8- Modern Bayesian Inference

Melih Kandemir

University of Southern Denmark
Department of Mathematics and Computer Science (IMADA)
kandemir@imada.sdu.dk

Fall 2022

Objective vs subjective interpretations of probability

Objective interpretation:

$$p(e) = \lim_{n \to +\infty} \frac{n_e}{n}$$

- n_e: Number of times the event of interest occurs
- n: Number of trials

This is the frequentist school.

Subjective interpretation:

- Express your prior belief (hypothesis H) about a possible outcome with a number in the scale $0 =$impossible and $1 =$sure.
- Observe the world via measurements M.
- Update your belief.

This is the Bayesian school.

Bayesian statistics

The Bayes rule is not the product rule itself! It uses the product rule to develop a framework for belief updates on hypotheses.

Figure: Thomas Bayes (1701-1761)

$$p(H|X) = \frac{p(X|H)p(H)}{p(X)}$$

H: Hypothesis
X: Measurement

The belief updating machinery

The Bayes rule:

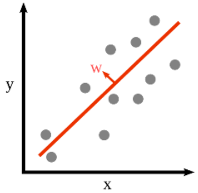
$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

- $x \in \mathcal{X}$ is an observation in the sample space $\mathcal{X}$.
- θ is a set of model parameters. It is an index to a frequentist, and a random variable for a Bayesian.
- $p(x|\theta)$: **likelihood** (how do model parameters describe data?)
- $p(\theta)$: **prior** (what is our prior belief about model parameters?)
- $p(x)$: **evidence** (what is the likelihood of data *regardless of* the model parameters?)
- $p(\theta|x)$: **posterior** (how do model parameters distribute after observations are taken into account?)

Prior? What does it really mean?

Who do you expect to win the tennis game and why?

What does it mean to be Bayesian in machine learning?



Model	Learning
Classical ML $y_n = \mathbf{w}^T \mathbf{x}_n + \epsilon$	$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\mathrm{argmin}}\ \lVert \mathbf{y} - \mathbf{w}^T \mathbf{X} \rVert_2^2$
Probabilistic ML $p(y_n \mid \mathbf{w}, \mathbf{x}_n) = \mathcal{N}(y_n \mid \mathbf{w}^T \mathbf{x}_n, \sigma^2)$	$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\mathrm{argmax}} \prod_{n=1}^{N} p(y_n \mid \mathbf{w}, \mathbf{x}_n)$
Bayesian ML $p(\mathbf{y} \mid \mathbf{w}, \mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mathbf{w}^T \mathbf{x}_n, \sigma^2)$ $p(\mathbf{w}) \leftarrow$ Prior belief	$\underbrace{p(\mathbf{w} \mid \mathbf{X}, \mathbf{y})}_{\text{Posterior}} = \dfrac{\underbrace{p(\mathbf{y} \mid \mathbf{w}, \mathbf{X})}_{\text{Likelihood}} \times \underbrace{p(\mathbf{w})}_{\text{Prior}}}{\underbrace{p(\mathbf{y} \mid \mathbf{X})}_{\text{Evidence}}}$

Motivation 1 for the Bayesian approach

The random variables $(x_1, x_2, \cdots, x_N)$ are exchangeable if for any permutation π, the following equality holds

$$p(x_1, x_2, \cdots, x_N) = p(x_{\pi_1}, x_{\pi_2}, \cdots, x_{\pi_N}).$$

De Finetti's theorem. A sequence of random variables is infinitely exchangeable, i.e. $p(x_1, x_2, \cdots, x_N) = p(x_{\pi_1}, x_{\pi_2}, \cdots, x_{\pi_N})$ iff $\forall N$,

$$p(x_1, x_2, \cdots, x_N) = \int \prod_{i=1}^{N} p(x_i|\theta) P(d\theta)$$

Implications:

- Exchangeability can be checked from right hand side.
- There must exist a parameter θ.
- There must exist a likelihood $p(x|\theta)$.
- There must exist a distribution P on θ.

Motivation 2 for the Bayesian approach

Model averaging. Given a posterior $p(\theta|x)$ and a new observation x^*, the posterior predictive distribution is

$$p(x^*|x) = \int p(x^*|\theta)p(\theta|x)d\theta = \mathbb{E}_{p(\theta|x)}[p(x^*|\theta)]$$

This distribution takes into account all possible values of θ with importance proportional to the probability of their occurrence.

Motivation 3 for the Bayesian approach

Model selection. We are given two hypotheses that claim to explain a certain data set.

