Laplace's Approximation, Model Selection/Averaging

CS772A: Probabilistic Machine Learning

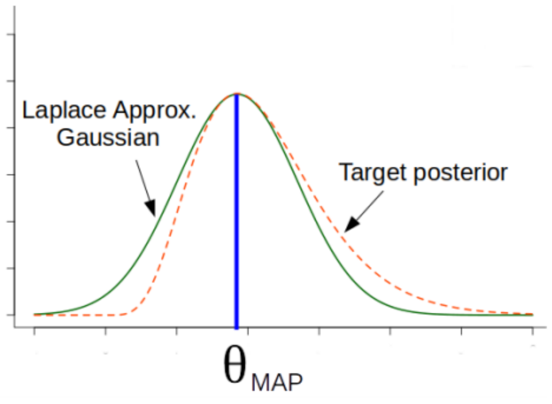
Piyush Rai

Laplace's Approximation

- Consider a posterior distribution that is intractable to compute

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D},\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

- Laplace approximation approximates the above using a Gaussian distribution



$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta|\theta_{MAP}, \mathbf{\Lambda}^{-1})$$

Tells us about the space (curvature) of the true posterior around θ_{MAP}

Related to the Fisher Information Matrix (FIM); will see shortly

$$\theta_{MAP} = \operatorname{argmax}_\theta \log p(\theta|\mathcal{D})$$

Negative of the Hessian, i.e., the second derivative of the log joint, at θ_{MAP}

$$\mathbf{\Lambda} = -\nabla_\theta^2 \log p(\theta|\mathcal{D})\Big|_{\theta=\theta_{MAP}} = -\nabla_\theta^2 \log p(\mathcal{D},\theta)\Big|_{\theta=\theta_{MAP}}$$

- Laplace's approx. is based on a second-order Taylor approx. of the posterior

Derivation of the Laplace's Approximation

$$\boxed{p(\mathcal{D}) \approx \exp\left(\log p(\mathcal{D}, \theta_{MAP})\right) \times (2\pi)^{D/2} \det(\mathbf{\Lambda})^{1/2}}$$

We also get a Laplace approximation of the marginal likelihood (for free!)

Note: Sometimes marginal likelihood is also called model evidence

- Let's write the Bayes rule as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D},\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D},\theta)}{\int p(\mathcal{D},\theta)d\theta} = \frac{\exp[\log p(\mathcal{D},\theta)]}{\int \exp[\log p(\mathcal{D},\theta)]d\theta}$$

- Consider second-order Taylor approximation of a function $f(\theta)$ around some θ_0

$$f(\theta) \approx f(\theta_0) + (\theta-\theta_0)^\top \nabla_\theta f(\theta_0) + \frac{1}{2}(\theta-\theta_0)^\top \nabla_\theta^2 f(\theta_0)(\theta-\theta_0)$$

- Assuming $f(\theta) = \log p(\mathcal{D},\theta)$ and $\theta_0 = \theta_{MAP}$

Constant w.r.t. θ

Same as $\nabla^2 \log p(\theta_{MAP}|\mathcal{D})$

$$\log p(\mathcal{D},\theta) \approx \log p(\mathcal{D},\theta_{MAP}) + \frac{1}{2}(\theta-\theta_{MAP})^\top \nabla_\theta^2 \log p(\mathcal{D},\theta_{MAP})(\theta-\theta_{MAP})$$

$$p(\theta|\mathcal{D}) \propto \exp\left[-\frac{1}{2}(\theta-\theta_{MAP})^\top(-\nabla_\theta^2 \log p(\mathcal{D},\theta_{MAP}))(\theta-\theta_{MAP})\right]$$

