Probabilistic Machine Learning Project: Bayesian Convolutional Neural Network

Bruno Bonaiuto Bolivar

Course of AA 2022-2023 - DSSC

1 Introduction

Artificial Neural Networks are a kind of systems that perform a given task, by learning on examples without having prior knowledge about the task. This is done by finding an optimal point estimate for the weights in every node. Generally, the network using point estimates as weights perform well with large datasets, but they fail to express uncertainty in regions with little or no data, leading to overconfident decisions. **Bayesian Convolutional Neural Network using Variational Inference introduces probability distribution over the weights**. and they can be applied to image classification, image super resolution and generative adversarial networks.

A Neural Network consists of three layers: input layer, to feed the data to the model to learn representation, hidden layer, that learns the representation and the output layer, that outputs the results or predictions, generally used to finds patterns in data which are too complex to be recognized by a human to teach to a machine.

A neural network starting layers learns simpler features like edges and corners and subsequent layers learn complex features like colours, textures and so on. However, **in multilayer perceptron where all neurons from one layer are connected with all the neurons in the next layer, instead weight sharing is the main idea behind a convolutional neural network.** The layers after the first one work similarly, taking the "local" features found in the previous hidden layer instead of pixel images. they can see larger portions of the image since they are combining information about increasingly larger subsets of the image.Finally, the final layer makes the correct prediction for the output class. then convolutional and pooling layers are the primary distinctions of Convolutional Neural Nets

The Convolution layer, convolve the input and pass its result to the next layer. And the pooling layer reduce the dimensions of data by computing the maximum or the average of input features from the previous layer and uses them as an output featured map.

2 The Probabilistic Approach

Variational inference: Using Bayesian inference, a prior distribution is used over the space of functions $p(f)$. This distribution represents our prior belief as to which functions are likely to have generated our data. The likelihood $p(Y|f,X)$ captures the process in which given a function observation is generated. Bayes rules allow us to find the posterior distribution given our dataset $p(f|X,Y)$. **the Predictive distribution** for a new point x^* is given by the following approximation taking a finite set of random variables w and conditioning the model on it.

$$p(y^*|x^*,X,Y) = \int p(y^*|f^*)p(f^*|x^*,w)p(w|X,Y)df^*dw. \tag{1}$$

Then the approximate distribution needs to be as close as possible to the posterior distribution obtained from the original model. In this way we minimise the Kullback-Leibler (KL) divergence, which is a measure of similarity between two distributions, resulting in the approximate predictive distribution.

$$q(y^*|x^*) = \int p(y^*|f^*)p(f^*|x^*,w)q(w)df^*dw. \tag{2}$$

Minimising the KL divergence is equivalent to maximising the **log evidence lower bound**

$$KL_{VI} := \int q(w)p(F|X,w)\log p(Y|F)dFdw - KL(q(w)||p(w)) \tag{3}$$

note: This is known as variational inference.

Maximizing the KL divergence between the posterior and the prior over w will result in a variational distribution that learns the data (obtained from log likelihood) and is closer to the prior distribution. This process is to prevent overfitting.

A Bayesian approach to Neural Networks account for uncertainty in parameter estimates and can propagate this uncertainty into predictions. Also, averaging over parameters instead of choosing single point this makes the model robust to overfitting. Several approaches has been proposed in the past for learning in Bayesian Networks: Laplace approximation , MC Dropout,and Variational Inference. We used Bayes by Back-prop.

Bayes by backprop: is a variational inference method to learn the posterior distribution on the weights of a neural network from which weights w can be sampled in backpropagation. It regularises the weights by minimising a compression cost, known as the expected lower bound on the marginal likelihood. then by measuring the Kullback-Leibler (KL) divergence, since the true posterior is typically intractable, an approximate distribution is defined that is aimed to be as similar as possible to the true posterior.

$$\begin{aligned}\theta^{opt} &= arg_\theta\ min\ KL[q_\theta(w|D)||p(w|D)] \\ &= arg_\theta\ min\ KL[q_\theta(w|D)||p(w)] \\ &- E_{q(w|\theta)}[\log p(D|w)] + \log p(D)\end{aligned} \tag{4}$$

where

$$KL[q_\theta(w|D)||p(w)] = \int q_\theta(w|D) \log \frac{q_\theta(w|D)}{p(w)} dw \tag{5}$$

Since the KL-divergence is also intractable to compute exactly, we follow a stochastic variational method. We sample the weights w from the variational distribution. The uncertainty afforded by Bayes by Backprop trained neural networks has been used successfully for training feedforward neural networks in both supervised and reinforcement learning environments, for training recurrent neural networks.

Model weights prunning: The whole idea of model pruning is to reduce the number of parameters without much loss in the accuracy of the model. There are several ways of achieving the pruned model, the most popular one is to map the low contributing weights to zero and reducing the number of overall non-zero valued weights. This can be achieved by training a large sparse model and pruning it further which makes it comparable to training a small dense model.

