

Variational inference

Introduction to VB

Helge Langseth and Thomas Dyhre Nielsen

Oct. 2018

Introduction

Day 1: Bayesian networks – Definition and inference

- Definition of Bayesian networks: Syntax and semantics
- Exact inference
- Approximate inference using MCMC

Day 2: Variational inference – Introduction and basis

- Approximate inference through the *Kullback-Leibler divergence*
- *Variational Bayes*
- The *mean-field* approach to Variational Bayes

Day 3: Variational Bayes – cont'd

- Solving the VB equations
- Introducing Exponential families

Day 4: Scalable Variational Bayes

- Variational message passing
- Stochastic gradient ascent
- Stochastic variational inference

Day 5: Current approaches and extensions

- Variational Auto Encoders
- Black Box variational inference
- Probabilistic Programming Languages

- Bayesian networks are used to represent (high-dim) distributions over random variables.
 - Simple syntax: Nodes, links, DAG, conditional distributions. Plate notation.
 - Clear semantics: Nodes, links, Markov properties
- Inference: Find $p(\mathbf{z} \mid \mathbf{x})$, where \mathbf{z} are variables of interest, \mathbf{x} are the observed variables.
 - Exact inference: We looked at variable elimination, but more clever methods are available.
 - Markov Chain Monte Carlo: We looked at Gibbs sampling, where each unobserved node is sampled based on its conditional given the Markov blanket.

Approximate inference as optimization

- The general setting for approximate inference in a BN is to somehow “resemble” the calculation of $p(\mathbf{z} \mid \mathbf{X} = \mathbf{x})$, but using less costly computational operations.
- We will call the approximation $q(\cdot)$, hence $q(\mathbf{z} \mid \mathbf{x}) \approx p(\mathbf{z} \mid \mathbf{X} = \mathbf{x})$
 - Often the conditioning part is dropped, hence $q(\mathbf{z})$ is a short-hand for $q(\mathbf{z} \mid \mathbf{x})$.

Formalization of approximate inference:

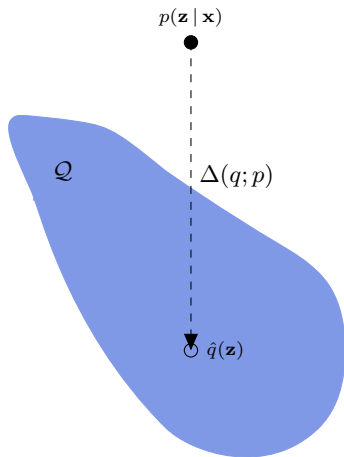
Given a family of tractable distributions \mathcal{Q} and a similarity measurement between distributions Δ , choose

$$\hat{q}(\mathbf{z}) = \arg \min_{q \in \mathcal{Q}} \Delta(q(\mathbf{z}); p(\mathbf{z} \mid \mathbf{x}))$$

Remaining decisions to be made:

- How to define a family of distributions \mathcal{Q} that is both flexible enough to generate good approximations, and at the same time restrictive enough to support efficient calculations?
- How to define Δ so that we end up with a high-quality solution from \mathcal{Q} .

- With a “target-set” \mathcal{Q} and a distance function $\Delta(q; p)$ we can think of the approximation as a projection of p onto the sub-space defined by \mathcal{Q} .
- The projection ends up at \hat{q} , which minimizes $\Delta(q; p)$ over all $q \in \mathcal{Q}$.
- We should choose Δ so that it actually captures the “relevant” difference between p and q ,
- ... and choose \mathcal{Q} so that it is flexible enough to capture “most of” $p(\cdot)$, meaning that $\Delta(q; p)$ is “typically small”.



Desiderata

To use Δ to measure the distance from a distribution f to a distribution g it would be relevant to require that Δ has the following properties:

Positivity: $\Delta(f; g) \geq 0$ and $\Delta(f; g) = 0$ if and only if $f = g$.

Symmetry: $\Delta(f; g) = \Delta(g; f)$

Triangle: For distributions f , g , and h we have that $\Delta(f; g) \leq \Delta(f; h) + \Delta(h; g)$.

