Bayesian linear classifiers: Laplace's method and variational Bayes

Brooks Paige

Week 4

Bayesian linear classifiers

For this lecture we're going to be focusing on a simple family of Bayesian linear classifiers. Suppose we have data $\mathcal{D} = \{\mathbf{x}_i, y_i\}$, $i = 1, \dots, N$, and a fixed feature map ϕ such that $\phi(\mathbf{x}_i) \in \mathbb{R}^D$.

For now we will focus only on binary classification problems, where $y_i \in \{0, 1\}$.

$$\begin{aligned} \mathbf{w} &\sim \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0) & \mathbf{w} &\in \mathbb{R}^D \\ y_i | \mathbf{x}_i &\sim \mathrm{Bernoulli}(\sigma(\mathbf{w}^\top \phi(\mathbf{x}_i))) & i &= 1, \dots, N \end{aligned}$$

where

$$\sigma(z) = \frac{1}{1 + \exp(z)}.$$

Generally, we will have a prior $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{S}_0 = \alpha^{-1}\mathbf{I}$.

Goals

Our goals for the moment are

1. Estimate the posterior

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathbf{w}) \prod_{i=1}^{N} p(y_i|\mathbf{w}, \mathbf{x}_i)}{p(y_{1:N}|\mathbf{x}_{1:N})}$$

2. Make predictions for new data $\mathbf{x}'$,

$$p(y'|\mathbf{x}', \mathcal{D}) = \int p(y'|\mathbf{w}, \mathbf{x}')p(\mathbf{w}|\mathcal{D})\mathrm{d}\mathbf{w}$$

MAP estimation

Now that we know all about optimization, we should have no trouble doing MAP estimation of $\mathbf{w}$! Define the log probability objective function

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \log p(\mathbf{w}) + \sum_{i=1}^{N} \log p(y_i|\mathbf{w}, \mathbf{x}_i) \\ &\equiv \log p(\mathbf{w}|\mathcal{D}) + \text{const}\end{aligned}$$

and find $\hat{\mathbf{w}}^{MAP} = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w})$, e.g. using gradient descent.

Warning: minibatches and priors

One thing to watch out for is to make sure the prior is properly scaled when doing mini-batch estimates! Suppose we would like to evaluate this objective function using a $M \ll N$ data point subset. This rescaling only affects the *likelihood*, not the entire objective!

$$\sum_{i=1}^{N} \log p(y_i|\mathbf{w}, \mathbf{x}_i) \approx \frac{N}{M} \sum_{j=1}^{M} \log p(y_j|\mathbf{w}, \mathbf{x}_j)$$

suggests that a mini-batch estimator for the whole loss should be

$$\hat{\mathcal{L}}_M(\mathbf{w}) = \log p(\mathbf{w}) + \frac{N}{M} \sum_{j=1}^{M} \log p(y_j|\mathbf{w}, \mathbf{x}_j).$$

Note in practice we often optimize $\frac{1}{N}\hat{\mathcal{L}}_M(\mathbf{w})$ instead!

What to do for non-Gaussian posteriors?

Two basic approaches:

- **Finite sample approximation** to the posterior
  - e.g., a collection of candidate weights $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(L)}$
  - So far: Markov chain Monte Carlo (we'll see some more approaches for deep learning, later!)

What to do for non-Gaussian posteriors?

Two basic approaches:

- **Finite sample approximation** to the posterior
  - e.g., a collection of candidate weights $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(L)}$
  - So far: Markov chain Monte Carlo (we'll see some more approaches for deep learning, later!)
- **Parametric approximation** to the posterior $p(\mathbf{w}|\mathcal{D})$
  - e.g., a Gaussian distribution $q(\mathbf{w})$ over weights
  - Variational Bayes; Laplace approximations

What to do for non-Gaussian posteriors?

Two basic approaches:

- **Finite sample approximation** to the posterior
  - e.g., a collection of candidate weights $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(L)}$
  - So far: Markov chain Monte Carlo (we'll see some more approaches for deep learning, later!)
- **Parametric approximation** to the posterior $p(\mathbf{w}|\mathcal{D})$
  - e.g., a Gaussian distribution $q(\mathbf{w})$ over weights
  - Variational Bayes; Laplace approximations

We know **the Gaussian distribution is wrong**. But we can still try to find some "best choice" parameters that get us "close", i.e. with $q(\mathbf{w}) \approx p(\mathbf{w}|\mathcal{D})$!

The Laplace approximation

Laplace approximation justification

In general: suppose we have found the value $\mathbf{w}^\star$ that maximizes of $\log p(\mathbf{w}|\mathcal{D})$.

- Take a Taylor expansion of $\log p(\mathbf{w}|\mathcal{D})$ in the vicinity of $\mathbf{w}^\star$. For small $\boldsymbol{\delta}$,

$$\log p(\mathbf{w}^\star + \boldsymbol{\delta}|\mathcal{D}) \approx \log p(\mathbf{w}^\star|\mathcal{D}) + \boldsymbol{\delta}^\top \mathbf{g} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{H}\boldsymbol{\delta},$$

where the gradient and Hessian are, respectively,

$$\mathbf{g} = \nabla \log p(\mathbf{w}|\mathcal{D})\Big|_{\mathbf{w}=\mathbf{w}^\star} \qquad \mathbf{H} = \nabla\nabla \log p(\mathbf{w}|\mathcal{D})\Big|_{\mathbf{w}=\mathbf{w}^\star}.$$

Laplace approximation justification

In general: suppose we have found the value $\mathbf{w}^\star$ that maximizes of $\log p(\mathbf{w}|\mathcal{D})$.