Some related work:

- Buntine and Weigend started to propose various maximum-a-posteriori (MAP) schemes for neural networks.

- Hinton and Van Camp, the first variational methods adding Gaussian noise to control the amount of information in weigth.

- Hochreiter and Schmidhuber suggested taking an information theory perspective into account and utilising a minimum description length (MDL) loss.

- Denker and LeCun, and also MacKay investigated the posterior probability distributions of neural networks and treated the search in the model space (the space of architectures, weight decay, regularizers, etc..) as an inference problem and tried to solve it using Laplace approximations.

- Neal investigated the use of hybrid Monte Carlo for training neural networks, although it has been difficult to apply these to the large sizes of neural networks.

- Graves derived a variational inference scheme for neural networks.

- Blundell et al extended this with an update for the variance that is unbiased and simpler to compute.

- Graves derives a similar algorithm in the case of a mixture posterior probability distribution.

- Soudry proposed a more scalable solution based on expectation propagation.

- **NOTE**: **While this method works for networks with binary weights, its extension to continuous weights is unsatisfying as it does not produce estimates of posterior variance.**

-Several authors have claimed that Dropout and Gaussian Dropout can be viewed as approximate variational inference schemes

-Gal and Ghahramani in 2015 provided a theoretical framework for modelling Bayesian uncertainty. Gal and Ghahramani obtained the uncertainty estimates by casting dropout training in conventional deep networks as a Bayesian approximation of a Gaussian Process. They showed that any network trained with dropout is an approximate Bayesian model, and uncertainty estimates can be obtained by computing the variance on multiple predictions with different dropout masks.

3 Bayesian convolutional neural network

The algorithm is a Cnn with probability distributions over its weights in each filter, and apply variational inference for instance, Bayes by Backprop, to compute the intractable true posterior probability distribution.

BUT: In most CNN architectures, a fully Bayesian perspective is not achieved simply by placing probability distributions on the weights in the convolutional layers; it also requires probability distributions over weights in fully-connected layers

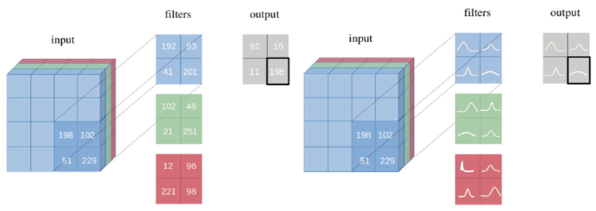


Figure 1: Example

- **The local reparameterization trick for convolutional layers** : we do not sample the weights w, but we sample instead the layer activation b due to its consequent computational acceleration. The variational posterior probability distribution allows to implement the local reparamerization trick in convolutional layers.

-Applying two Sequential Convolutional Operations (Mean and Variance): to make a Cnn with probability distributions over weights instead of single point-estimates and being able to update the variational posterior probability distribution by backpropagation lies in applying two convolutional operations whereas filters with single point-estimates apply one. As explained in the last chapter, we deploy the local reparametrization trick and sample from the output b. Since the output b is a function of mean and variance among others, we are then able to compute the two variables determining a Gaussian probability distribution, namely mean and variance, separately.

-In the two convolutional operations we do: in the first, we treat the output b as an output of a Cnn updated by frequentist inference. We optimize with Adam towards a single point-estimate. We interpret this single point-estimate as the mean of the variational posterior probability distributions. In the second convolutional operation, we learn the variance.In this way, we ensure that only one parameter is updated per convolutional operation.

In other words, while we learn in the first convolutional operation the MAP of the variational posterior probability distribution, we observe in the second convolutional operation how much values for weights w deviate from this MAP. This procedure is repeated in the fully-connected layers. In addition, to accelerate computation, to ensure a positive nonzero variance, and to enhance accuracy, we learn log of the variance and use the Softplus activation function. **note:** We use a Gaussian distribution and we store mean and variance values instead of just one weight.Variance cannot be negative and it is ensured by using softplus as the activation function

-Model prunning: means the reduction in the model weights parameters to reduce the model overall non-zero weights, inference time and computation cost. a Bayesian Convolutional Network learns two weights, i.e: the mean and the variance compared to point estimate learning one single weight, This makes the overall number of parameters of a Bayesian Network twice as compared to the parameters of a point estimate similar architecture.To make the Bayesian Cnns parameters equivalent to point estimate architecture, the number of filters in the Bayesian architectures is reduced to half. This compensates for the doubly learned parameters (mean and variance) against one in point estimates and makes the overall parameters equal for both networks.

4 Practical application *the Bayesian CNN code description*

-The data sets are MNIST and MNIST Corrupted.

