Bayesian Estimation Overview

Frequentist vs Bayesian?

Frequentist: Frequentist machine learning a.k.a. the general machine learning is when our main goal is to train the model such that we get the best accuracy on the dataset. We choose a single best model parameter θ that minimizes the training loss $L_D(\theta)$. We assume that θ always exists but in real world scenarios there can be multiple θ values that perform well on training data. Some limitations would be:

- Behave very differently on testing data (unseen)

- Can make overconfident decisions – when data is limited

- Disregard any uncertainty about θ

- High confidence in wrong predictions

- Focuses on minimizing the log loss:

$$\theta_{ML} = \arg\min_{\theta} L_D(\theta)$$
$$L_D = \frac{1}{N}\sum_{n=1}^{n} -\log p\,(y_n|x_n, \theta)$$

Bayesian: Bayesian estimation treats θ as a random variable and trains on the distribution of the model parameters rather than a single model parameter. It captures *epistemic uncertainty* (uncertainty due to lack of knowledge or data). Final predictions are made by averaging over the distribution. Some of the advantages that Bayesian offers over frequentist would be:

- Good for small data scenarios

- Avoids overfitting and improves generalization

- Naturally incorporates model uncertainty and avoids overconfident decisions

Calibration

A well calibrated parameter means how well the probability outputs match reality. Let's say in an ideal scenario when the model predicts an event with a probability of 0.7 the actual frequency of the event should also be 70%.

$$q(t|x) \approx p(t|x)$$

$q(t|x)$: models predicted probability for label t given input x

$p(t|x)$: true probability for label t given input x

Bayesian Learning

Bayesian Learning as Inference

Bayesian inference revolves around the Bayes Theorem, which combines a prior distribution $P(\theta)$ with the likelihood of the observed data $P(D|\theta)$ to produce a posterior distribution $P(\theta|D)$ over all the parameters. We assume the input as a training set $D = \left\{ (x_{n\,,} t_n)_{n=1}^{N} \right\}$ where t is the labels and x are the fixed constants. The target is the model parameter vector θ.

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

$P(D)$: Marginal Likelihood (normalizing agent and ensures that all the probabilities add up to 1)

$P(D|\theta)$: Likelihood of observed data

$P(\theta)$: Prior distribution

The joint distribution of the input and output would be: $p(\theta, D) = P(\theta) * P(D|\theta)$

Using the assumption that the labels t is conditionally independent given the covariates X = x and the model parameter vector θ we can write the likelihood as

$$P(D|\theta) = P(t_D|x_D, \theta) = \prod_{n=1}^{N} P(t_n|x_n, \theta)$$

Our main goal can be formulated as the ***minimization of free energy.***

Minimization of free energy

The goal of Bayesian learning is to find the best distribution of $q(\theta|D)$ over the model performance. This can be done by minimizing the free energy.

$$\min_{q(.|D)} F\big(q(\theta|D)||p(\theta, D)\big)$$

$$\mathrm{F}(q(\theta|\mathcal{D})||p(\theta,\mathcal{D})) = N \underbrace{\mathrm{E}_{\theta\sim q(\theta|\mathcal{D})}\left[L_{\mathcal{D}}(\theta)\right]}_{\text{average training loss}} + \underbrace{\mathrm{KL}(q(\theta|\mathcal{D})||p(\theta))}_{\text{information-theoretic regularizer}}$$

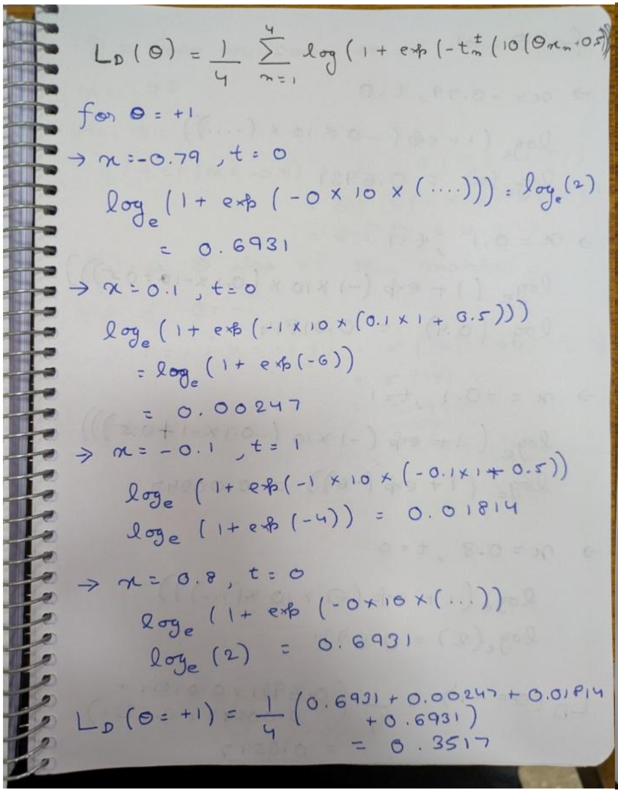
$$p(\theta|\mathcal{D}) = p(\theta|x_{\mathcal{D}}, t_{\mathcal{D}}) = \frac{p(\theta, t_{\mathcal{D}}|x_{\mathcal{D}})}{p(t_{\mathcal{D}}|x_{\mathcal{D}})} = \frac{\overbrace{\underbrace{p(\theta)}_{\text{prior}} \prod_{n=1}^{N} p(t_n|x_n,\theta)}^{}}{\underbrace{p(t_{\mathcal{D}}|x_{\mathcal{D}})}_{\text{marginal likelihood}}}$$

Here the marginal likelihood is given by

$$p(t_{\mathcal{D}}|x_{\mathcal{D}}) = \mathrm{E}_{\theta\sim p(\theta)}\left[\prod_{n=1}^{N} p(t_n|x_n,\theta)\right]$$

Example 1 – Bayesian vs Frequentist

Consider a binary classification problem with label t ∈ {0,1} and covariate x ∈ R. Assume that the model class H includes two models identified with the binary parameter θ ∈ {−1, +1}. The models define the likelihoods (t | x = x, θ) ~ p(t | x, θ) = Bern (t σ (10(θ x + 0.5))), and the points in the dataset is given as D = {(-0.79, 0), (0.1, 1), (-0.1, 1), (0.8,0)}



$L_D(\theta) = \frac{1}{4}\sum_{n=1}^{4} \log(1+\exp(-t_n^{\pm}(10(\theta x_n + 0.5)$

for $\theta = +1$

→ $x = -0.79$, $t = 0$

$\log_e(1+\exp(-0 \times 10 \times (....))) = \log_e(2)$

$= 0.6931$

→ $x = 0.1$, $t = 0$

$\log_e(1+\exp(-1 \times 10 \times (0.1 \times 1 + 0.5)))$

$= \log_e(1+\exp(-6))$

$= 0.00247$

→ $x = -0.1$, $t = 1$

$\log_e(1+\exp(-1 \times 10 \times (-0.1 \times 1 + 0.5))$

$\log_e(1+\exp(-4)) = 0.01814$

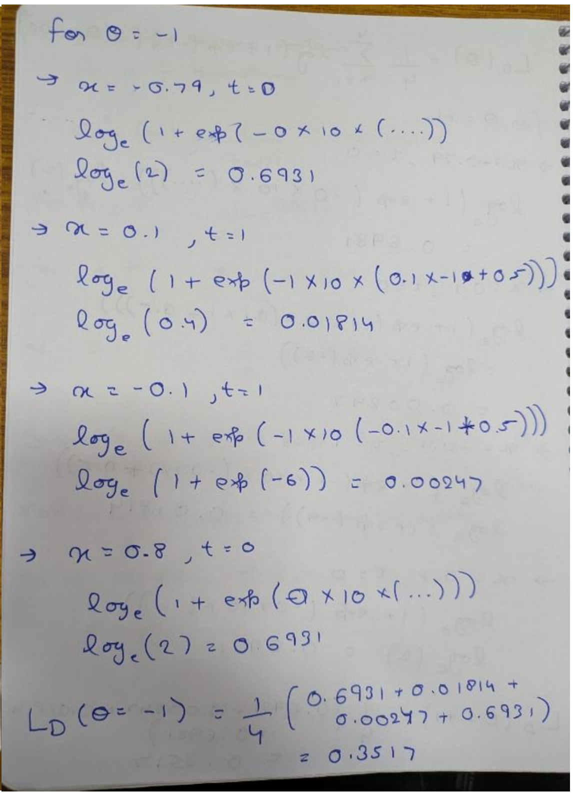
→ $x = 0.8$, $t = 0$

$\log_e(1+\exp(-0 \times 10 \times (...))$

$\log_e(2) = 0.6931$

$L_D(\theta = +1) = \frac{1}{4}(0.6931 + 0.00247 + 0.01814 + 0.6931)$

$= 0.3517$



for $\theta = -1$

→ $x = -0.79$, $t = 0$

$\log_e(1+\exp(-0 \times 10 \times (....))$

$\log_e(2) = 0.6931$

→ $x = 0.1$, $t = 1$

$\log_e(1+\exp(-1 \times 10 \times (0.1 \times -1 + 0.5)))$

$\log_e(0.4) = 0.01814$

→ $x = -0.1$, $t = 1$

$\log_e(1+\exp(-1 \times 10(-0.1 \times -1 + 0.5)))$

$\log_e(1+\exp(-6)) = 0.00247$

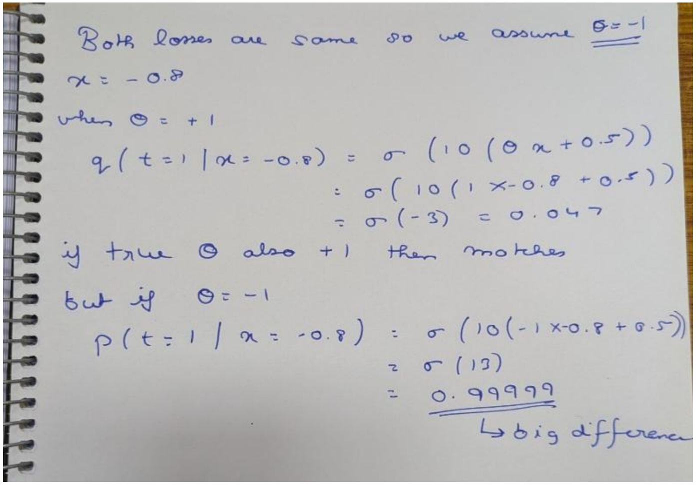
→ $x = 0.8$, $t = 0$

$\log_e(1+\exp(0 \times 10 \times (...)))$

$\log_e(2) = 0.6931$

$L_D(\theta = -1) = \frac{1}{4}(0.6931 + 0.01814 + 0.00247 + 0.6931)$

$= 0.3517$



Both losses are same so we assume $\underline{\underline{\theta = -1}}$

$x = -0.8$

when $\theta = +1$

$q(t=1 \mid x = -0.8) = \sigma(10(\theta x + 0.5))$

$= \sigma(10(1 \times -0.8 + 0.5))$

$= \sigma(-3) = 0.047$

if true θ also $+1$ then matches

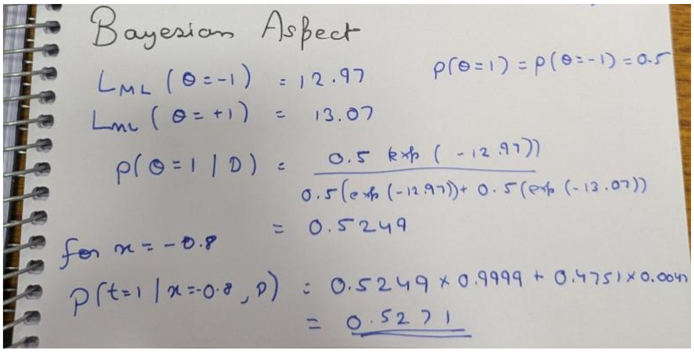
but if $\theta = -1$

$p(t=1 \mid x = -0.8) = \sigma(10(-1 \times -0.8 + 0.5))$

$= \sigma(13)$

$= \underline{\underline{0.99999}}$

↳ big difference



Bayesian Aspect

$L_{ML}(\theta = -1) = 12.97$ $p(\theta = 1) = p(\theta = -1) = 0.5$

$L_{ML}(\theta = +1) = 13.07$

$p(\theta = 1 \mid D) = \frac{0.5\exp(-12.97))}{0.5(\exp(-12.97)) + 0.5(\exp(-13.07))}$

$= 0.5249$

for $x = -0.8$

$p(t=1 \mid x = -0.8, D) = 0.5249 \times 0.9999 + 0.4751 \times 0.0047$

$= \underline{\underline{0.5271}}$

As we can see in this example when we implemented the frequentist approach the losses at 2 different values of *θ* is the same so the ML can choose any model but when we try to predict the outcome, we can see that one value of *θ* is clearly much better than the other. The machine could choose the other value which will make it work better in training data but create false predictions on test data basically overfitting on the data when *θ* = -1.

Looking at the Bayesian aspect, its clear that when we used Bayesian estimation since we trained on the entire set of *θ* as parameters we can see that the result we get is more centred and believable and more accurate than frequentist. This is the main difference.