- Take a Taylor expansion of $\log p(\mathbf{w}|\mathcal{D})$ in the vicinity of $\mathbf{w}^\star$. For small $\boldsymbol{\delta}$,

$$\log p(\mathbf{w}^\star + \boldsymbol{\delta}|\mathcal{D}) \approx \log p(\mathbf{w}^\star|\mathcal{D}) + \boldsymbol{\delta}^\top \mathbf{g} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{H}\boldsymbol{\delta},$$

  where the gradient and Hessian are, respectively,

$$\mathbf{g} = \nabla \log p(\mathbf{w}|\mathcal{D})\Big|_{\mathbf{w}=\mathbf{w}^\star} \qquad \mathbf{H} = \nabla\nabla \log p(\mathbf{w}|\mathcal{D})\Big|_{\mathbf{w}=\mathbf{w}^\star}.$$

- Since $\mathbf{w}^\star$ is a mode of $\log p(\mathbf{w}|\mathcal{D})$, $\mathbf{g} = \mathbf{0}$, so taking $\mathbf{w} = \mathbf{w}^\star + \boldsymbol{\delta}$ we have

$$\log p(\mathbf{w}|\mathcal{D}) \approx \log p(\mathbf{w}^\star|\mathcal{D}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top \mathbf{H}(\mathbf{w} - \mathbf{w}^\star).$$

Laplace approximation justification

Now note that the first term is constant w.r.t. $\mathbf{w}$,

$$\log p(\mathbf{w}|\mathcal{D}) \approx \underbrace{\log p(\mathbf{w}^\star|\mathcal{D})}_{\text{const}} + \frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top \mathbf{H}(\mathbf{w} - \mathbf{w}^\star).$$

Laplace approximation justification

Now note that the first term is constant w.r.t. $\mathbf{w}$,

$$\log p(\mathbf{w}|\mathcal{D}) \approx \underbrace{\log p(\mathbf{w}^\star|\mathcal{D})}_{\text{const}} + \frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top \mathbf{H}(\mathbf{w} - \mathbf{w}^\star).$$

This suggests a Gaussian approximation with $\boldsymbol{\Lambda} = -\mathbf{H} = -\nabla\nabla \log p(\mathbf{w}|\mathcal{D})$,

$$\log p(\mathbf{w}|\mathcal{D}) \approx \log \mathcal{N}(\mathbf{w}|\mathbf{w}^\star, \boldsymbol{\Lambda}^{-1}) = -\frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top \boldsymbol{\Lambda}(\mathbf{w} - \mathbf{w}^\star) + \text{const}.$$

Laplace approximation justification

Now note that the first term is constant w.r.t. $\mathbf{w}$,

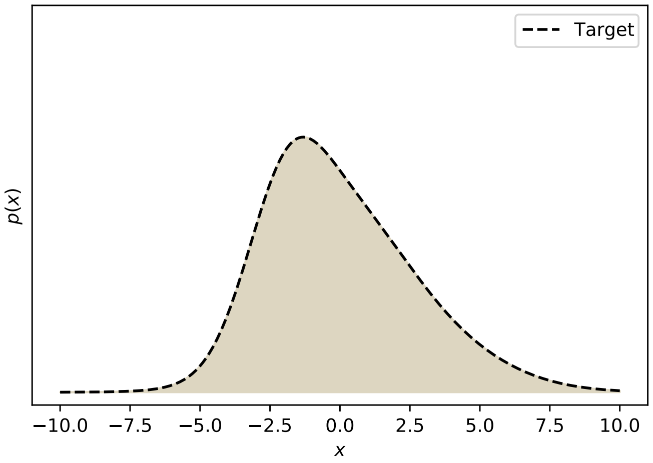
$$\log p(\mathbf{w}|\mathcal{D}) \approx \underbrace{\log p(\mathbf{w}^\star|\mathcal{D})}_{\text{const}} + \frac{1}{2}(\mathbf{w}-\mathbf{w}^\star)^\top \mathbf{H}(\mathbf{w}-\mathbf{w}^\star).$$

This suggests a Gaussian approximation with $\boldsymbol{\Lambda} = -\mathbf{H} = -\nabla\nabla \log p(\mathbf{w}|\mathcal{D})$,

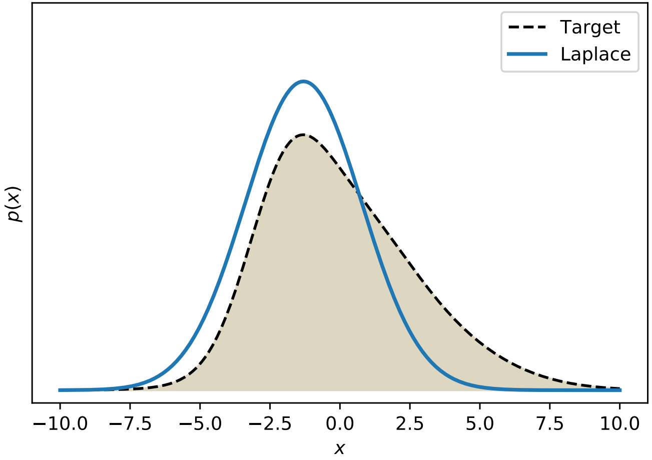
$$\log p(\mathbf{w}|\mathcal{D}) \approx \log \mathcal{N}(\mathbf{w}|\mathbf{w}^\star, \boldsymbol{\Lambda}^{-1}) = -\frac{1}{2}(\mathbf{w}-\mathbf{w}^\star)^\top \boldsymbol{\Lambda}(\mathbf{w}-\mathbf{w}^\star) + \text{const}.$$

Works nicely if posterior is more-or-less Gaussian and the Hessian is not too expensive to compute!

