Introduction to Bayesian methods and Bayesian Neural Networks

Tomasz Kuśmierczyk

2024-05-29 & 2024-06-12

This research is part of the project No. ***2022/45/P/ST6/02969*** *co-funded by the National Science Centre and the European Union Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie grant agreement No. 945339. For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission;*

Part 1

Part 1: Introduction to Bayesian methods

Coin tossing example

Task: for a biased (=unfair) coin, find the probability of heads θ :

Coin tossing example

Task: for a biased (=unfair) coin, find the probability of heads θ :

1. Observations $\mathcal{D} = \{H, H, H, T, H, H, T, H, H, H, H, T\}$

Coin tossing example

Task: for a biased (=unfair) coin, find the probability of heads θ :

2. Observations $\mathcal{D} = \{H, H, H, T, H, H, T, H\}$

Coin tossing example

Task: for a biased (=unfair) coin, find the probability of heads θ :

3. Observations $\mathcal{D} = \{H, H, H, T\}$

Coin tossing example

Task: for a biased (=unfair) coin, find the probability of heads θ :

4. Observations $\mathcal{D} = \{T, T\}$

Maximum Likelihood Estimate

1. Specify model.
2. Select loss.
3. Find parameters minimizing loss (=maximizing data likelihood).

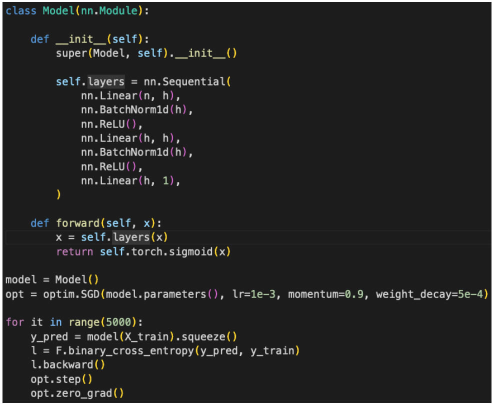
Coin tossing example:

- Parameter: θ
- Option 1: MSE loss: $\mathbb{L}(D, \theta) = \sum_{y \in \mathcal{D}} (y - \theta)^2$
- Option 2: Bernoulli negative log-likelihood: $\mathbb{L}(D, \theta) = -\sum_{y \in D} \log p(y|\theta)$ where $p =$ Bernoulli pmf
- Solution: $\hat{\theta} = \text{argmin}_\theta \mathbb{L}(D, \theta)$
  $\implies \hat{\theta} = 0.75$ for 1.,2.,3.
  $\implies \hat{\theta} = 0.0$ for 4.

Few problems with MLE

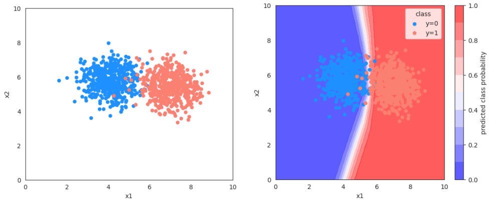
- **Overfitting with Small Sample Size**: If we toss the coin a very small number of times (e.g., once or twice), MLE can give misleading results.
- **Bias in Estimation with Limited Data**: MLE is highly sensitive to the sample size. With few observations, the estimation can be heavily biased.
- **Zero Probability Issue**: If an outcome is not observed in the sample, MLE assigns it a probability of zero.
- **Doesn't Account for Prior Knowledge**: MLE only uses the observed data and does not incorporate any prior knowledge or beliefs.
- **Variance in Estimates**: The variance of the MLE estimate is high for small sample sizes, leading to unstable predictions.
- **Sensitivity to Outliers**: Outliers or rare events can disproportionately affect the MLE.

Neural Network Classifier (likelihood=Bernoulli) predicting $p(y = 1|x, \mathcal{D}) = 1 - p(y = 0|x, \mathcal{D})$



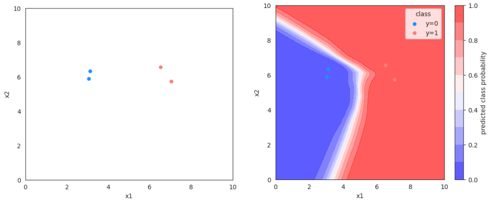
```
class Model(nn.Module):

    def __init__(self):
        super(Model, self).__init__()

        self.layers = nn.Sequential(
            nn.Linear(n, h),
            nn.BatchNorm1d(h),
            nn.ReLU(),
            nn.Linear(h, h),
            nn.BatchNorm1d(h),
            nn.ReLU(),
            nn.Linear(h, 1),
        )

    def forward(self, x):
        x = self.layers(x)
        return self.torch.sigmoid(x)

model = Model()
opt = optim.SGD(model.parameters(), lr=1e-3, momentum=0.9, weight_decay=5e-4)

for it in range(5000):
    y_pred = model(X_train).squeeze()
    l = F.binary_cross_entropy(y_pred, y_train)
    l.backward()
    opt.step()
    opt.zero_grad()
```

MLE solution $p(y|x, \mathcal{D})$ for 1k data points



Based on: `https://github.com/wiseodd/last_layer_laplace`
accompanying: *Kristiadi et al. "Being bayesian, even just a bit, fixes overconfidence in relu networks." ICML 2020.*

MLE solution $p(y|x, \mathcal{D})$ for 4 data points



Based on: `https://github.com/wiseodd/last_layer_laplace`
accompanying: *Kristiadi et al. "Being bayesian, even just a bit, fixes overconfidence in relu networks." ICML 2020.*

Goal

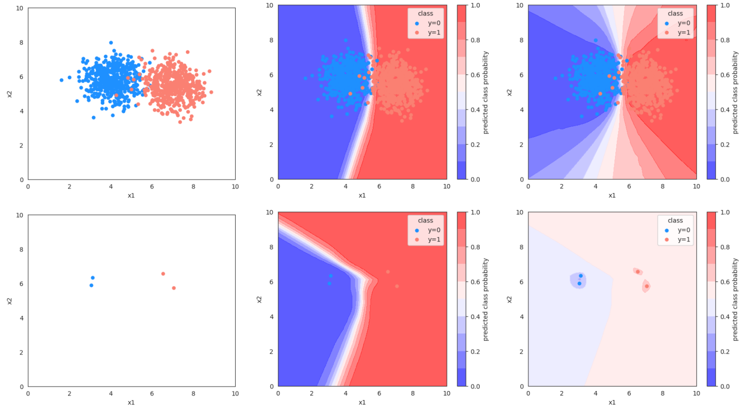
We want ML models that:

- are uncertain about unseen things
- become more certain with more data

MLE vs Bayes: Latent variable inference

- point-wise - find one value $\hat{\theta}$,
  e.g., by minimizing loss = maximizing likelihood (MLE) / maximum a posteriori (MAP): $\cancel{\hat{\theta} = \text{argmin}_{\theta} \mathcal{L}(\mathcal{D}, \theta)}$
- distributional - find a posterior $p(\theta|\mathcal{D})$
  - we get uncertainty about θ (e.g., variance in addition to the mean)

MLE vs Bayesian (LLLA) solution



Based on: `https://github.com/wiseodd/last_layer_laplace`

Bayes Theorem

Everything follows from two basic rules:

- Sum rule: $p(A) = \sum_b p(A, B = b)$
- Product rule: $p(A, B) = p(B|A) \cdot p(A) = p(A|B) \cdot p(B)$

Bayes Theorem

Everything follows from two basic rules:

- Sum rule: $p(A) = \sum_b p(A, B = b)$
- Product rule: $p(A, B) = p(B|A) \cdot p(A) = p(A|B) \cdot p(B)$

Bayes' Theorem:

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)} = \frac{p(B|A) \cdot p(A)}{\sum_a p(B, A = a)} = \frac{p(B|A) \cdot p(A)}{\sum_a p(B|A = a) p(A = a)}$$

Basic Concepts of Bayesian Methods

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)}$$

Let's rename $A \to \theta$, $B \to \mathcal{D}$:

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

- **Prior** $p(\theta)$**:** Initial belief on parameters before seeing data
- **Likelihood** $p(\mathcal{D}|\theta)$**:** Probability of data given parameters of the model
- **Posterior** $p(\theta|\mathcal{D})$**:** Updated belief after seeing (more) data

Bayes Theorem Example: Determining Stroke Type Based on MRI Scan Intensity

- The MRI scan of a patient shows an intensity value of $y = 140$ units ($\mathcal{D} = \{140\}$).