Gibbs Sampling

Gibbs sampling is an MCMC algorithm for drawing samples from a multivariate posterior when direct sampling is difficult. The key idea is to sequentially sample each variable (or block of variables) from its conditional distribution given the others. Since these conditionals are (by assumption) easier to sample from, Gibbs sampling can be implemented even in complex models.

Basically, when the posterior $p(\theta|D)$ is too complex to calculate we use Gibbs Sampling. For every new test point x, we draw a sample θ from the posterior distribution $q(\theta|D)$. Using this compute the prediction $p(t|x, \theta)$ and repeat this for each input drawing new θ every time. Simple and cheap per sample and it mimics single model behaviour per input. Some of its limitations would be it has high variance and it's not stable.

$$\theta \sim q(\theta|D)$$

After a burn-in period, the Gibbs chain produces samples that can be treated as (correlated) draws from the true posterior. For example, in a Bayesian mixture model, one could alternate sampling the cluster assignments given current parameters, and then sampling parameters given assignments. Over many iterations the chain explores the joint space, yielding marginal samples and allowing estimates of any quantity (like means or variances) via sample averages.

Bayesian Ensemble

Bayesian ensembles refer to combining multiple model predictions to capture uncertainty. One natural form is Bayesian model averaging, where different models (or sets of parameters) are weighted by their posterior probabilities and averaged. In practice this can mean taking an ensemble of predictive models: for example, drawing many parameter samples from the posterior and averaging their predictions. This ensemble captures model uncertainty because it aggregates predictions from all plausible parameter settings.

Here, for each test point we compute the prediction $p(t|x,\theta)$ over many different θ sampled from the posterior and averaged them. It results in lower variance but is computationally expensive. The ensemble predictor averages the predictions obtained under all model parameters θ, each weighted by corresponding score $q(\theta|D)$ which generally doesn't belong to class H of soft predictors of the form $p(t|x,\theta)$, but is instead a **mixture** of such parameters

$$q(t|x,D) = E_{\theta \sim q(\theta|D)}\big(p(t|x,\theta)\big)$$

Example 2 – Analytical intuition using logistic regression

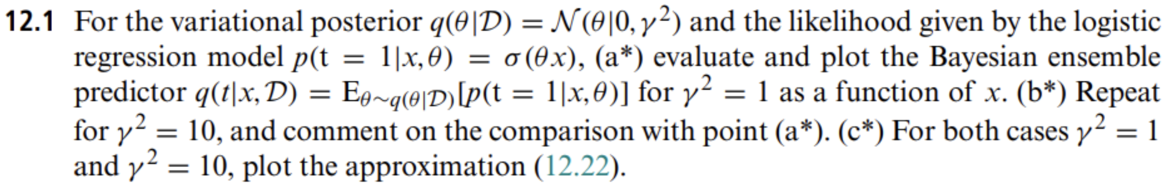
Consider logistic regression under the assumption that the posterior is the Gaussian Distribution $p(\theta \mid D) = \mathcal{N}(\theta \mid \nu, \sigma^2 I)$ for some mean ν and variance σ^2. Logistic regression sets the likelihood as $p(t = 1 \mid x, \theta) = \sigma\big(\theta^\top u(x)\big)$ for a feature of vectors u(x). For this model, what shall be the ensemble predictive distribution

$$p(t \mid x, D) = E_{\theta \sim \mathbb{N}\left(\theta|\nu, \sigma^2 I\right)}\left[\sigma\big(\theta^\top u(x)\big)\right]$$

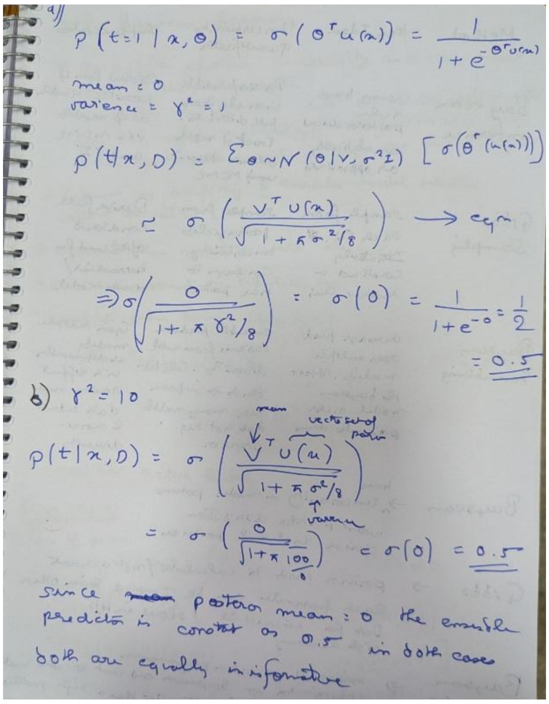
$$p(t \mid x, D) \approx \sigma\left(\frac{\nu^{\mathsf{T}} u(x)}{\sqrt{1+\frac{\pi\sigma^2}{8}}}\right)$$

A larger variance σ^2 of the posterior decreases the confidence of the prediction by reducing the absolute value of the logit (i.e. of the argument of the sigmoid function). In simple words – The probability that the label t = 1, given input x and model parameters θ, is computed using a sigmoid function applied to the dot product of θ and the feature vector *u(x).* When we perform Bayesian Prediction, we average the probability over all the likely values of θ drawn from a Gaussian posterior.

Question 12.1



12.1 For the variational posterior $q(\theta|\mathcal{D}) = \mathcal{N}(\theta|0,\gamma^2)$ and the likelihood given by the logistic regression model $p(\mathrm{t} = 1|x,\theta) = \sigma(\theta x)$, (a*) evaluate and plot the Bayesian ensemble predictor $q(t|x,\mathcal{D}) = \mathrm{E}_{\theta\sim q(\theta|\mathcal{D})}[p(\mathrm{t} = 1|x,\theta)]$ for $\gamma^2 = 1$ as a function of x. (b*) Repeat for $\gamma^2 = 10$, and comment on the comparison with point (a*). (c*) For both cases $\gamma^2 = 1$ and $\gamma^2 = 10$, plot the approximation (12.22).



a)

$p(t=1 \mid x, \theta) = \sigma(\theta^T u(x)) = \dfrac{1}{1+e^{-\theta^T u(x)}}$

mean = 0

variance = $\gamma^2 = 1$

$p(t|x, D) = \mathcal{E}_{\theta \sim \mathcal{N}(\theta|\nu, \sigma^2 I)} \left[\sigma(\theta^T(u(x)))\right]$

$\simeq \sigma\left(\dfrac{\nu^T u(x)}{\sqrt{1+\pi\sigma^2/8}}\right) \longrightarrow$ eqn

$\Rightarrow \sigma\left(\dfrac{0}{\sqrt{1+\pi\gamma^2/8}}\right) = \sigma(0) = \dfrac{1}{1+e^{-0}} = \dfrac{1}{2}$

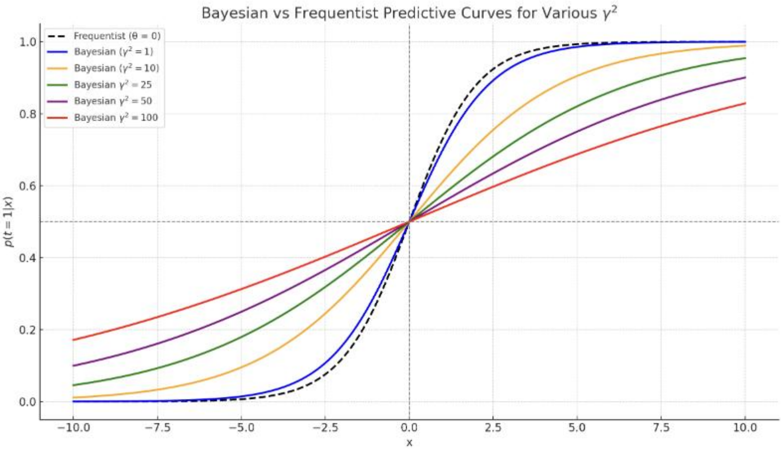
$= \underline{\underline{0.5}}$

b) $\gamma^2 = 10$

$p(t|x, D) = \sigma\left(\dfrac{\overset{\text{mean}}{\nu}^T \overbrace{u(x)}^{\text{vector set of param}}}{\sqrt{1+\pi\underset{\text{variance}}{\sigma^2}/8}}\right)$

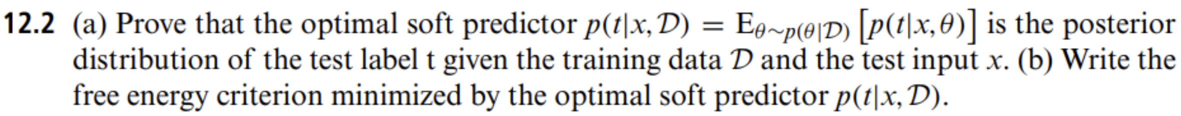
$= \sigma\left(\dfrac{0}{\sqrt{1+\pi\,\frac{100}{8}}}\right) = \sigma(0) = \underline{\underline{0.5}}$

Since ~~mean~~ posterior mean = 0 the ensemble predictor is constant as 0.5 in both cases both are equally informative



As we can see in this question when we took the posterior mean as 0 we get the ensemble prediction at 0.5 constant whereas if we plot the values of gamma, we can see how higher values of gamma the model is more cautious to assign extreme probability values and all these can be compared to a basic logistic regression frequentist approach.

Question 12.2



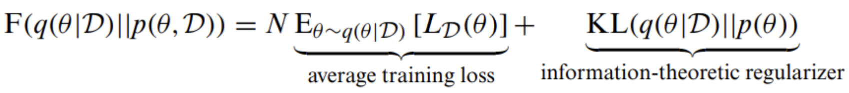
12.2 (a) Prove that the optimal soft predictor $p(t|x,\mathcal{D}) = \mathrm{E}_{\theta\sim p(\theta|\mathcal{D})}\left[p(t|x,\theta)\right]$ is the posterior distribution of the test label t given the training data $\mathcal{D}$ and the test input x. (b) Write the free energy criterion minimized by the optimal soft predictor $p(t|x,\mathcal{D})$.

According to law of probability

$$P(t|n,D) = \int p(t|x,\theta)p(\theta|D)\,d\theta$$

$$p(t|x,D) = E_{\theta\sim p(\theta|D)}\big(p(t|x,\theta)\big)$$

And,



$$\mathrm{F}(q(\theta|\mathcal{D})||p(\theta,\mathcal{D})) = N\,\underbrace{\mathrm{E}_{\theta\sim q(\theta|\mathcal{D})}\left[L_{\mathcal{D}}(\theta)\right]}_{\text{average training loss}} + \underbrace{\mathrm{KL}(q(\theta|\mathcal{D})||p(\theta))}_{\text{information-theoretic regularizer}}$$

Why Bayesian Learning?

- **Quantifies Epistemic uncertainty and can detect out-of-distribution data**
- **Robust to overfitting and enables model selection without validation**
- **It is optimal if model is well specified**
- **Minimizes a bound on the population loss (even for mis specified models)**

$$\mathcal{L}_p\big(q(\theta \mid D)\big) = E_{\theta\sim q(\theta|D)}\Big[E_{(x,t)\sim p(x,t)}[-\log p(t \mid x,\theta)]\Big]$$

$q(\theta \mid D)$: Posterior distribution$\theta \sim q(\theta \mid D)$: gibbs predictor
$(x,t) \sim p(x,t)$: true population$p(t \mid x,\theta)$: Likelihood

- **Facilitates Online Learning (streaming data)**
  Bayesian learning allows for a clean and incremental update to the model when a new data point arrives. Instead of retraining from scratch, we can use the current posterior as the new prior, multiply it by the likelihood of the new data point $p(t_{N+1} \mid x_{N+1},\theta)$ and the result is the updated posterior. This makes Bayesian methods naturally suited for online learning, especially in streaming settings where data keeps on arriving continuously

$$p(\theta \mid D) = p(\theta \mid \{t_n\},\{x_n\}) \propto p(\theta)\prod_{n=1}^{N} p(t_n \mid x_n,\theta)$$

$$p(\theta \mid D, x_{N+1}, t_{N+1}) \propto p(\theta)\left(\prod_{n=1}^{N} p(t_n \mid x_n,\theta)\right)p(t_{N+1} \mid x_{N+1},\theta)$$

$$p(\theta \mid D, x_{N+1}, t_{N+1}) \propto p(\theta \mid D)\cdot p(t_{N+1} \mid x_{N+1},\theta)$$

- **Facilitates Active Learning**
  In active learning learner selects data points adaptively given D the learner chooses next input x_{N+1} so he observes the corresponding random value t_{N+1}. An important property of Bayesian learning is ensemble prediction provides a well calibrated measure of uncertainty.

Exact Bayesian Learning

As we have seen that Bayesian learning solves an optimal inference problem by computing the posterior distribution $p(\theta, D)$ of the model parameters given in the training set. Therefore, Exact Bayesian Learning is possible on a limited set of models as we need to know the prior of the model parameters which is not feasible. Now we will talk about the scenarios where it is feasible.

