CSC421/2516 Lecture 19: Bayesian Neural Nets

Roger Grosse and Jimmy Ba
Overview

- Some of our networks have used probability distributions:
  - Cross-entropy loss is based on a probability distribution over categories.
  - Generative models learn a distribution over $x$.
  - Stochastic computations (e.g. dropout).
- But we’ve always fit a point estimate of the network weights.
- Today, we see how to learn a distribution over the weights in order to capture our uncertainty.
- **This lecture will not be on the final exam.**
  - Depends on CSC411/2515 lectures on Bayesian inference, which some but not all of you have seen.
  - We can’t cover BNNs properly in 1 hour, so this lecture is just a starting point.
Overview

Why model uncertainty?

- Smooth out the predictions by averaging over lots of plausible explanations (just like ensembles!)
- Assign confidences to predictions (i.e. calibration)
- Make more robust decisions (e.g. medical diagnosis)
- Guide exploration (focus on areas you’re uncertain about)
- Detect out-of-distribution examples, or even adversarial examples
Two types of uncertainty

- **Aleatoric uncertainty**: inherent uncertainty in the environment’s dynamics
  - E.g., output distribution for a classifier or a language model (from the softmax)
  - Alea = Latin for “dice”
- **Epistemic uncertainty**: uncertainty about the model parameters
  - We haven’t yet considered this type of uncertainty in this class.
  - This is where Bayesian methods come in.
Recap: Full Bayesian Inference

- Recall: full Bayesian inference makes predictions by averaging over all likely explanations under the posterior distribution.
- Compute posterior using Bayes’ Rule:
  \[ p(w | D) \propto p(w)p(D | w) \]
- Make predictions using the posterior predictive distribution:
  \[ p(t | x, D) = \int p(w | D) p(t | x, w) \, dw \]
- Doing this lets us quantify our uncertainty.
Bayesian Linear Regression

- **Bayesian linear regression** considers various plausible explanations for how the data were generated.
- It makes predictions using all possible regression weights, weighted by their posterior probability.

Prior distribution: \( w \sim \mathcal{N}(0, S) \)

Likelihood: \( t \mid x, w \sim \mathcal{N}(w^\top \psi(x), \sigma^2) \)

Assuming fixed/known \( S \) and \( \sigma^2 \) is a big assumption. There are ways to estimate them, but we'll ignore that for now.
Bayesian Linear Regression
Bayesian Linear Regression

- Example with radial basis function (RBF) features

\[
\phi_j(x) = \exp \left( -\frac{(x - \mu_j)^2}{2s^2} \right)
\]

— Bishop, Pattern Recognition and Machine Learning
Bayesian Linear Regression

Functions sampled from the posterior:

— Bishop, Pattern Recognition and Machine Learning
Bayesian Linear Regression

Here we visualize confidence intervals based on the posterior predictive mean and variance at each point:

---

Bishop, Pattern Recognition and Machine Learning

Roger Grosse and Jimmy Ba

CSC421/2516 Lecture 19: Bayesian Neural Nets
As we know, fixed basis functions are limited. Can we combine the advantages of neural nets and Bayesian models?

**Bayesian neural networks (BNNs)**

- Place a prior on the weights of the network, e.g. \( p(\theta) = \mathcal{N}(\theta; 0, \eta I) \)
  - In practice, typically separate variance for each layer
- Define an observation model, e.g. \( p(t \mid x, \theta) = \mathcal{N}(t; f_{\theta}(x), \sigma^2) \)
- Apply Bayes’ Rule:

\[
p(\theta \mid D) \propto p(\theta) \prod_{i=1}^{N} p(t^{(i)} \mid x^{(i)}, \theta)
\]
Samples from the Prior

- We can understand a Bayesian model by looking at prior samples of the functions.
- Here are prior samples of the function for BNNs with one hidden layer and 10,000 hidden units.

In the 90s, Radford Neal showed that under certain assumptions, an infinitely wide BNN approximates a Gaussian process.
- Just in the last few years, similar results have been shown for deep BNNs.
One way to use posterior uncertainty is to sample a set of values $\theta_1, \ldots, \theta_K$ from the posterior $p(\theta | D)$ and then average their predictive distributions:

$$p(t | x, D) \approx \sum_{k=1}^{K} p(t | x, \theta_k).$$

We can’t sample exactly from the posterior, but we can do so approximately using Markov chain Monte Carlo (MCMC), a class of techniques covered in CSC412/2506.

In particular, an MCMC algorithm called Hamiltonian Monte Carlo (HMC). This is still the “gold standard” for doing accurate posterior inference in BNNs.

