Variational inference

Introduction to VB

Helge Langseth and Thomas Dyhre Nielsen

Introduction

Plan for this weeks

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

Recap from last time

- 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

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

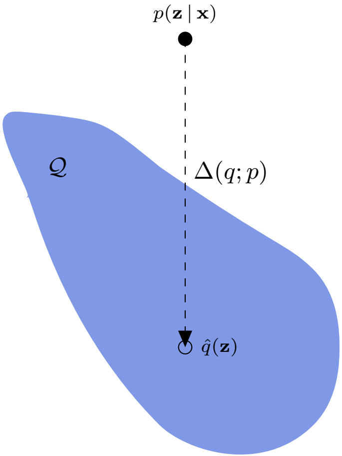
$$\hat{q}(\mathbf{z}) = \arg\min_{q \in \mathcal{Q}} \Delta(q(\mathbf{z}); p(\mathbf{z} \mid \mathbf{x}))$$

Remaining decisions to be made:

- How to define a family of distributions $\mathcal{Q}$ that is both flexible enough to generate god 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 $\mathcal{Q}$.

Approximate inference as projections in “distribution space”

- 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 $\Delta(f;g) \leq \Delta(f;h) + \Delta(h;g)$.

Distance measure

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 whe working with probability distributions

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

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

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

Two alternative KL projections

Information-projection

- Minimizes $\mathrm{KL}\left(q||p\right)$, that is, the expected information loss under q if p is used instead of q.
- Factorizes as $\mathrm{KL}\left(q||p\right) = -\,\mathbb{E}_q[\ln p(\mathbf{z})] - \mathcal{H}_q$.
- 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 $\mathrm{KL}\left(p||q\right)$, that is, the expected information loss under p if q is used instead of p.
- Factorizes as $\mathrm{KL}\left(p||q\right) = -\,\mathbb{E}_p[\ln q(\mathbf{z})] - \mathcal{H}_p$.
- 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

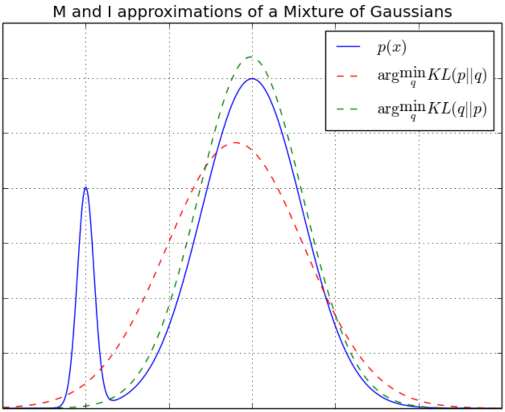
Cheat-sheet:

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

Code Task: Moment and Information projection in Python

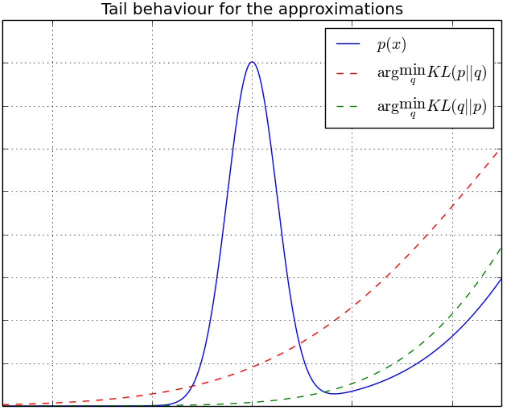
- We define a target function `p_distribution_pdf`, in this case a Mixture of Gaussians.
- Furthermore, we have an approximation `q_distribution_pdf`, a single Gaussians parameterized by μ and σ.
- Use `scipy.minimize`, which implements numerical minimization of scalar functions, to find the pair (μ,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



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

Moment and Information projection – main difference



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

Variational Bayes w/ Mean Field

Variational Bayes setup

VB uses information projections:

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

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

- **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.

ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

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

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

ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

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

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

VB focuses on ELBO:

$$\log p(\mathbf{x}) = \mathcal{L}\left(q\right) + \mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right)$$

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

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

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

Bayesian learning using VB:

- We posit a model $p(\mathbf{y} \mid \boldsymbol{\theta})$, and want to learn $\boldsymbol{\theta}$ from a data-set $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$.
- 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

$$\hat{q}(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta} \mid \mathcal{D})\right).$$

- This is identical to maximizing $\mathcal{L}\left(q\right)$ – 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 $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$.
- 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

$$\hat{q}(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta} \mid \mathcal{D})\right).$$

- 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.