Laplace approximation cartoon



Laplace approximation cartoon



Hessian for Bayesian logistic regression

To apply this to logistic regression, first write the negative log target density,

$$-\log p(\mathbf{w}|\mathcal{D}) = -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(y_i|\mathbf{w}, \mathbf{x}_i) + \text{const}$$

Hessian for Bayesian logistic regression

To apply this to logistic regression, first write the negative log target density,

$$\begin{aligned}
-\log p(\mathbf{w}|\mathcal{D}) &= -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(y_i|\mathbf{w}, \mathbf{x}_i) + \text{const} \\
&= \frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^\top \mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0) \\
&\quad - \sum_{i=1}^{N} \big(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\big) + \text{const}
\end{aligned}$$

where $\hat{y}_i = \sigma(\mathbf{w}^\top \phi(\mathbf{x}_i))$.

Hessian for Bayesian logistic regression

To apply this to logistic regression, first write the negative log target density,

$$\begin{aligned}
-\log p(\mathbf{w}|\mathcal{D}) &= -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(y_i|\mathbf{w},\mathbf{x}_i) + \text{const} \\
&= \frac{1}{2}(\mathbf{w}-\mathbf{m}_0)^\top \mathbf{S}_0^{-1}(\mathbf{w}-\mathbf{m}_0) \\
&\quad - \sum_{i=1}^{N} \big(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\big) + \text{const}
\end{aligned}$$

where $\hat{y}_i = \sigma(\mathbf{w}^\top \phi(\mathbf{x}_i))$. The covariance $\mathbf{S}_N$ is then its inverse Hessian

$$\mathbf{S}_N^{-1} = -\nabla\nabla \log p(\mathbf{w}|\mathcal{D}) = \mathbf{S}_0^{-1} + \sum_{i=1}^{N} \hat{y}_i(1-\hat{y}_i)\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top.$$

Making predictions (1/2)

This gives us a posterior approximation $p(\mathbf{w}|\mathcal{D}) \approx q(\mathbf{w}) \equiv \mathcal{N}(\mathbf{w}|\hat{\mathbf{w}}_{MAP}, \mathbf{S}_N)$.

How do we actually make predictions for some new point $\mathbf{x}$? Unfortunately,

$$p(y=1|\mathbf{x},\mathcal{D}) \approx \int p(y=1|\mathbf{x},\mathbf{w})q(\mathbf{w})\mathrm{d}\mathbf{w} = \int \sigma(\mathbf{w}^\top \phi(\mathbf{x}))q(\mathbf{w})\mathrm{d}\mathbf{w}$$

is still something we can't compute exactly.

Making predictions (1/2)

This gives us a posterior approximation $p(\mathbf{w}|\mathcal{D}) \approx q(\mathbf{w}) \equiv \mathcal{N}(\mathbf{w}|\hat{\mathbf{w}}_{MAP}, \mathbf{S}_N)$.

How do we actually make predictions for some new point $\mathbf{x}$? Unfortunately,

$$p(y=1|\mathbf{x},\mathcal{D}) \approx \int p(y=1|\mathbf{x},\mathbf{w})q(\mathbf{w})\mathrm{d}\mathbf{w} = \int \sigma(\mathbf{w}^\top\phi(\mathbf{x}))q(\mathbf{w})\mathrm{d}\mathbf{w}$$

is still something we can't compute exactly.

Easy approach: simple Monte Carlo. Sample $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(L)} \sim q(\mathbf{w})$, and evaluate

$$\mathbb{E}_{q(\mathbf{w})}[\sigma(\mathbf{w}^\top\phi(\mathbf{x}))] \approx \frac{1}{L}\sum_{\ell=1}^{L}\sigma(\mathbf{w}^{(\ell)\top}\phi(\mathbf{x})).$$

Making predictions (2/2)

Interesting alternative: It turns out this D-dimensional integral can be re-written as a one-dimensional integral, by defining $a = \mathbf{w}^\top \phi(\mathbf{x})$, and just computing an integral along this projection. Let

$$\mu_a = \hat{\mathbf{w}}_{MAP}^\top \phi(\mathbf{x}), \qquad \sigma_a^2 = \phi(\mathbf{x})^\top \mathbf{S}_N \phi(\mathbf{x}).$$

Making predictions (2/2)

Interesting alternative: It turns out this D-dimensional integral can be re-written as a one-dimensional integral, by defining $a = \mathbf{w}^\top \phi(\mathbf{x})$, and just computing an integral along this projection. Let

$$\mu_a = \hat{\mathbf{w}}_{MAP}^\top \phi(\mathbf{x}), \qquad \sigma_a^2 = \phi(\mathbf{x})^\top \mathbf{S}_N \phi(\mathbf{x}).$$

Then with $p(a) = \mathcal{N}(a|\mu_a, \sigma_a^2)$,

$$\mathbb{E}_{q(\mathbf{w})}[\sigma(\mathbf{w}^\top \phi(\mathbf{x}))] = \mathbb{E}_{p(a)}[\sigma(a)] = \int \sigma(a) \mathcal{N}(a|\mu_a, \sigma_a^2) \mathrm{d}a.$$

This still doesn't have a closed form solution — but it could be easily numerically integrated.

