference.

Week 3 - Applications of Deep Learning Robustness

I.1 - Most uncertain vs confident samples

We show some confident and uncertain samples in Fig. 9 and Fig. 10 respectively.

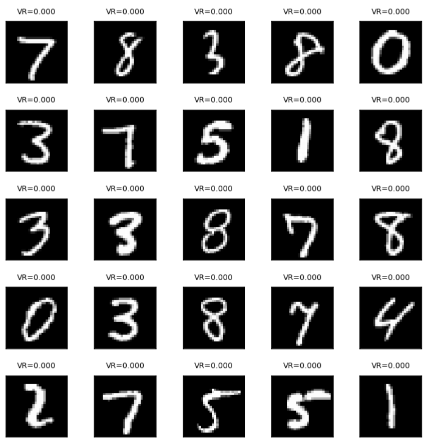

Figure 9: Confident samples

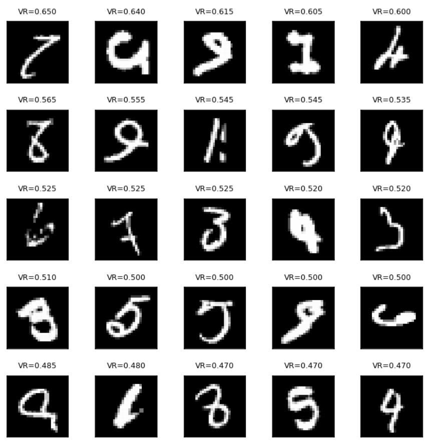

Figure 10: Uncertain samples

The digits from Fig. 9 are indeed well written, easy to understand.

However, the digits from Fig. 10 are hard to read: the one in the first column, third row, is clearly a "6", but it is not very readable. The digits in first row, first and fourth columns could be "1" or "7", the digit in the last column, fourth row, could be a rotated "6" or a rotated "9".

So the results are quite good: the samples for which the model is confident are indeed neat, whereas the ones for which the model is not confident can be hard to read.

We can have a look at a few more examples.

For instance, in Fig. 11, the model is very confident that the digit is a "6", which is indeed the case.

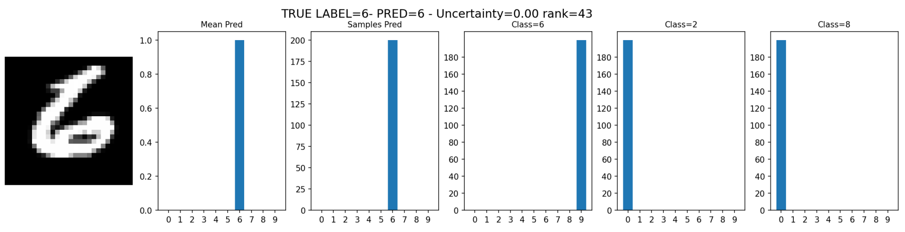

Figure 11: Very confident sample, correctly classified

In Fig. 12, the model is not really sure whether the digit is a "3" or a "5" (which is quite reasonable when we look at the image), but it tends to predict a "5", which is correct. We see that the histogram is rather flat (whereas the previous one was a Dirac).

In Fig. 13, the model is not really sure whether the digit is a "7" or a "9", but it tends to predict a "9", which is incorrect. The histogram is once again flat (even more than before).

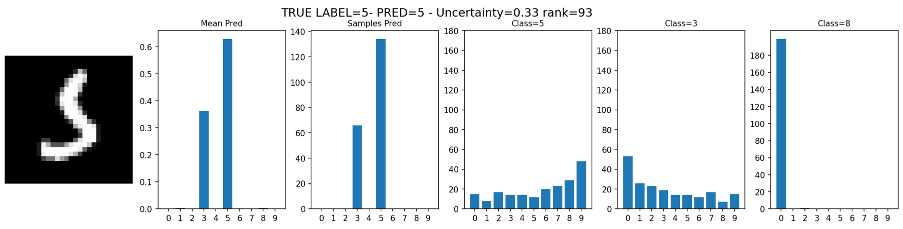

Figure 12: Confident sample, correctly classified

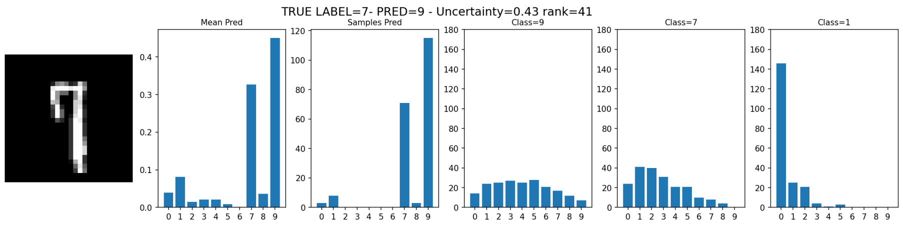

Figure 13: Wrongly classified sample

II.1 and II.2 - Failure prediction

- The goal of failure prediction is to detect, for a given sample, whether the predictions of a model can be trusted. The idea is to somehow have a reliable measure of confidence; when this measure is higher than some threshold, we accept the model's prediction, otherwise we reject it (and require external help).