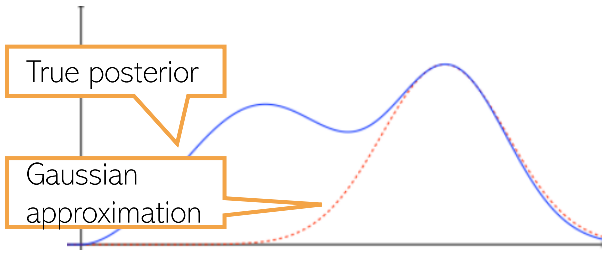
$$= \mathcal{N}(\theta|\theta_{MAP}, \mathbf{\Lambda}^{-1}) \quad (\text{where } \mathbf{\Lambda} = -\nabla_\theta^2 \log p(\mathcal{D},\theta_{MAP}) = -\mathbf{H})$$

Properties of Laplace's Approximation

- Straightforward if posterior's derivatives (first/second) can be computed easily
- Expensive if parameter $\boldsymbol{\theta}$ is very high dimensional

E.g., a deep neural network, or even in simpler models (e.g., logistic reg with a very large number of features

  - Reason: We need to compute and invert Hessian of size $D \times D$ (D is the # of params)
- Can do badly if the (true) posterior is multimodal



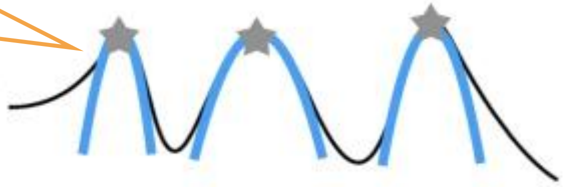
For multimodal posteriors, can use a mixture of Laplace approximations*

Useful for deep learning models

If K local modes, then define the approx. posterior as a mixture of K Gaussians

$$p(\theta|D) \approx \sum_{k=1}^{K} \pi^{(k)} \mathcal{N}(\theta|\theta_{MAP}^{(k)}, H^{(k)^{-1}})$$

(see paper cited below for details)



- Used only when $\boldsymbol{\theta}$ is a real-valued vector (because of Gaussian approximation)
- Note: Even if we have a <u>non-probabilistic</u> model (loss function + regularization), we can obtain an approx "posterior" for that model using the Laplace's approximation
  - Optima of the regularized loss function will be Gaussian's mean
  - Inverse of the second derivative of the regularized loss function will be covariance matrix

*Mixtures of Laplace Approximations for Improved Post-Hoc Uncertainty in Deep Learning (Eschenhagen et al, 2021)

Detour: Hessian and Fisher Information Matrix

- Hessian is related to the Fisher Information Matrix (FIM)
- Gradient of the log likelihood is also called score function: $s(\theta) = \nabla_\theta \log p(y|\theta)$
  - Note: At some places (some generative models) $\nabla_y \log p(y|\theta)$ also called score function
- Expectation of score function is zero: $\mathbb{E}_{p(y|\theta)}[s(\theta)] = 0$ (exercise)
- Fisher Information Matrix (FIM) is covariance matrix of score function

$$\mathbf{F} = \mathbb{E}_{p(y|\theta)}[(s(\theta)-0)(s(\theta)-0)^\top] = \mathbb{E}_{p(y|\theta)}[\nabla_\theta \log p(y|\theta) \nabla_\theta \log p(y|\theta)^\top]$$

Note: If we have a prior $p(\theta)$ too, then also add the second derivative of $\log p(\theta)$

- $\mathbf{F} = -\mathbb{E}_{p(y|\theta)}\left[\nabla_\theta^2 \log p(y|\theta)\right]$, i.e., negative of expected Hessian (exercise)
- Each entry F_{ij} tells us how "sensitive" the model is w.r.t. the pair (θ_i, θ_j)
  - Each diagonal entry $F_{ii} = (\nabla_{\theta_i} \log p(y|\theta))^2$ tells "important" θ_i is by itself
- Can compute empirical FIM using data: $\hat{\mathbf{F}} = \frac{1}{N}\sum_{n=1}^{N}[\nabla_\theta \log p(y_n|\theta) \nabla_\theta \log p(y_n|\theta)^\top]$