[Note: for proof, see Bishop PRML Chapter 4.5.2]

Making predictions (2/2)

Interesting alternative: It turns out this D-dimensional integral can be re-written as a one-dimensional integral, by defining $a = \mathbf{w}^\top \phi(\mathbf{x})$, and just computing an integral along this projection. Let

$$\mu_a = \hat{\mathbf{w}}_{MAP}^\top \phi(\mathbf{x}), \qquad \sigma_a^2 = \phi(\mathbf{x})^\top \mathbf{S}_N \phi(\mathbf{x}).$$

Then with $p(a) = \mathcal{N}(a|\mu_a, \sigma_a^2)$,

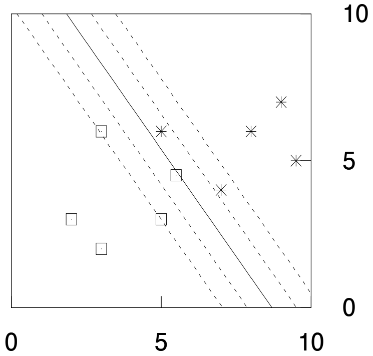
$$\mathbb{E}_{q(\mathbf{w})}[\sigma(\mathbf{w}^\top \phi(\mathbf{x}))] = \mathbb{E}_{p(a)}[\sigma(a)] = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2)\mathrm{d}a.$$

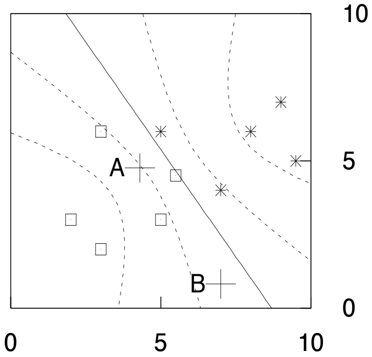
This still doesn't have a closed form solution — but it could be easily numerically integrated.

It also can be closely approximated by $p(y = 1|\mathbf{x}, \mathcal{D}) \approx \sigma\left(\frac{\mu_a}{\sqrt{1+\pi\sigma_a^2/8}}\right)$.

[Note: for proof, see Bishop PRML Chapter 4.5.2]

What's the difference?





Note that the symmetric "decision boundary", at $p(y = 1) = 0.5$, is the same.
But this will still have an effect on a decision boundary, if losses are asymmetric.

Figure: MacKay Chapter 41

Thoughts on Laplace approximations

- Nice improvement over "doing nothing", but quite limited: approximations are Gaussian, which may be a poor match for the true posterior shape
- Can be used for model selection, and for online learning (not discussed at this point)
- There's no reason to believe the curvature of the mode characterizes any "global" properties of the distribution
- Computing Hessian matrices will be unpleasant in higher-dimensional problems
- Can be applied to multi-class problems as well — just requires changing the Hessian matrix (though I'm not aware of any alternatives to Monte Carlo integration for predictions)

Variational Bayes

Motivation

In doing the Laplace approximation, we found a Gaussian posterior approximation. But:

1. Is this the **best** Gaussian approximation? Or is another one better? How would we compare?

2. What if we want an approximating distribution that is different than a Gaussian?

Variational inference

The Laplace approximation isn't the only way to find a Gaussian posterior approximation. Another approach is **variational inference**, in which we directly search for a Gaussian

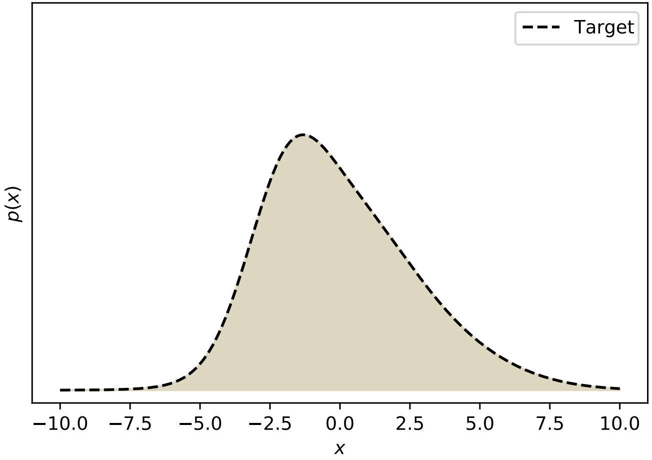
$$q_\lambda(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \approx p(\mathbf{w}|\mathcal{D}).$$

The most popular way to do this (for deep networks) is to minimize the **Kullback-Leibler** divergence w.r.t. $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$,