Hypothesis 1 ($\mathcal{H}_1$): Likelihood: $p_{\mathcal{H}_1}(x|\theta_1)$, Prior: $p_{\mathcal{H}_1}(\theta_1)$
Hypothesis 2 ($\mathcal{H}_2$): Likelihood: $p_{\mathcal{H}_2}(x|\theta_2)$, Prior: $p_{\mathcal{H}_2}(\theta_2)$

We can alternatively treat the hypothesis as a random variable $\mathcal{H} = \{1, 2\}$ that determines the type of the distribution $p(\cdot)$:

$$p_{\mathcal{H}_1}(x|\theta_1) = p(x|\theta_1, \mathcal{H} = 1), p_{\mathcal{H}_2}(x|\theta_2) = p(x|\theta_2, \mathcal{H} = 2)$$

Let us place a prior on also on the hypothesis variable, e.g. $P(\mathcal{H} = 1) = P(\mathcal{H} = 2)$, and consider all possible model parameter realizations for both hypotheses (i.e. calculate the evidence):

$$p(x|\mathcal{H} = 1) = \int p(x|\theta_1, \mathcal{H} = 1)p(\theta_1|\mathcal{H} = 1)d\theta_1$$
$$p(x|\mathcal{H} = 2) = \int p(x|\theta_2, \mathcal{H} = 2)p(\theta_2|\mathcal{H} = 2)d\theta_2$$

Bayesian model selection

- Apply Bayes theorem to calculate the posterior on hypotheses

$$P(\mathcal{H}|x) = \frac{p(x|\mathcal{H})P(\mathcal{H})}{p(x)}$$

- Choose the hypothesis with higher posterior probability. Compare $p(\mathcal{H}=1|x)$ and $p(\mathcal{H}=2|x)$.
  - Since $p(x)$ does not depend on $\mathcal{H}$, its magnitude does not have an effect on the comparison.
  - Since we chose a **uniform prior** on the hypotheses ($P(\mathcal{H}=1) = P(\mathcal{H}=2)$), the magnitude of $P(\mathcal{H})$ also does not have an effect.
- Hence, it suffices to calculate $p(x|\mathcal{H}=1)/p(x|\mathcal{H}=2)$. This metric is called the **Bayes factor** [Kass and Raftery, 1995]. Choose $\mathcal{H}_1$ if Bayes factor is greater than 1, choose $\mathcal{H}_2$ otherwise.
- The model **evidence** serves as a **quantitative** score for model selection in the Bayesian setting.

MLE vs MAP

Given observed data X and the assumption that $X \sim p(X|\theta)$, the **Maximum Likelihood Estimate (MLE)** is

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \; p(X|\theta) = \underset{\theta}{\text{argmax}} \;\; \log \; p(X|\theta)$$

Since $\log(\cdot)$ is monotonically increasing

$$\begin{aligned} \underset{w}{\text{argmax}} \; \log \; p(w|X,y) &= \underset{w}{\text{argmax}} \; \log \frac{p(y|w,X)p(w)}{p(y|X)} \\ &= \underset{w}{\text{argmax}} \; \log \; p(y|w,X) + \log p(w) \underbrace{- \log p(y|X)}_{\text{const}} \\ &= \underset{w}{\text{argmax}} \; \log \; p(y|w,X) + \log p(w) \end{aligned}$$

The found mode is called the **Maximum A Posteriori (MAP)** estimate of the model. This is a technique a true Bayesian largely avoids, though there are specific cases where it is useful.

What are priors for?

- To incorporate prior beliefs
- To avoid overfitting
  - Controlling model complexity:
    1. inducing sparsity=regularization
    2. marginal likelihood
  - Marginalization of model parameters (represented as a distribution, not a point estimate.
- To attain posterior uncertainty, which is essential for
  - active learning
  - decision making (medicine, finance, etc.)

Types of priors [1]

- **Non-informative priors:** Priors that allow the model and the data speak for themselves.
- **Informative priors:** Priors that reflect beliefs. They are subjective but not arbitrary.
- **Hierarchical priors:** Multiple levels of priors

  $$p(\theta) = \int p(\theta|\alpha)p(\alpha)d\alpha$$

  where $p(\alpha)$ is called a *hyperprior*.
- **Empirical priors:** Learn some of the parameters of the prior from the data (i.e. Empirical Bayes!)

[1] Z. Ghahramani's lecture

Empirical priors [2]

- **Given:**

$$p(\mathcal{D}|\alpha) = \int p(\mathcal{D}|\theta)p(\theta|\alpha)d\theta$$

  where α is the vector of *hyperparameters*.