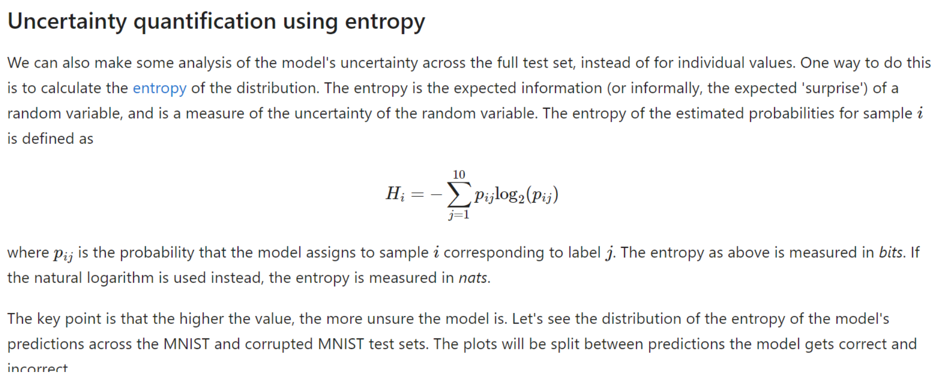
-The deterministic approach we had train a standard deterministic CNN classifier model as a base model before implementing the probabilistic and Bayesian neural networks.

-The probabilistic approach You'll start by turning this deterministic network into a probabilistic one, by letting the model output a distribution instead of a deterministic tensor. This model will capture the aleatoric uncertainty on the image labels. You will do this by adding a probabilistic layer (ONE HOT CATEGORICAL) to the end of the model and training using the negative loglikelihood.

Note that the target data now uses the one-hot version of the labels, instead of the sparse version. This is to match the categorical distribution you added at the end.

-bayesian cnn model The probabilistic model you just created considered only aleatoric uncertainty, assigning probabilities to each image instead of deterministic labels. The model still had deterministic weights. However, as you've seen, there is also 'epistemic' uncertainty over the weights, due to uncertainty about the parameters that explain the training data. Custom prior For the parameters of the DenseVariational layer, we will use a custom prior: the "spike and slab" (also called a scale mixture prior) distribution. This distribution has a density that is the weighted sum of two normally distributed ones: one with a standard deviation of 1 and one with a standard deviation of 10. In this way, it has a sharp spike around 0 (from the normal distribution with standard deviation 1), but is also more spread out towards far away values (from the contribution from the normal distribution with standard deviation 10). The reason for using such a prior is that it is like a standard unit normal, but makes values far away from 0 more likely, allowing the model to explore a larger weight space. Run the code below to create a "spike and slab" distribution and plot its probability density function, compared with a standard unit normal.

-Uncertainty using entropy



Uncertainty quantification using entropy

We can also make some analysis of the model's uncertainty across the full test set, instead of for individual values. One way to do this is to calculate the entropy of the distribution. The entropy is the expected information (or informally, the expected 'surprise') of a random variable, and is a measure of the uncertainty of the random variable. The entropy of the estimated probabilities for sample i is defined as

$$H_i = -\sum_{j=1}^{10} p_{ij} \log_2(p_{ij})$$

where p_{ij} is the probability that the model assigns to sample i corresponding to label j. The entropy as above is measured in *bits*. If the natural logarithm is used instead, the entropy is measured in *nats*.

The key point is that the higher the value, the more unsure the model is. Let's see the distribution of the entropy of the model's predictions across the MNIST and corrupted MNIST test sets. The plots will be split between predictions the model gets correct and incorrect

Figure 2:

There are two main conclusions this was for the prob approach, not the Bayesian:

The model is more unsure on the predictions it got wrong: this means it "knows" when the prediction may be wrong. The model is more unsure for the corrupted MNIST test than for the uncorrupted version. Futhermore, this is more pronounced for correct predictions than for those it labels incorrectly. In this way, the model seems to "know" when it is unsure. This is a great property to have in a machine learning model, and is one of the advantages of probabilistic modelling.

Uncertainty quantification using entropy (this was for the bayesian approach) We also again plot the distribution of distribution entropy across the different test sets below. In these plots, no consideration has been made for the epistemic uncertainty, and the conclusions are broadly similar to those for the previous model.

-optional The model predictions for MNIST We will now do some deeper analysis by looking at the probabilities the model assigns to each class instead of its single prediction. The model is very confident that the first image is a 6, which is correct. For the second image, the model struggles, assigning nonzero probabilities to many different classes.

-optional The model predictions for MNIST Corrupted The first is the same 6 as you saw above, but the second image is different. Notice how the model can still say with high certainty that the first image is a 6, but struggles for the second, assigning an almost uniform distribution to all possible labels.

References

[1] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," 2016.

[2] Z. Ul Abideen, M. Ghafoor, K. Munir, M. Saqib, A. Ullah, T. Zia, S. A. Tariq, G. Ahmed, and A. Zahra, "Uncertainty assisted robust tuberculosis identification with bayesian convolutional neural networks," *IEEE Access*, vol. 8, pp. 22812–22825, 2020.

[3] K. Shridhar, F. Laumann, and M. Liwicki, "A comprehensive guide to bayesian convolutional neural network with variational inference," 2019.