Bayes Theorem Example: Determining Stroke Type Based on MRI Scan Intensity

- The MRI scan of a patient shows an intensity value of $y = 140$ units ($\mathcal{D} = \{140\}$).
- Ischemic Stroke (Ischemic): In an ischemic stroke, affected brain areas may show lower signal intensity due to reduced blood flow. Assume these intensity values follow a normal distribution with a mean of 100 units and a standard deviation of 20:
  $y \sim \mathcal{N}(100, 20^2)|_{\theta=I}$.
- Hemorrhagic Stroke (Hemorrhagic): In a hemorrhagic stroke, affected areas show higher signal intensity due to bleeding. Assume these intensity values follow a normal distribution with a mean of 180 units and a standard deviation of:
  $y \sim \mathcal{N}(180, 30^2)|_{\theta=H}$.

Bayes Theorem Example: likelihood

$$p(y|\theta) = \mathcal{N}(y|100, 20^2) \cdot \mathbb{I}[\theta = I] + \mathcal{N}(y|180, 30^2) \cdot \mathbb{I}[\theta = H]$$

Bayes Theorem Example: priors

Prior Beliefs about Parameters:

- Ischemic Stroke (Ischemic): This is the most common type of stroke, accounting for about 85% of all strokes (prior probability $= 0.85$):
  $p(\theta = I) = 0.85$
- Hemorrhagic Stroke (Hemorrhagic): Less common, these strokes account for the remaining 15% of stroke cases (prior probability $= 0.15$).
  $p(\theta = H) = 0.15$

Bayes Theorem Example: solution

- Data: $\mathcal{D} = \{140\}$
- Likelihood: $\mathcal{N}(y|100, 20^2)|_{\theta=l}$, $\mathcal{N}(y|180, 30^2)|_{\theta=H}$
- Priors: $p(\theta = l) = 0.85$, $p(\theta = H) = 0.15$

Bayes Theorem Example: solution

- Data: $\mathcal{D} = \{140\}$
- Likelihood: $\mathcal{N}(y|100, 20^2)|_{\theta=I}$, $\mathcal{N}(y|180, 30^2)|_{\theta=H}$
- Priors: $p(\theta = I) = 0.85$, $p(\theta = H) = 0.15$
- Computation:
  - $p(\mathcal{D}|\theta = I) = \mathcal{N}(140|100, 20^2) = 0.0027$
  - $p(\mathcal{D}|\theta = H) = \mathcal{N}(140|180, 30^2) = 0.0055$

Bayes Theorem Example: solution

- Data: $\mathcal{D} = \{140\}$
- Likelihood: $\mathcal{N}(y|100, 20^2)|_{\theta=I}$, $\mathcal{N}(y|180, 30^2)|_{\theta=H}$
- Priors: $p(\theta = I) = 0.85$, $p(\theta = H) = 0.15$
- Computation:
  - $p(\mathcal{D}|\theta = I) = \mathcal{N}(140|100, 20^2) = 0.0027$
  - $p(\mathcal{D}|\theta = H) = \mathcal{N}(140|180, 30^2) = 0.0055$
  - $p(\mathcal{D}) = 0.0027 * 0.85 + 0.15 * 0.0055 = 0.002295 + 0.000825 = 0.00312$
  - $p(\theta = I|\mathcal{D}) = 0.0027 * 0.85/0.00312 = 0.736$
  - $p(\theta = H|\mathcal{D}) = 0.15 * 0.0055/0.00312 = 0.264$

Understanding priors: posteriors as priors

$$p(\theta|\mathcal{D}_0) = \frac{p(\mathcal{D}_0|\theta) \cdot p(\theta)}{p(\mathcal{D}_0)}$$

Understanding priors: posteriors as priors

$$p(\theta|\mathcal{D}_0) = \frac{p(\mathcal{D}_0|\theta) \cdot p(\theta)}{p(\mathcal{D}_0)}$$

$$p(\theta|\mathcal{D}_1 \cup \mathcal{D}_0) = \frac{p(\mathcal{D}_1 \cup \mathcal{D}_0|\theta) \cdot p(\theta)}{p(\mathcal{D}_1 \cup \mathcal{D}_0)} =$$

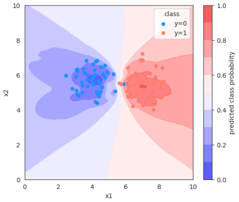
$$= \frac{p(\mathcal{D}_1|\theta)p(\mathcal{D}_0|\theta)p(\theta)}{p(\mathcal{D}_1)p(\mathcal{D}_0)} = \frac{p(\mathcal{D}_1|\theta)p(\theta|\mathcal{D}_0)}{p(\mathcal{D}_1)}$$

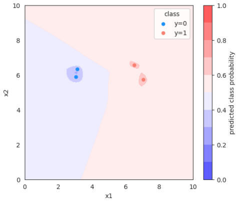
Understanding priors: posteriors as priors

$$p(\theta|\mathcal{D}_0) = \frac{p(\mathcal{D}_0|\theta) \cdot p(\theta)}{p(\mathcal{D}_0)}$$

$$p(\theta|\mathcal{D}_1 \cup \mathcal{D}_0) = \frac{p(\mathcal{D}_1 \cup \mathcal{D}_0|\theta) \cdot p(\theta)}{p(\mathcal{D}_1 \cup \mathcal{D}_0)} =$$

$$= \frac{p(\mathcal{D}_1|\theta)p(\mathcal{D}_0|\theta)p(\theta)}{p(\mathcal{D}_1)p(\mathcal{D}_0)} = \frac{p(\mathcal{D}_1|\theta)p(\theta|\mathcal{D}_0)}{p(\mathcal{D}_1)}$$





$$p(\theta|\mathcal{D}_1 \cup \mathcal{D}_0) \quad \leftarrow \quad p(\theta|\mathcal{D}_0) \leftarrow \quad p(\theta) = p(\theta|\emptyset)$$

Bayes Theorem Example: repeated measurement

Previous solution for $\mathcal{D}_0 = \{140\}$:

- $p(\theta = I|\mathcal{D}) = 0.0027 * 0.85/0.00312 = 0.736$
- $p(\theta = H|\mathcal{D}) = 0.15 * 0.0055/0.00312 = 0.264$

After additional measurement $y = 160$: $\mathcal{D}_1 = \{160\}$; $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_1 = \{140, 160\}$:

