Model form of Bayes' Theorem

Bayesian neural networks replace the deterministic network's weight parameters with distributions over these parameters, and instead of optimising the network weights directly we average over all possible weights (referred to as marginalisation).
When the model form of Bayes' theorem is stated in the literature different symbols are often used. Let's introduce them.

Instead of event A, we'll typically see Θ, this symbol is called Theta. Theta is what we're interested in, it represents the set of parameters/weights.

Instead of event B, we'll see *data* or ***y*** = *{y1, y2, …, yn}*. These represent the data, i.e. the set of observations that we have. I'll explicitly use *data* in the equation to hopefully make the equation a little less cryptic.

So now Bayes' theorem in model form is written as:

$$P(\mathbf{\Theta}|data) = \frac{P(data|\mathbf{\Theta}) \times P(\mathbf{\Theta})}{P(data)}.$$

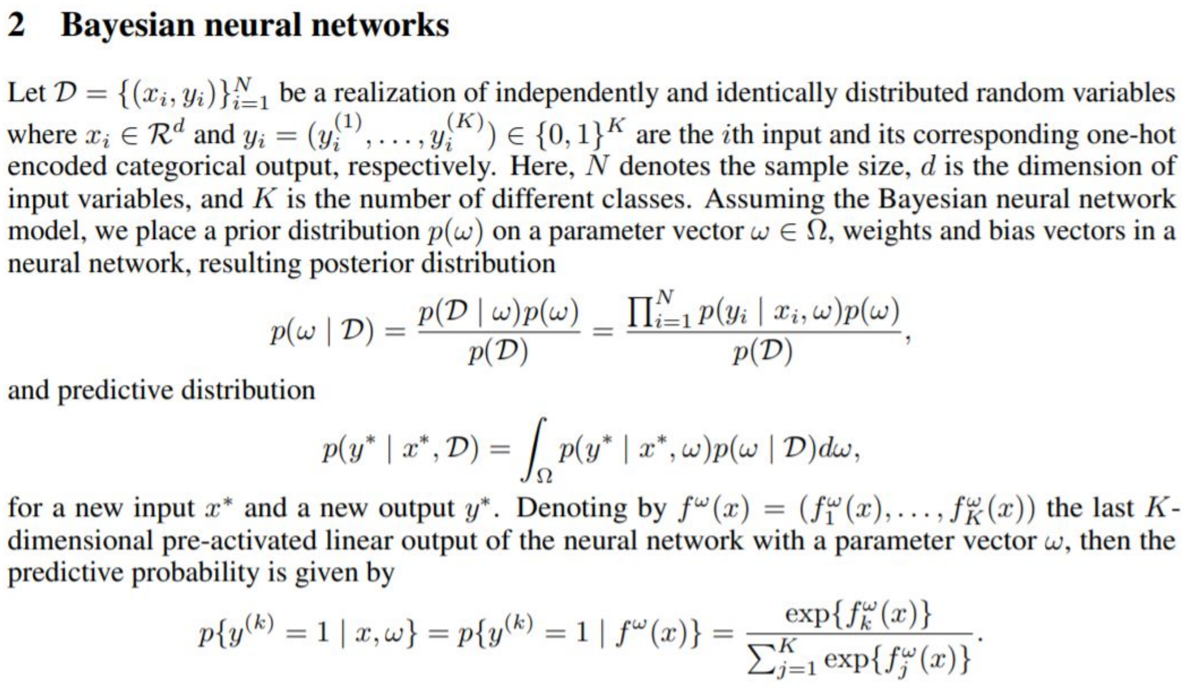
1) We‘ve seen that P(Θ) is the **prior distribution**. It represents our beliefs about the true value of the parameters, just like we had distributions representing our belief about the probability of selling ice cream.
2) *P(*Θ|*data)* on the left hand side is known as the **posterior distribution.** This is the distribution representing our belief about the parameter values after we have calculated everything on the right hand side taking the observed data into account.
3) *L(data; μ, σ)* is the likelihood distribution (for a Gaussian distribution). Well *P(data|* Θ*)* is exactly this, it's the **likelihood distribution** in disguise. Sometimes it's written as *ℒ(*Θ; *data)* but it's the same thing here**.**
4) P(data) is called **evidence**. It is the **normalising constant.**

In some cases we don't care about this property of the distribution. We only care about where the peak of the distribution occurs, regardless of whether the distribution is normalised or not.This makes it explicit that the true posterior distribution is not equal to the right hand side because we haven't accounted for the normalisation constant *P(data)*. In this case many people write the model form of Bayes' theorem as

$$P(\mathbf{\Theta}|data) \propto P(data|\mathbf{\Theta}) \times P(\mathbf{\Theta})$$

Remember, we're only interested in the parameter values but P(data) doesn't have any reference to them. In fact, P(data) doesn't even evaluate to a distribution. It's just a number, we've already observed the data so we can calculate P(data). In general, it turns out that calculating **P(data) is very hard** and so many methods exist to calculate it. The reason why P(data) is important is because the number that comes out is a normalising constant.

