where $0 < \eta_t \leq 1$ is the step size, and

$$\boldsymbol{g}_t = \frac{\partial}{\partial \boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_t)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \tag{6.166}$$

$$\mathbf{H}_t = \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\mathsf{T}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_t)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \tag{6.167}$$

If $\eta_t = 1$, [Lan95a] cals this the **gradient EM algorithm**. However, it is possible to use a larger step size to speed up the algorithm, as in the **quasi-Newton EM algorithm** of [Lan95b]. This method also replaces the Hessian in Equation (6.167), which may not be negative definite (for non exponential family models), with a BFGS approximation. This ensures the overall algorithm is an ascent algorithm. Note, however, when the M step cannot be computed in closed form, EM loses some of its appeal over directly optimizing the marginal likelihood with a gradient based solver.

6.6.6.5 ECM algorithm

The **ECM** algorithm stands for "expectation conditional maximization", and refers to optimizing the parameters in the M step sequentially, if they turn out to be dependent. The **ECME** algorithm, which stands for "ECM either" [LR95], is a variant of ECM in which we maximize the expected complete data log likelihood (the Q function) as usual, or the observed data log likelihood, during one or more of the conditional maximization steps. The latter can be much faster, since it ignores the results of the E step, and directly optimizes the objective of interest. A standard example of this is when fitting the Student T distribution. For fixed ν, we can update $\boldsymbol{\Sigma}$ as usual, but then to update ν, we replace the standard update of the form $\nu^{t+1} = \arg\max_\nu Q((\boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1}, \nu), \boldsymbol{\theta}^t)$ with $\nu^{t+1} = \arg\max_\nu \log p(\mathcal{D}|\boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1}, \nu)$. See [MK97] for more information.

6.6.6.6 Online EM

When dealing with large or streaming datasets, it is important to be able to learn online, as we discussed in Section 19.7.5. There are two main approaches to **online EM** in the literature. The first approach, known as **incremental EM** [NH98a], optimizes the lower bound $Q(\boldsymbol{\theta}, q_1, \ldots, q_N)$ one q_n at a time; however, this requires storing the expected sufficient statistics for each data case.

The second approach, known as **stepwise EM** [SI00; LK09; CM09], is based on stochastic gradient descent. This optimizes a local upper bound on $\ell_n(\boldsymbol{\theta}) = \log p(\boldsymbol{x}_n|\boldsymbol{\theta})$ at each step. (See [Mai13; Mai15] for a more general discussion of stochastic and incremental bound optimization algorithms.)

6.7 The Bayesian learning rule

in this section, we discuss the "**Bayesian learning rule**" [KR21a], which provides a unified framework for deriving many standard (and non-standard) optimization and inference algorithms used in the ML community.

To motivate the BLR, recall the standard **empirical risk minimzation** or **ERM** problem, which has the form $\boldsymbol{\theta}_* = \text{argmin}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta})$, where

$$\bar{\ell}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) + R(\boldsymbol{\theta}) \tag{6.168}$$

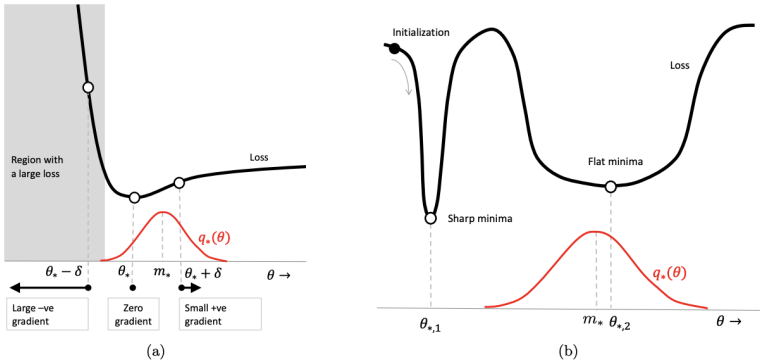

Figure 6.9: Illustration of the robustness obtained by using a Bayesian approach to parameter estimation. (a) When the minimum $\boldsymbol{\theta}_*$ lies next to a "wall", the Bayesian solution shifts away from the boundary to avoid large losses due to perturbations of the parameters. (b) The Bayesian solution prefers flat minima over sharp minima, to avoid large losses due to perturbations of the parameters. From Figure 1 of [KR21a]. Used with kind permission of Emtiyaz Khan.

where $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a prediction function, $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is a loss function, and $R(\boldsymbol{\theta})$ is some kind of regularizer.