When x and t are Independent – SPECIAL CASE

When x and t are independent, it means that the likelihood function does not depend on the input covariates x, and we can write:

$$p(t|x,\theta) = p(t|\theta)$$

This simplification leads to a scenario where you're essentially modelling the distribution of targets t directly, rather than modelling the conditional distribution given inputs x. This setup is most common in density estimation problems such as:

- Estimating a distribution from samples (e.g. Estimating a gaussian distribution from data)
- Learning a generative model for the data

Basically, what happens since x doesn't influence the likelihood, the learning problem becomes independent of the covariates. We only update the beliefs about θ based on observed data t, without needing to condition on x. In such a scenario the posterior $p(\,\theta \mid D\,)$ is computed simply by updating the natural parameters of the prior with sufficient statistics from the data.

$$t_n \sim \mathcal{N}(\mu, \sigma^2)$$

Here μ is unknown, σ^2 is known and there is no input x as $p(t|\mu)$ is independent of x

STEP 1: We define a prior on mean of known variance

$$p(\mu) = \mathcal{N}(\,\mu \mid \mu_0, \sigma_0^2\,)$$

STEP 2: Likelihood Function

$$p(\,t_1, \dots, t_N \mid \mu\,) = \prod_{n=1}^{N} \mathcal{N}(\,t_n \mid \mu, \sigma^2\,)$$

STEP 3: Posterior Calculation – our posterior is also gaussian as our prior is gaussian

$$p(\,\mu \mid \{t_n\}\,) = \mathcal{N}(\,\mu \mid \mu_N, \sigma_N^2\,)$$

Where:

- Posterior variance: $\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}$
- Posterior Mean: $\mu_N = \sigma_N^2 \left(\frac{\mu_0}{\sigma_0^2} + \frac{N\bar{t}}{\sigma^2}\right)$
- With $\bar{t} = \frac{1}{N}\sum t_n$

Since we used a conjugate prior, we can compute the posterior exactly.

When x and t are dependent – Gaussian-Gaussian GLM

In some simple cases, Bayesian learning has an exact solution. A classic example is a Gaussian–Gaussian model: assume the data have a Gaussian likelihood with known variance, and the prior on the mean is also Gaussian. Because these are conjugate distributions, the posterior is analytically Gaussian as well . In this setting one can derive closed-form expressions for the posterior mean and variance. Thus no approximations or sampling are needed – we get the exact posterior directly.

This is a general case in which the likelihood $p(t|x, \theta)$ depends on x. Lets assume a **regression model**:

Input: x *Feature Vector:* u(x) *Output/ Labels:* $t = \theta^T u(x) + \mathcal{N}(0, \beta^{-1})$

Likelihood will be given by:

$$p(\, t \mid x, \theta\,) = \mathcal{N}(\, t \mid \theta^T u(x), \beta^{-1}\,)$$

Our prior is derived from a gaussian given as:

$$p(\theta) = \mathcal{N}(\, \theta \mid 0, \alpha^{-1} I\,)$$

So, with the help of Bayes, we have our posterior as:

$$p(\, \theta \mid \mathcal{D}\,) = \mathcal{N}\big(\, \theta \mid \theta_D^{\text{MAP}}, \Sigma_D^{\text{MAP}}\,\big)$$

θ_D^{MAP}: most likely value of θ and $\lambda = \frac{\alpha}{\beta}$

Σ_D^{MAP}: how uncertain we are about our θ

$$\theta_D^{\text{MAP}} = \beta(\alpha I + \beta X_D^T X_D)^{-1} X_D^T t_D = (\lambda I + X_D^T X_D)^{-1} X_D^T t_D$$

$$\Sigma_D^{\text{MAP}} = (\alpha I + \beta X_D^T X_D)^{-1}$$

So our predictive distribution will look somewhat like:

$$p(\, t \mid x, \mathcal{D}\,) = \mathcal{N}\left(\, t \,\middle|\, \left(\theta_D^{\text{MAP}}\right)^T u(x), \sigma^2(x)\right)$$

$$\sigma^2(x) = \beta^{-1} + u(x)^T \Sigma_D^{\text{MAP}} u(x)$$

Mean prediction using most likely weights and variance is the uncertainty in the model parameters.

Whereas, if we try to derive the same in Frequentist Machine Learning we observe:

$$p\big(\, t \mid x, \theta_D^{\text{MAP}}\,\big) = \mathcal{N}\left(\, t \,\middle|\, \left(\theta_D^{\text{MAP}}\right)^T u(x), \beta^{-1}\right)$$

β^{-1}: only aleatoric uncertainty no epistemic

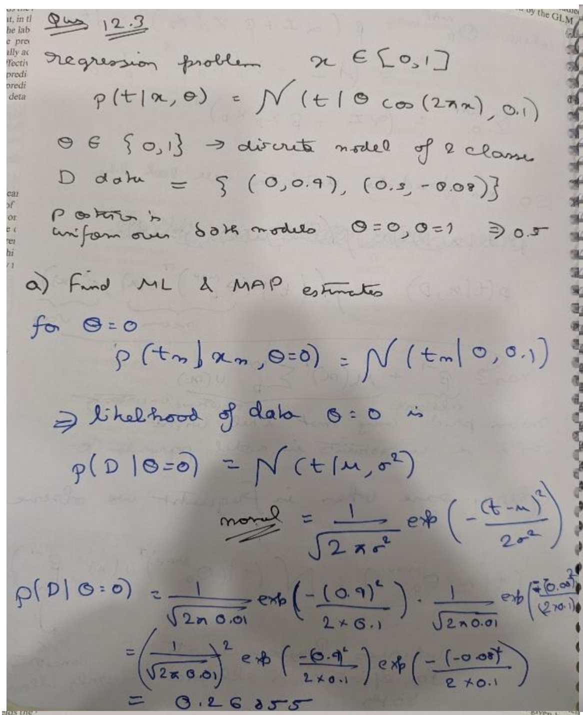
Whereas we can see that in Bayesian it accounts for both – aleatoric and epistemic uncertainty

Moral of the story:

Exact Bayesian learning for gaussian-gaussian GLM yields a predictive distribution that incorporates both aleatoric (data uncertainty) and epistemic uncertainty through a covariate-dependent variance. In contrast, the frequentist MAP approach only provides a point estimate with fixed uncertainty, ignoring model uncertainty.

Question 12.3 – Frequentist vs Bayesian Ensemble

12.3 We have a regression problem with input x constrained in the interval $[0, 1]$ and model class $\mathcal{H}$ consisting of soft predictors $p(t|x, \theta) = \mathcal{N}(t|\theta \cos(2\pi x), 0.1)$ with $\theta \in \{0, 1\}$. Note that there are only two models in the class. Assume that the prior puts equal probability on both models. A limited data set $\mathcal{D} = \{(0, 0.9), (0.5, -0.08)\}$ is available. (a) Obtain the ML and MAP solutions $\theta_{\mathcal{D}}^{ML}$ and $\theta_{\mathcal{D}}^{MAP}$. (b) Compute the posterior distribution $p(\theta|\mathcal{D})$. (c*) For $x = 1$, plot the ensemble predictive distribution $p(t|x, \mathcal{D})$ as a function of t. In the same plot, also show the corresponding frequentist predictor $p(t|x, \theta_{\mathcal{D}}^{ML})$.



Qus 12.3

regression problem $x \in [0,1]$

$$p(t|x,\theta) = \mathcal{N}(t|\theta\cos(2\pi x), 0.1)$$

$\theta \in \{0,1\}$ → discrete model of 2 classes

D data $= \{(0,0.9), (0.5,-0.08)\}$

P osterior is uniform over both models $\theta=0, \theta=1$ ⇒ 0.5

a) Find ML & MAP estimates

for $\theta = 0$

$$p(t_n|x_n, \theta=0) = \mathcal{N}(t_n|0, 0.1)$$

⇒ likelihood of data $\theta = 0$ is

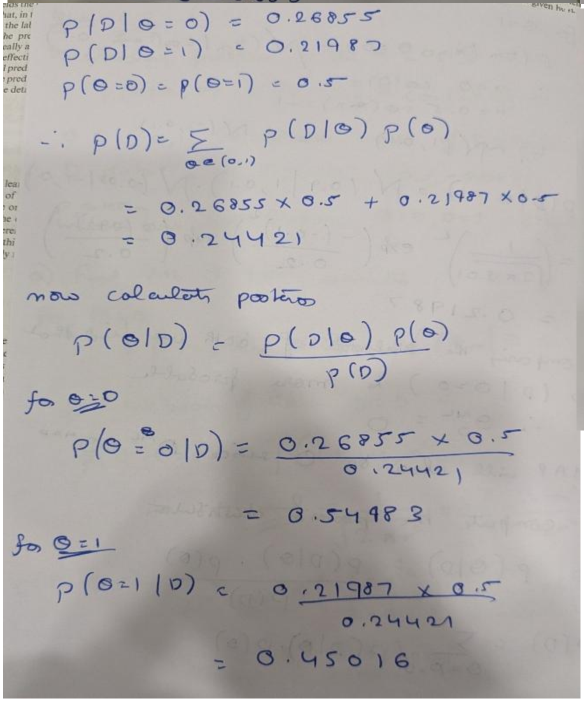
$$p(D|\theta=0) = \mathcal{N}(t|\mu, \sigma^2)$$

$$\underline{\underline{\text{normal}}} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right)$$

$$p(D|\theta=0) = \frac{1}{\sqrt{2\pi\, 0.01}} \exp\left(-\frac{(0.9)^2}{2\times 0.1}\right) \cdot \frac{1}{\sqrt{2\pi\, 0.01}} \exp\left(\frac{-(0.08)^2}{(2\times 0.1)}\right)$$

$$= \left(\frac{1}{\sqrt{2\pi\, 0.01}}\right)^2 \exp\left(\frac{-(0.9)^2}{2\times 0.1}\right) \exp\left(-\frac{(-0.08)^2}{2\times 0.1}\right)$$

$$= 0.26855$$



$$p(D|\theta=0) = 0.26855$$

$$p(D|\theta=1) = 0.21987$$

$$p(\theta=0) = p(\theta=1) = 0.5$$

$$\therefore p(D) = \sum_{\theta\in(0,1)} p(D|\theta)\, p(\theta)$$

$$= 0.26855 \times 0.5 + 0.21987 \times 0.5$$

$$= 0.24421$$

now calculate posterior

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

for $\underline{\underline{\theta=0}}$

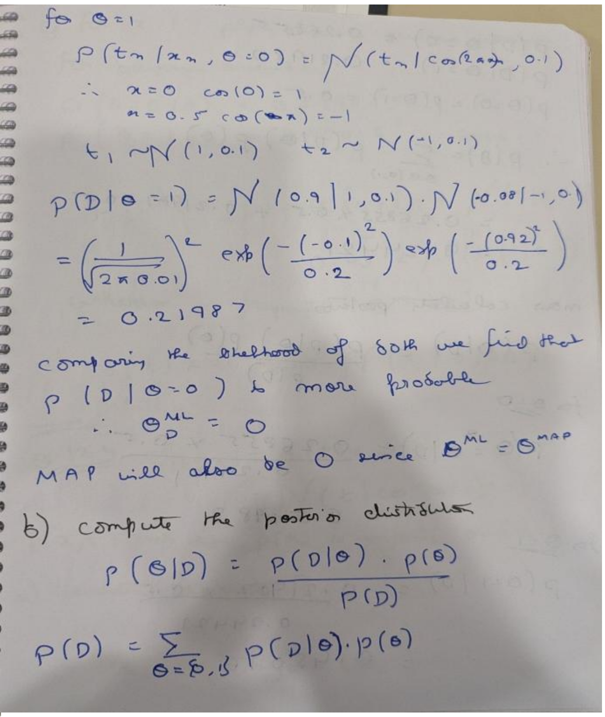
$$p(\theta=0|D) = \frac{0.26855 \times 0.5}{0.24421}$$

$$= 0.54983$$

for $\underline{\underline{\theta=1}}$

$$p(\theta=1|D) = \frac{0.21987 \times 0.5}{0.24421}$$

$$= 0.45016$$



for $\theta = 1$

$$p(t_n|x_n, \theta=0) = \mathcal{N}(t_n|\cos(2\pi x), 0.1)$$

$\therefore$ $x = 0$ $\cos(0) = 1$

$x = 0.5$ $\cos(\pi) = -1$

$$t_1 \sim \mathcal{N}(1, 0.1) \qquad t_2 \sim \mathcal{N}(-1, 0.1)$$

$$p(D|\theta=1) = \mathcal{N}(0.9|1, 0.1) \cdot \mathcal{N}(-0.08|-1, 0.1)$$

$$= \left(\frac{1}{\sqrt{2\pi\, 0.01}}\right)^2 \exp\left(-\frac{(-0.1)^2}{0.2}\right) \exp\left(\frac{-(0.92)^2}{0.2}\right)$$