The mean field assumption

What we have ...

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

$$\Delta(q;p) = \mathrm{KL}\left(q(\mathbf{z}) || p(\mathbf{z} \mid \mathbf{x})\right).$$

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

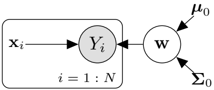
$$q(\mathbf{z}) = \prod_i q_i\left(z_i\right)$$

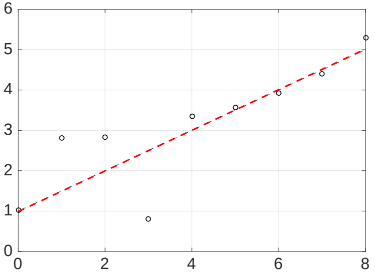
Note! This may seem like a very restricted set. However, we can choose any $q(\mathbf{z}) \in \mathcal{Q}$, and this is how the magic ($\sim$ “absorbing information from $\mathbf{x}$”) happens.

VB-MF example – “sanity check”

Simple example:

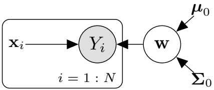
- Regression model:
  $Y_i \mid \{\mathbf{w}, \mathbf{x}_i\} = \mathbf{w}^\mathsf{T}\mathbf{x}_i + \epsilon_i$.
  - For notational convenience we write $\mathbf{x}_i$ for $[1, x_i]^\mathsf{T}$, x_i is a scaler.
- $\epsilon \sim N(0, 1/\gamma)$; γ known.
- $\mathbf{w} \sim \mathcal{N}\left(\boldsymbol{\mu}_0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

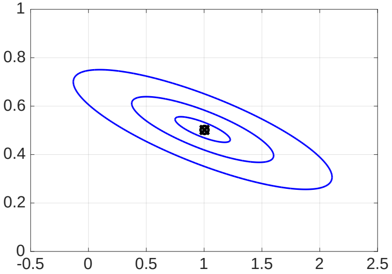




Simple example:

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





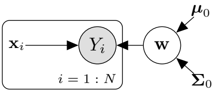
Exact Bayesian solution:

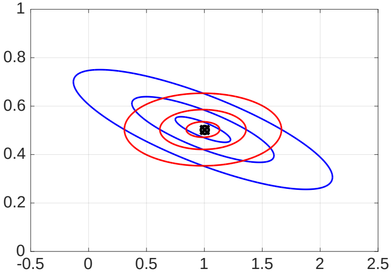
$$\mathbf{w} \mid \{\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\} \sim \mathcal{N}\left(\gamma(\mathbf{I}_d + \gamma \mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}, (\mathbf{I}_d + \gamma \mathbf{X}^\intercal \mathbf{X})^{-1}\right).$$

VB-MF example – “sanity check”

Simple example:

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





Exact Bayesian solution:

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

Variational Bayesian solution w/ Mean Field:

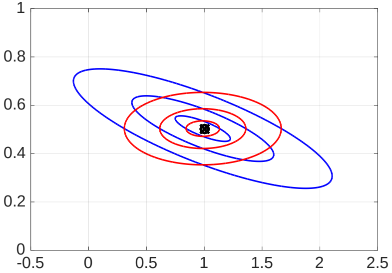
- Iterative approach; assumes factorized posterior.

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:
  - $\texttt{q\_0\_prec} \leftarrow 1 + \gamma \cdot N$.
  - $\texttt{q\_0\_mean} \leftarrow (\gamma \cdot \sum_i y_i - \texttt{q\_1\_mean} \cdot \sum_i x_i)/\texttt{q\_0\_prec}$.
  - $\texttt{q\_1\_prec} \leftarrow 1 + \gamma \cdot \sum_i x_i^2$.
  - $\texttt{q\_1\_mean} \leftarrow \gamma \cdot (\sum_i x_i y_i - \texttt{q\_0\_mean} \cdot \sum_i x_i)/\texttt{q\_1\_prec}$.
- 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_projections.ipynb`.

Important observations from the example



- The VB-MF solution approximates the mean of the true posterior well.
  - Not always the case – depends on $\mathcal{Q}$ 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.

Summary

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

$$q(\mathbf{z}) = \prod_i q_i(z_i \mid \boldsymbol{\lambda}_i).$$

- In VB, parameters ($\boldsymbol{\lambda}_i$ for each q_i) are chosen to minimize $\mathrm{KL}\left(q || p\right)$.
- Practically we define the Evidence Lower Bound (ELBO),

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

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