Laplace Approx. for High-Dimensional Problems

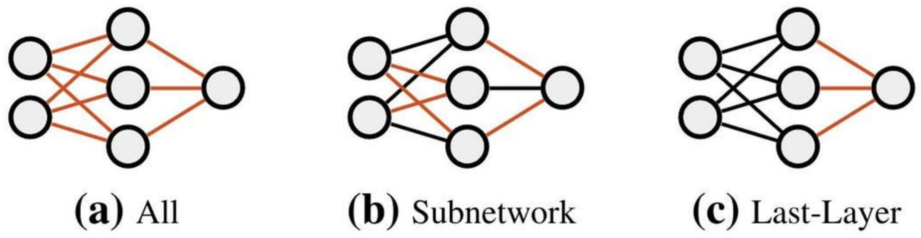
- For high-dim $\boldsymbol{\theta}$, Laplace's approx $p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta|\theta_{MAP}, \boldsymbol{\Lambda}^{-1})$ can be expensive
- Many methods to address this, e.g.,
  - Use a diagonal of (empirical) Fisher as the precision

$$\boldsymbol{\Lambda} \approx \mathbf{diag}(\mathbf{F})$$

  - Use a block-diagonal approximation* of $\boldsymbol{\Lambda}$ (better than diagonal approx)

Diagonal approximation assumes that the weights are all independent whereas block-diagonal assumes that the weights within each block may have correlations

- For deep nets, use LA only for some weights + point estimates for others
  - Option 1: Use LA only for last layer weights - "last layer Laplace's approximation" (LLLA)
  - Option 2: Use LA for weights from an identified "subnetwork"



  - See the "Laplace Redux" paper for more options and discussion on scalability of LA

*KFAC paper: "A Scalable Laplace Approximation for Neural Networks" (Ritter et al, ICLR 2018) *Laplace Redux -- Effortless Bayesian Deep Learning " (Daxberger et al, NeurIPS 2021)

PPD when using Laplace's Approximation

- The PPD when using the Laplace's approximation of the posterior

$$p(y_*|\boldsymbol{x}_*,\mathcal{D}) = \int p(y_*|\boldsymbol{x}_*,\theta)p(\theta|\mathcal{D})d\theta$$

This PPD is an approximation because we are using an approximation of the posterior

$$\approx \int p(y_*|\boldsymbol{x}_*,\theta)\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}^{-1})d\theta$$

- PPD may be intractable depending on the form of $p(y_*|\boldsymbol{x}_*,\theta) = p(y_*|f(\boldsymbol{x}_*,\theta))$
- We can use further approximations if the integral is intractable. Two options:
  - Generate M samples $\{\theta^{(i)}\}_{i=1}^{M}$ from $\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}^{-1})$ and compute a Monte Carlo approx.

$$\int p(y_*|\boldsymbol{x}_*,\theta)\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}^{-1})d\theta \approx \frac{1}{M}\sum_{i=1}^{M} p\left(y_*|\boldsymbol{x}_*,\theta^{(i)}\right)$$

Using MC approximation is the general purpose option when computing intractable PPD

Generalized Gauss-Newton method

  - Use the GGN approximation of LA. Equivalent to using a "linearlized" model for $p(y_*|\boldsymbol{x}_*,\theta)$, using which we can easily compute PPD using linear Gaussian model results

Detour: Gradient and Hessian

- For LA (and for optimization general), we need $\nabla_\theta \log p(\mathcal{D},\theta)$ and $\nabla^2_\theta \log p(\mathcal{D},\theta)$
- These depend on the likelihood function, $p(\boldsymbol{y}|\boldsymbol{x},\theta) = p(\boldsymbol{y}|f(\boldsymbol{x},\theta))$
- The form of the function f depends on the likelihood model. Some examples:

$$p(\boldsymbol{y}|\boldsymbol{x},\theta) = \mathcal{N}(\boldsymbol{y}|\theta^\top \boldsymbol{x},\sigma^2) \qquad p(\boldsymbol{y}|\boldsymbol{x},\theta) = \text{multinoulli}(\boldsymbol{y}|\text{softmax}(\theta^\top \boldsymbol{x}))$$

$$p(\boldsymbol{y}|\boldsymbol{x},\theta) = \mathcal{N}(\boldsymbol{y}|\text{NN}(\boldsymbol{x},\theta),\sigma^2) \qquad p(\boldsymbol{y}|\boldsymbol{x},\theta) = \text{multinoulli}(\boldsymbol{y}|\text{softmax}(\text{NN}(\boldsymbol{x},\theta)))$$