$$= 0.21987$$

comparing the likelihood of both we find that $p(D|\theta=0)$ is more probable

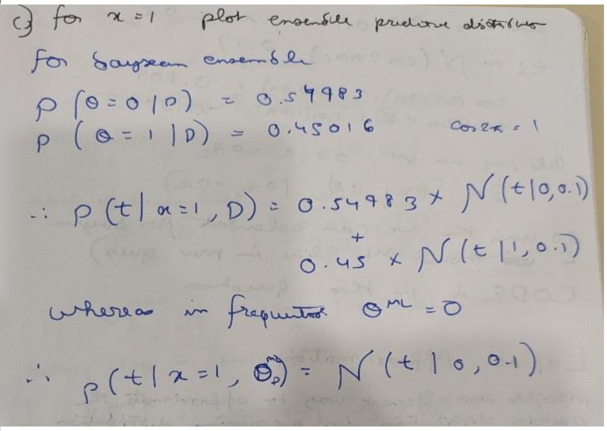
$$\therefore \theta_D^{ML} = 0$$

MAP will also be 0 since $\theta^{ML} = \theta^{MAP}$

b) compute the posterior distribution

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

$$p(D) = \sum_{\theta=\{0,1\}} p(D|\theta) \cdot p(\theta)$$



c) for $x = 1$ plot ensemble predictive distribution

For Bayesian ensemble

$$p(\theta=0|D) = 0.54983$$

$$p(\theta=1|D) = 0.45016 \qquad \cos 2\pi = 1$$

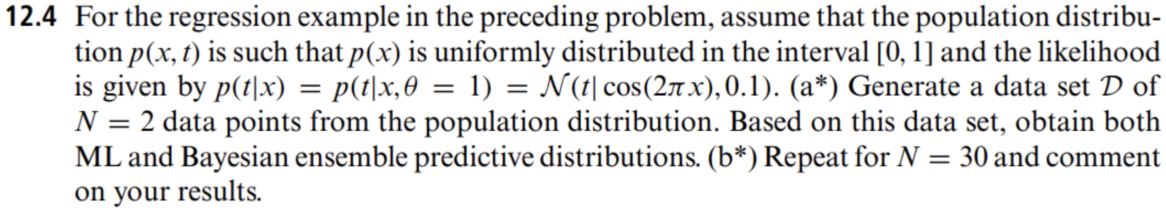
$$\therefore p(t|x=1, D) = 0.54983 \times \mathcal{N}(t|0, 0.1) + 0.45 \times \mathcal{N}(t|1, 0.1)$$

whereas in frequentist $\theta^{ML} = 0$

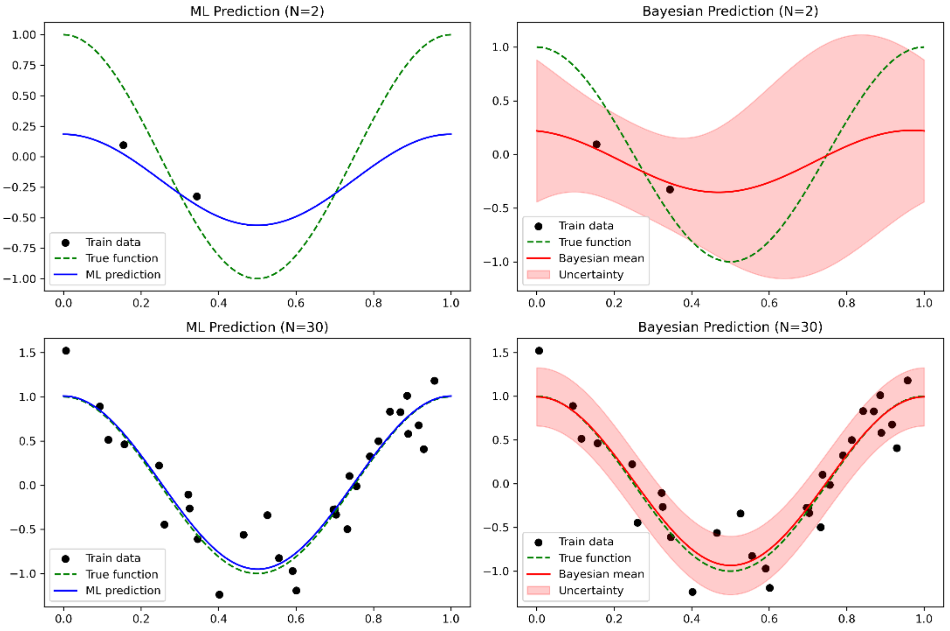
$$\therefore p(t|x=1, \theta_D^{ML}) = \mathcal{N}(t|0, 0.1)$$

As we can see that the frequentist uses only θ=0 for the final posterior prediction whereas in Bayesian ensemble we are taking an average of both of the θ values and getting a more general approach considering a weightage of both of the parameters

Ques 12.4 and Ques 12.5 (Replication of Figure)

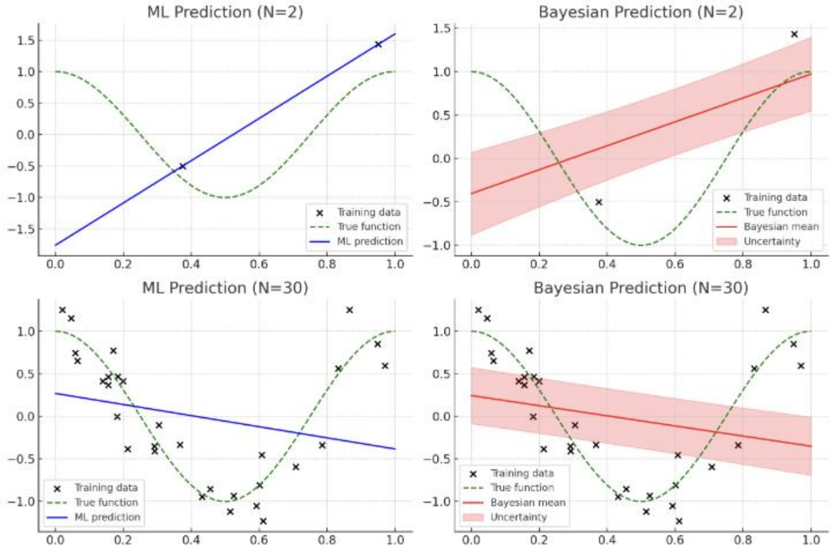


12.4 For the regression example in the preceding problem, assume that the population distribution $p(x,t)$ is such that $p(x)$ is uniformly distributed in the interval $[0,1]$ and the likelihood is given by $p(t|x) = p(t|x,\theta = 1) = \mathcal{N}(t|\cos(2\pi x), 0.1)$. (a*) Generate a data set $\mathcal{D}$ of $N = 2$ data points from the population distribution. Based on this data set, obtain both ML and Bayesian ensemble predictive distributions. (b*) Repeat for $N = 30$ and comment on your results.



a) N = 2 - **ML:** the blue line poorly captures the true cosine function because there's too little data basically it's a linear fir that overfits the 2 points. **Bayesian:** The red mean is also rough but comes with a large uncertainty band, showing epistemic uncertainty due to data scarcity

b) N = 30 - **ML:** Still a polynomial model and cannot capture the non-linearity of the cosine shape basically it's trying to fit but will have a lot of inaccuracy. **Bayesian:** The predictive mean is closer to the true function and the uncertainty band is a little narrower reflecting confidence due to more data



This is the same experiment but instead of a Fourier basis I have used a linear function and you can clearly see the difference. Our ML is basically either overfitting or underfitting and our Bayesian is much closer and have a bit more confidence

Laplace Approximation

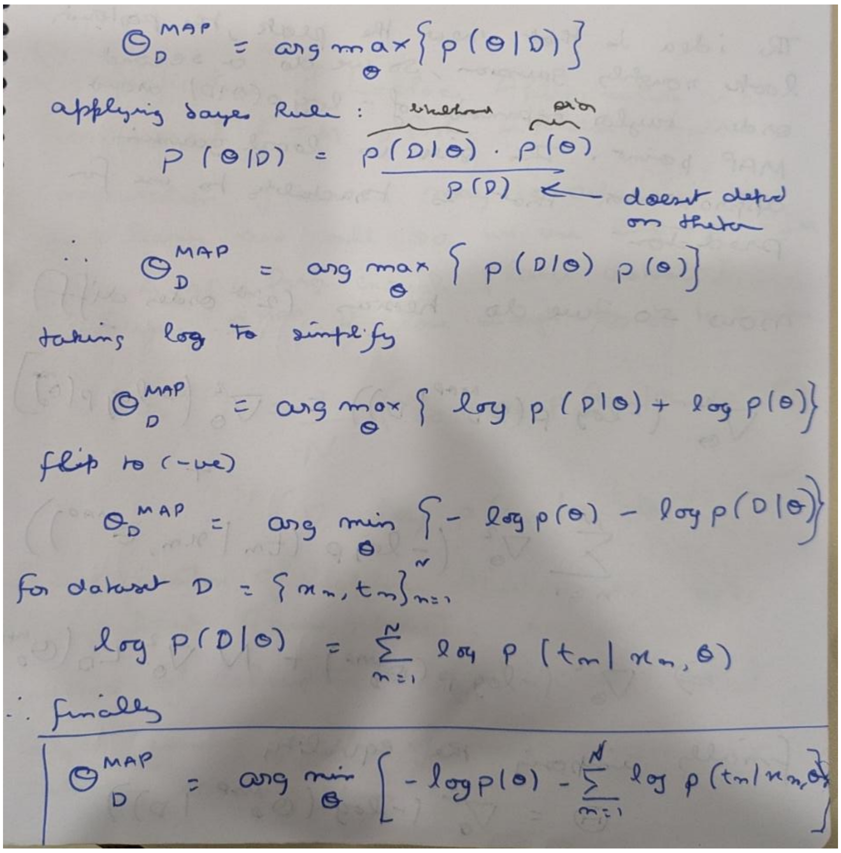
The Laplace approximation is a way to approximate an intractable posterior with a Gaussian around its mode. Intuitively, one finds the maximum a posteriori (MAP) point (the peak of the posterior), and then fits a Gaussian there using the curvature (second derivative) of the log-posterior. This gives a local Gaussian approximation that can be much easier to handle. For many problems, Laplace is the simplest alternative to exact integration.

Laplace Approximation provides an efficient way to approximate the posterior distribution via a gaussian distribution with mean given by MAP solution. Till now we have used:

$$p(\theta, D) \propto p(D|\theta)p(\theta)$$

But this is very challenging in high dimensional data so we will now approximate $p(\theta|D)$ as a gaussian centred at the MAP estimate. This is a 2-step process:

STEP 1: Find the MAP estimate (lets derive it)



$$\Theta_D^{MAP} = \arg\max_{\Theta} \{ p(\Theta|D) \}$$

applying bayes Rule :

$$P(\Theta|D) = \frac{\overbrace{p(D|\Theta)}^{\text{likelihood}} \cdot \overbrace{p(\Theta)}^{\text{prior}}}{p(D)} \xleftarrow{} \text{doesnt depend on theta}$$

$$\therefore \quad \Theta_D^{MAP} = \arg\max_{\Theta} \{ p(D|\Theta)\, p(\Theta) \}$$

taking log to simplify

$$\Theta_D^{MAP} = \arg\max_{\Theta} \{ \log p(D|\Theta) + \log p(\Theta) \}$$

flip to (-ve)

$$\Theta_D^{MAP} = \arg\min_{\Theta} \{ -\log p(\Theta) - \log p(D|\Theta) \}$$

for dataset $D = \{x_n, t_n\}_{n=1}^{N}$

$$\log p(D|\Theta) = \sum_{n=1}^{N} \log p(t_n | x_n, \Theta)$$

$\therefore$ finally

$$\boxed{\Theta_D^{MAP} = \arg\min_{\Theta} \left[-\log p(\Theta) - \sum_{n=1}^{N} \log p(t_n | x_n, \Theta) \right\}}$$

STEP 2: fit the gaussian to the posterior

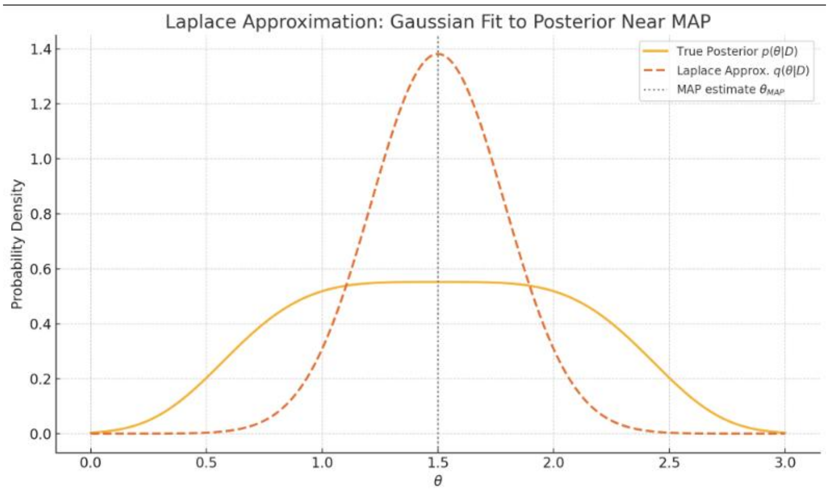
$$q(\theta \mid D) = \mathcal{N}\left(\theta \mid \theta_D^{\text{MAP}}, \Theta^{-1}\right)$$

θ_D^{MAP}: is the MAP solution

Θ^{-1}: optimized precision matrix

The main idea here is that near the peak, the posterior looks roughly gaussian, so we do a second order Taylor expansion of $-logp(\theta|D)$ around the MAP point. This gives us a local gaussian approximation that is tractable to use for prediction.

One of the things that I noticed here is that in practice Laplace is nothing but it assumes the posterior is roughly gaussian near the mode. The advantage that it has is that it is very simple. But here the strength is also its weakness like let's say the true posterior is far from being gaussian let's say skewed dataset or something. Laplace will give us an approximate answer when exact sampling is hard.