- **Estimation:**

$$\hat{\alpha} = \arg\max_{\alpha} \; p(\mathcal{D}|\alpha)$$

  This method is called *Type II Maximum Likelihood*.

- **Prediction:**

$$p(x^*|\mathcal{D}, \hat{\alpha}) = \int p(x|\theta)p(\theta|\mathcal{D}, \hat{\alpha})d\theta$$

- **Plus:** Tuning the prior belief to data.
- **Minus:** Double accounting of data $\rightarrow$ Overfitting.

[2] Z. Ghahramani's lecture

Yin-Yang in statistics

Regularization: Frequentist way of being Bayesian
Non-informative priors: Bayesian way of being Frequentist

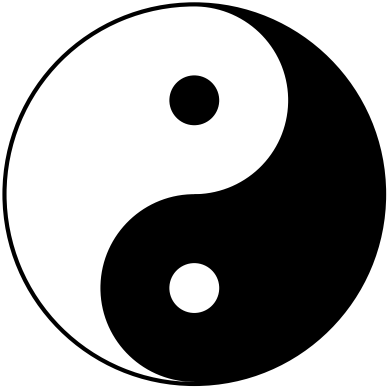


Image: `https://en.wikipedia.org/wiki/Yin_and_yang`

Regularization in linear regression

We intend to have this to satisfy the Occam's razor principle:

$$\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} \Big(y_n - \sum_{d=1}^{D} w_d x_{nd}\Big)^2 \text{subject to } \sum_{d=1}^{D} w_d^2 < t$$

which amounts to this

$$\underset{w}{\operatorname{argmin}}\ \underset{\lambda}{\operatorname{argmax}} \sum_{n=1}^{N} \Big(y_n - \sum_{d=1}^{D} w_d x_{nd}\Big)^2 + \lambda\Big(\sum_{d=1}^{D} w_d^2 - t\Big)$$

But in practice we simplify by this

$$\underset{w}{\operatorname{argmin}}\ \cancel{\underset{\lambda}{\operatorname{argmax}}} \sum_{n=1}^{N} \Big(y_n - \sum_{d=1}^{D} w_d x_{nd}\Big)^2 + \lambda\Big(\sum_{d=1}^{D} w_d^2 - \cancel{t}\Big)$$

and solve this: $\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} \Big(y_n - w^T X_n\Big)^2 + \lambda||w||_2^2$.

l_2-norm regularization $\rightarrow$ ridge regression

$$\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} \Big(y_n - w^T X_n \Big)^2 + \lambda \underbrace{||w||_2^2}_{w^T w}$$

Solution:

$$\begin{aligned}
\nabla_w \Big\{ (y - Xw)^T (y - Xw) + \lambda w^T w \Big\} &= 0 \\
w^T X^T X w - 2 y^T X w + y^T y + \lambda w^T w &= 0 \\
2 X^T X w - 2 X^T y + 2 \lambda w &= 0 \\
(X^T X + \lambda I) w &= X^T y \\
(X^T X + \lambda I)^{-1} X^T y &= \hat{w}
\end{aligned}$$

Thanks to λI, the matrix inverse exists even though $N < D$.

Conjugacy

If $p(\theta|\mathcal{D})$ is in the same family as $p(\theta)$, then $p(\theta)$ is called a **conjugate prior** for $p(\mathcal{D}|\theta)$. **Example.** Normal distribution with known variance

$$\begin{aligned} x_1, \cdots, x_N|\mu, \sigma^2 &\sim \mathcal{N}(x|\mu, \sigma^2) \\ \mu &\sim \mathcal{N}(\mu|\mu_0, \sigma_0^2) \\ \sigma^2 &\rightarrow known \end{aligned}$$

The posterior is also normal distributed:

$$p(\mu|x, \sigma^2) = \mathcal{N}\left(\mu \left| \frac{\frac{\sum_{i=1}^N x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}}, \frac{N}{\sigma^2} + \frac{1}{\sigma_0^2} \right.\right).$$

Bayesian linear regression

$$y, X|w \sim \prod_{n=1}^{N} \mathcal{N}(y_n|w^T X_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I),$$
$$w \sim \mathcal{N}(w|0, \alpha^{-1}I).$$

We would like to infer $p(w|X, y)$.

