[Paper review 2]

Bayesian Learning For Neural Networks (Radford M. Neal, 1995)

[Contents]

1. Introduction

1-1. Bayesian and frequentists views of learning

- Estimate "Uncertainty"
- Not interested in the value of θ it self,

but rather the "Value of some quantity that may be observed in the future" (= $x^{(n+1)}$)

Likelihood, Posterior, Posterior Predictive distribution

- likelihood : $L(\theta)=L\left(\theta \mid x^{(1)}, \ldots, x^{(n)}\right) \propto P\left(x^{(1)}, \ldots, x^{(n)} \mid \theta\right)=\prod_{i=1}^{n} P\left(x^{(i)} \mid \theta\right)$
- posterior : $P\left(\theta \quad x^{(1)}, \ldots, x^{(n)}\right)=\frac{P\left(x^{(1)}, \ldots, x^{(n)} \mid \theta\right) P(\theta)}{P\left(x^{(1)}, \ldots, x^{(n)}\right)} \quad \propto \quad L\left(\theta \quad x^{(1)}, \ldots, x^{(n)}\right) P(\theta)$
- posterior predictive : $P\left(x^{(n+1)} \mid x^{(1)}, \ldots, x^{(n)}\right)=\int P\left(x^{(n+1)} \mid \theta\right) P\left(\theta \mid x^{(1)}, \ldots, x^{(n)}\right) d \theta$

Bayesian View

- prediction "by INTEGRATION" , "rather than MAXIMIZATION"
- avoids jumping to conclusion by considering not just the value of θ (that explains the data best), but also

  other values of θ that explain the data reasonably well
- maximum a posteriori probability (MAP)

  MAP estimation is better characterized as a form of maximum PENALIZED likelihood estimation, with penalty being the "PRIOR" density

Hierarchical Models

- θ_k are independent given γ

  $P(\theta)=P\left(\theta_{1}, \ldots, \theta_{p}\right)=\int P(\gamma) \prod_{k=1}^{p} P\left(\theta_{k} \mid \gamma\right) d \gamma$
- give the top-level hyperparameters prior distribution "that are very vague, or even improper"

  (data is sufficiently informative!)

Learning Complex models

- From Bayesian view, adjusting the complexity of the model based on the amount of data "MAKES NO SENSE"
- if model and prior are correct for 1000 observation $\rightarrow$ also correct for 10 observation
- Occam's Razor will be applied automatically!

1-2. BNN (Bayesian Neural Networks)

1-2-1. MLP (Multi-layer Perceptron Networks)

neural networks are "NON-parametric"

- do have a parameter
- just more numerous, less interpretable than "parametric" models

MLP (Multi-layer Perceptron Networks)

$$f_k(x) = b_k + \sum_j v_{jk} h_j(x)$$

$$h_j(x) = \tanh\left(a_j + \sum_i u_{ij} x_i\right)$$

- u_{ij} : weight from "input unit i" to "hidden unit j"
- v_{jk} : weight from "hidden unit j" to "output unit k"

(MLP can be used to define probabilistic models for Regression & Classification)

Regression

- Gaussian ($y_k \sim N(f_k(x), \sigma_k^2)$)
- $P(y \mid x) = \prod_k \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-(f_k(x) - y_k)^2 / 2\sigma_k^2\right)$

Classification

- Softmax
- $P(y = k \mid x) = \exp(f_k(x)) / \sum_{k'} \exp(f_{k'}(x))$

Minimization of error function = MLE for the Gaussian noise model

If we use Cross Validation to find the appropriate weight penalty...

- problem 1) computationally expensive
- problem 2) if n networks find different local minima, for which different weight penalties are appropriate

Bayesian Goal : "to find the predictive distribution" in new case

- Posterior predictive distribution :
  $P\left(y^{(n+1)} \mid x^{(n+1)}, \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right)$
  $= \int P\left(y^{(n+1)} \mid x^{(n+1)}, \theta\right) P\left(\theta \mid \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right) d\theta$
- Likelihood :
  $L\left(\theta \mid \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right) = \prod_{i=1}^n P\left(y^{(i)} \mid x^{(i)}, \theta\right)$
- single-point prediction :
  $\hat{y}_k^{(n+1)} = \int f_k\left(x^{(n+1)}, \theta\right) P\left(\theta \mid \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right) d\theta$
  (f_k : network output functions)

1-2-2. Selecting a network "model" & "prior"

give weights & biases a "Gaussian prior distribution"

advantages of hierarchical models

- hyperparameter : "variance" of the Gaussian prior for the weight & biases
  (allows the model to adapt to whatever degree of smoothness)

- role of hyperparameter (controlling the priors for weights) = role of "weight decay"
- without the need for validation set!