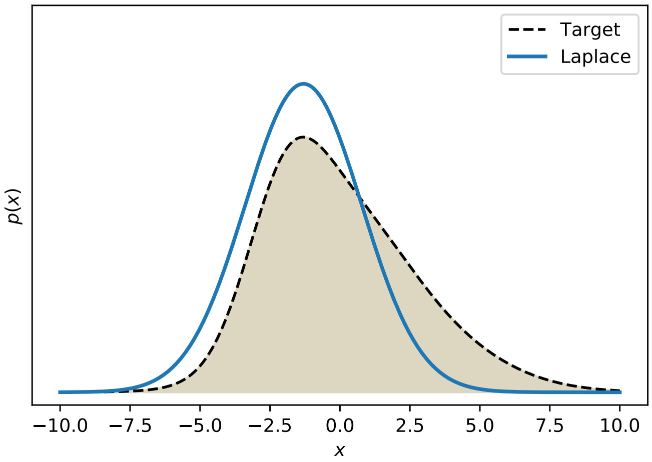
$$D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D})) = \int q_\lambda(\mathbf{w}) \log \frac{q_\lambda(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})} d\mathbf{w}.$$

The **KL divergence** isn't a "distance", but it measures dissimilarity between distributions: $D_{KL} \geq 0$, and $D_{KL}(q\|p) = 0$ only if $q = p$ almost everywhere.

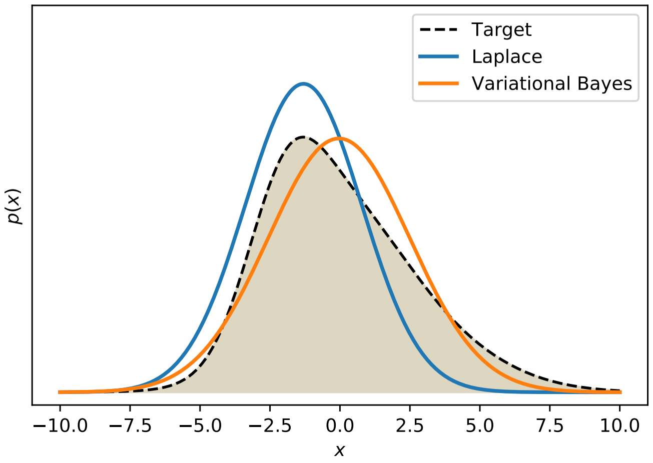
Variational Bayes cartoon



Variational Bayes cartoon



Variational Bayes cartoon



Evidence lower bound

We can re-arrange the KL divergence so the integrand only contains quantities we can evaluate pointwise:

$$D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D})) = \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{q_\lambda(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})}\right]$$

Evidence lower bound

We can re-arrange the KL divergence so the integrand only contains quantities we can evaluate pointwise:

$$\begin{aligned} D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D})) &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{q_\lambda(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})}\right] \\ &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{q_\lambda(\mathbf{w})p(\mathcal{D})}{p(\mathbf{w},\mathcal{D})}\right] \end{aligned}$$

Evidence lower bound

We can re-arrange the KL divergence so the integrand only contains quantities we can evaluate pointwise:

$$\begin{aligned}
D_{KL}(q_\lambda(\mathbf{w})\|p(\mathbf{w}|\mathcal{D})) &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log\frac{q_\lambda(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log\frac{q_\lambda(\mathbf{w})p(\mathcal{D})}{p(\mathbf{w},\mathcal{D})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log p(\mathcal{D})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log\frac{q_\lambda(\mathbf{w})}{p(\mathbf{w},\mathcal{D})}\right]
\end{aligned}$$

Evidence lower bound

We can re-arrange the KL divergence so the integrand only contains quantities we can evaluate pointwise:

$$\begin{aligned}
D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D})) &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{q_\lambda(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{q_\lambda(\mathbf{w})p(\mathcal{D})}{p(\mathbf{w},\mathcal{D})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log p(\mathcal{D})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{q_\lambda(\mathbf{w})}{p(\mathbf{w},\mathcal{D})}\right] \\
&= \log p(\mathcal{D}) - \underbrace{\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w},\mathcal{D})}{q_\lambda(\mathbf{w})}\right]}_{ELBO}
\end{aligned}$$

Evidence lower bound (ELBO)

Re-arranging the result on the last slide,

$$\underbrace{\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right]}_{ELBO,\ \mathcal{L}(\lambda,\mathcal{D})} = \underbrace{\log p(\mathcal{D})}_{\text{const.}} - \underbrace{D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D}))}_{\geq 0}$$

Evidence lower bound (ELBO)

Re-arranging the result on the last slide,

$$\underbrace{\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right]}_{ELBO,\ \mathcal{L}(\lambda,\mathcal{D})} = \underbrace{\log p(\mathcal{D})}_{\text{const.}} - \underbrace{D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D}))}_{\geq 0}$$

$$\mathcal{L}(\lambda, \mathcal{D}) \leq \log p(\mathcal{D}).$$

This is why it is called the **ELBO**.

Evidence lower bound (ELBO)

Re-arranging the result on the last slide,

$$\underbrace{\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right]}_{ELBO,\ \mathcal{L}(\lambda,\mathcal{D})} = \underbrace{\log p(\mathcal{D})}_{\text{const.}} - \underbrace{D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}|\mathcal{D}))}_{\geq 0}$$

$$\mathcal{L}(\lambda, \mathcal{D}) \leq \log p(\mathcal{D}).$$

This is why it is called the **ELBO**.

Our main question for now, though, is how to find $\arg\max_\lambda \mathcal{L}(\lambda, \mathcal{D})$!