$$\begin{aligned}
&\log p(w|X, y) \\
&\quad = -\frac{\alpha}{2}w^T w - \frac{1}{2}(y - Xw)^T(\beta^{-1}I)^{-1}(y - Xw) + \text{const} \\
&\quad = -\frac{\alpha}{2}w^T w - \frac{\beta}{2}(y^T - w^T X^T)(y - Xw) + \text{const} \\
&\quad = -\frac{\alpha}{2}w^T w - \frac{\beta}{2}w^T X^T X w + \beta y^T X w + \text{const} \\
&\quad = -\frac{1}{2}w^T \underbrace{\left(\alpha I + \beta X^T X\right)}_{\Sigma_p^{-1}} w + \underbrace{\beta y^T X}_{\mu_p^T \Sigma_p^{-1}} w + \text{const.}
\end{aligned}$$

Bayesian linear regression cont'd

Hence, $\Sigma_p = \Big(\alpha I + \beta X^T X\Big)^{-1}$ and $\mu_p = \Sigma_p(\beta X^T y)$. Then the posterior reads

$$p(w|X, y) = \mathcal{N}(w|\mu_p, \Sigma_p),$$

where $\Sigma_p = \Big(\alpha I + \beta X^T X\Big)^{-1}$ and $\mu_p = \beta \Sigma_p X^T y$.
Take a closer look at the posterior mean:

$$\begin{aligned}
\mu_p = \beta \Sigma_p X^T y &= \beta \Big(\alpha I + \beta X^T X\Big)^{-1} X^T y \\
&= \beta \left(\beta \Big(\frac{1}{\beta}\alpha I + \frac{1}{\beta}\beta X^T X\Big)\right)^{-1} X^T y \\
&= \left(\frac{\alpha}{\beta} I + X^T X\right)^{-1} X^T y.
\end{aligned}$$

This is the solution of the **ridge regression** with regularization parameter set to $\frac{\alpha}{\beta}$!

MAP Example

Let us take again the Bayesian linear regression case.

$$y|w, X \sim \mathcal{N}(y|Xw, \beta^{-1}I),$$
$$w \sim \mathcal{N}(w|0, \alpha^{-1}I).$$

Our aim is to solve

$$\begin{aligned}
&\underset{w}{\operatorname{argmax}}\ \log \mathcal{N}(y|Xw, \beta^{-1}I) + \log \mathcal{N}(w|0, \alpha^{-1}I) \\
&= \underset{w}{\operatorname{argmax}} \left\{ -\frac{\beta}{2}(y - Xw)^T(y - Xw) - \frac{\alpha}{2} w^T w \right\} \\
&= \underset{w}{\operatorname{argmax}} \left\{ -\frac{1}{2} w^T \Big(\beta X^T X + \alpha I \Big) w + \beta y^T X w \right\}
\end{aligned}$$

MAP Example

Set the gradient of the variable of interest to zero:

$$\nabla_w \left\{ -\frac{1}{2} w^T \Big(\beta X^T X + \alpha I\Big) w + \beta y^T X w \right\} \triangleq 0$$
$$-\Big(\beta X^T X + \alpha I\Big) w + \beta X^T y \triangleq 0$$

Solving for w and rearranging β in the same way as above gives

$$\hat{w} \triangleq \left(X^T X + \frac{\alpha}{\beta} I \right)^{-1} X^T y.$$

Once more we recapitulate the ridge regression.

Calculus of variations

Typically we have scalars or vectors as variables. Then we operate on mappings from these variables to other entities. For instance in $f(X) : \mathbb{R}^D \rightarrow \mathbb{R}$, the vector X is our variable of interest and $f(\cdot)$ is a *function* of it.

There are some cases where we take *functions* as *variables* of interest and operate on mappings from functions to other entities:

$$\mathbb{F} : f(X) \rightarrow \mathbb{R}.$$

Such mappings are called **functionals**. One example is the *KL divergence*. The branch of mathematics that has functionals in its focus is named as the *calculus of variations*.

What if we have non-conjugate priors?

Assume we are given a data set $X = \{X_1, \cdots, X_N\}$ and a Bayesian model

$$X|\theta \sim \prod_{n=1}^{N} p(X_n|\theta),$$
$$\theta \sim p(\theta).$$

with a non-conjugate prior $p(\theta)$ on the set of **latent variables** wrt likelihood $p(X_n|\theta)$. We are interested in the posterior

$$p(\theta|X)$$

for which an analytical expression is not available. What shall we do then?

Approximating the posterior

Choose a $q(\theta|\Omega)$, a density parameterized by Ω, and construct an optimization problem to make $q(\theta|\Omega)$ as similar as possible to the true posterior $p(\theta|X)$.