- $p(\mathcal{D}_1|\theta = I) = \mathcal{N}(160|100, 20^2) = 0.0002$
- $p(\mathcal{D}_1|\theta = H) = \mathcal{N}(160|180, 30^2) = 0.0106$
- $p(\mathcal{D}) = 0.0002 * 0.736 + 0.0106 * 0.264 = 0.0029456$
- $p(\theta = I|\mathcal{D}) = 0.0002 * 0.736/0.0029456 = 0.05$
- $p(\theta = H|\mathcal{D}) = 0.0106 * 0.264/0.0029456 = 0.95$

Conjugate priors

- Problem: how to find $p(\theta|\mathcal{D})$?
- For some pairs of prior+likelihood, the posterior takes the same form as prior

Conjugate priors: whiteboard example

Back to the coin-tossing example:

Let's consider Beta-Bernoulli model:

- prior $p(\theta) = Beta(\theta|\alpha, \beta) = \frac{\theta^{\alpha-1} \cdot (1-\theta)^{\beta-1}}{B(\alpha,\beta)}$
- likelihood
  $p(\mathcal{D}|\theta) = \prod_{y \in \mathcal{D}} Bernoulli(y|\theta) = \prod_{y} \left(\theta^{y} \cdot (1-\theta)^{(1-y)}\right)$
- example data: $\mathcal{D} = \{T, T\}$

Conjugate priors: whiteboard example

Back to the coin-tossing example:

Let's consider Beta-Bernoulli model:

- prior $p(\theta) = Beta(\theta|\alpha, \beta) = \frac{\theta^{\alpha-1} \cdot (1-\theta)^{\beta-1}}{B(\alpha,\beta)}$
- likelihood
  $p(\mathcal{D}|\theta) = \prod_{y \in \mathcal{D}} Bernoulli(y|\theta) = \prod_y \left(\theta^y \cdot (1-\theta)^{(1-y)}\right)$
- example data: $\mathcal{D} = \{T, T\}$

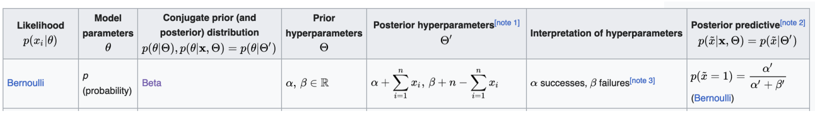
Find $p(\theta|\mathcal{D})$:

1. Let's start with $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$
2. ...
3. `https://homepage.divms.uiowa.edu/~mbognar/applets/beta.html`
4. Importance of priors: $\alpha = \beta = 1$ vs $\alpha = \beta = 10$

Conjugate priors

Conjugate priors:

`https://en.wikipedia.org/wiki/Conjugate_prior`



Likelihood $p(x_i \mid \theta)$	Model parameters θ	Conjugate prior (and posterior) distribution $p(\theta \mid \Theta), p(\theta \mid \mathbf{x}, \Theta) = p(\theta \mid \Theta')$	Prior hyperparameters Θ	Posterior hyperparameters[note 1] Θ'	Interpretation of hyperparameters	Posterior predictive[note 2] $p(\tilde{x} \mid \mathbf{x}, \Theta) = p(\tilde{x} \mid \Theta')$
Bernoulli	p (probability)	Beta	$\alpha, \beta \in \mathbb{R}$	$\alpha + \sum_{i=1}^{n} x_i, \beta + n - \sum_{i=1}^{n} x_i$	α successes, β failures[note 3]	$p(\tilde{x} = 1) = \frac{\alpha'}{\alpha' + \beta'}$ (Bernoulli)

BNNs: Typical Practical Setting

- data $\mathcal{D} = \{y\}$
  (or $\mathcal{D} = \{(x, y)\}$ but x (e.g., features vector) is not modeled)
- parameters vector θ
  - parameters steering a generative process
  - a lower-dimensional representation (like in VAE)
  - set of weights and biases in a NN

BNNs: Typical Practical Setting

- data $\mathcal{D} = \{y\}$
  (or $\mathcal{D} = \{(x, y)\}$ but x (e.g., features vector) is not modeled)
- parameters vector θ
  - parameters steering a generative process
  - a lower-dimensional representation (like in VAE)
  - set of weights and biases in a NN
- likelihood $p(\mathcal{D}|\theta)$ or $p(y|\theta, x)$
  - **NN structure (layers, activations etc.) is part of the likelihood**
  - iid: $p(\mathcal{D}|\theta) = \prod_{(x,y)\in\mathcal{D}} p(y|\theta, x)$ (same as $= \prod_i p(y_i|\theta, x_i)$)
    - e.g. for the stroke example with $\mathcal{D} = \{140, 160\}$:
      $p(\mathcal{D}|\theta = I) = \mathcal{N}(140|100, 20^2) \cdot \mathcal{N}(160|100, 20^2)$,
      $p(\mathcal{D}|\theta = H) = \mathcal{N}(140|180, 30^2) \cdot \mathcal{N}(160|180, 30^2)$
- prior $p(\theta)$

BNNs: Typical Practical Setting

- data $\mathcal{D} = \{y\}$
  (or $\mathcal{D} = \{(x, y)\}$ but x (e.g., features vector) is not modeled)
- parameters vector θ
  - parameters steering a generative process
  - a lower-dimensional representation (like in VAE)
  - set of weights and biases in a NN
- likelihood $p(\mathcal{D}|\theta)$ or $p(y|\theta, x)$
  - **NN structure (layers, activations etc.) is part of the likelihood**
  - iid: $p(\mathcal{D}|\theta) = \prod_{(x,y)\in\mathcal{D}} p(y|\theta, x)$ (same as $= \prod_i p(y_i|\theta, x_i)$)
    - e.g. for the stroke example with $\mathcal{D} = \{140, 160\}$:
      $p(\mathcal{D}|\theta = I) = \mathcal{N}(140|100, 20^2) \cdot \mathcal{N}(160|100, 20^2)$,
      $p(\mathcal{D}|\theta = H) = \mathcal{N}(140|180, 30^2) \cdot \mathcal{N}(160|180, 30^2)$
- prior $p(\theta)$
- ultimate goal: find posterior predictive $p(y|\mathcal{D})$ or $p(y|\mathcal{D}, x)$
- intermediate goal: find posterior $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$

The simplest BNN: linear regression with fixed noise variance $\sigma^2 = 1$.

- Likelihood of linear regression:
  $p(\mathcal{D}|\theta) = \prod_{(x,y)\in\mathcal{D}} p(y|x,\theta)$
  $p(y|x,\theta) = \mathcal{N}(y|\theta^T \cdot x, 1)$
  (i.e. $y = \theta^T x + \epsilon, \epsilon \sim \mathcal{N}(0,1)$)

The simplest BNN: linear regression with fixed noise variance $\sigma^2 = 1$.

- Likelihood of linear regression:
  $p(\mathcal{D}|\theta) = \prod_{(x,y)\in\mathcal{D}} p(y|x,\theta)$
  $p(y|x,\theta) = \mathcal{N}(y|\theta^T \cdot x, 1)$
  (i.e. $y = \theta^T x + \epsilon, \epsilon \sim \mathcal{N}(0,1)$)
- Factorized Gaussian prior:
  $p(\theta) = \mathcal{N}(\theta|0, I) = \prod_d \mathcal{N}(\theta_d|0,1)$

The simplest BNN: linear regression with fixed noise variance $\sigma^2 = 1$.