1-2-3. The ARD(Automatic Relevance Determination) Model

Problem

- Another dimension of complexity in NN = "number of input variables"

  (including more inputs $\rightarrow$ lead to poor generalization)
- Must limit the number of input variables, "based on our assessment of which attributes are most likely to be relevant"

  (So, which one is relevant ?)
- ARD : a model that can automatically determine the degree!

ARD

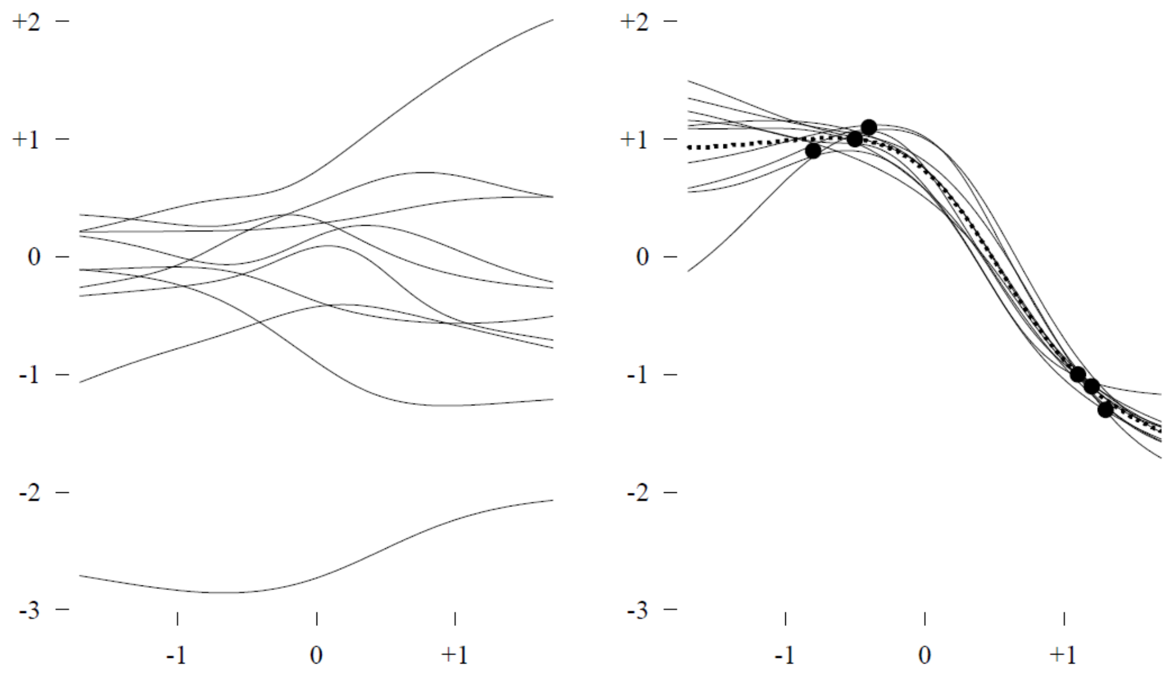
- each input variable has associated with it a "hyperparameter" that controls the magnitudes of the weights

  (these hyperparameters are given some prior distribution)
- If hyperparameter associated with an input specifies small standard deviation for weights out of that input $\rightarrow$ weight will be small
- intended for use with a complex model (with each input associated with many weights)

  (role of hyperparamter : introduce dependencies between these weights)
- will talk about more in Chapter 4

1-2-4. Illustration of Bayesian learning for Neural Networks

example) 1 input unit, 16 hidden units, 1 output units



(L) 10 networks, weight & bias from "Gaussian prior"

- easy to sample

(R) 10 networks, weight & bias from "Posterior distribution derived from prior & likelihood due to 6 data points"

- step 1) generate many networks from "prior"
- step 2) compute likelihood (for 6 points) of each 10 network
- accept each network (from step 1) with a probability proportional to its likelihood (step 2)

Best way to guess the targets?