Maximizing the ELBO

Doing this by gradient descent requires computing gradients of an expectation:

$$\nabla_\lambda \mathcal{L}(\lambda, \mathcal{D}) = \nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})} \left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})} \right].$$

Score function estimator (1/3)

Interestingly, it's possible to construct a "score function" estimator in nearly any setting. To do this, we will need the following two useful identities.

From the chain rule for logarithms:

$$\nabla_\lambda q_\lambda(\mathbf{w}) = q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}).$$

Score function estimator (1/3)

Interestingly, it's possible to construct a "score function" estimator in nearly any setting. To do this, we will need the following two useful identities.

From the chain rule for logarithms:

$$\nabla_\lambda q_\lambda(\mathbf{w}) = q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}).$$

Then, we have $\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] = 0$.

Score function estimator (1/3)

Interestingly, it's possible to construct a "score function" estimator in nearly any setting. To do this, we will need the following two useful identities.

From the chain rule for logarithms:

$$\nabla_\lambda q_\lambda(\mathbf{w}) = q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}).$$

Then, we have $\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] = 0$.

$$\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] = \int q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w}$$

Derivation in e.g. Ranganath et al. (2014), *Black box variational inference*

Score function estimator (1/3)

Interestingly, it's possible to construct a "score function" estimator in nearly any setting. To do this, we will need the following two useful identities.

From the chain rule for logarithms:

$$\nabla_\lambda q_\lambda(\mathbf{w}) = q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}).$$

Then, we have $\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] = 0$.

$$\begin{aligned}\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] &= \int q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w} \\ &= \int \nabla_\lambda q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w}\end{aligned}$$

Derivation in e.g. Ranganath et al. (2014), *Black box variational inference*

Score function estimator (1/3)

Interestingly, it's possible to construct a "score function" estimator in nearly any setting. To do this, we will need the following two useful identities.

From the chain rule for logarithms:

$$\nabla_\lambda q_\lambda(\mathbf{w}) = q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}).$$

Then, we have $\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] = 0$.

$$\begin{aligned}
\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] &= \int q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&= \int \nabla_\lambda q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&= \nabla_\lambda \int q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w}
\end{aligned}$$

Derivation in e.g. Ranganath et al. (2014), *Black box variational inference*

Score function estimator (1/3)

Interestingly, it's possible to construct a "score function" estimator in nearly any setting. To do this, we will need the following two useful identities.

From the chain rule for logarithms:

$$\nabla_\lambda q_\lambda(\mathbf{w}) = q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}).$$

Then, we have $\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] = 0$.

$$\begin{aligned}
\mathbb{E}_{q_\lambda(\mathbf{w})}[\nabla_\lambda \log q_\lambda(\mathbf{w})] &= \int q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&= \int \nabla_\lambda q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&= \nabla_\lambda \int q_\lambda(\mathbf{w}) \mathrm{d}\mathbf{w} = \nabla_\lambda 1 = 0.
\end{aligned}$$

Derivation in e.g. Ranganath et al. (2014), *Black box variational inference*

Score function estimator (2/3)

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] &= \int \nabla_\lambda q_\lambda(\mathbf{w}) f(\mathbf{w},\lambda) \mathrm{d}\mathbf{w} \\ &= \end{aligned}$$

Score function estimator (2/3)

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] &= \int \nabla_\lambda q_\lambda(\mathbf{w}) f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\ &= \int f(\mathbf{w},\lambda)\nabla_\lambda q_\lambda(\mathbf{w}) + q_\lambda(\mathbf{w})\nabla_\lambda f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w}\end{aligned}$$

Score function estimator (2/3)

$$\begin{aligned}
\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] &= \int \nabla_\lambda q_\lambda(\mathbf{w}) f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\
&= \int f(\mathbf{w},\lambda)\nabla_\lambda q_\lambda(\mathbf{w}) + q_\lambda(\mathbf{w})\nabla_\lambda f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\
&= \int q_\lambda(\mathbf{w})\big(f(\mathbf{w},\lambda)\nabla_\lambda \log q_\lambda(\mathbf{w}) + \nabla_\lambda f(\mathbf{w},\lambda)\big)\mathrm{d}\mathbf{w}
\end{aligned}$$

Score function estimator (2/3)

$$\begin{aligned}
\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] &= \int \nabla_\lambda q_\lambda(\mathbf{w}) f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\
&= \int f(\mathbf{w},\lambda)\nabla_\lambda q_\lambda(\mathbf{w}) + q_\lambda(\mathbf{w})\nabla_\lambda f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\
&= \int q_\lambda(\mathbf{w})\big(f(\mathbf{w},\lambda)\nabla_\lambda \log q_\lambda(\mathbf{w}) + \nabla_\lambda f(\mathbf{w},\lambda)\big)\mathrm{d}\mathbf{w} \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\nabla_\lambda \log q_\lambda(\mathbf{w})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\nabla_\lambda f(\mathbf{w},\lambda)\right].
\end{aligned}$$

Score function estimator (2/3)

$$\begin{aligned}
\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] &= \int \nabla_\lambda q_\lambda(\mathbf{w}) f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\
&= \int f(\mathbf{w},\lambda)\nabla_\lambda q_\lambda(\mathbf{w}) + q_\lambda(\mathbf{w})\nabla_\lambda f(\mathbf{w},\lambda)\mathrm{d}\mathbf{w} \\
&= \int q_\lambda(\mathbf{w})\big(f(\mathbf{w},\lambda)\nabla_\lambda \log q_\lambda(\mathbf{w}) + \nabla_\lambda f(\mathbf{w},\lambda)\big)\mathrm{d}\mathbf{w} \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\nabla_\lambda \log q_\lambda(\mathbf{w})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\nabla_\lambda f(\mathbf{w},\lambda)\right].
\end{aligned}$$