- Likelihood of linear regression:
  $p(\mathcal{D}|\theta) = \prod_{(x,y)\in\mathcal{D}} p(y|x,\theta)$
  $p(y|x,\theta) = \mathcal{N}(y|\theta^T \cdot x, 1)$
  (i.e. $y = \theta^T x + \epsilon, \epsilon \sim \mathcal{N}(0,1)$)
- Factorized Gaussian prior:
  $p(\theta) = \mathcal{N}(\theta|0, I) = \prod_d \mathcal{N}(\theta_d|0,1)$
- Computation: evidence (normalizer):
  $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta =$
  $\int \cdots \int \prod_{(x,y)\in\mathcal{D}} p(y|x,\theta_1,\ldots\theta_N) \cdot \prod_d \mathcal{N}(\theta_d|0,1)d\theta_1\ldots d\theta_N$

Predictions

- Point-wise:

$$\underbrace{p(y|x,\mathcal{D})}_{\text{predictive distribution}} = \underbrace{p(y|\hat{\theta},x)}_{\text{likelihood}}$$

- Bayesian Model Averaging (average of all possible models weighted by the posterior):

$$\underbrace{p(y|x,\mathcal{D})}_{\text{posterior predictive}} = \int \underbrace{p(y|\theta,x)}_{\text{likelihood}}\,\underbrace{p(\theta|\mathcal{D})}_{\text{posterior}}\,d\theta$$

Predictions

- Point-wise:

$$\underbrace{p(y|x,\mathcal{D})}_{\text{predictive distribution}} = \underbrace{p(y|\hat{\theta},x)}_{\text{likelihood}}$$

- Bayesian Model Averaging (average of all possible models weighted by the posterior):

$$\underbrace{p(y|x,\mathcal{D})}_{\text{posterior predictive}} = \int \underbrace{p(y|\theta,x)}_{\text{likelihood}} \underbrace{p(\theta|\mathcal{D})}_{\text{posterior}} d\theta$$

Statistics of $p(y|x,\mathcal{D})$ can be obtained using the Monte-Carlo estimates by two-step sampling:

- sample $\theta \sim p(\theta|\mathcal{D})$
- for the fixed θ, sample $y \sim p(y|\theta,x)$

Predictions

- Point-wise:

$$\underbrace{p(y|x,\mathcal{D})}_{\text{predictive distribution}} = \underbrace{p(y|\hat{\theta},x)}_{\text{likelihood}}$$

- Bayesian Model Averaging (average of all possible models weighted by the posterior):

$$\underbrace{p(y|x,\mathcal{D})}_{\text{posterior predictive}} = \int \underbrace{p(y|\theta,x)}_{\text{likelihood}} \underbrace{p(\theta|\mathcal{D})}_{\text{posterior}} d\theta$$

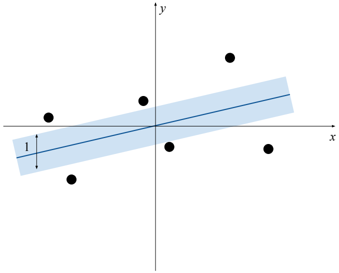
Statistics of $p(y|x,\mathcal{D})$ can be obtained using the Monte-Carlo estimates by two-step sampling:

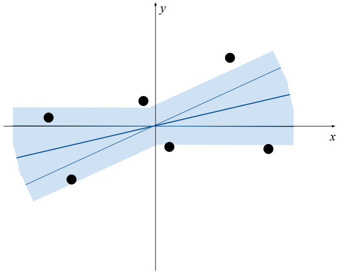
- sample $\theta \sim p(\theta|\mathcal{D})$
- for the fixed θ, sample $y \sim p(y|\theta,x)$

For example, $\mathbb{E}_{p(y|x,\mathcal{D})}[y] \approx \frac{1}{S_\theta}\frac{1}{S_y}\sum_{\theta\sim p(\theta|\mathcal{D})}\sum_{y\sim p(y|\theta,x)} y$

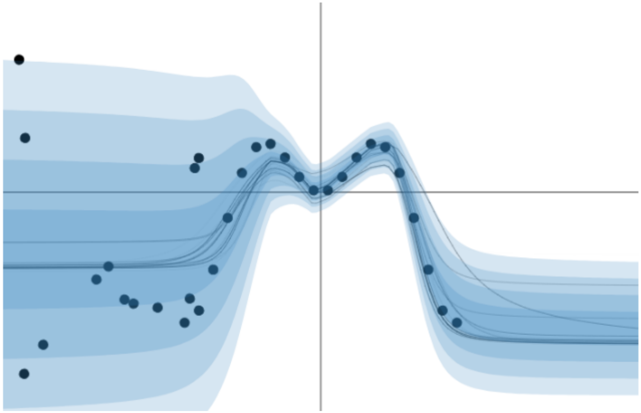
Toy example

- point-wise - find $\hat{\theta}$,
  e.g., maximizing likelihood (MLE) or maximum a posteriori (MAP)
- distributional - find a posterior $p(\theta|D)$





Uncertainty: non-linear regression



http://mlg.eng.cam.ac.uk/yarin/blog_2248.html#demo

Figure: Multiple draws of a regression model.

Side note: Types of uncertainty

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

- aleatoric (via likelihood $p(y|\theta, x)$)
  - by nature of the data
  - for example: imprecise measurement
  - irreducible
  - can be captured with point inference
- epistemic (via posterior $p(\theta|\mathcal{D})$)
  - uncertainty in a model
  - for example: regression weights
  - can be reduced with more data
  - handled by, e.g., Bayesian inference

Types of uncertainty continued: Uncertainty decomposition 1

Following the law of the total variance, the total predictive uncertainty can be expressed as:

$$\mathrm{Var}(y \mid x, \mathcal{D}) = \underbrace{\mathbb{E}_{p(\theta|\mathcal{D})}[\mathrm{Var}(y \mid \theta, x)]}_{\text{Aleatoric Uncertainty}} + \underbrace{\mathrm{Var}_{p(\theta|\mathcal{D})}[\mathbb{E}(y \mid \theta, x)]}_{\text{Epistemic Uncertainty}}$$

- Aleatoric uncertainty is the **expected variance of the predictions**, while epistemic uncertainty is the **variance of the expected predictions**
- This decomposition is particularly useful in regression problems where understanding the contribution of noise versus model uncertainty to the total predictive uncertainty is important.

Types of uncertainty continued: Uncertainty decomposition 2

$$H(y \mid x, \mathcal{D}) = \underbrace{\mathbb{E}_{p(\theta \mid \mathcal{D})}[H(y \mid \theta, x)]}_{\text{Aleatoric Uncertainty}} + \underbrace{I(y; \theta \mid \mathcal{D})}_{\text{Epistemic Uncertainty}}$$

Where:

- $H(y \mid x, \mathcal{D})$ is the predictive entropy.
- $H(y \mid \theta, x)$ is the entropy of the predictions given the model parameters θ.
- $I(y; \theta \mid \mathcal{D})$ is the mutual information between the predictions and the model parameters given the training data.

This decomposition is particularly useful in classification problems and when understanding the uncertainty in probabilistic predictions is important. It provides a more granular view of uncertainty in terms of information gain and entropy.