Continuing with the derivation we now calculate the hessian (2nd order differential):

$$\nabla_\theta^2[-\log p\,(\,\theta_D^{\text{MAP}} \mid D\,)] = \nabla_\theta^2[-\log p\,(\theta_D^{\text{MAP}})] + \nabla_\theta^2\left[\sum_{n=1}^{N} -\log p\,(\,t_n \mid x_n, \theta_D^{\text{MAP}}\,)\right]$$

$$\nabla_\theta^2[-\log p\,(\,\theta_D^{\text{MAP}} \mid D\,)] = \nabla_\theta^2[-\log p\,(\theta_D^{\text{MAP}})] + N \cdot \nabla_\theta^2 \mathcal{L}_{\mathcal{D}}(\theta_D^{\text{MAP}})$$

Finally imposing the equality

$$\Theta = \nabla_\theta^2[-\log p\,(\,\theta_D^{\text{MAP}} \mid D\,)]$$

$$p(\,\theta \mid D\,) \approx q(\,\theta \mid D\,) = \mathcal{N}(\,\theta \mid \theta_D^{\text{MAP}}, \Theta^{-1}\,)$$

Basically, we use the Taylor expansion to smooth the functions locally at the mode. It gives us the simplest possible local approximation that results in a Gaussian form. Its efficient and analytically tractable. We use the **second-order Taylor expansion** to locally approximate the log-posterior around its mode. This transforms a complex posterior into a **Gaussian** — which we can compute, sample from, and use for downstream prediction.

Simplification of Laplace Approximation via Empirical FIM

Computation of hessian may be expensive in deep learning and so we use Empirical FIM (Fisher Information Matrix). It is the expected curvature under the model and used in deep learning a lot. It serves to simplify the Laplace approximation by providing an approximate covariance matrix for the Gaussian Posterior.

The key assumptions we make here would be:

- Assuming N is large enough (large dataset)
- Assuming θ_D^{MAP} is close enough to the truth value

$$\widehat{\mathcal{FIM}}\left(\theta_D^{\text{MAP}}\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\nabla_\theta \log p\left(t_n \mid x_n, \theta_D^{\text{MAP}}\right)\right)\left(\nabla_\theta \log p\left(t_n \mid x_n, \theta_D^{\text{MAP}}\right)\right)^\top$$

Simplification of Laplace Approximation via GGN

We can also use Generalized Gauss – Newton Method to simplify the Laplace approximation. Instead of calculating $\nabla_\theta^2\left[-\log p\left(D \middle| \theta_D^{\text{MAP}}\right)\right]$, GGN uses a structure of Neural networks to simplify it using the chain rule

$$\nabla_\theta^2\left(-\log p\left(t_n \mid x_n, \theta\right)\right) \cong \left.\frac{\partial^2\left(-\log p\left(t_n \mid a\right)\right)}{\partial a^2}\right|_{a=a_n^L} \cdot \nabla_\theta a_n^L \cdot \left(\nabla_\theta a_n^L\right)^\top$$

This equation gives the Hessian of the negative log-likelihood using the second derivative with respect to the model output a, and then applying the chain rule through the model parameters θ. Its commonly used in neural networks and generalized linear models to compute curvature for optimization or uncertainty estimation.

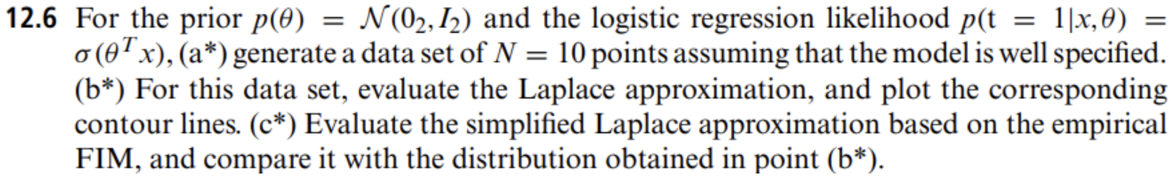
Applying logistic regression to this we would get:

$$l(a_n^L) = \log\left(1 + \exp\left(-t_n^{\pm}. a_n^L\right)\right)$$

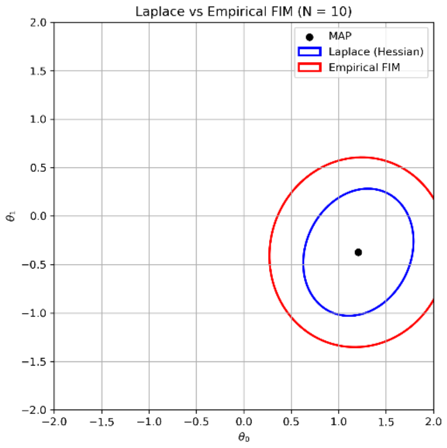
$$a_n^L = w^\top u(x_n)$$

$$\nabla_\theta^2\left(-\log p\left(t_n \mid x_n, \theta\right)\right) = \sigma(a_n^L)\left(1 - \sigma(a_n^L)\right) \cdot \nabla a_n^L \cdot \left(\nabla a_n^L\right)^\top$$

Ques 12.6



12.6 For the prior $p(\theta) = \mathcal{N}(0_2, I_2)$ and the logistic regression likelihood $p(\mathrm{t} = 1|x, \theta) = \sigma(\theta^T x)$, (a*) generate a data set of $N = 10$ points assuming that the model is well specified. (b*) For this data set, evaluate the Laplace approximation, and plot the corresponding contour lines. (c*) Evaluate the simplified Laplace approximation based on the empirical FIM, and compare it with the distribution obtained in point (b*).



When we calculated the exact hessian, we can see we get the exact centre point with the proper estimation whereas in a large dataset scenario when we calculate the using the empirical FIM we get a wider and overestimates the uncertainty and gives a conservative approximation. In simple words This figure shows that the Laplace approximation using the exact Hessian gives a more precise posterior, while the empirical FIM is a cruder but simpler method that tends to overestimate uncertainty. Both agree on he same parameters and share the same elliptical centre.

When we can't compute the posterior directly?

For complex models or high-dimensional datasets, the evidence *p(D)* becomes **intractable** to compute (i.e., the integral is impossible to evaluate analytically), so the posterior $p(\theta|D)$ cannot be **normalized** or **computed explicitly**. There are mainly 2 methods we use if we can't compute the posterior directly. They would be:

- **Variational Inference (VI):** approximate a true posterior from a simple distribution by minimizing the KL divergence between $q(\theta)$ and $p(\theta|D)$. It is easy to compute, fast but biased.

- **Monte Carlo (MC) Sampling:** Draw samples directly from the true posterior $p(\theta|D)$. Use those samples to compute posterior expectations. These methods are powerful, asymptomatically exact but can be slow/ converge poorly in high dimensions.

MC Sampling Based Bayesian Learning

GOAL

Samples s $\theta_1 \dots \dots .. \theta_S$ i.i.d. posterior $p(\theta|D)$ and compute the posterior expectations:

$$E_{\theta\sim p(\,\theta|D\,)}[f(\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} f\big(\theta_{(s)}\big)$$

$f(\theta)$: $p(t|x,\theta)$

Note we are not sampling from the actual posterior $p(\theta|D)$ as

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

$P(D)$: is difficult to calculate for complex datasets (evidence)

We can't sample directly because we can't normalize it. We rely on unnormalized posterior which is just $P(D|\theta)P(\theta)$ so we sample our θ from here as it follows the same distribution but unnormalized. Basically, we only have access to the joint distribution. We don't need to know $p(D)$ to perform posterior sampling. As long as we can compute the product $P(D|\theta)P(\theta)$, we can construct Markov chains that converge to the correct posterior distribution up to normalization. This allows us to perform Bayesian inference without computing the evidence.

Rejection Sampling

Rejection sampling is a simple Monte Carlo method to draw samples from a target distribution by using an easier "proposal" distribution as an envelope. The idea is to sample a candidate point from the proposal distribution, then accept it with probability proportional to the ratio of the target density to the proposal density at that point. Geometrically, one can imagine graphing the target density curve and throwing "darts" under a bounding curve: points falling under the target curve are accepted, others are rejected. As one description puts it, we draw uniform points under the proposal curve and keep those below the target curve to simulate the target distribution. Rejection sampling is easy to implement when a suitable envelope is found, but can be inefficient if the proposal does not closely fit the target (leading to many rejections). It is mostly used in low dimensions or as a building block in more complex algorithms.

Basically, we often can't sample directly from the posterior. So, we design an accept – reject mechanism that samples from the prior and accepts samples with a specific probability such that the accepted samples follow the posterior distribution.

Generate samples $\theta_1 \ldots \ldots . \theta_s \sim p(\theta|D)$ using accept/ reject logic

$$p(\theta|\mathrm{acc}) = \frac{p(\theta)p(\mathrm{acc}|\theta)}{p(\mathrm{acc})}$$

$$p(\mathrm{acc}) = \int p(\theta')p(\mathrm{acc}|\theta')d\,\theta'$$

Here $\theta'_1 \ldots \ldots \theta'_s$ i.i.d. from prior $p(\theta)$ [known]

$p(\mathrm{acc}|\theta')$: acceptance probability

Our main goal is to satisfy

$$p(\theta|\mathrm{acc}) = p(\theta|D) \propto p(\theta)p(D|\theta)$$

Combining we get $p(\mathrm{acc}|\theta) \propto p(D|\theta)$

$$p(\mathrm{acc}|\theta) = \frac{p(D|\theta)}{B}$$

Where B should satisfy

- It doesn't depend on θ
- It should ensure the equality $p(acc|\theta) \leq 1$ for all possible values of sample θ

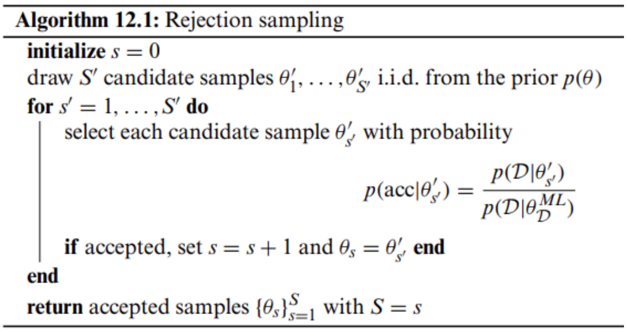
If yes then,

$$B = max_{\theta}p(D|\theta) = p(D\big|\theta_D^{ML})$$

Finally, we can say that:

$$p(acc|D) = \frac{p(\theta|D)}{p(D\big|\theta_D^{ML})}$$

- Very inefficient when the acceptance probability is small for samples drawn from prior $p(\theta)$
- Large number of samples are required
- Used for only small dimensional problems



Algorithm 12.1: Rejection sampling

```
initialize $s = 0$
draw $S'$ candidate samples $\theta'_1, \ldots, \theta'_{S'}$ i.i.d. from the prior $p(\theta)$
for $s' = 1, \ldots, S'$ do
    select each candidate sample $\theta'_{s'}$ with probability
        $p(\mathrm{acc}|\theta'_{s'}) = \frac{p(\mathcal{D}|\theta'_{s'})}{p(\mathcal{D}|\theta_{\mathcal{D}}^{ML})}$
    if accepted, set $s = s + 1$ and $\theta_s = \theta'_{s'}$ end
end
return accepted samples $\{\theta_s\}_{s=1}^{S}$ with $S = s$
```

MORAL:

- Rejection sampling converts sampling from prior -> posterior using a simple accept/ reject rule.
- Its simple and exact (theoretically) but can be very inefficient when a> the prior and posterior are very different and b> in high dimensions as it causes low acceptance rate.

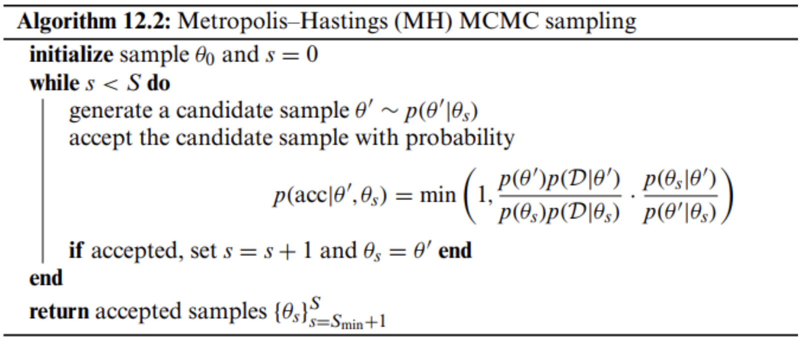
Markov Chain Monte Carlo: Metropolis Hastings Algorithm

The Metropolis–Hastings (MH) algorithm is a general MCMC technique that overcomes the need for an envelope distribution. It works by proposing a new point (from any chosen proposal distribution) and then accepting it with a probability that depends on the ratio of posterior densities. Concretely, starting from a current sample, MH draws a candidate and then computes an acceptance probability (typically $min\,(1,\ target(candidate)/target(current))$, adjusting for proposal asymmetry). If accepted, the chain moves to the candidate; if rejected, it stays at the current point. Over many steps, this Markov chain's stationary distribution is exactly the desired posterior.