- Assume $\boldsymbol{y}$ and $\mathbf{f} = f(\boldsymbol{x},\theta)$ both to be vectors of size C, $\theta \in \mathbb{R}^P$ and define

Jacobian of size $C \times P$ with $[\mathcal{J}_\theta(\boldsymbol{x})]_{ci} = \nabla_{\theta_i} f_c(\boldsymbol{x},\theta)$

$$\mathcal{J}_\theta(\boldsymbol{x}) = \nabla_\theta f$$

Hessian of size $C \times P \times P$ with $[\mathcal{H}_\theta(\boldsymbol{x})]_{cij} = \nabla_{\theta_i}\nabla_{\theta_j} f_c(\boldsymbol{x},\theta)$

$$\mathcal{H}_\theta(\boldsymbol{x}) = \nabla^2_\theta f$$

$$\boldsymbol{r}(\boldsymbol{y};\mathbf{f}) = \nabla_f \log p(\boldsymbol{y}|\mathbf{f})$$
$$\mathbf{L}(\boldsymbol{y};\mathbf{f}) = -\nabla^2_f \log p(\boldsymbol{y}|\mathbf{f})$$

$$\nabla_\theta \log p(\boldsymbol{y}|f(\boldsymbol{x},\theta)) = \mathcal{J}_\theta(\boldsymbol{x})^\top \boldsymbol{r}(\boldsymbol{y};\mathbf{f})$$

$$\nabla^2_\theta \log p(\boldsymbol{y}|f(\boldsymbol{x},\theta)) = \mathcal{H}_\theta(\boldsymbol{x})^\top \boldsymbol{r}(\boldsymbol{y};\mathbf{f}) - \mathcal{J}_\theta(\boldsymbol{x})^\top \mathbf{L}(\boldsymbol{y};\mathbf{f})\mathcal{J}_\theta(\boldsymbol{x})$$

Generalized Gauss-Newton (GGN) Approximation

- The Hessian of the log-likelihood turned out to be

$$\nabla^2_\theta \log p(\boldsymbol{y}|f(\boldsymbol{x},\theta)) = \mathcal{H}_\theta(\boldsymbol{x})^\top \boldsymbol{r}(\boldsymbol{y};\mathbf{f}) - \mathcal{J}_\theta(\boldsymbol{x})^\top \mathbf{L}(\boldsymbol{y};\mathbf{f})\mathcal{J}_\theta(\boldsymbol{x})$$

- Ignoring the term involving $\mathcal{H}_\theta(\boldsymbol{x}) = \nabla^2_\theta f$, we have an approximation

$$\boxed{\nabla^2_\theta \log p(\boldsymbol{y}|f(\boldsymbol{x},\theta)) \approx -\mathcal{J}_\theta(\boldsymbol{x})^\top \mathbf{L}(\boldsymbol{y};\mathbf{f})\mathcal{J}_\theta(\boldsymbol{x})}$$

This approximation of the Hessian is guaranteed to be positive semi-definite unlike the original Hessian because $-\log p(y|f(x,\theta))$ may not be convex in θ

- This is called the Generalized Gauss-Newton (GGN) approximation* of the precision matrix used in Laplace Approximation
  - We can further apply diagonal or block-diagonal approximations for efficiency*

Reason: $\mathcal{H}_\theta(\boldsymbol{x})$ will be 0 for a linear function

- GGN is also equivalent to approximating $\mathbf{f} = f(\boldsymbol{x},\theta)$ by a linear function of $\boldsymbol{\theta}$