Uncertainty: aleatoric vs epistemic

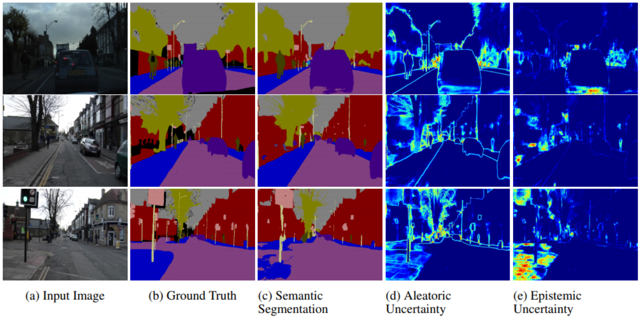

(a) Input Image
(b) Ground Truth
(c) Semantic Segmentation
(d) Aleatoric Uncertainty
(e) Epistemic Uncertainty

Figure 1: **Illustrating the difference between aleatoric and epistemic uncertainty** for semantic segmentation on the CamVid dataset [8]. *Aleatoric* uncertainty captures noise inherent in the observations. In (d) our model exhibits increased aleatoric uncertainty on object boundaries and for objects far from the camera. *Epistemic* uncertainty accounts for our ignorance about which model generated our collected data. This is a notably different measure of uncertainty and in (e) our model exhibits increased epistemic uncertainty for semantically and visually challenging pixels. The bottom row shows a failure case of the segmentation model when the model fails to segment the footpath due to increased epistemic uncertainty, but not aleatoric uncertainty.

Source: Kendall et al.: *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* NIPS2017

Summary: Challenges in Bayesian learning

Design:

- likelihood and network structure
- priors $p(\theta)$

Learning:

- posterior $p(\theta|\mathcal{D}) = \frac{p(\mathcal{D},\theta)}{p(\mathcal{D})}$
- evidence $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta$
- posterior predictive $p(y|\mathcal{D}) = \int p(y|\theta)p(\theta|\mathcal{D})d\theta$
- model selection = hyperparameters

Part 2

Part 2: Bayesian Neural Networks

NNs: Typical Regression/Classification Setting

- Data: $\mathcal{D} = \{(x_i, y_i)\}$
  but x_i (e.g., features vector) are not modeled (=are fixed)
- Task: predict unknown y for some input x
- Model:
  - parameters vector θ
  - likelihood $p(\mathcal{D}|\theta) = \prod_i p(y_i|\theta, x_i)$

NNs: likelihood $p(\mathcal{D}|\theta) = \prod_i p(y_i|\theta, x_i)$

- NN structure (layers, activations etc.) may be hidden inside of likelihood or we can write explicitly: $p(y|\theta, x) = p(y|\mathrm{NN}(\theta^{\mathrm{NN}}, x), \theta^{lik})$
  - $\theta = \theta^{\mathrm{NN}} \cup \theta^{lik}$

NNs: likelihood $p(\mathcal{D}|\theta) = \prod_i p(y_i|\theta, x_i)$

- NN structure (layers, activations etc.) may be hidden inside of likelihood or we can write explicitly:
  $p(y|\theta, x) = p(y|\mathrm{NN}(\theta^{\mathrm{NN}}, x), \theta^{lik})$
  - $\theta = \theta^{\mathrm{NN}} \cup \theta^{lik}$
  - where NN is the network
  - p "interprets" logits as parameters of a probability distribution e.g. softmax, normal
  - θ^{lik} are additional likelihood parameters not included in NN
- $\mathrm{NN}(\theta, x) = \phi^L(\theta^L, \phi^{L-1}(\theta^{L-1}, \dots, \phi^1(\theta^1, x)))$
  - where $\theta^{\mathrm{NN}} = \theta^1 \cup \dots \cup \theta^L$ consists of weights and biases in NN
  - ϕ^l are layers
    e.g. $\phi^l(\text{weights} \cup \text{biases}, \text{inputs}) = a^l(\text{weights} \cdot \text{inputs} + \text{biases})$
  - a^l are activations

Example: Homoscedastic Gaussian regression

- $p(y_i|\mathrm{NN}(\theta, x_i), \sigma) = \mathcal{N}(y_i|\mu_i, \sigma)$
- $\mu_i := \mathrm{NN}(x_i, \theta) = \phi^L(\theta^L, \phi^{L-1}(\theta^{L-1}, \ldots, \phi^1(\theta^1, x_i)))$
- $\theta^{lik} = \{\sigma\}$
- $a^L =$ identity (i.e., $a^L(v) = v$)

Note: we look for $\mathbb{E}_{p(y_i|\mathrm{NN}(\theta,x_i),\sigma)}[y_i] = \mu_i$, and it is coincidental that (for Gaussian regression) the NN is returning exactly μ_i.

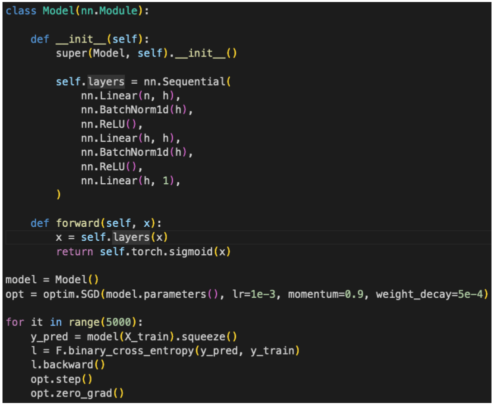
Example: Heteroscedastic Gaussian regression

- $p(y_i|\mathrm{NN}(\theta, x_i)) = \mathcal{N}(y_i|\mu_i, \sigma_i)$
- $[\mu_i, \sigma_i] := \mathrm{NN}(x_i, \theta) = \phi^L(\theta^L, \phi^{L-1}(\theta^{L-1}, \ldots, \phi^1(\theta^1, x_i)))$
- $\theta^{lik} = \emptyset$
- $a^L(v) = concat(v[0 \ldots (H/2-1)], \exp(v[H/2 \ldots H]) + \epsilon)$
  where exp makes sure $\sigma_i > 0$

Example: Classification

- $p(y_i|\mathrm{NN}(\theta, x_i)) = \mathrm{Bernoulli}(y_i|p_i)$
- $p_i := \mathrm{NN}(x_i, \theta) = \phi^L(\theta^L, \phi^{L-1}(\theta^{L-1}, \ldots, \phi^1(\theta^1, x_i)))$
- $\theta^{lik} = \emptyset$
- $a^L(v) = \mathrm{sigmoid}(v)$
  sigmoid makes sure $p_i \in [0, 1]$

MLE (likelihood=Bernoulli) for a NN Classifier



```
class Model(nn.Module):

    def __init__(self):
        super(Model, self).__init__()

        self.layers = nn.Sequential(
            nn.Linear(n, h),
            nn.BatchNorm1d(h),
            nn.ReLU(),
            nn.Linear(h, h),
            nn.BatchNorm1d(h),
            nn.ReLU(),
            nn.Linear(h, 1),
        )

    def forward(self, x):
        x = self.layers(x)
        return self.torch.sigmoid(x)

model = Model()
opt = optim.SGD(model.parameters(), lr=1e-3, momentum=0.9, weight_decay=5e-4)

for it in range(5000):
    y_pred = model(X_train).squeeze()
    l = F.binary_cross_entropy(y_pred, y_train)
    l.backward()
    opt.step()
    opt.zero_grad()
```

Point-wise (MLE/MAP) solution

- Parameters: find one value $\hat{\theta}$,
  e.g., by minimizing loss = maximizing likelihood (MLE) / maximum a posteriori (MAP):

$$\hat{\theta} = \text{argmin}_{\theta}\mathbb{L}(\mathcal{D}, \theta)$$

- Predictions:

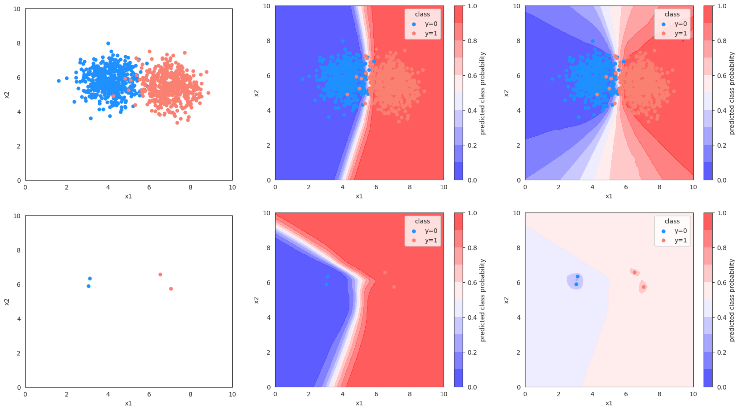
$$\underbrace{p(y|x, \mathcal{D})}_{\text{predictive distribution}} = \underbrace{p(y|\hat{\theta}, x)}_{\text{likelihood}}$$

Bayesian (distributional) solution

- Parameters: find a posterior $p(\theta|\mathcal{D})$
  - we get uncertainty about θ (e.g., variance in addition to the mean)
- Predictions: Bayesian Model Averaging (average of all possible models weighted by the posterior):

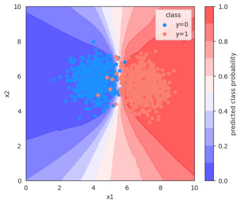
$$\underbrace{p(y|x,\mathcal{D})}_{\text{posterior predictive}} = \int \underbrace{p(y|\theta,x)}_{\text{likelihood}} \underbrace{p(\theta|\mathcal{D})}_{\text{posterior}} d\theta$$

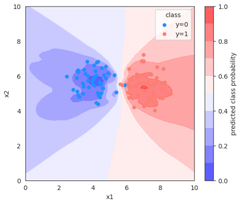
Classification: MLE vs Bayesian (LLLA) solution

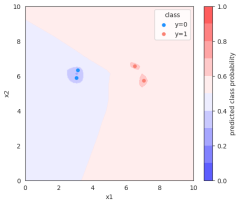


Based on: `https://github.com/wiseodd/last_layer_laplace`

Classification: Bayesian (LLLA) solution for varying data sizes

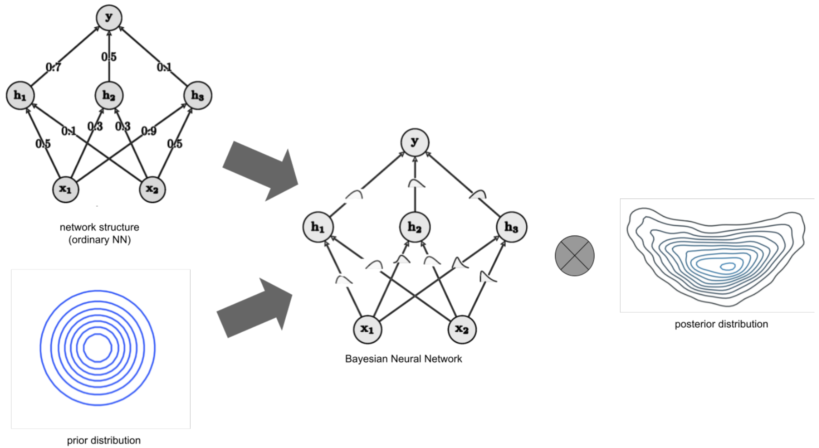






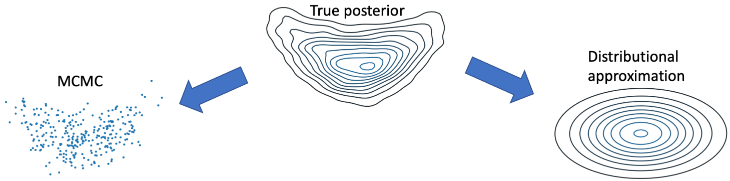
Based on: `https://github.com/wiseodd/last_layer_laplace`

Example: (Basic Feed-Forward) Bayesian Neural Network



Objective: find the posterior distribution $p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta) \cdot p(\theta)}{p(\mathcal{D})}$

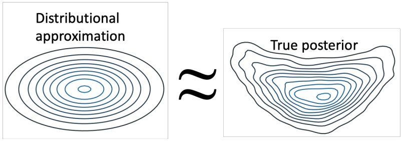
Posteriors for complex Bayesian models



- *MCMC methods 'kind of' work, but are slow*
- *Distributional approaches are fast but require tricks to make them work*

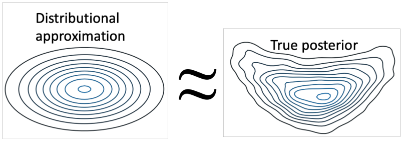
Distributional approximations

- Laplace approximation: find $\mathcal{N}(\theta|\mu, \Sigma) \approx p(\theta|\mathcal{D})$.
- Variational:
  - Postulate $q(\theta|\lambda) \approx p(\theta|\mathcal{D})$



Distributional approximations

- Laplace approximation: find $\mathcal{N}(\theta|\mu, \Sigma) \approx p(\theta|\mathcal{D})$.
- Variational:
  - Postulate $q(\theta|\lambda) \approx p(\theta|\mathcal{D})$



  - Minimize divergence between the approximation and the true posterior:
    - EP: $q(\theta|\lambda) = \text{argmin}_q KL(p|q)$
    - **VI:** $q(\theta|\lambda) = \textbf{argmin}_q KL(q|p)$

Approximate inference: variational objective - ELBO

$$KL(q|p) = \int q(\theta|\lambda) \log\left(\frac{q(\theta|\lambda)}{p(\theta|\mathcal{D})}\right) d\theta = \int q(\theta|\lambda) \log\left(\frac{q(\theta|\lambda)p(\mathcal{D})}{p(\theta|\mathcal{D})p(\mathcal{D})}\right) d\theta$$

$$= \int q(\theta|\lambda) \log\left(\frac{q(\theta|\lambda)}{p(\theta, \mathcal{D})}\right) d\theta + \log p(\mathcal{D})$$

$$= -\underbrace{\int q(\theta|\lambda) \left[\log\left(p(\mathcal{D}|\theta)p(\theta)\right) - \log q(\theta|\lambda)\right] d\theta}_{ELBO} + \log p(\mathcal{D})$$

$$= -\underbrace{\left(\mathbb{E}_q \log\left(p(\mathcal{D}|\theta)p(\theta)\right) \underbrace{-\mathbb{E}_q \log q(\theta|\lambda)}_{H(q)}\right)}_{ELBO} + \log p(D)$$

Approximate inference: variational objective - ELBO

$$
\begin{aligned}
KL(q|p) &= \int q(\theta|\lambda) \log\left(\frac{q(\theta|\lambda)}{p(\theta|\mathcal{D})}\right) d\theta = \int q(\theta|\lambda) \log\left(\frac{q(\theta|\lambda)p(\mathcal{D})}{p(\theta|\mathcal{D})p(\mathcal{D})}\right) d\theta \\
&= \int q(\theta|\lambda) \log\left(\frac{q(\theta|\lambda)}{p(\theta,\mathcal{D})}\right) d\theta + \log p(\mathcal{D}) \\
&= -\underbrace{\int q(\theta|\lambda)\left[\log\left(p(\mathcal{D}|\theta)p(\theta)\right) - \log q(\theta|\lambda)\right] d\theta}_{ELBO} + \log p(\mathcal{D}) \\
&= -\underbrace{\left(\mathbb{E}_q \log\left(p(\mathcal{D}|\theta)p(\theta)\right) \underbrace{-\mathbb{E}_q \log q(\theta|\lambda)}_{H(q)}\right)}_{ELBO} + \log p(D)
\end{aligned}
$$