We know Rejection sampling draws i.i.d. samples from posterior which is highly inefficient when the prior is very different from the posterior. This leads to low acceptance rate. Here we make 2 main approximations:

- Produced samples are correlated, not i.i.d. $(\theta_1 \ldots\ldots \theta_s)$
- The marginal distributions of samples converge to posterior asymptomatically as $s \rightarrow \infty$

MCMCMH methods generate candidate samples in an adaptive way with the next sample θ_{s+1} being in the neighbourhood of last accepted sample θ_s. These are correlated and may require long time to explore the entire posterior domain. It uses transition distribution $p(\theta'|\theta)$



```
Algorithm 12.2: Metropolis–Hastings (MH) MCMC sampling
initialize sample $\theta_0$ and $s = 0$
while $s < S$ do
    generate a candidate sample $\theta' \sim p(\theta'|\theta_s)$
    accept the candidate sample with probability
        $p(\text{acc}|\theta', \theta_s) = \min\left(1, \frac{p(\theta')p(\mathcal{D}|\theta')}{p(\theta_s)p(\mathcal{D}|\theta_s)} \cdot \frac{p(\theta_s|\theta')}{p(\theta'|\theta_s)}\right)$
    if accepted, set $s = s + 1$ and $\theta_s = \theta'$ end
end
return accepted samples $\{\theta_s\}_{s=S_{\min}+1}^{S}$
```

Posterior ratio:

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

Accept $\theta > 1$. Still accept $\theta < 1$ but with room for exploration.

Transition Ratio: $\frac{p(\theta_s|\theta')}{p(\theta'|\theta_s)}$ = 1 (symmetric)

Our probability of acceptance is $\propto$ improvement in the posterior obtained by θ' as opposed to θ_s. Some main use – cases would be:

- $p(\theta|D)$ is known up to a constant
- $Direct\ sampling\ isint\ possible$
- High dimensional data or complex shape
- Any target distribution

We should avoid using it when:

- Conjugate posterior is possible
- Very high dimension with local minima

Stochastic Gradient Markov Chain Monte Carlo

MCMC-MH requires full access of the dataset at each iteration to compute posterior which is very slow for large datasets. Stochastic Gradient MCMC uses

- Mini Batches of data instead of full dataset
- Combining ideas from stochastic gradient descent with MCMC

Stochastic Gradient Langevin Dynamics (SGLD)

Stochastic Gradient Langevin Dynamics (SGLD) is a modern MCMC method that scales Bayesian sampling to large datasets and models (often called SGMCMC for "stochastic gradient MCMC"). SGLD combines ideas from optimization and sampling: it performs gradient updates on the log-posterior (like stochastic gradient descent) but injects carefully scaled noise so that in the limit it samples from the posterior. In effect, one treats the optimization trajectory itself as a sampling process. Practically, SGLD uses mini-batches to compute a noisy gradient of the log-likelihood, adds Gaussian noise, and slowly decays the step size. The result is that the parameters "wander" around a local mode, producing approximate samples of the posterior. It can be viewed as a variant of Langevin dynamics where the gradient is estimated from a minibatch. The big advantage is efficiency: like SGD, SGLD only looks at part of the data each step, making it feasible for very large datasets. This blurs the line between optimization and inference – one gets approximate posterior samples by essentially running a noisy SGD. SGLD has been applied, for example, to Bayesian deep learning, letting neural networks capture uncertainty by treating the training updates as a sampling process.

GD -> moves deterministically in direction of gradient + LD -> adds gaussian noise to simulate diffusion
. process -> leads to sample from distribution

$$\theta_{s+1} \leftarrow \theta_s + \gamma_s \left(\frac{1}{S_s} \sum_{n \in \mathcal{S}_s} \nabla_\theta \log p\,(t_n | x_n, \theta_s) + \frac{1}{N} \nabla_\theta \log p\,(\theta_s) \right) + \nu_s$$

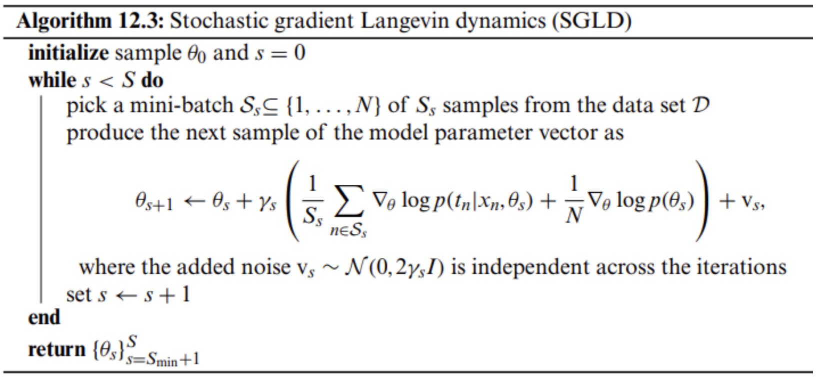
θ_s: current model parameter

γ_s: step size

$n \in \mathcal{S}_s$: mini batch

$\nabla_\theta \log p\,(\theta_s)$: Gradient of log prior

ν_s: Gaussian noise



Algorithm 12.3: Stochastic gradient Langevin dynamics (SGLD)

```
initialize sample $\theta_0$ and $s = 0$
while $s < S$ do
    pick a mini-batch $\mathcal{S}_s \subseteq \{1, \ldots, N\}$ of $S_s$ samples from the data set $\mathcal{D}$
    produce the next sample of the model parameter vector as
        $\theta_{s+1} \leftarrow \theta_s + \gamma_s \left( \frac{1}{S_s} \sum_{n \in \mathcal{S}_s} \nabla_\theta \log p(t_n | x_n, \theta_s) + \frac{1}{N} \nabla_\theta \log p(\theta_s) \right) + \mathrm{v}_s,$
     where the added noise $\mathrm{v}_s \sim \mathcal{N}(0, 2\gamma_s I)$ is independent across the iterations
    set $s \leftarrow s + 1$
end
return $\{\theta_s\}_{s=S_{\min}+1}^{S}$
```

SGLD samples from an approximation from the posterior $p(\theta|D)$ especially when run with decision step size. Under proper conditions it will converge to true posterior asymptomatically.

Monte Carlo Methods for Likelihood free models

In many real-world models, the likelihood function $p(D|\theta)$ is either intractable or unknown, but it is still possible to simulate data from the generative model given a parameter θ. In such likelihood-free scenarios, traditional MCMC methods like Metropolis-Hastings or Stochastic Gradient Langevin Dynamics (SGLD) cannot be applied directly.

To overcome this, we use Approximate Bayesian Computation (ABC), a Monte Carlo technique that:

- Avoids the explicit computation of the likelihood,
- Instead, simulates datasets using proposed parameters,
- And accepts parameters if the simulated data are "close enough" to the observed data, according to some distance metric and tolerance.

This allows us to perform Bayesian inference even when the likelihood is unavailable, making ABC an essential tool for complex scientific models, e.g., in genetics, ecology, and epidemiology.

Importance sampling

Importance sampling is a Monte Carlo technique for approximating expectations with respect to one distribution by sampling from another. One draws samples from a convenient proposal distribution and then weights each sample by the ratio of the target density to the proposal density. These weighted samples can then estimate expectations under the target distribution. In Bayesian inference, importance sampling can estimate integrals for the posterior when direct sampling is not possible. Formally, it is "a Monte Carlo method for evaluating properties of a particular distribution while only having samples from a different distribution". The weights correct for the mismatch between the proposal and target. Importance sampling is unbiased and simple, but it can suffer from high variance if the proposal does not cover the target well. In practice, it is often used as a building rather than a standalone posterior sampler.

Basically, sometimes we can't/ don't want to sample from the true posterior $p(\theta|D)$, but we can sample from a different, easier distribution $q(\theta)$. Importance sampling helps us to correct the mismatch. We sample from $q(\theta)$, and correct the mismatch to target $p(\theta|D)$ using importance weights. So, we need to calculate:

$$E_{\theta\sim p(\theta|\mathbb{D})}[f(\theta)] \simeq \sum_{s=1}^{S} w_s f(\theta_s)$$

$$p(t \mid x, \mathcal{D}) \simeq \sum_{s=1}^{S} w_s p(t \mid x, \theta_s)$$

Unnormalized weights: $v_s = \frac{p(\theta_s,\mathcal{D})}{q(\theta_s)}$

Normalized weights: $w_s = \frac{v_s}{\sum_{s'=1}^{S} v_{s'}}$

We can combine it with other MCMC methods which will be called annealed importance sampling.

Variational Bayesian Learning

Basically, in Bayesian we calculate:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

$P(D)$: is difficult to calculate for complex datasets (evidence)

Our variational idea says that we will approximate the posterior $p(\theta|D)$ with a simple distribution $q(\theta|D) \rightarrow \emptyset$. Choose $q(\theta|\emptyset)$ closest to $p(\theta|D)$. We use KL divergence for it

$$\phi^* = \arg\min_{\phi} \mathrm{KL}\left(q(\theta \mid \phi)|p(\theta \mid D)\right) \qquad \text{(this has intractable denominator)}$$

We minimize evidence lower bound:

$$\mathcal{F}(q) = \mathrm{KL}\big(q(\theta \mid \phi)|p(\theta)\big) - E_{q(\theta|\phi)}[\log p\,(D \mid \theta)]$$

Basically:

$$min_{\phi}\, \mathcal{F}\big(q(\theta \mid \phi)|p(\theta \mid D)\big)$$

Reparameterization Based Variational Bayesian Learning

Variational inference (VI) turns Bayesian inference into an optimization problem: we posit a family of approximate posteriors and then adjust them to be close to the true posterior. A popular trick in modern VI (especially in variational autoencoders) is the reparameterization trick. The idea is to express a random variable as a deterministic transform of a fixed noise source. For example, a Gaussian random sample $z \sim \mathcal{N}(\mu, \sigma^2)$ can be written as $z = \mu + \sigma\epsilon$ with $\epsilon \sim \mathcal{N}(0,1)$. This reparameterization lets us backpropagate gradients through the randomness: we optimize the parameters (mean and variance) by gradient descent using samples. In general, the reparameterization trick allows computing gradients of stochastic objectives with low variance. In VI, this means we can maximize the evidence lower bound (ELBO) efficiently using mini-batches and SGD, almost like training a neural network.

Assumptions:

- Prior over parameter:

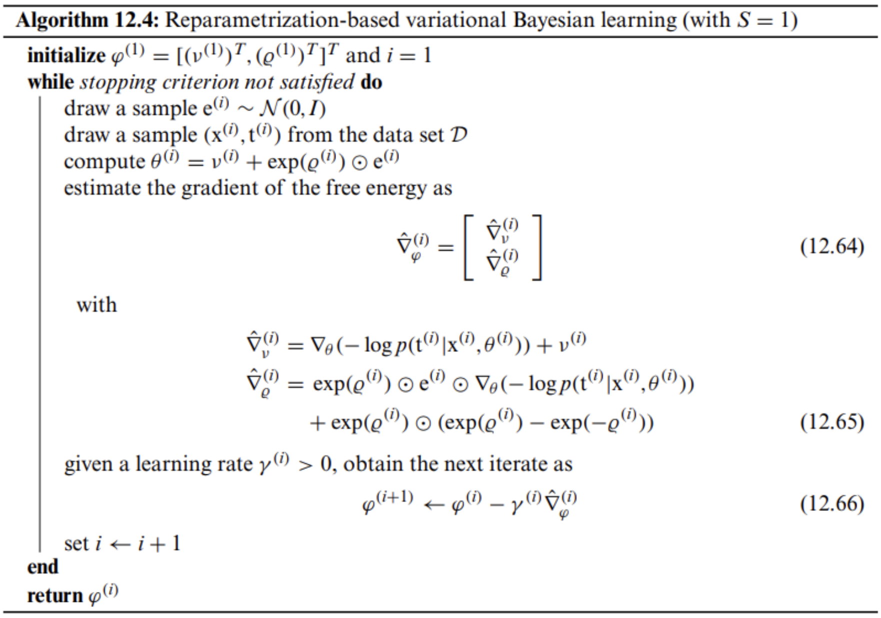
$$p(\theta) = \mathcal{N}(\theta \mid 0, I)$$

- Variational Posterior:

$$q(\theta \mid \varphi) = \mathcal{N}\big(z \mid \nu, \mathrm{Diag}(\exp(2\varrho))\big)$$

Where:

$$\varphi = \quad [\![\{bmatrix\}\, \nu\text{^}T,\ \ \varrho\text{^}T\ ]\!]\ \{bmatrix\}\text{^}T$$



Algorithm 12.4: Reparametrization-based variational Bayesian learning (with $S = 1$)

```
initialize $\varphi^{(1)} = [(\nu^{(1)})^T, (\varrho^{(1)})^T]^T$ and $i = 1$
while stopping criterion not satisfied do
    draw a sample $e^{(i)} \sim \mathcal{N}(0, I)$
    draw a sample $(x^{(i)}, t^{(i)})$ from the data set $\mathcal{D}$
    compute $\theta^{(i)} = \nu^{(i)} + \exp(\varrho^{(i)}) \odot e^{(i)}$
    estimate the gradient of the free energy as
        $\hat{\nabla}_{\varphi}^{(i)} = \begin{bmatrix} \hat{\nabla}_{\nu}^{(i)} \\ \hat{\nabla}_{\varrho}^{(i)} \end{bmatrix}$    (12.64)
    with
        $\hat{\nabla}_{\nu}^{(i)} = \nabla_{\theta}(-\log p(t^{(i)}|x^{(i)}, \theta^{(i)})) + \nu^{(i)}$
        $\hat{\nabla}_{\varrho}^{(i)} = \exp(\varrho^{(i)}) \odot e^{(i)} \odot \nabla_{\theta}(-\log p(t^{(i)}|x^{(i)}, \theta^{(i)}))$
            $+ \exp(\varrho^{(i)}) \odot (\exp(\varrho^{(i)}) - \exp(-\varrho^{(i)}))$    (12.65)
    given a learning rate $\gamma^{(i)} > 0$, obtain the next iterate as
        $\varphi^{(i+1)} \leftarrow \varphi^{(i)} - \gamma^{(i)} \hat{\nabla}_{\varphi}^{(i)}$    (12.66)
    set $i \leftarrow i + 1$
end
return $\varphi^{(i)}$
```

Now let's see in practice how these two hold up in real life scenarios.