This does not solve

$$D_{KL}[p(\theta|X)||q(\theta|\Omega)] = \int p(\theta|X) \log \frac{p(\theta|X)}{q(\theta|\Omega)} d\theta.$$

because the loss function depends on $p(\theta|X)$, which we do not know. Try the other way around.

Variational Bayes

$$\begin{aligned}
D_{KL}[q(\theta|\Omega)||p(\theta|X)] &= \int q(\theta|\Omega) \log \frac{q(\theta|\Omega)}{\underbrace{p(\theta|X)}_{\frac{p(\theta,X)}{p(X)}}} d\theta \\
&= \int q(\theta|\Omega) \log \frac{q(\theta|\Omega)p(X)}{p(\theta,X)} d\theta \\
&= \int q(\theta|\Omega) \log q(\theta|\Omega) d\theta \\
&+ \int q(\theta|\Omega) \log p(X) d\theta \\
&- \int q(\theta|\Omega) \log p(\theta,X) d\theta
\end{aligned}$$

Variational Bayes

$$D_{KL}[q(\theta|\Omega)||p(\theta|X)] = \underbrace{\mathbb{E}_{q(\theta|\Omega)}[\log q(\theta|\Omega)]}_{-\mathbb{H}_{q(\theta|\Omega)}[\theta]} + \underbrace{\mathbb{E}_{q(\theta|\Omega)}[\log p(X)]}_{\log p(X)}$$
$$- \mathbb{E}_{q(\theta|\Omega)}[\log p(\theta, X)]$$

Arranging the terms, we get the interesting outcome below

$$\underbrace{\log p(X)}_{\text{const}} = \underbrace{\mathbb{E}_{q(\theta|\Omega)}[\log p(\theta, X)] + \mathbb{H}_{q(\theta|\Omega)}[\theta]}_{\mathcal{L}} + \underbrace{D_{KL}[q(\theta|\Omega)||p(\theta|X)]}_{\geq 0}.$$

As $\mathcal{L}$ is a lower bound on the $\log$ evidence, it is called the **Evidence Lower Bound (ELBO)**. ELBO equals to the log-evidence iff $q(\theta|\Omega) = p(\theta|X)$.

Inference as optimization

Let us take a closer look at the generic form and contemplate on the feasibility of the approach

$$\begin{aligned}
&\arg\max_{q(\theta|\Omega)} \mathcal{L}(\Omega) \\
&= \arg\max_{\Omega} \left\{ \sum_{n=1}^{N} \mathbb{E}_{q(\theta|\Omega)}[\log p(X_n|\theta)] + \mathbb{E}_{q(\theta|\Omega)}[\log p(\theta)] + \mathbb{H}_{q(\theta|\Omega)}[\theta] \right\} \\
&= \arg\max_{\Omega} \left\{ \sum_{n=1}^{N} \underbrace{\mathbb{E}_{q(\theta|\Omega)}[\log p(X_n|\theta)]}_{\text{Data fit}} - \underbrace{D_{KL}[q(\theta|\Omega)||p(\theta)]}_{\text{Complexity penalizer}} \right\}
\end{aligned}$$

Calculate $\mathbb{E}_{q(\theta|\Omega)}[\log p(X_n|\theta)]$ and look up $\mathbb{H}_{q(\theta|\Omega)}[\theta]$ or alternatively $D_{KL}[q(\theta|\Omega)||p(\theta)]$. Take the gradient of the ELBO wrt Ω and optimize. Choosing $(\theta) = \prod_{i \in \mathcal{P}} q(\theta_i)$ is called **mean-field** variational Bayes. This is in contrast to **structured** variational Bayes.

Bayesian Neural Nets (BNN)

Given data $\mathcal{D} = \{(x_n, y_n) | n = 1, \ldots, N\}$, a **Bayesian neural net** is defined as the data generating process below

$$p(\mathcal{D}|\theta) = \prod_{n=1}^{N} \mathcal{N}(y_n | f_\theta(x_n), g_\theta(x_n))$$
$$p(\theta) = \mathcal{N}(\theta | 0, \kappa^{-1} I),$$

where

$$f_\theta(x) = W_2^T \sigma(W_1^T x), \qquad g_\theta(x) = \exp(W_3^T \sigma(W_1^T x)),$$

$\kappa \in \mathbb{R}_+$ and $\theta = \{W_1, W_2, W_3\}$. Such weight sharing is called the **head-split** design. It is still a BNN when the design of the likelihood function or the architectures of the neural nets f, g changes.