With $f(\mathbf{w},\lambda) = \log p(\mathbf{w},\mathcal{D}) - \log q_\lambda(\mathbf{w})$, then the second term

$$\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\nabla_\lambda\left(\log p(\mathbf{w},\mathcal{D}) - \log q_\lambda(\mathbf{w})\right)\right] = -\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\nabla_\lambda \log q_\lambda(\mathbf{w})\right] = 0.$$

Score function estimator (3/3)

This leaves us with

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] = \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w},\mathcal{D})}{q_\lambda(\mathbf{w})} \nabla_\lambda \log q_\lambda(\mathbf{w})\right].$$

Since this is an expectation under $q_\lambda(\mathbf{w})$, we can Monte Carlo evaluate it with a finite set of samples.

Score function estimator (3/3)

This leaves us with

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w}, \lambda)\right] = \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})} \nabla_\lambda \log q_\lambda(\mathbf{w})\right].$$

Since this is an expectation under $q_\lambda(\mathbf{w})$, we can Monte Carlo evaluate it with a finite set of samples.

Unfortunately, this typically leads to very high-variance estimates of the gradient. Intuitively, this estimator involves

1. Draw samples from the approximation $q_\lambda(\mathbf{w})$
2. Compute the gradients $\nabla_\lambda \log q_\lambda(\mathbf{w})$
3. Estimate the expectation by "weighting" those according to $\log \frac{p(\mathbf{w},\mathcal{D})}{q_\lambda(\mathbf{w})}$.

Score function estimator (3/3)

This leaves us with

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[f(\mathbf{w},\lambda)\right] = \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log\frac{p(\mathbf{w},\mathcal{D})}{q_\lambda(\mathbf{w})}\nabla_\lambda \log q_\lambda(\mathbf{w})\right].$$

Since this is an expectation under $q_\lambda(\mathbf{w})$, we can Monte Carlo evaluate it with a finite set of samples.

Unfortunately, this typically leads to very high-variance estimates of the gradient. Intuitively, this estimator involves

1. Draw samples from the approximation $q_\lambda(\mathbf{w})$
2. Compute the gradients $\nabla_\lambda \log q_\lambda(\mathbf{w})$
3. Estimate the expectation by “weighting” those according to $\log\frac{p(\mathbf{w},\mathcal{D})}{q_\lambda(\mathbf{w})}$.

Note that **nowhere** in here did we use $\nabla_{\mathbf{w}} p(\mathbf{w},\mathcal{D})\ldots$!

Re-parameterization

The "trick" here is to re-parameterize the expectation so that it does not involve λ. For Gaussians, this is easy:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

is the same as

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}), \qquad \mathbf{w} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}.$$

Here, $\boldsymbol{\Sigma}^{1/2}$ could be a Cholesky decomposition of $\boldsymbol{\Sigma}$, or (for larger models) $\boldsymbol{\Sigma}$ will often be a diagonal matrix with entries $\mathrm{diag}(\boldsymbol{\sigma}^2)$, in which case $\boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon} = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$.

Re-parameterization

With that in mind, define $p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$, define

$$\mathbf{w} = r(\lambda, \boldsymbol{\epsilon}) \equiv r(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}.$$

Then

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] = \nabla_\lambda \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right]$$

Re-parameterization

With that in mind, define $p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$, define

$$\mathbf{w} = r(\lambda, \boldsymbol{\epsilon}) \equiv r(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}.$$

Then

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] &= \nabla_\lambda \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\lambda \log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right].\end{aligned}$$

Re-parameterization

With that in mind, define $p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$, define

$$\mathbf{w} = r(\lambda, \boldsymbol{\epsilon}) \equiv r(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}.$$

Then

$$\begin{aligned} \nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] &= \nabla_\lambda \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\lambda \log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right]. \end{aligned}$$

I know it feels like cheating! But, in this case we can exchange the order of the expectation and the gradient. Now just use **autodiff**. . .

How does this now work?

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\lambda \log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right]$$

- In practice, draw a single sample $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, and use it to compute $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$
- The computation of $\boldsymbol{\mu}$ is **exactly the same** as if it were a fixed parameter $\mathbf{w}$ we were optimizing
- Using a diagonal approximation $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma})$, then at a cost of $\times 2$ parameters relative to MAP optimization, we can also estimate a per-weight posterior standard deviation

Use mini-batch estimates of the gradient

For models where $\log p(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$, then we can happily use mini-batch estimators:

$$\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right]$$

Use mini-batch estimates of the gradient

For models where $\log p(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$, then we can happily use mini-batch estimators:

$$\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] = \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i, \mathbf{w}) + \log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right]$$

Use mini-batch estimates of the gradient

For models where $\log p(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$, then we can happily use mini-batch estimators:

$$
\begin{aligned}
\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i, \mathbf{w}) + \log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i, \mathbf{w})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right]
\end{aligned}
$$

Use mini-batch estimates of the gradient

For models where $\log p(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$, then we can happily use mini-batch estimators:

$$\begin{aligned}
\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log p(y_i|\mathbf{x}_i, \mathbf{w})\right] - D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}))
\end{aligned}$$