Desiderata

To use Δ to measure the distance from a distribution f to a distribution g it would be relevant to require that Δ has the following properties:

Positivity: $\Delta(f; g) \geq 0$ and $\Delta(f; g) = 0$ if and only if $f = g$.

Symmetry: $\Delta(f; g) = \Delta(g; f)$

Triangle: For distributions f , g , and h we have that $\Delta(f; g) \leq \Delta(f; h) + \Delta(h; g)$.

Standard choice when working with probability distributions

It has become standard to choose the *Kullback-Leibler divergence* as the distance measure, where

$$\text{KL}(f||g) = \int_{\mathbf{z}} f(\mathbf{z}) \log \left(\frac{f(\mathbf{z})}{g(\mathbf{z})} \right) d\mathbf{z} = \mathbb{E}_f \left[\log \left(\frac{f(\mathbf{z})}{g(\mathbf{z})} \right) \right].$$

Notice that while $\text{KL}(f||g)$ obeys the positivity criterion, it satisfies neither symmetry nor the triangle inequality. It is thus **not a proper distance measure**.

Information-projection

- Minimizes $\text{KL}(q||p)$, that is, the expected information loss under q if p is used instead of q .
- Factorizes as
$$\text{KL}(q||p) = -\mathbb{E}_q[\ln p(\mathbf{z})] - \mathcal{H}_q.$$
- Preference given to q that has:
 - High probability at p -probable regions.
 - Very small probability given to any region where p is small.
 - High entropy ("large variance")

Moment-projection

- Minimizes $\text{KL}(p||q)$, that is, the expected information loss under p if q is used instead of p .
- Factorizes as
$$\text{KL}(p||q) = -\mathbb{E}_p[\ln q(\mathbf{z})] - \mathcal{H}_p.$$
- Preference given to q that has:
 - High probability allocated to regions that are probable according to p .
 - Non-zero probability given to any region with p is non-negligible.
 - No explicit focus of entropy

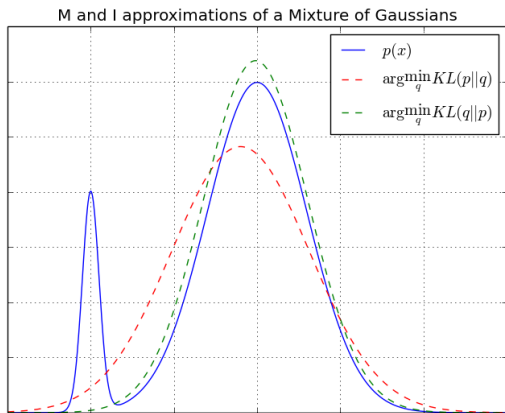
Cheat-sheet:

- **KL-divergence:** $\text{KL}(f||g) = \int_{\mathbf{z}} f(\mathbf{z}) \log \left(\frac{f(\mathbf{z})}{g(\mathbf{z})} \right) d\mathbf{z} = \mathbb{E}_f \left[\log \left(\frac{f(\mathbf{z})}{g(\mathbf{z})} \right) \right].$
- **Entropy:** $\mathcal{H}_f = - \int_{\mathbf{z}} f(\mathbf{z}) \log (f(\mathbf{z})) d\mathbf{z} = -\mathbb{E}_f [\log (f(\mathbf{z}))].$