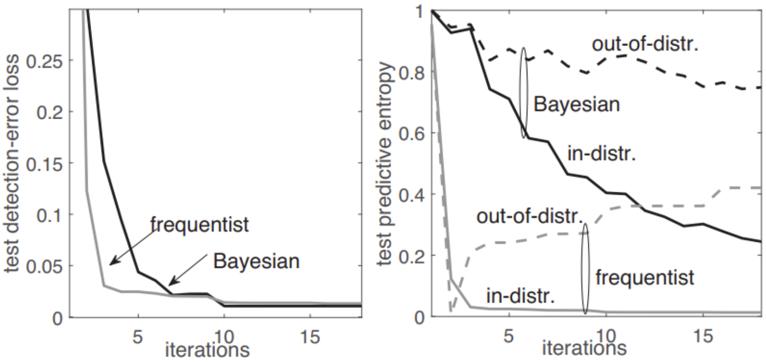


Figure 12.5 Test detection-error loss (left) and test predictive entropy for in-distribution and out-of-distribution samples (right) under frequentist (MAP) and Bayesian VI-based learning.

As we can see in the left figure at approx. 6 or 7 iterations both frequentist and Bayesian have the test detection error loss almost exactly same but in the right figure we can see how Bayesian is more self-aware whereas frequentist is adamant with its predictions.

Sparsity and Parametric Variational Methods

Some variational methods incorporate sparsity by using priors or approximations that encourage many parameters to be zero. For example, a spike-and-slab prior or a Laplace prior can induce a sparse approximate posterior, effectively "turning off" irrelevant parameters. Parametric variational methods typically assume a particular form (e.g. fully factorized Gaussian) for the approximate posterior. These approximations trade expressiveness for tractability: a simpler variational family means faster optimization but potentially poorer fit. In practice, one chooses a variational family that balances flexibility and efficiency. For instance, sparse variational Gaussian processes use a small set of "inducing points" to reduce complexity, while deep learning methods may prune network weights to encourage sparsity. The goal is to capture the most important uncertainty without blowing up the number of variational parameters.

Basically, in real world scenarios, many parameters may not be that important or necessary. Sparsity helps

- Compress the model
- Improve interpretability
- Reduce overfitting

Encourage or enforce the variational posterior $q(\theta|\phi)$ to place the most probability mass around zero for many entities in θ.

Spike and Slab distributions

Spike: A delta mass at 0 -> remove parameters completely

Slab: A wide gaussian -> allows for real values when parameter is important

$$\theta_i = b_i \cdot \overline{\theta_i}$$

In Neural Network: $b_i = 0$

Adversarial Bayesian Learning

Adversarial variational Bayes is a technique that uses ideas from adversarial learning (as in GANs) to improve variational inference. Instead of choosing a fixed simple variational family, one trains an expressive inference model (often a neural network) to produce samples, and uses a discriminator to measure how close those samples are to the true posterior. The adversarial "game" guides the variational parameters so that the generated samples fool the discriminator into thinking they come from the true posterior. This allows very flexible approximate posteriors at the cost of a more complex training setup. While advanced, the core idea is to use a learned loss (via an adversarial network) to drive the variational approximation beyond simple analytic forms.

We don't explicitly write down the posterior $q(\theta|\phi)$. Instead, we define a sample procedure:

$$\theta = G(\epsilon; \phi)$$

Where $\epsilon \sim p(\epsilon)$

We don't know the density $q(\theta|\phi)$ so we can't directly compute the KL divergence.

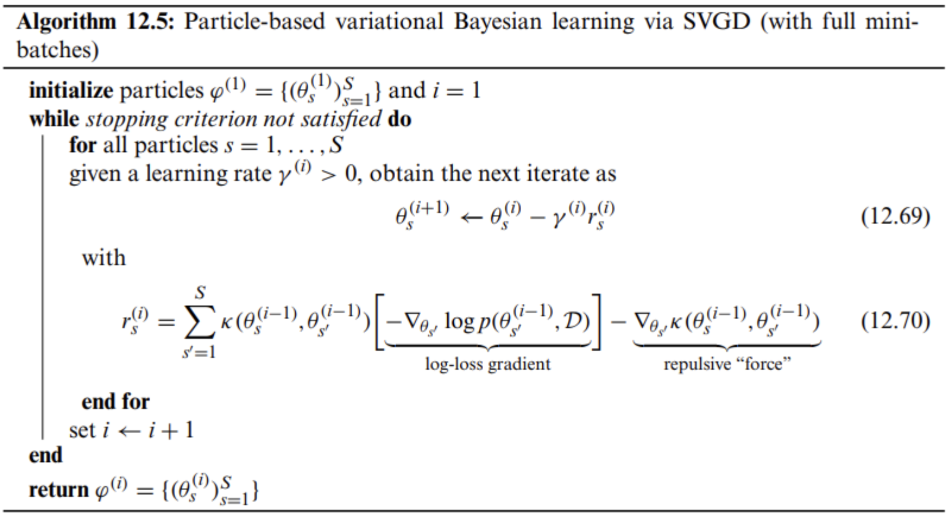
$$\theta' \sim p(\theta) \rightarrow prior$$

$$\theta = G(\theta'|\phi)$$

Particle based Variational Bayesian Learning

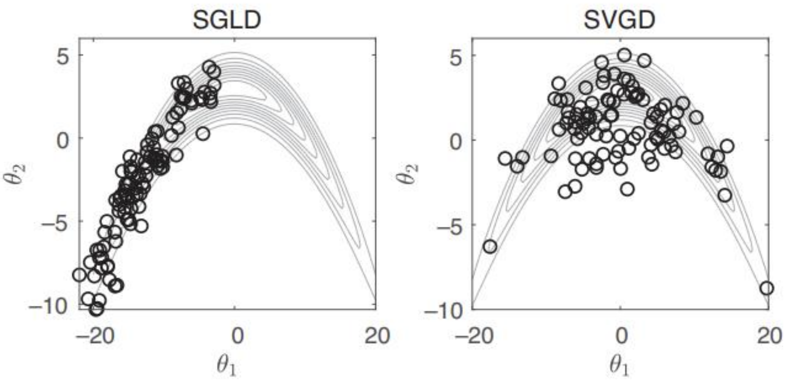
Particle-based variational inference combines sampling and optimization. These methods represent the approximate posterior with a set of particles (samples) and then move the particles to better fit the target. One prominent example is Stein Variational Gradient Descent (SVGD): it treats each particle as a point in parameter space and applies a deterministic update that pushes particles toward high-posterior-density regions while repelling them from each other. In effect, SVGD repeatedly adjusts all particles to minimize the KL divergence to the true posterior. The result is a set of particles that, together, approximate the full posterior. These methods can capture rich distributions because they are nonparametric (the quality depends on the number of particles) but still use gradient-based updates for efficiency.

Basically, instead of a parametric posterior, we use a set of particles $\{\theta^{(s)}\}_{s=1}^{S}$ to approximate the posterior.



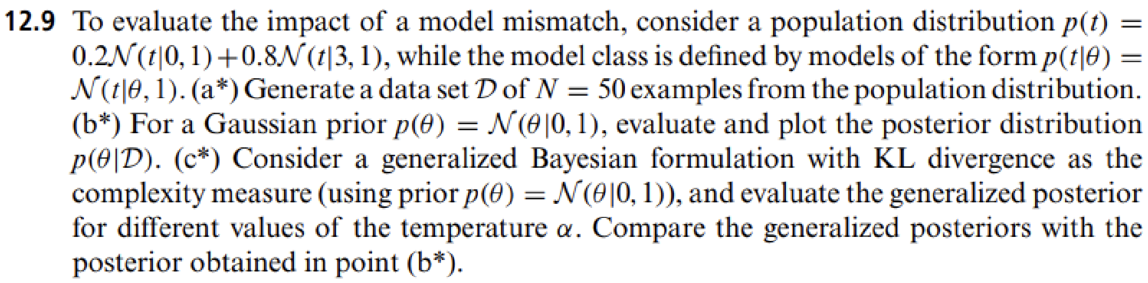
Algorithm 12.5: Particle-based variational Bayesian learning via SVGD (with full mini-batches)

```
initialize particles $\varphi^{(1)} = \{(\theta_s^{(1)})_{s=1}^{S}\}$ and $i = 1$
while stopping criterion not satisfied do
    for all particles $s = 1, \dots, S$
    given a learning rate $\gamma^{(i)} > 0$, obtain the next iterate as
        $\theta_s^{(i+1)} \leftarrow \theta_s^{(i)} - \gamma^{(i)} r_s^{(i)}$    (12.69)
     with
        $r_s^{(i)} = \sum_{s'=1}^{S} \kappa(\theta_s^{(i-1)}, \theta_{s'}^{(i-1)}) \Big[ \underbrace{-\nabla_{\theta_{s'}} \log p(\theta_{s'}^{(i-1)}, \mathcal{D})}_{\text{log-loss gradient}} \Big] - \underbrace{\nabla_{\theta_{s'}} \kappa(\theta_s^{(i-1)}, \theta_{s'}^{(i-1)})}_{\text{repulsive "force"}}$    (12.70)
     end for
    set $i \leftarrow i + 1$
end
return $\varphi^{(i)} = \{(\theta_s^{(i)})_{s=1}^{S}\}$
```

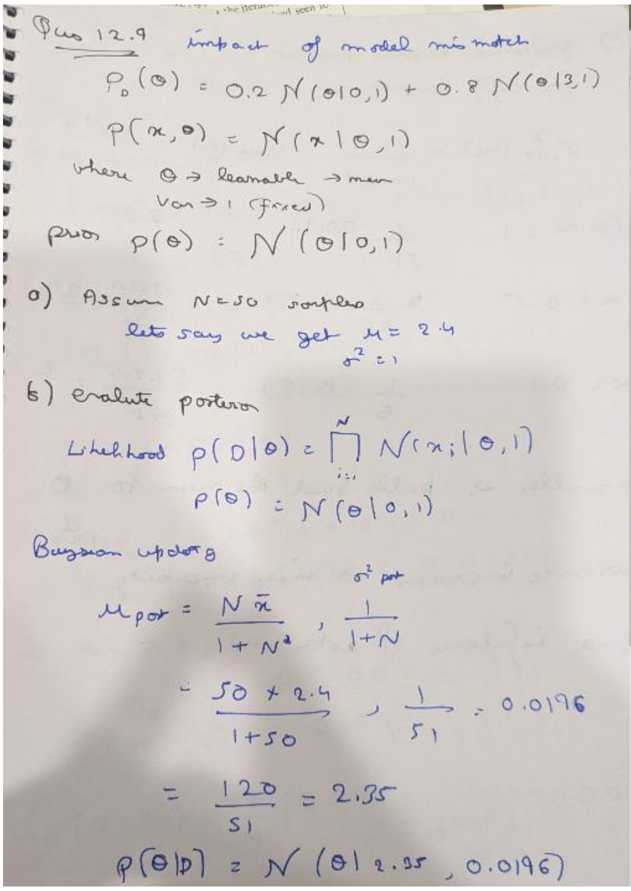


Basically, in SGLD it's like explore a gradient and its noisy from the gradient and it doesn't go past high probability regions whereas in SVGD we push past the high probability zones and explore the entire posterior.

Question 12.9 – Impact of Model mismatch



12.9 To evaluate the impact of a model mismatch, consider a population distribution $p(t) = 0.2\mathcal{N}(t|0,1) + 0.8\mathcal{N}(t|3,1)$, while the model class is defined by models of the form $p(t|\theta) = \mathcal{N}(t|\theta,1)$. (a*) Generate a data set $\mathcal{D}$ of $N = 50$ examples from the population distribution. (b*) For a Gaussian prior $p(\theta) = \mathcal{N}(\theta|0,1)$, evaluate and plot the posterior distribution $p(\theta|\mathcal{D})$. (c*) Consider a generalized Bayesian formulation with KL divergence as the complexity measure (using prior $p(\theta) = \mathcal{N}(\theta|0,1)$), and evaluate the generalized posterior for different values of the temperature α. Compare the generalized posteriors with the posterior obtained in point (b*).