Use mini-batch estimates of the gradient

For models where $\log p(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$, then we can happily use mini-batch estimators:

$$
\begin{aligned}
\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log p(y_i|\mathbf{x}_i, \mathbf{w})\right] - D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w})) \\
&\approx \frac{N}{M} \sum_{j=1}^{M} \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log p(y_j|\mathbf{x}_j, \mathbf{w})\right] - D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}))
\end{aligned}
$$

Use mini-batch estimates of the gradient

For models where $\log p(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$, then we can happily use mini-batch estimators:

$$\begin{aligned}
\mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w}, \mathcal{D})}{q_\lambda(\mathbf{w})}\right] &= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w})\right] + \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log \frac{p(\mathbf{w})}{q_\lambda(\mathbf{w})}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\log p(y_i|\mathbf{x}_i, \mathbf{w})\right] - D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w})) \\
&\approx \frac{N}{M} \sum_{j=1}^{M} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p(y_j|\mathbf{x}_j, r(\lambda, \boldsymbol{\epsilon}))\right] - D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}))
\end{aligned}$$

In Bayesian logistic regression

To apply this to Bayesian logistic regression, just drop in our likelihood, prior, and approximate posterior.

$$\begin{aligned}\mathcal{L}(\lambda, \mathcal{D}) \approx \mathbb{E}_{p(\boldsymbol{\epsilon})}\Bigg[&\frac{N}{M}\sum_{j=1}^{M} \log \mathrm{Bernoulli}(y_j | \mathbf{x}_j, r(\lambda, \boldsymbol{\epsilon})) \\ &+ \log \mathcal{N}(r(\lambda, \boldsymbol{\epsilon}) | \mathbf{m}_0, \mathbf{S}_0) - \log \mathcal{N}(r(\lambda, \boldsymbol{\epsilon}) | \boldsymbol{\mu}, \boldsymbol{\Sigma})\Bigg].\end{aligned}$$

Sample a value of $\boldsymbol{\epsilon}$, compute an estimate of the ELBO, autodiff, and then update with your favourite gradient descent algorithm.

Dealing with constraints

Suppose we want to fit an approximate posterior $q(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

While $\boldsymbol{\mu}$ can take any value at all, for $\boldsymbol{\Sigma}$ to be a valid covariance matrix it must be symmetric and positive definite. How can we ensure that gradient updates don't violate the constraint?

Transforming parameters

Current best practice: **transform parameters** to an **unconstrained** space, yielding an unconstrained optimization problem.

- For $\mathbf{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$, consider optimizing with respect to an “untransformed” variable z_d, such that $\sigma_d^2 = T(z_d)$, where T could be e.g. $T(z) = \exp(z)$ or $T(z) = \log(1 + \exp(z))$.

Transforming parameters

Current best practice: **transform parameters** to an **unconstrained** space, yielding an unconstrained optimization problem.

- For $\mathbf{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$, consider optimizing with respect to an "untransformed" variable z_d, such that $\sigma_d^2 = T(z_d)$, where T could be e.g. $T(z) = \exp(z)$ or $T(z) = \log(1 + \exp(z))$.
- For arbitrary $\mathbf{\Sigma}$, consider parameterizing as e.g.

$$\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^\top + \mathrm{diag}(\boldsymbol{\sigma}^2),$$

  where $\mathbf{A} \in \mathbb{R}^{D \times D}$ is an unconstrained matrix, and $\sigma \geq 0$ is enforced as in the diagonal case.

Transforming parameters

Current best practice: **transform parameters** to an **unconstrained** space, yielding an unconstrained optimization problem.

- For $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$, consider optimizing with respect to an "untransformed" variable z_d, such that $\sigma_d^2 = T(z_d)$, where T could be e.g. $T(z) = \exp(z)$ or $T(z) = \log(1 + \exp(z))$.
- For arbitrary $\boldsymbol{\Sigma}$, consider parameterizing as e.g.

  $$\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^\top + \mathrm{diag}(\boldsymbol{\sigma}^2),$$

  where $\mathbf{A} \in \mathbb{R}^{D \times D}$ is an unconstrained matrix, and $\sigma \geq 0$ is enforced as in the diagonal case.
- "Low-rank plus diagonal" approximate posteriors are also a common choice, i.e. using a rectangular matrix $\mathbf{A}$. This allows non-diagonal covariance matrices with far fewer than $\mathcal{O}(D^2)$ parameters.

Now what?

- As it turns out: $D_{KL}(q_\lambda(\mathbf{w}) \| p(\mathbf{w}))$ actually has a closed form solution if both distributions are Gaussians! Its gradient can be computed analytically.
- This means it's only necessary to Monte Carlo estimate the gradient of the $\mathbb{E}_{p(\epsilon)}\left[\log p(y_j|\mathbf{x}_j, r(\lambda, \boldsymbol{\epsilon}))\right]$ terms
- Note that the way we presented variational Bayes here is completely different than most textbooks, which instead derive closed-form coordinate ascent updates in conjugate exponential family models.
- Instead, all we did was assume our model was differentiable with respect to its latent variables. This approach might be less efficient, but we can apply it immediately to find Gaussian posteriors for any differentiable model…
- Also, there is nothing stopping us from picking alternative non-Gaussian distributions for $q_\lambda(\mathbf{w})$

Break

That’s actually probably enough for this week — let me know if you have any questions, and we’ll pick up from here next time.