Variational inference of BNNs

Make the mean-field assumption for simplicity and choose the variational distribution below

$$q(\theta; \Omega) = \prod_{j \in \theta} \mathcal{N}(\theta_j | m_j, s_j^2)$$

where $\Omega = \{m, S\}$ with $m = \{m_j | j \in \theta\}$ and $S = \{s_j^2 | j \in \theta\}$. Then

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \mathbb{E}_{\mathcal{N}(\theta|m,S)}[\log \mathcal{N}(y_n | f_\theta(x_n), g_\theta(x_n))] \\ - D_{KL}(\mathcal{N}(\theta|m,S) || \mathcal{N}(\theta|0, \kappa^{-1} I))$$

Deep dive into the KL penalizer

$$
\begin{aligned}
D_{KL}(\mathcal{N}(\theta|m,S)||\mathcal{N}(\theta|0,\kappa^{-1}I)) &= \int \log \frac{\mathcal{N}(\theta|m,S)}{\mathcal{N}(\theta|0,\kappa^{-1}I)} \mathcal{N}(\theta|m,S) d\theta \\
&= \int \sum_{j \in \theta} \log \frac{\mathcal{N}(\theta_j|m,S)}{\mathcal{N}(\theta_j|0,\kappa^{-1}I)} \prod_{j \in \theta} \mathcal{N}(\theta_j|m_j,s_j^2) d\theta_1, \dots \theta_{|\theta|} \\
&= \sum_{j \in \theta} \int \log \frac{\mathcal{N}(\theta_j|m,S)}{\mathcal{N}(\theta_j|0,\kappa^{-1}I)} \mathcal{N}(\theta_j|m_j,s_j^2) d\theta_j \\
&= \sum_{j \in \theta} D_{KL}(\mathcal{N}(\theta_j|m_j,s_j^2)||\mathcal{N}(\theta_j|0,\kappa^{-1}I)) \\
&= \sum_{j \in \theta} \left\{ \log \left(\frac{\kappa^{-1}}{s_j^2} \right) + \frac{s_j^2 + m_j^2}{2\kappa^{-1}} - \frac{1}{2} \right\}
\end{aligned}
$$

Deep dive into the data fit term

Use log-scale reparameterization $\mathcal{N}(\theta|m, S)$ as

$$\epsilon \sim \mathcal{N}(0, I), \quad \theta = m + \sqrt{S}\epsilon.$$

$$\begin{aligned}
&\mathbb{E}_{\theta \sim \mathcal{N}(\theta|m,S)} \left[\sum_{n=1}^{N} \log \mathcal{N}(y_n | f_{m+\sqrt{S}\epsilon}(x_n), g_{m+\sqrt{S}\epsilon}(x_n)) \right] \\
&= -\mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} \left[\frac{1}{2} \log g_{m+\sqrt{S}\epsilon}(x_n) + \frac{1}{2 g_{m+\sqrt{S}\epsilon}(x_n)} (y_n - f_{m+\sqrt{S}\epsilon}(x_n))^2 \right] \\
&\qquad - \frac{1}{2} \log(2\pi)
\end{aligned}$$

where the last term is constant wrt m, S and the expectaction can now be taken simply by MC integration.

Local reparameterization

Plain MC may incur high estimator variance in calculation of the term

$$-\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\sum_{n=1}^{N}\frac{1}{2}\log g_{m+\sqrt{S}\epsilon}(x_n)+\frac{1}{2g_{m+\sqrt{S}\epsilon}(x_n)}(y_n-f_{m+\sqrt{S}\epsilon}(x_n))^2\right]$$

as a single sample ϵ is passed on to the data fit terms of all data points. We can reduce its variance if we can sample for each data point separately.

Denote by $h_n := [h_n^1, \ldots, h_n^K]$ the activation map of an intermediate layer of a neural net consisting of K neurons for data point x_n and by $q(w) := \prod_j \mathcal{N}(w_j^r | m_j^r, (s_j^r)^2)$ the approximate weight posterior of the rth neuron of that layer. We are interested in the following intermediate random variable in the MC integration process: $v_n^r = w_r^T h_n, w_r \sim q(w_r)$. Affine transform of a normal distribution is another normal, hence

$$v_n^r = w^T h_n \sim \mathcal{N}\left(\sum_{j=1}^{K} m_j h_n^j, \sum_{j=1}^{K}(h_n^j)^2 s_j^2\right).$$

Local reparameterization

Apply the log-scale reparameterization on the **local** (i.e. data point specific) random variable v_n

$$\epsilon_n^r \sim \mathcal{N}(0,1), \qquad v_n^r = \sum_{j=1}^{K} m_j^r h_n^j + \epsilon_n^r \sqrt{(h_n^j)^2 s_j^2}$$

Then compute the activation map of the next layer as

$$h' := \sigma([v_n^1, \ldots, v_n^{K'}])$$

for each of its K' neurons. Then repeat the same process until the output layer is reached. This is called the **local reparameterization trick**[3].