Code Task: Moment and Information projection in Python

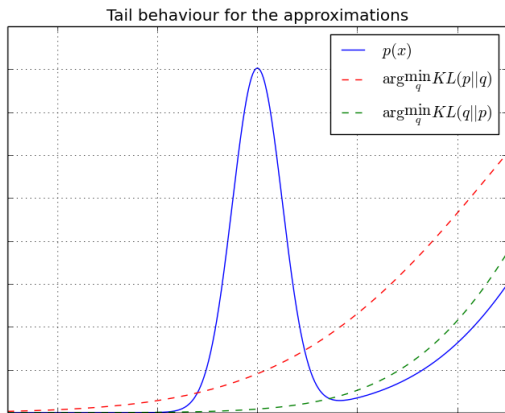
- We define a target function `p_distribution_pdf`, in this case a Mixture of Gaussians.
- Furthermore, we have an approximation `q_distribution_pdf`, a single Gaussians parameterized by μ and σ .
- Use `scipy.minimize`, which implements numerical minimization of scalar functions, to find the pair (μ, σ) that minimizes
 - Information-projection $KL(q||p)$.
 - Moment-projection $KL(p||q)$.
- Start from the partial implementation

```
students_M_and_I_projections.ipynb.
```
- Things to check:
 - What happens if `p_distribution_pdf` is a Gaussian (obtain this using, e.g., by setting `w = 0.` in `p_distribution_pdf`?)
 - Do you see general patterns showing differences in behaviour of the two solutions?



- **Moment projection** – optimizing $KL(p||q)$ – has slightly larger variance.
- Similar means, but **Information projection** – optimizing $KL(q||p)$ – focuses mainly on the most prominent mode.

Moment and Information projection – main difference



- **Moment projection** – optimizing $KL(p||q)$ – has slightly larger variance.
- Similar means, but **Information projection** – optimizing $KL(q||p)$ – focuses mainly on the most prominent mode.
- **I-projection** is *zero-forcing*, while **M-projection** is *zero-avoiding*.

Variational Bayes w/ Mean Field

VB uses information projections:

Variational Bayes uses *information projections*, i.e., approximates $p(\mathbf{x} | \mathbf{z})$ by

$$\hat{q}(\mathbf{z}) = \arg \min_{q \in \mathcal{Q}} \text{KL}(q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x}))$$

- **Positives:**

- Very efficient inference when combined with cleverly chosen \mathcal{Q} .
- General formulation for exponential family distributions (a specific distributional families we will cover next time).
- Clever interpretation when used for (Bayesian) learning.

- **Negatives:**

- As we have seen, this may result in *zero-forcing* behaviour.
 - The typical choice of \mathcal{Q} can make this issue even more prominent.
- Somewhat difficult when working outside the class of exponential family distributions.

Notice how we can rearrange the KL divergence as follows:

$$\begin{aligned}\text{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right] \\ &= \mathbb{E}_q \left[\log \frac{q(\mathbf{z}) \cdot p(\mathbf{x})}{p(\mathbf{z}|\mathbf{x}) \cdot p(\mathbf{x})} \right] \\ &= \log p(\mathbf{x}) - \mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} \right] = \log p(\mathbf{x}) - \mathcal{L}(q)\end{aligned}$$

where the Evidence Lower Bound (ELBO) is $\mathcal{L}(q) = -\mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} \right]$.

Notice how we can rearrange the KL divergence as follows:

$$\begin{aligned}
 \text{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right] \\
 &= \mathbb{E}_q \left[\log \frac{q(\mathbf{z}) \cdot p(\mathbf{x})}{p(\mathbf{z}|\mathbf{x}) \cdot p(\mathbf{x})} \right] \\
 &= \log p(\mathbf{x}) - \mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} \right] = \log p(\mathbf{x}) - \mathcal{L}(q)
 \end{aligned}$$

where the Evidence Lower Bound (ELBO) is $\mathcal{L}(q) = -\mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} \right]$.

VB focuses on ELBO:

$$\log p(\mathbf{x}) = \mathcal{L}(q) + \text{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$$

Since $\log p(\mathbf{x})$ is constant wrt. q and $\text{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) \geq 0$ it follows:

- We can minimize $\text{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$ by maximizing $\mathcal{L}(q)$
- This is **computationally simpler** because it uses $p(\mathbf{z}, \mathbf{x})$ instead of $p(\mathbf{z}|\mathbf{x})$.
- $\mathcal{L}(q)$ is a *lower bound* of the marginal data log likelihood $\log p(\mathbf{x})$.

↪ During inference, we will look for $\hat{q}(\mathbf{z}) = \arg \max_{q \in \mathcal{Q}} \mathcal{L}(q)$.