Gradient vector acting as "features" in this linear model

Gradient of f at $\theta = \theta_{MAP}$

Not the gradient of the (log)likelihood – that gradient is zero at θ_{MAP}

A linear function of θ

Makes PPD easy to compute when using Laplace approximation

A nonlinear function, e.g., a neural net, approximated by a linear function

$$\boxed{f(\boldsymbol{x}_*,\theta) \approx f(\boldsymbol{x}_*,\theta_{MAP}) + \nabla_{\theta_{MAP}} f^\top(\theta - \theta_{MAP})} = f_{\text{lin}}(\boldsymbol{x}_*,\theta)$$

*Improving predictions of Bayesian neural nets via local linearization (Immer et al, 2021)

PPD with GGN/Linearized Laplace's Approximation

- Assuming $p(\boldsymbol{y}|\boldsymbol{x},\theta) = p(\boldsymbol{y}|f(\boldsymbol{x},\theta))$, LA based PPD is

$$p(\boldsymbol{y}_*|\boldsymbol{x}_*,\mathcal{D}) = \int p(\boldsymbol{y}_*|f(\boldsymbol{x}_*,\theta))p(\theta|\mathcal{D})d\theta \approx \int p(\boldsymbol{y}_*|f(\boldsymbol{x}_*,\theta))\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}^{-1})d\theta$$

- We can use GGN and Linearized Laplace idea in two ways for the above PPD
- Use $f(\boldsymbol{x}_*,\theta)$ but use $\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}_{\text{GGN}}^{-1})$ as approx post instead $\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}^{-1})$
  - May require Monte Carlo integration if PPD integral is intractable (e.g., if f is a neural net or non-lin func)
  - Less commonly used and is less accurate*
- Use $f_{\text{lin}}(\boldsymbol{x}_*,\theta)$ instead of $f(\boldsymbol{x}_*,\theta)$ and also use $\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}_{\text{GGN}}^{-1})$ as approx. post.
  - Assuming $p(\boldsymbol{y}_*|f(\boldsymbol{x}_*,\theta)) = \mathcal{N}(y_*|f_{\text{lin}}(\boldsymbol{x}_*,\theta),\beta^{-1})$ for scalar-valued regression

Linear transformation of θ with $p(\theta|\mathcal{D}) = \mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}_{\text{GGN}}^{-1})$ and Gaussian noise $\epsilon \sim \mathcal{N}(0,\beta^{-1})$

$$y_* \approx f_{\text{lin}}(\boldsymbol{x}_*,\theta) + \epsilon$$

Even though $f(\boldsymbol{x}_*,\theta)$ is a complex function like neural net, using linearlized Laplace approx, we get PPD in closed form

$$p(y_*|\boldsymbol{x}_*,\mathcal{D}) \approx \mathcal{N}(y_*|f(\boldsymbol{x}_*,\theta_{MAP}), \nabla_{\theta_{MAP}}f^\top\boldsymbol{\Lambda}_{\text{GGN}}^{-1}\nabla_{\theta_{MAP}}f + \beta^{-1})$$

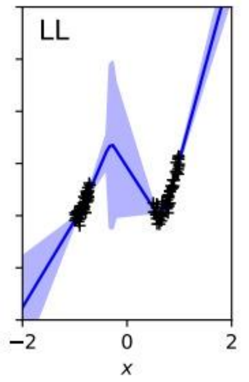
- *'In-Between' Uncertainty in Bayesian Neural Networks (Foong et al, 2019),
- *Improving predictions of Bayesian neural nets via local linearization (Immer et al, 2021)