- use the AVERAGE of the network functions over the posterior distribution

  (how to average? $\rightarrow$ Monte Carlo Approximation)
- but Bayesian gives more than just a single-valued guess! also UNCERTAINTY

  (Uncertainty increases rapidly beyond the region where the training points are located)

1-2-5. Implementations based on Gaussian Approximation

"Finding the Predictive Distribution" = "Evaluating the Integral of Equation"

How to find the integral of equation?

- Ch 1) by Gaussian Approximation
- Ch 3) MCMC

Gaussian Approximation

- step 1) find (one or more) modes
- step 2) approximate posterior distribution

  (covariance matrix is chosen to match the second derivatives of the log posterior probability at the mode)
- step 3) If more than one mode $\rightarrow$ decide how much weight to give to each
- step 4) Approximate predictive distribution of equation

  (model that are linear in the vicinity of a mode, it is easy. If not, hard!)

Gaussian Approximation will fail to be good for LARGE networks

That's when we should use MCMC.

1-3. MCMC (Markov Chain Monte Carlo)

Make NO ASSUMPTION concerning the form of the distribution

deal with "Hybrid Monte Carlo" (= Hamiltonian Monte Carlo)

1-3-1. Monte Carlo integration using Markov Chains

posterior : $Q(\theta)$

then, expectation of $a(\theta)$: $E[a]=\int a(\theta) Q(\theta) d\theta$

(let $a(\theta)=f_k\left(x^{(n+1)}, \theta\right)$ to find the best guess for $y_k^{(n+1)}$)

How to find $E[a]=\int a(\theta) Q(\theta) d\theta$?

- use Monte Carlo Approximation!

Monte Carlo Approximation

- $E[a] \approx \frac{1}{N} \sum_{t=1}^N a\left(\theta^{(t)}\right)$
- $\theta^{(1)} \ldots \theta^{(N)}$ are generated by distribution Q (they are all independent)

  (even if Q is complicated .... would be possible to generate independent samples)
- gives UNBIASED estimate of $E[a]$ (even the samples are dependent!)
- still converge to true value (as long as the dependence is not great)

Invariance(=Stationary)

- $Q\left(\theta^{\prime}\right)=\int T\left(\theta^{\prime} \mid \theta\right) Q(\theta) d\theta$
- stronger condition of "detailed balance"

Detailed balance

- $T\left(\theta^{\prime} \mid \theta\right) Q(\theta)=T\left(\theta \mid \theta^{\prime}\right) Q\left(\theta^{\prime}\right)$
- "detailed balance" $\rightarrow$ REVERSIBLE

"Ergodic"

- meaning that is has unique invariant(stationary) distribution (= "Equilibrium distribution")
- this distribution converges from ANY INITIAL state
- our goal : "construct a Markov chain which is ERGODIC"

To construct chain for complex problem :

- combine the transitions for simpler chains!
- sequnce of it will also be invariant(stationary)

Check effect of dependencies of MC estimate? by auto-correlation !

Autocorrelation of a at lag s : $\rho(s)=\frac{E\left[\left(a\left(\theta^{(t)}\right)-E[a]\right)\left(a\left(\theta^{(t-s)}\right)-E[a]\right)\right]}{\operatorname{Var}[a]}$

1-3-2. Gibbs Sampling

- skip

1-3-3. Metropolis Algorithm

- skip

1-4. Outline of the remainder of the thesis

Chapter 2)

- properties of prior distribution
- limit as the "number of hidden units $\rightarrow \infty$ "
- goal) reasonable priors (for such infinite networks) can be defined

Chapter 3)

- computational problems of BNN
- estimate using "hybrid MC"

Chapter 4)

- evaluate how good the predictions of BNNs are!
- evaluate the effectiveness of Hierarchical Models (in particular, the ARD model)