  This is particularly useful in applications such as health or transportation, where mistakes can be very costly.

  It can also be seen as a way to improve a model: once the model is trained, we can use the confidence of the model to detect which samples are problematic, and maybe to figure out how to improve the model on such samples (for instance, train the model with more samples from a given class).

- The class `LeNetConfidNet` contains a `LeNet` network, as well as a fully-connected neural network which predicts the True Class Probability.

  We can notice that the attributes of the `LeNet` class are present in `LeNetConfidNet` (`self.conv1`, `self.conv2`, `self.fc1` and `self.fc2`). So it is possible to load weights from a `LeNet` model into a `LeNetConfidNet` model, as done with the line:

  `lenet_confidnet.load_state_dict(torch.load('lenet_final.cpkt'), strict=False).`

  Moreover, during training, these previous layers are frozen, and only the layers computing the TCP are optimized.

- We see in Fig. 14 that the AUPR score is higher for ConfidNet than for MCP, which itself has a higher AUPR than MCDropout. However, in some executions, MCP can perform better than ConfidNet.

III.1 - OOD detection

We can see in Fig. 15 that for OOD detection, ODIN performs better than MCP, which performs better than MCDropout.

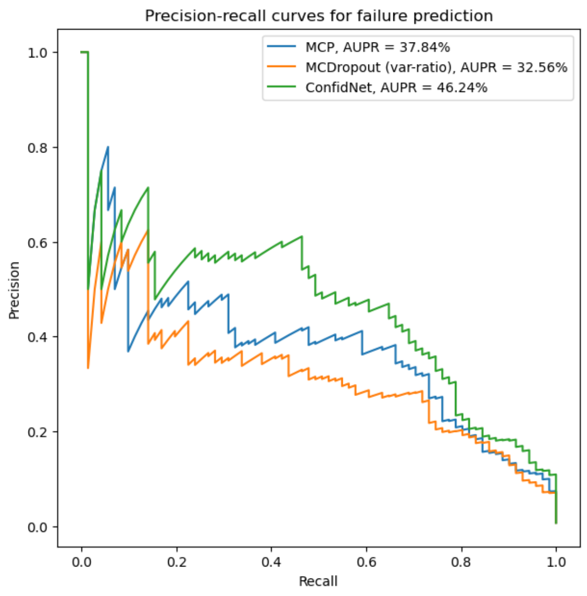

Figure 14: Precision-recall curves for MCP, MCDropout and ConfidNet

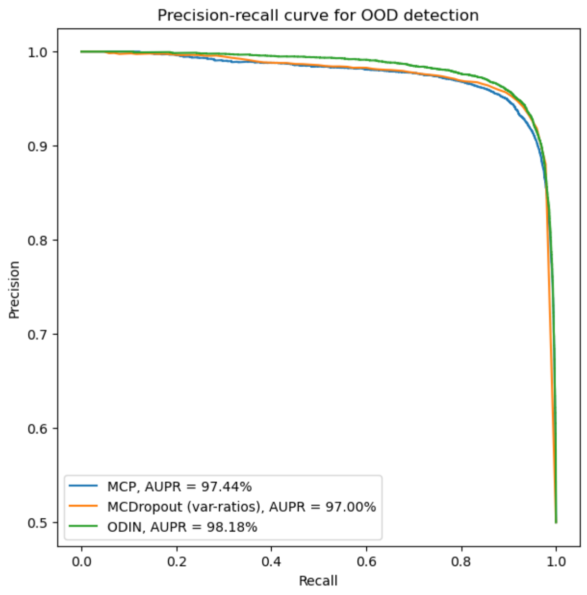

Figure 15: Precision-recall curve for OOD detection

- MCP is based on a very simple idea, which is to train the model to predict the uncertainty.
- MCDropout is also very easy to implement (we just add some Dropout layers), and there is some theoretical background.
- For ODIN, we add both temperature scaling and adversarial attack, which is something that can be particulary useful in OOD detection (as seen in Fig. 15).