Unfortunately, HMC doesn’t scale to large datasets, because it is inherently a batch algorithm, i.e. requires visiting the entire training set for every update.
A less accurate, but more scalable, approach is variational inference, just like we used for VAEs. Variational inference for Bayesian models is called variational Bayes. We approximate a complicated posterior distribution with a simpler variational approximation. E.g., assume a Gaussian posterior with diagonal covariance (i.e. fully factorized Gaussian):

$$q(\theta) = \mathcal{N}(\theta; \mu, \Sigma) = \prod_{j=1}^{D} \mathcal{N}(\theta_j; \mu_j, \sigma_j)$$

This means each weight of the network has its own mean and variance.
Posterior Inference: Variational Bayes

- The marginal likelihood is the probability of the observed data (targets given inputs), with all possible weights marginalized out:

\[
p(D) = \int p(\theta) p(D | \theta) d\theta
= \int p(\theta) p(\{t^{(i)}\} | \{x^{(i)}\}, \theta) d\theta.
\]

- Analogously to VAEs, we define a variational lower bound:

\[
\log p(D) \geq F(q) = E_{q(\theta)}[\log p(D | \theta)] - D_{KL}(q(\theta) \parallel p(\theta))
\]

- Unlike with VAEs, \( p(D) \) is fixed, and we are only maximizing \( F(q) \) with respect to the variational posterior \( q \) (i.e. a mean and standard deviation for each weight).
\[ \log p(D) \geq \mathcal{F}(q) = \mathbb{E}_{q(\theta)}[\log p(D | \theta)] - D_{KL}(q(\theta) \| p(\theta)) \]

- Same as for VAEs, the gap equals the KL divergence from the true posterior:
  \[ \mathcal{F}(q) = \log p(D) - D_{KL}(q(\theta) \| p(\theta | D)). \]
  Hence, maximizing \( \mathcal{F}(q) \) is equivalent to approximating the posterior.
Posterior Inference: Variational Bayes

- **Likelihood term:**
  \[
  \mathbb{E}_{q(\theta)}[\log p(D | \theta)] = \mathbb{E}_{q(\theta)} \left[ \sum_{i=1}^{N} \log p(t^{(i)} | x^{(i)}, \theta) \right]
  \]
  This is just the usual likelihood term (e.g. minus classification cross-entropy), except that \( \theta \) is sampled from \( q \).

- **KL term:**
  \[D_{KL}(q(\theta) \parallel p(\theta))\]
  This term encourages \( q \) to match the prior, i.e. each dimension to be close to \( \mathcal{N}(0, \eta^{1/2}) \).

- Without the KL term, the optimal \( q \) would be a point mass on \( \theta_{ML} \), the maximum likelihood weights.
  - Hence, the KL term encourages \( q \) to be more spread out (i.e. more stochasticity in the weights).
Posterior Inference: Variational Bayes

- We can train a variational BNN using the same reparameterization trick as from VAEs.
  \[ \theta_j = \mu_j + \sigma_j \epsilon_j, \]
  where \( \epsilon_j \sim \mathcal{N}(0, 1) \).
- Then the \( \epsilon_j \) are sampled at the beginning, independent of the \( \mu_j, \sigma_j \), so we have a deterministic computation graph we can do backprop on.
- If all the \( \sigma_j \) are 0, then \( \theta_j = \mu_j \), and this reduces to ordinary backprop for a deterministic neural net.
- Hence, variational inference injects stochasticity into the computations. This acts like a regularizer, just like with dropout.
  - The difference is that it’s stochastic activations, rather than stochastic weights.
  - See Kingma et al., “Variational dropout and the local reparameterization trick”, for the precise connections between variational BNNs and dropout.
Posterior Inference: Variational Bayes

- Bad news: variational BNNs aren’t a good match to Bayesian posterior uncertainty.
- The BNN posterior distribution is complicated and high dimensional, and it’s really hard to approximate it accurately with fully factorized Gaussians.

So what are variational BNNs good for?

— Hernandez-Lobato et al., Probabilistic Backpropagation

---

Roger Grosse and Jimmy Ba  
CSC421/2516 Lecture 19: Bayesian Neural Nets  
19 / 22
What variational BNNs are really doing is regularizing the description length of the weights.

Intuition: the more concentrated the posterior is, the more bits it requires to describe the location of the weights to adequate precision.

A more concentrated $q$ generally implies a higher KL from the prior.
Description Length Regularization

- The KL term $D_{KL}(q(\theta) \parallel p(\theta))$ can be interpreted as the number of bits required to describe $\theta$ to adequate precision.
  - This can be made precise using the bits-back argument. This is beyond the scope of the class, but see here for a great explanation: https://youtu.be/0IoLKnAg6-s

- A classic result from computational learning theory ("Occam’s Razor") bounded the generalization error a learning algorithm that selected from $K$ possible hypotheses.
  - It requires $\log K$ bits to specify the hypothesis.
  - PAC-Bayes gives analogous bounds for the generalization error of variational BNNs, where $D_{KL}(q(\theta) \parallel p(\theta))$ behaves analogously to $\log K$.
    - This is one of the few ways we have to prove that neural nets generalize.
    - See Dziugaite et al., “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data”.

Uses of BNNs

- Guiding exploration
  - Curriculum learning: Graves et al., 2017. Automated curriculum learning for neural networks
  - Intrinsic motivation in reinforcement learning: Houthooft et al., 2016. Variational information maximizing exploration

- Network compression: Louizos et al., 2017. Bayesian compression for deep learning

- Lots more references in CSC2541, “Scalable and Flexible Models of Uncertainty”
  - https://csc2541-f17.github.io/