Bayesian learning using VB:

- We posit a model $p(\mathbf{y} \mid \boldsymbol{\theta})$, and want to learn $\boldsymbol{\theta}$ from a data-set $\mathcal{D} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$.
- We cast the problem in a Bayesian setting by including a prior $p(\boldsymbol{\theta})$.
- We seek the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$, which is approximated by

$$\hat{q}(\boldsymbol{\theta}) = \arg \min_{q \in \mathcal{Q}} \text{KL}(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta} \mid \mathcal{D})).$$

- This is identical to maximizing $\mathcal{L}(q)$ – a lower-bound of $p(\mathcal{D})$.

Bayesian learning using VB:

- We posit a model $p(\mathbf{y} \mid \boldsymbol{\theta})$, and want to learn $\boldsymbol{\theta}$ from a data-set $\mathcal{D} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$.
- We cast the problem in a Bayesian setting by including a prior $p(\boldsymbol{\theta})$.
- We seek the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$, which is approximated by

$$\hat{q}(\boldsymbol{\theta}) = \arg \min_{q \in \mathcal{Q}} \text{KL}(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta} \mid \mathcal{D})).$$

- This is identical to maximizing $\mathcal{L}(q)$ – a lower-bound of $p(\mathcal{D})$.

Things to take notice of:

- When using variational Bayes to approximate $p(\boldsymbol{\theta} \mid \mathcal{D})$, the resulting $\hat{q}(\boldsymbol{\theta})$ is chosen as the $q \in \mathcal{Q}$ that makes the data most probable.
- The VB objective is therefore reasonable in a learning-setting.
- The ELBO is a natural “quality score”, much like expected log likelihood is in an EM-setting.

What we have ...

We now have the first building-block of the approximation:

$$\Delta(q; p) = \text{KL} (q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x})) .$$

We still need the set \mathcal{Q} :

We will use the **mean field assumption**, which states that \mathcal{Q} consists of all distributions that *factorizes* according to the equation

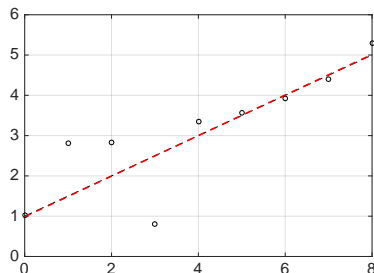
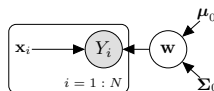
$$q(\mathbf{z}) = \prod_i q_i (z_i)$$

Note! This may seem like a very restricted set. However, we can choose any $q(\mathbf{z}) \in \mathcal{Q}$, and this is how the magic (\sim “absorbing information from \mathbf{x} ”) happens.

Simple example:

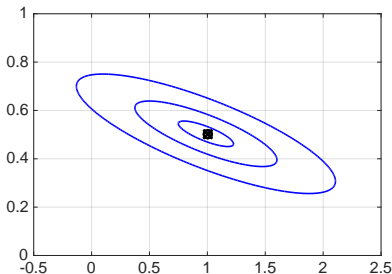
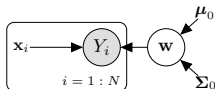
- Regression model:

$$Y_i \mid \{\mathbf{w}, \mathbf{x}_i\} = \mathbf{w}^\top \mathbf{x}_i + \epsilon_i.$$
 - For notational convenience we write \mathbf{x}_i for $[1, x_i]^\top$, x_i is a scalar.
- $\epsilon \sim N(0, 1/\gamma)$; γ known.
- $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2 \times 2})$.