[3] `https://arxiv.org/abs/1506.02557`

Gaussian dropout

Consider plain dropout for dropout rate $\rho \in (0,1)$:

$$z_n^r \sim \mathrm{Bernoulli}(1-\rho), \qquad v_n^r = \frac{1}{(1-\rho)} \sum_{j=1}^{K} w_j h_n^j z_n^r.$$

Here v_n^r is sum of K independent random variables and K is typically large. Due to the Central Limit Theorem, v_n^r will be approximately normal distributed. Calculate its first two moments and sample from the resulting normal distribution. This is called **Gaussian dropout**:

$$\mathbb{E}[v_n^r] = \frac{1}{(1-\rho)} \sum_{j=1}^{K} w_j h_n^j \mathbb{E}[z_n^r] = \sum_{j=1}^{K} w_j h_n^j,$$

$$\mathrm{Var}[v_n^r] = \sum_{j=1}^{K} \mathrm{Var}\left[\frac{1}{(1-\rho)} w_j (h_n^j) z_n^r\right] = \frac{1}{(1-\rho)} \sum_{j=1}^{K} w_j^2 (h_n^j)^2 \mathrm{Var}[z_n^r]$$

$$= \frac{1}{(1-\rho)^2} \sum_{j=1}^{K} w_j^2 (h_n^j)^2 \rho(1-\rho) = \frac{\rho}{1-\rho} \sum_{j=1}^{K} w_j^2 (h_n^j)^2.$$

Variational dropout

Compare the resulting distribution

$$p(v_n^r) \approx \mathcal{N}\left(\sum_{j=1}^{K} w_j h_n^j, \frac{\rho}{1-\rho}\sum_{j=1}^{K} w_j^2 (h_n^j)^2\right).$$

to what we got for variational inference of BNNs

$$v_n^r = w^T h_n \sim \mathcal{N}\left(\sum_{j=1}^{K} m_j h_n^j, \sum_{j=1}^{K} (h_n^j)^2 s_j^2\right).$$

Match $m_j = w_j$ and $\frac{\rho}{(1-\rho)} w_j^2 = \alpha m_j^2 = s_j^2$ where $\alpha = \frac{\rho}{(1-\rho)}$. Hence

$$\log \alpha = \log s_j^2 - \log m_j^2,$$

which is a commonplace quantity to set thresholds for pruning a synaptic connection. This means mean-field variational inference of a BNN corresponds to learning an individual dropout rate for each neuron! That is why it is also referred to as **variational dropout**.

Torch implementation

```python
class VBLinear(torch.nn.Module):
    def __init__(self, in_features, out_features):
        super(VBLinear, self).__init__()
        self.n_in = in_features; self.n_out = out_features
        self.prior_prec = 10
        self.bias = torch.nn.Parameter(torch.Tensor(out_features))
        self.mu_w = torch.nn.Parameter(torch.Tensor(out_features, in_features))
        self.logsig2_w = torch.nn.Parameter(torch.Tensor(out_features, in_features))
        self.reset_parameters()

    def reset_parameters(self):
        stdv = 1.0 / torch.sqrt(self.mu_w.size(1))
        self.mu_w.data.normal_(0, stdv)
        self.logsig2_w.data.zero_().normal_(-9, 0.001)
        self.bias.data.zero_()

    def KL(self, loguniform=False):
        logsig2_w = self.logsig2_w.clamp(-11, 11)
        kl = (0.5* (self.prior_prec * (self.mu_w.pow(2)
            + logsig2_w.exp()) - logsig2_w - 1- torch.log(self.prior_preci)).sum())
        return kl

    def forward(self, input):
        mu_out = torch.nn.functional.linear(input, self.mu_w, self.bias)
        s2_w = self.logsig2_w.clamp(-11, 11).exp()
        var_out = torch.nn.functional.linear(input.pow(2), s2_w) + 1e-8
        return mu_out + var_out.sqrt() * torch.randn_like(mu_out)
```

Variational Auto-Encoders (VAEs) [4]