Ques 12.9 impact of model mis match

$$P_D(\theta) = 0.2\,\mathcal{N}(\theta|0,1) + 0.8\,\mathcal{N}(\theta|3,1)$$

$$P(x,\theta) = \mathcal{N}(x|\theta,1)$$

where θ → learnable → mean

Var → 1 (fixed)

prior $p(\theta) = \mathcal{N}(\theta|0,1)$

a) Assume $N=50$ samples

lets say we get $\mu = 2.4$

$\sigma^2 = 1$

b) evalute posterior

Likelihood $p(D|\theta) = \prod_{i=1}^{N} \mathcal{N}(x_i|\theta,1)$

$p(\theta) = \mathcal{N}(\theta|0,1)$

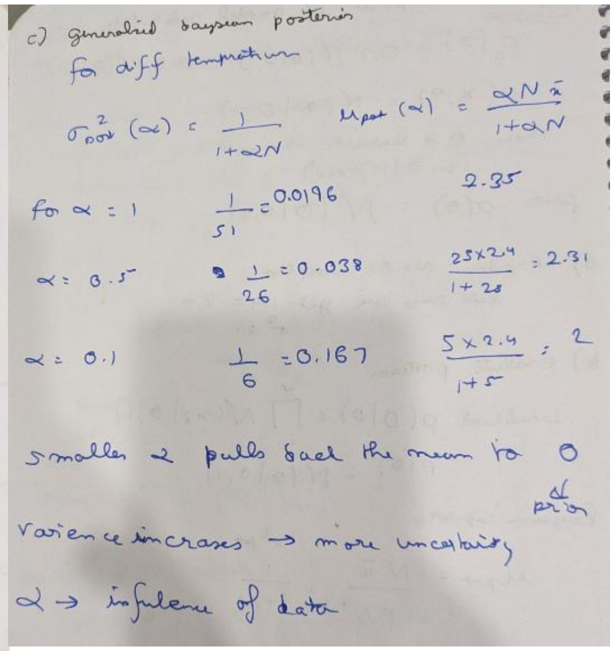
Bayesian updating

$$\mu_{post} = \frac{N\bar{x}}{1+N^2}, \quad \overset{\sigma^2_{post}}{\frac{1}{1+N}}$$

$$= \frac{50 \times 2.4}{1+50}, \quad \frac{1}{51} = 0.0196$$

$$= \frac{120}{51} = 2.35$$

$$p(\theta|D) = \mathcal{N}(\theta|2.35, 0.0196)$$



c) generalized bayesian posteriors for diff temprature

$$\sigma^2_{post}(\alpha) = \frac{1}{1+\alpha N} \qquad \mu_{post}(\alpha) = \frac{\alpha N\bar{x}}{1+\alpha N}$$

for $\alpha = 1$ $\quad \frac{1}{51} = 0.0196 \quad 2.35$

$\alpha = 0.5$ $\quad \frac{1}{26} = 0.038 \quad \frac{25\times2.4}{1+25} = 2.31$

$\alpha = 0.1$ $\quad \frac{1}{6} = 0.167 \quad \frac{5\times2.4}{1+5} = 2$

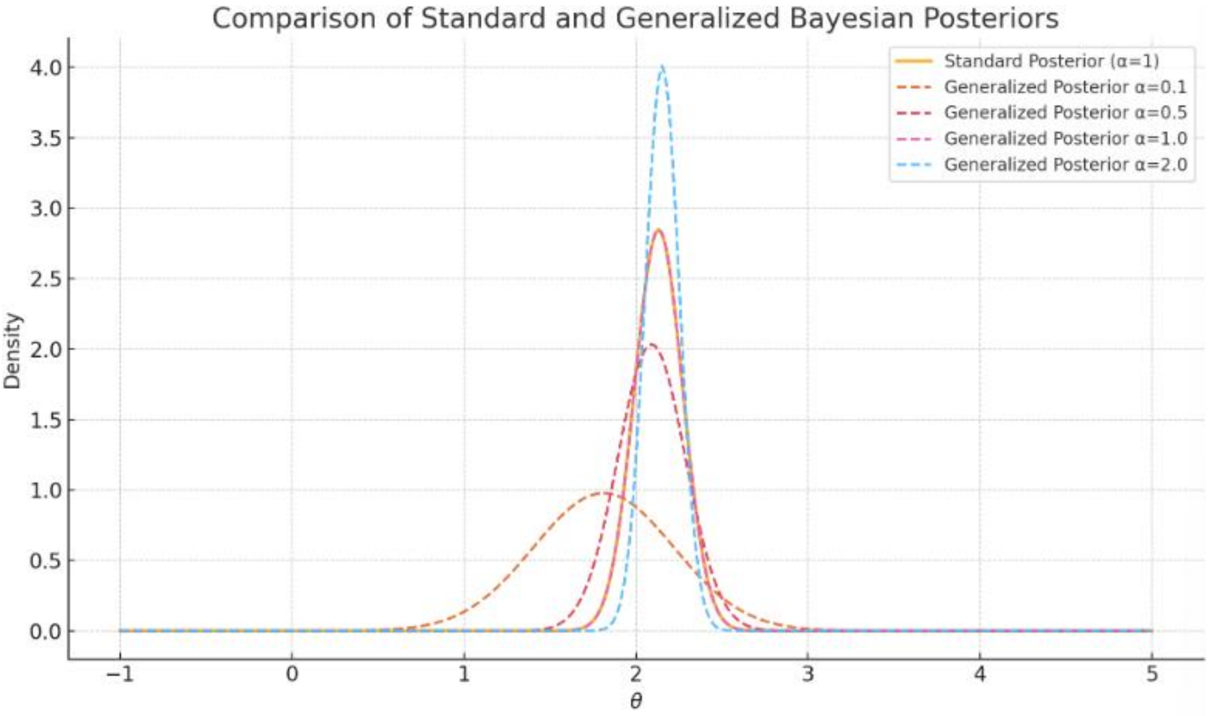
smaller α pulls back the mean to 0 of prior

varience increases → more uncertainty

α → influence of data

As we can see smaller alpha pulls back the mean to 0 -> prior, our variance increases -> more uncertainty, alpha is just the influence n the data as we can see in the below graph for the same question.

We can see less trust on data and the posterior is flatter on alpha = 0.1 whereas as we go up the posterior is sharper at alpha = 2. This shows how temperature scaling affects the robustness and certainty of posterior beliefs when there is a model mismatch.



Bayesian Model Selection

Model Selection without validation: Empirical Bayes

Empirical Bayes is a pragmatic way to set prior hyperparameters using the data itself. Rather than fixing the prior a priori, Empirical Bayes methods estimate the prior distribution (or its parameters) by maximizing the marginal likelihood of the observed data. In effect, one treats the prior's parameters as unknown "hyperparameters" and finds values that make the data most probable. This is akin to Type-II maximum likelihood. It provides a form of automatic model selection: rather than holding out data for validation, we use the same data to tune priors in a hierarchical model. In hierarchical terms, Empirical Bayes approximates a fully Bayesian treatment by plugging in the best hyperparameters instead of integrating them out. For example, in a Gaussian–Gaussian model one can compute the marginal likelihood as a function of the prior variance and choose the value that maximizes it. This often yields results similar to cross-validation but in a principled (albeit approximate) Bayesian way.

We knew in frequentist we require validation which is not required in Bayesian as it can do model selection using just the training data, by computing marginal likelihood. Empirical Bayes takes this idea further by instead of integrating over the full prior, we estimate the hyperparameters from the data – Type II ML.

$$-\log p\left(\mathcal{D} \mid \theta_{\mathcal{D}}^{\mathrm{ML}}\right) = -\log p\left(t_{\mathcal{D}} \mid x_{\mathcal{D}}, \theta_{\mathcal{D}}^{\mathrm{ML}}\right) = -\sum_{n=1}^{N} \log p\left(t_n \mid x_n, \theta_{\mathcal{D}}^{\mathrm{ML}}\right)$$

Marginal likelihood would be:

$$p(\mathcal{D}) = p(t_{\mathcal{D}} \mid x_{\mathcal{D}}) = \int p(\theta) \prod_{n=1}^{N} p(t_n \mid x_n, \theta)\ d$$

How well the model class fits the data, increasing the model capacity doesn't always help, bigger model -> priors get more spaced, less density to well-fitting regions

The problem of maximizing the marginal training likelihood over the model capacity, or more generally any other hyperparameter that determine prior or likelihood is known as empirical bayes or Type II ML

$$p(\mathcal{D}) = \prod_{n=1}^{n} p(t_n \mid \mathcal{D}, t^{n-1}) = \prod_{n=1}^{n} p(t_n \mid x_n, \mathcal{D}^{n-1})$$

Autoregressive decomposition of marginal likelihood

$$-\log p\left(\mathcal{D}\right) = -\sum_{n=1}^{n} \log p\left(t_n \mid x_n, \mathcal{D}^{n-1}\right)$$

Marginal likelihood as a kind of leave-one-out-cross-validation but computed on the fly

$$p(t_n \mid x_n, \mathcal{D}^{n-1}) = E_{\theta \sim p\left(\theta \middle| \mathbb{D}^{\mathbb{m}-\mathbb{1}}\right)}[p(t_n \mid x_n, \theta)]$$

Each predicting term is a Bayesian prediction distribution integrating over the posterior

Gaussian – Gaussian GLM: Empirical Bayes Version

In the Gaussian–Gaussian linear model, Empirical Bayes can be done analytically. One computes the marginal likelihood of the data (integrating out the Gaussian parameter) and finds the best prior variance or signal-to-noise ratio. This ends up relating to classical estimators: for instance, maximizing the marginal likelihood can yield a shrinkage estimator for the regression coefficients (similar to ridge regression), automatically choosing the regularization strength. In essence, Empirical Bayes in this setting lets the data tell you how much to trust the prior vs the observations.

$$t_D = X_D\theta + \mathcal{N}(0, \beta^{-1}I_N) \rightarrow (Linear\ regression\ with\ gaussian\ noise)$$

$$p(\mathcal{D}) = p(\, t_D \mid x_D \,) = \mathcal{N}(\, t_D \mid 0, C_D \,) \ \rightarrow \ marginal\ training\ likelihood$$

$$C_D = \alpha^{-1}X_D X_D^T + \beta^{-1}I_N \rightarrow covarience\ matrix$$

And the marginal training log loss would be:

$$-\log p\,(\mathcal{D}) = -\log p\,(\, t_D \mid x_D \,) = \frac{1}{2} t_D^T C_D^{-1} t_D + \frac{1}{2}\log\det(2\pi C_D)$$

$\frac{1}{2} t_D^T C_D^{-1} t_D$: data fir by mahalanobis distance

$\frac{1}{2}\log\det(2\pi C_D)$: complexity penalty vis determinant of covariance

Laplace Approximation for Empirical Bayes

When the marginal likelihood is not tractable, one can use the Laplace approximation to estimate it. That is, we approximate the integral over parameters by a Gaussian at the MAP (the evidence integral). The Laplace approximation of the marginal likelihood then becomes a criterion to tune hyperparameters. This is a common trick: find hyperparameters that maximize the Laplace-evidence, which is cheaper than full integration. It inherits the limitations of Laplace (e.g. if the posterior is not well-approximated by a Gaussian, the evidence estimate can be off), but it often works well in practice for model selection.

$$p(\theta, \mathcal{D}) \simeq p\big(\theta_{\mathcal{D}}^{\mathrm{MAP}}, \mathcal{D}\big) \cdot \exp\left(\frac{1}{2}\big(\theta - \theta_{\mathcal{D}}^{\mathrm{MAP}}\big)^T \nabla_\theta^2 \left(-\log p\left(\theta_{\mathcal{D}}^{\mathrm{MAP}}, \mathcal{D}\right)\right)\left(\theta - \theta_{\mathcal{D}}^{\mathrm{MAP}}\right)\right)$$

$$-\log p\,(\mathcal{D}) \simeq -\log p\left(\theta_{\mathcal{D}}^{\mathrm{MAP}}, \mathcal{D}\right) + \frac{1}{2}\log\det\left(2\pi\nabla_\theta^2\left(-\log p\left(\theta_{\mathcal{D}}^{\mathrm{MAP}}, \mathcal{D}\right)\right)\right)$$

Generalized Bayesian Learning

“Generalized Bayesian” refers to methods that alter the usual likelihood or prior structure to achieve robustness or computational benefits. For example, one can raise the likelihood to a fractional power (power posteriors) to temper its influence, leading to a generalized posterior that down weights extreme data. Alternatively, one might use alternative divergences (beyond KL) to define the update rule. These generalizations can improve robustness to model misspecification or heavy-tailed data. In essence, generalized Bayes methods extend the standard Bayes rule by modifying its components; the goal is often to make inference more robust or to incorporate regularization directly into the posterior. These ideas go beyond the scope of classical Bayesian inference but are an active research area for practical modelling.

Basically, it’s the optimization of generalized free energy

$$\min_{q(\theta|\mathcal{D})} \left\{N\, E_{\theta\sim q(\theta|\mathbb{D})}[L_D(\theta)] + \alpha \cdot C\big(q(\theta|\mathcal{D})\big)\right\}$$

Generalized posterior

Solving these minimization problems are generalized posteriors. They are

$$q(\theta|\mathcal{D}) \propto p(\theta)\left(\prod_{n=1}^{N} p(t_n|x_n,\theta)\right)^{\frac{1}{\alpha}}$$

Referred to as $(1/\alpha)$ – posterior

- Robustness to mis specified models
- Broader applicability
- Gibbs vs ensemble prediction

We can say that Bayesian can be generalized by allowing arbitrary loss function and complexity regularizes to enhance robustness to misspecifications.