Simple example:

- Regression model:
 $Y_i \mid \{\mathbf{w}, \mathbf{x}_i\} = \mathbf{w}^\top \mathbf{x}_i + \epsilon_i.$
 - For notational convenience we write \mathbf{x}_i for $[1, x_i]^\top$, x_i is a scalar.
- $\epsilon \sim N(0, 1/\gamma)$; γ known.
- $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2 \times 2}).$

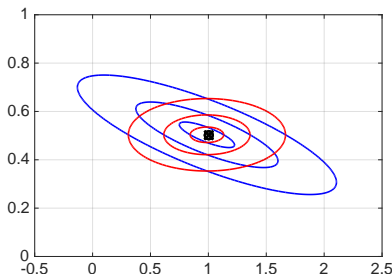
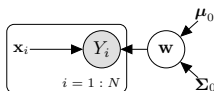


Exact Bayesian solution:

$$\mathbf{w} \mid \{\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\} \sim \mathcal{N} \left(\gamma(\mathbf{I}_d + \gamma \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, (\mathbf{I}_d + \gamma \mathbf{X}^\top \mathbf{X})^{-1} \right).$$

Simple example:

- Regression model:
 $Y_i | \{\mathbf{w}, \mathbf{x}_i\} = \mathbf{w}^\top \mathbf{x}_i + \epsilon_i.$
 - For notational convenience we write \mathbf{x}_i for $[1, x_i]^\top$, x_i is a scalar.
- $\epsilon \sim N(0, 1/\gamma)$; γ known.
- $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2 \times 2})$.

**Exact Bayesian solution:**

$$\mathbf{w} | \{\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\} \sim \mathcal{N} \left(\gamma(\mathbf{I}_d + \gamma \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, (\mathbf{I}_d + \gamma \mathbf{X}^\top \mathbf{X})^{-1} \right).$$

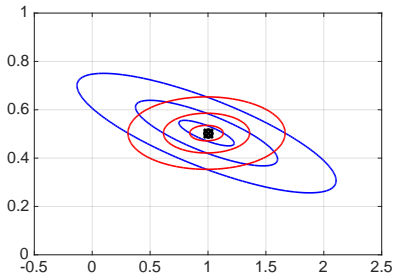
Variational Bayesian solution w/ Mean Field:

- Iterative approach; assumes factorized posterior.

Code Task: Implement VB for the regression example

- We are to find the approximation $q(w_0, w_1)$, which under the MF is defined by $q(w_0, w_1) = q(w_0) \cdot q(w_1)$.
- Each $q(w_j)$ is a Gaussian, parameterized by its mean and precision (inverse variance). We will optimize the parameters, that are denoted `q_0_mean`, `q_0_prec`, `q_1_mean`, and `q_1_prec` in the code.
- The update rules are given as follows:
 - $q_0_prec \leftarrow 1 + \gamma \cdot N$.
 - $q_0_mean \leftarrow (\gamma \cdot \sum_i y_i - q_1_mean \cdot \sum_i x_i) / q_0_prec$.
 - $q_1_prec \leftarrow 1 + \gamma \cdot \sum_i x_i^2$.
 - $q_1_mean \leftarrow \gamma \cdot (\sum_i x_i y_i - q_0_mean \cdot \sum_i x_i) / q_1_prec$.
- Notice how the different elements interact. To converge we need to run a couple of iterations (say, around 25).
- Start from the partial implementation

`students_linreg_vb.ipynb`.



- The **VB-MF** solution approximates the mean of the **true posterior** well.
 - Not always the case – depends on Q as well.
- **VB-MF** totally disregards correlation between W_1 and W_2 .
- **VB-MF** under-estimates the uncertainty of the **true posterior**, as shown by the 10%, 50% and 90% credibility intervals.
 - **VB-MF**'s tendency to under-estimate the uncertainty is evident for each marginal, as well as for the full joint.

- Approximate inference can be seen as *optimization*, where we look for a “simple” distribution $q(\mathbf{z})$ that is close to a “difficult” distribution $p(\mathbf{z} | \mathbf{x})$.
- First we define a class \mathcal{Q} of functions that are considered simple. In particular the *mean-field* assumption focus on distributions that *factorize*:

$$q(\mathbf{z}) = \prod_i q_i(z_i | \lambda_i).$$

- In VB, parameters (λ_i for each q_i) are chosen to minimize $\text{KL}(q||p)$.
- Practically we define the Evidence Lower Bound (ELBO),

$$\mathcal{L}(q) = -\mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} \right],$$

and look for $\hat{q}(\mathbf{z}) = \arg \max_{q \in \mathcal{Q}} \mathcal{L}(q)$.