Since $\log p(D) = const$: $\text{argmin}_q KL(q|p) = \text{argmax}_q ELBO$

ELBO: 3 forms

- with entropy

$$\mathcal{L} = \mathbb{E}_q[\log(p(D|\theta)p(\theta))] + H(q)$$

- under one integral:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_q[\log(p(D|\theta)p(\theta)) - \log q(\theta|\lambda)] \\ &= \mathbb{E}_q[\log p(D|\theta) + \log p(\theta) - \log q(\theta|\lambda)]\end{aligned}$$

- with KL (like in VAEs):

$$\mathcal{L} = \mathbb{E}_q[\log p(D|\theta)] - KL(q(\theta|\lambda)|p(\theta))$$

note KL here is between q and the prior and it's a different KL than the one between q and the true posterior we started from

Practical implementation: minibatches

$$\begin{aligned}
ELBO &= \int q(\theta|\lambda) \log \left(p(\mathcal{D}|\theta)\right) d\theta + KL(q|p) \\
&= \int q(\theta|\lambda) \log \left(\prod_{(x,y)\in\mathcal{D}} p(y|\theta,x) \right) d\theta + KL(q|p) \\
&= \sum_{(x,y)\in\mathcal{D}} \int q(\theta|\lambda) \log \left(p(y|\theta,x)\right) d\theta + KL(q|p) \\
&\approx \frac{|\mathcal{D}|}{|\mathcal{D}'|} \sum_{(x,y)\in\mathcal{D}'} \int q(\theta|\lambda) \log \left(p(y|\theta,x)\right) d\theta + KL(q|p)
\end{aligned}$$

independen

minibatches

for a minibatch $\mathcal{D}' \subset \mathcal{D}$

Practical implementation: Monte-Carlo of Integrals

$$\begin{aligned} ELBO &= \int q(\theta|\lambda) \left[\log\left(p(\mathcal{D}|\theta)p(\theta)\right) - \log q(\theta|\lambda)\right] d\theta \\ &\approx \frac{1}{S} \sum_{\theta \sim q(\theta|\lambda)} \log\left(p(\mathcal{D}|\theta)p(\theta)\right) - \log q(\theta|\lambda) \end{aligned}$$

Gradient-based optimization

- **Objective:** $\text{argmax}_q ELBO$
- **Gradient-based optimization:**
  - variational: $q(\theta|\lambda)_{next} := q(\theta|\lambda) + \eta \nabla_q ELBO$
  - implementation: $\lambda_{next} := \lambda + \eta \nabla_\lambda ELBO$

Gradient-based optimization

- **Objective:** $\text{argmax}_q ELBO$
- **Gradient-based optimization:**
  - variational: $q(\theta|\lambda)_{next} := q(\theta|\lambda) + \eta \nabla_q ELBO$
  - implementation: $\lambda_{next} := \lambda + \eta \nabla_\lambda ELBO$
  - reparameterization allows us to *push* gradient through the sampling operation $(\theta \sim q(\theta|\lambda))$:
    - $\nabla_\lambda \sum_{\theta \sim q(\theta|\lambda)} g(\theta) = \sum_{\epsilon \sim q_0(\epsilon)} \nabla_\lambda g(\theta(\epsilon, \lambda))$

Reparameterization for 1D normal distribution

Sample from a distribution without any parameters to generate samples from a target distribution by a deterministic transformation:

- let's consider $q(\theta|\lambda) \equiv \mathcal{N}(\theta|\mu, \sigma)$ (where $\lambda \equiv \{\mu, \sigma\}$) [target]
- reparameterize $\theta(\epsilon, \mu, \sigma) = \mu + \sigma\epsilon$, $\epsilon \sim \mathcal{N}(\epsilon|0, I)$ [sampling] (note: $N(\epsilon|0, I)$ has all parameters fixed)
- then $\theta(\epsilon, \mu, \sigma) \sim N(\theta|\mu, \sigma)$

Reparameterized gradient of ELBO

$$\nabla_\lambda ELBO \approx \nabla_\lambda \left[\frac{1}{S} \sum_{\theta \sim q(\theta|\lambda)} \log p(\mathcal{D}, \theta) - \log q(\theta|\lambda) \right]$$

reparameter

$$= \nabla_\lambda \left[\frac{1}{S} \sum_{\epsilon \sim q_0(\epsilon)} \log p(\mathcal{D}, \theta(\epsilon, \lambda)) - \log q(\theta(\epsilon, \lambda)|\lambda) \right]$$

$$= \frac{1}{S} \sum_{\epsilon \sim q_0(\epsilon)} \nabla_\lambda \left[\log p(\mathcal{D}, \theta(\epsilon, \lambda)) - \log q(\theta(\epsilon, \lambda)|\lambda) \right]$$

$\log p(\mathcal{D}, \theta(\epsilon, \lambda)) - \log q(\theta(\epsilon, \lambda)|\lambda)$ is a deterministic function of $\epsilon, \lambda, \mathcal{D}$

$\rightarrow$ gradient of it can be computed either analytically or numerically

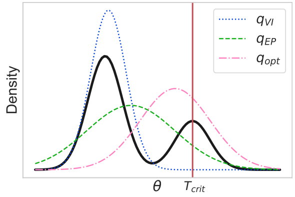
Summary: variational inference for practitioners

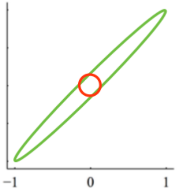
We look for $q(\theta|\lambda) \approx p(\theta|\mathcal{D})$:

1. Choose model likelihood $p(\mathcal{D}|\theta)$ and prior $p(\theta)$
2. Assume $q(\theta|\lambda)$ in some family parametrized by λ;
   often: $q(\theta|\lambda) \equiv N(\mu, diag(\sigma))$ so $\lambda \equiv \{\mu\} \cup \{\sigma\}$
3. Optimize with gradients: $\lambda_{next} := \lambda + \eta \nabla_\lambda \mathcal{L}$;
   usually $\mathcal{L} \equiv ELBO$
4. Use $q(\theta|\lambda)$ in place of $p(\theta|\mathcal{D})$ wherever needed

Problems

- additional assumptions needed to learn for high-dimensional posteriors,
  e.g., factorized (mean-field) posteriors $q(\theta|\lambda) = \prod_d q(\theta_d|\lambda)$
- mode collapsing and underestimating variance for $KL(q|p)$





What need to do better for BNNs

- Learn full posterior only for last-layer methods
- Define priors in functions space
- ...

Last layer VI

Idea: Bayesian learning just for the last layer. MAP for the rest.

Last layer VI: Variational Bayesian Last Layers (ICLR 2024)

- **Likelihood:** $p(y|\theta, x) = p(y|\phi^L(\theta^L, \phi^{L-1}(\omega^{L-1}, \dots, \phi^1(\omega^1, x)))) = p(y|\mathrm{NN}(\theta, \omega, x))$ where $\theta = \theta^L$, $\omega = \{\omega^1, \dots, \omega^{L-1}\}$
  ω are weights and biases learned point-wise
- **Prior:** $p(\theta^L)$, $p(\omega)$ (only for MAP regularization)
- **Goal:** find $p(\theta|\mathcal{D})$ and $\hat{\omega}$.