Consider the unlabeled data set $X = \{x_1, \ldots, x_N\}$. Assume it follows the generating process below

$$
\begin{aligned}
p(z_n) &= \mathcal{N}(z_n|0, I), \qquad\qquad \forall n = 1, \ldots, N \\
p(x_n|z_n) &= \mathcal{N}(x_n|f_\theta(z_n), g_\theta(z_n)),
\end{aligned}
$$

where $z_n \in \mathbb{R}^D$ is a latent representation of observation x_n. The true posterior factorizes across data points

$$
\begin{aligned}
p(Z|X) &= \frac{\prod_{n=1}^N p(x_n|z_n)p(z_n)}{\int \prod_{n=1}^N p(x_n|z_n)p(z_n) dz_1, \ldots, dz_n} \\
&= \frac{\prod_{n=1}^N p(x_n|z_n)p(z_n)}{\prod_{n=1}^N \int p(x_n|z_n)p(z_n) dz_n} = \prod_{n=1}^N \frac{p(x_n|z_n)p(z_n)}{p(x_n)} \\
&= \prod_{n=1}^N p(z_n|x_n).
\end{aligned}
$$

[4] https://arxiv.org/abs/1312.6114

Amortization

Reflect the factorization of the true posterior to variational distribution:

$$q(Z|X) = \prod_{n=1}^{N} q(z_n|\Omega_n).$$

This neat factorization comes at the expense of the parameter size to grow proportional to N. Assume $q(z_n|\Omega_n) = \mathcal{N}(m_n, s_n^2)$, then we have $2N$ free parameters for a data set with N data points! Inspire by the fact that $q(z_n|\Omega_n) \approx p(z_n|x_n)$ and do

$$q(z_n; \omega, x_n) = \mathcal{N}(h_\omega(x_n), v_\omega(x_n)).$$

This way we use the observed sample to obtain a parametric expression of the variational posterior. This technique has three names in the literature:

- **amortization** (arguably the most widespread one)
- **inference networks**
- **recognition models**

The VAE ELBO

$$\mathcal{L}(\omega) = \sum_{n=1}^{N} \left\{ \mathbb{E}_{q(z_n;\omega,x_n)}[\log p(x_n|z_n)] - D_{KL}(q(z_n;\omega,x_n)||p(z_n)) \right\}$$
$$= \sum_{n=1}^{N} \left\{ \mathbb{E}_{\mathcal{N}(h_\omega(x_n),v_\omega(x_n))}[\log p(x_n|z_n)] \right.$$
$$\left. - D_{KL}(\mathcal{N}(h_\omega(x_n),v_\omega(x_n))||\mathcal{N}(0,I)) \right\}$$

We apply once again the log-scale reparameterization

$$\epsilon_n \sim N(0,I), \quad z_n = h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)}.$$

Thanks to the factorized posterior, our reparameterization is already local. No further tricks required.

Why is VAE an auto-encoder?

$$\mathcal{L}(\omega) = \sum_{n=1}^{N} \Bigg\{ \mathbb{E}_{\epsilon_n \sim \mathcal{N}(0,I)} \Bigg[\log \mathcal{N}\Bigg(x_n \Bigg| f_\theta(h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)}), g_\theta(h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)}) \Bigg) \Bigg] - D_{KL}(\mathcal{N}(h_\omega(x_n), v_\omega(x_n)) || \mathcal{N}(0, I)) \Bigg\}$$

Variational because of the inference technique, auto-encoder because

- The inference network $h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)}$ maps from observation space to latent space, hence **encodes**.
- The likelihood networks $f_\theta(\cdot), g_\theta(\cdot)$ map from the latent space to observation space, hence **decode**.

The VAE loss in full details

$$
\begin{aligned}
\arg\min_{\omega} \sum_{n=1}^{N} \mathbb{E}_{\epsilon_n \sim \mathcal{N}(0,I)} \Bigg[& \log g_\theta\Big(h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)}\Big) \\
& + \frac{\big(y_n - f_\theta(h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)})\big)^2}{g_\theta\Big(h_\omega(x_n) + \epsilon_n \sqrt{v_\omega(x_n)}\Big)} \\
& + \sum_{j=1}^{D} \Big[v_\omega^j(x_n) + h_\omega^j(x_n)^2 - 2\log(v_\omega^j(x_n)) \Big] \Bigg]
\end{aligned}
$$

This is the negative ELBO, which is sometimes referred to as the **Variational Free Energy (VFE)**. The index j runs over the latent space dimensions.