Standard Laplace vs Linearlized Laplace

- Standard LA based PPD is usually computed using Monte Carlo sampling

$$p(y_*|\boldsymbol{x}_*,\mathcal{D}) \approx \int p(y_*|f(\boldsymbol{x}_*,\theta))\mathcal{N}(\theta|\theta_{MAP},\boldsymbol{\Lambda}^{-1})d\theta \approx \frac{1}{M}\sum_{i=1}^{M} p\left(y_*|f(\boldsymbol{x}_*,\theta^{(i)})\right)$$

- If the samples $\boldsymbol{\theta}^{(i)}$ don't come from high-prob regions of the posterior, the above PPD may have poor accuracy (often happens for high-dim posteriors)



- Linearlized Laplace based PPD is computed as

$$p(y_*|\boldsymbol{x}_*,\mathcal{D}) \approx \mathcal{N}\left(y_*|f(\boldsymbol{x}_*,\theta_{MAP}), \nabla_{\theta_{MAP}} f^{\top} \boldsymbol{\Lambda}_{\text{GGN}}^{-1} \nabla_{\theta_{MAP}} f + \beta^{-1}\right)$$

- Linearlized Laplace based PPD typically is reasonably accurate and sometimes even more accurate than standard LA with PPD computed using MC sampling*

- 'In-Between' Uncertainty in Bayesian Neural Networks (Foong et al, 2019),
- Improving predictions of Bayesian neural nets via local linearization (Immer et al, 2021)

More on Marginalization

Note: m is just a model identifier; can ignore when writing

- PPD is a marginalization using the posterior. For a model $p(y_*|\boldsymbol{x}_*, \theta, m)$

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}, m) = \int p(y_*|\boldsymbol{x}_*, \theta, m) p(\theta|\mathcal{D}, m) d\theta$$

$$\approx \frac{1}{S} \sum_{i=1}^{S} p\left(y_*|\boldsymbol{x}_*, \theta^{(i)}, m\right)$$

Will look at these ideas in more depth later

Each $\theta^{(i)}$ is drawn i.i.d. from the distribution $p(\theta|\mathcal{D}, m)$

Above integral replaced by a "Monte-Carlo Averaging"

- Marginalization can be done even over several choices of models

Marginalization over all weights of a single model m

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}, m) = \int p(y_*|\boldsymbol{x}_*, \theta, m) p(\theta|\mathcal{D}, m) d\theta$$

Marginalization over all finite choices $m = 1,2, \ldots, M$ of the model

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}) = \sum_{m=1}^{M} p(y_*|\boldsymbol{x}_*, \mathcal{D}, m) p(m|\mathcal{D})$$

For example, deep nets with different architectures

Like a double averaging (over all model choices, and over all weights of each model choice)

Haven't yet told you how to compute this quantity but will see shortly

Model Selection and Model Averaging

- Can use Bayes rule to find the best model from a set of models $m = 1,2,\ldots,M$

Posterior probability of model m

Marginal likelihood of model m

Prior probability of choosing model m

Will discuss later how to compute marginal likelihood

$$p(m|\mathbf{X}) = \frac{p(\mathbf{X}|m)p(m)}{p(\mathbf{X})} = \frac{p(\mathbf{X}|m)p(m)}{\sum_{m=1}^{M} p(\mathbf{X}|m)p(m)}$$

Marginal likelihood over all models

In general, intractable to compute exactly

$$p(\mathbf{X}|m) = \int p(\mathbf{X}|\theta, m)p(\theta|m)d\theta$$

Integrating out all possible parameter values under model m

Best model

$$\hat{m} = \arg\max_m p(m|\mathbf{X}) = \arg\max_m p(\mathbf{X}|m)p(m)$$

- If all models equally likely a priori then $\hat{m} = \arg\max_m p(\mathbf{X}|m)$

- For PPD, can use either the best model $\hat{m}$ or can average over all models

Test data

Training data

$$p(x_*|\mathbf{X}) \approx p(x_*|\mathbf{X}, \hat{m}) \quad \underline{\text{OR}} \quad p(x_*|\mathbf{X}) = \sum_{m=1}^{M} p(x_*|\mathbf{X}, m)p(m|\mathbf{X})$$