Last layer VI: Variational Bayesian Last Layers (ICLR 2024)

- **Likelihood:** $p(y|\theta, x) = p(y|\phi^L(\theta^L, \phi^{L-1}(\omega^{L-1}, \dots, \phi^1(\omega^1, x)))) = p(y|\mathrm{NN}(\theta, \omega, x))$ where $\theta = \theta^L$, $\omega = \{\omega^1, \dots, \omega^{L-1}\}$

  ω are weights and biases learned point-wise
- **Prior:** $p(\theta^L)$, $p(\omega)$ (only for MAP regularization)
- **Goal:** find $p(\theta|\mathcal{D})$ and $\hat{\omega}$.
- $\mathcal{L} = \mathbb{E}_{q(\theta|\lambda)}\left[\log p(D|\theta, \omega) p(\theta)\right] + H(q) + p(\omega)$
- Solve for: $\hat{\lambda}, \hat{\omega} = argmax_{\lambda,\omega} \mathcal{L}$

Last layer VI: Variational Bayesian Last Layers (ICLR 2024)

- **Likelihood:** $p(y|\theta, x) = p(y|\phi^L(\theta^L, \phi^{L-1}(\omega^{L-1}, \dots, \phi^1(\omega^1, x)))) = p(y|\text{NN}(\theta, \omega, x))$ where $\theta = \theta^L$, $\omega = \{\omega^1, \dots, \omega^{L-1}\}$
  ω are weights and biases learned point-wise
- **Prior:** $p(\theta^L)$, $p(\omega)$ (only for MAP regularization)
- **Goal:** find $p(\theta|\mathcal{D})$ and $\hat{\omega}$.
- $\mathcal{L} = \mathbb{E}_{q(\theta|\lambda)}\left[\log p(D|\theta, \omega)p(\theta)\right] + H(q) + p(\omega)$
- Solve for: $\hat{\lambda}, \hat{\omega} = argmax_{\lambda,\omega}\mathcal{L}$
- Trick (collapsed VI): for fixed ω and carefully selected $q(\theta|\lambda)$, the expectation in ELBO has a close-form solution (no sampling is needed anymore = lower variance):
  $\mathbb{E}_{q(\theta|\lambda)}\left[\log p(D|\theta, \omega)p(\theta)\right] = \mathcal{P}(\mathcal{D}, \omega, \lambda)$

Last layer VI: Variational Bayesian Last Layers (ICLR 2024)

- **Likelihood:** $p(y|\theta, x) =$ $p(y|\phi^L(\theta^L, \phi^{L-1}(\omega^{L-1}, \dots, \phi^1(\omega^1, x)))) = p(y|\text{NN}(\theta, \omega, x))$ where $\theta = \theta^L$, $\omega = \{\omega^1, \dots, \omega^{L-1}\}$
  ω are weights and biases learned point-wise
- **Prior:** $p(\theta^L)$, $p(\omega)$ (only for MAP regularization)
- **Goal:** find $p(\theta|\mathcal{D})$ and $\hat{\omega}$.
- $\mathcal{L} = \mathbb{E}_{q(\theta|\lambda)}\left[\log p(D|\theta, \omega)p(\theta)\right] + H(q) + p(\omega)$
- Solve for: $\hat{\lambda}, \hat{\omega} = argmax_{\lambda,\omega}\mathcal{L}$
- Trick (collapsed VI): for fixed ω and carefully selected $q(\theta|\lambda)$, the expectation in ELBO has a close-form solution (no sampling is needed anymore = lower variance): $\mathbb{E}_{q(\theta|\lambda)}\left[\log p(D|\theta, \omega)p(\theta)\right] = \mathcal{P}(\mathcal{D}, \omega, \lambda)$
- $\hat{\lambda}, \hat{w} = argmax_{\lambda,w}\mathcal{P}(\mathcal{D}, \omega, \lambda) + H(q) + p(\omega)$, where $H(q)$ is also a function of only λ

Last layer VI: Variational Bayesian Last Layers (ICLR 2024)

Regression example:

- Let's denote $\phi = \phi^{L-1}(\omega^{L-1}, \dots \phi^1(\omega^1, x))))$
- Normal likelihood $N(y|\theta^T\phi, \sigma^2)$, $a^L =$ identity
- Factorized prior $N(\theta|0, I)$
- Normal posterior assumption: $q(\theta|\mu, \Sigma) = N(\theta|\mu, \Sigma)$, where $\lambda = \{\mu, \Sigma\}$.
  - Note: Σ may be full-rank.
- We may learn (point-wise) the likelihood hyperparameter σ as well: $\omega = \{\omega^1, \dots, \omega^{L-1}\} \cup \{\sigma\}$

$$\int \log\left(N(y|\theta^T\phi, \sigma^2)N(\theta|0, I)\right) N(\theta|\mu, \Sigma)d\theta = \dots$$

$$= \mathcal{P}(\mathcal{D}, \{\psi, \sigma\}, \{\mu, \Sigma\})$$

Priors

- $\theta = \theta^L \cup \dots \theta^1 \cup \theta^{lik}$
- Factorized priors $p(\theta) = \prod_d p(\theta_d)$
  Note: lower vs upper indices
- often $p(\theta_d) = N(\theta_d | 0, 1)$

Function-space view on BNNs: Matching priors

1. Function view on NNs (whiteboard 1D example)
   - Prior predictive distribution: $p(y|x) = \int p(y|x,\theta) p(\theta|\zeta) d\theta$ where ζ denotes hyperparameters of the priors
   - Let $p_{\text{NN}}(f|\zeta)$ be functions induced by NN
2. Let $p_T(f)$ denote target distribution on functions
3. Let's use Wasserstein loss between function samples $f_{\text{NN}} \sim p_{\text{NN}}(f|\zeta)$ and $f_T \sim p_T(f)$
4. Matching the target: gradient-based optimization on index-set χ:

$$\hat{\zeta} = \text{argmin}_{\zeta} \mathbb{L}\left(p_{\text{NN}}(f|\zeta), p_T(f)\right)|_{\chi}$$

Appendix

Disclaimer on notation

Continuous r.v.-s $\longleftrightarrow$ Discrete r.v.-s:

- integral $\int \longleftrightarrow$ sum $\sum$
- probability density $p \longleftrightarrow$ probability mass P

We may abuse notation by writing e.g. $r.v. \sim p(r.v.|params)$ (instead of $r.v. \sim p(params)$) to explicitly inform about $r.v.$

Loss: $\mathbb{L}(\mathcal{D}, \theta) =$ negative log-likelihood: $-\mathcal{L}(\mathcal{D}, \theta)$

Distributions: $q \equiv q(\theta|\lambda)$, $p \equiv p(\theta|\mathcal{D})$
e.g. $KL(q|p) \equiv KL(q(\theta|\lambda)|p(\theta|\mathcal{D})) = \int q(\theta|\lambda) \log \frac{q(\theta|\lambda)}{p(\theta|\mathcal{D})} d\theta$,
$H(q) \equiv H(q(\theta|\lambda)) = -\int q(\theta|\lambda) \log q(\theta|\lambda) d\theta$

Some useful identities

- $\log e^A = A$
- $\log(A \cdot B) = \log(A) + \log(B)$
- $\log(\prod_i A_i) = \sum_i \log(A_i)$
- $\log(A/B) = \log(A) - \log(B)$
- $\int g(B)p(A)dA = g(B)$