Therefore we can calculate the *posterior distribution* of our parameters using our *prior beliefs* updated with our *likelihood*.



2 Bayesian neural networks

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a realization of independently and identically distributed random variables where $x_i \in \mathcal{R}^d$ and $y_i = (y_i^{(1)}, \ldots, y_i^{(K)}) \in \{0,1\}^K$ are the ith input and its corresponding one-hot encoded categorical output, respectively. Here, N denotes the sample size, d is the dimension of input variables, and K is the number of different classes. Assuming the Bayesian neural network model, we place a prior distribution $p(\omega)$ on a parameter vector $\omega \in \Omega$, weights and bias vectors in a neural network, resulting posterior distribution

$$p(\omega \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \omega)p(\omega)}{p(\mathcal{D})} = \frac{\prod_{i=1}^N p(y_i \mid x_i, \omega)p(\omega)}{p(\mathcal{D})},$$

and predictive distribution

$$p(y^* \mid x^*, \mathcal{D}) = \int_\Omega p(y^* \mid x^*, \omega)p(\omega \mid \mathcal{D})d\omega,$$

for a new input x^* and a new output y^*. Denoting by $f^\omega(x) = (f_1^\omega(x), \ldots, f_K^\omega(x))$ the last K-dimensional pre-activated linear output of the neural network with a parameter vector ω, then the predictive probability is given by

$$p\{y^{(k)} = 1 \mid x, \omega\} = p\{y^{(k)} = 1 \mid f^\omega(x)\} = \frac{\exp\{f_k^\omega(x)\}}{\sum_{j=1}^K \exp\{f_j^\omega(x)\}}.$$

Conjugate prior:

If the posterior distributions *p*(*θ* | *x*) are in the same probability distribution family as the prior probability distribution *p*(θ), the prior and posterior are then called conjugate distributions, and the prior is called a conjugate prior for the likelihood function. For example, the Gaussian family is conjugate to itself (or *self-conjugate*) with respect to a Gaussian likelihood function: if the likelihood function is Gaussian, choosing a Gaussian prior over the mean will ensure that the posterior distribution is also Gaussian.

Bayesian(probabilistic) Inference:-

Variational Bayesian method-

We'll typically see Θ, this symbol is called Theta. Theta is what we're interested in, it represents the set of parameters. So if we're trying to estimate the parameter values of a Variational distribution then Θ represents both the mean, μ and the standard deviation, σ (written mathematically as $\Theta = \{\mu, \sigma\}$, ($P(w|Data) \sim Q(w)$)

In variational inference, the posterior distribution over a set of unobserved variables Z={Z1,Z2,,,Zn} is approximated by a so-called **variational distribution**,Q(w).

Variational Bayesian methods are primarily used for two purposes:

1) To provide an analytical approximation to the posterior probability of the unobserved variables, in order to do statistical inference over these variables.
2) To derive a lower bound for the marginal likelihood (sometimes called the "evidence") of the observed data (i.e. the marginal probability of the data given the model, with marginalization performed over unobserved variables). This is

typically used for performing model selection, the general idea being that a higher marginal likelihood for a given model indicates a better fit of the data by that model and hence a greater probability that the model in question was the one that generated the data.

For further reading- https://en.wikipedia.org/wiki/Variational_Bayesian_methods

Markov chain Monte Carlo (MCMC) algorithm;-

Markov chain Monte Carlo (MCMC) algorithms make educated guesses about the unknown weights, computing the likelihood of the set of weights. By repeating this process many times, MCMC builds a distribution of likely parameters. Constructing this distribution is the goal of probabilistic inference.

For further reading;- https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo

Dropout as Bayesian approximation:-

The “dropout as a Bayesian Approximation” proposes a simple approach to quantify the neural network uncertainty. It employs dropout during ***both training and testing***.

The same input is passed Many times and the network regresses

Thus, the first and second moment (mean and variance) provides the network's output and uncertainty respectively.

So by simply adding Dropout, we now learn a Bayesian model where weights are sampled from a (kind of) Bernoulli distribution on the rows (or columns, depending on how the matrix multiplication is done). One thing to notice is that training will be exactly the same as before, the only difference is during inference, where we do not turn off the stochasticity, but instead, sample several weights, pass it for the forward pass, and obtain now a distribution of the output. The fact that we use several samples of weights during inference is called Monte Carlo Dropout (or MC Dropout).

For further info-
https://medium.com/@ahmdtaha/dropout-as-a-bayesian-approximation-representing-model-uncertainty-in-deep-learning-7a2e49e64a15