Although the regularizer can prevent overfitting, the ERM method can still result in parameter estimates that are not robust. A better approach is to fit a *distribution* over possible parameter values, $q(\boldsymbol{\theta})$. If we minimize the expected loss, we will find parameter settings that will work well even if they are slightly perturbed, as illustrated in Figure 6.9, which helps with robustness and generalization. Of course, if the distribution q collapses to a single delta function, we will end up with the ERM solution. To prevent this, we add a penalty term, that measures the KL divergence from $q(\boldsymbol{\theta})$ to some prior $\pi_0(\boldsymbol{\theta}) \propto \exp(R(\boldsymbol{\theta}))$. This gives rise to the following BLR objective:

$$\mathcal{L}(q) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\sum_{n=1}^{N} \ell(\boldsymbol{y}_n, f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))\right] + D_{\mathbb{KL}}\left(q(\boldsymbol{\theta}) \parallel \pi_0(\boldsymbol{\theta})\right) \tag{6.169}$$

We can rewrite the KL term as

$$D_{\mathbb{KL}}\left(q(\boldsymbol{\theta}) \parallel \pi_0(\boldsymbol{\theta})\right) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[R(\boldsymbol{\theta})\right] - \mathbb{H}(q(\boldsymbol{\theta})) \tag{6.170}$$

and hence can rewrite the BLR objective as follows:

$$\mathcal{L}(q) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\bar{\ell}(\boldsymbol{\theta})\right] - \mathbb{H}(q(\boldsymbol{\theta})) \tag{6.171}$$

Below we show that different approximations to this objective recover a variety of different methods in the literature.

6.7.1 Deriving inference algorithms from BLR

In this section we show how to derive several different inference algorithms from BLR. (We discuss such algorithms in more detail in Chapter 10.)

6.7.1.1 Bayesian inference as optimization

The BLR objective includes standard exact Bayesian inference as a special case, as first shown in [Opt88]. To see this, let us assume the loss function is derived from a log-likelihood:

$$\ell(y, f_{\boldsymbol{\theta}}(\boldsymbol{x})) = -\log p(\boldsymbol{y}|f_{\boldsymbol{\theta}}(\boldsymbol{x})) \tag{6.172}$$

Let $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n) : n = 1 : N\}$ be the data we condition on. The Bayesian posterior can be written as

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{Z(\mathcal{D})}\pi_0(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{y}_n|f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) \tag{6.173}$$

This can be derived by minimizing the BLR, since

$$\mathcal{L}(q) = -\mathbb{E}_{q(\boldsymbol{\theta})}\left[\sum_{n=1}^{N} \log p(\boldsymbol{y}_n|f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))\right] + D_{\mathbb{KL}}\left(q(\boldsymbol{\theta}) \parallel \pi_0(\boldsymbol{\theta})\right) \tag{6.174}$$

$$= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log \frac{q(\boldsymbol{\theta})}{\frac{\pi_0(\boldsymbol{\theta})}{Z(\mathcal{D})}\prod_{n=1}^{N} p(\boldsymbol{y}_n|f_{\boldsymbol{\theta}}(\boldsymbol{x}_n))}\right] - \log Z(\mathcal{D}) \tag{6.175}$$

$$= D_{\mathbb{KL}}\left(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D})\right) - \log Z(\mathcal{D}) \tag{6.176}$$

Since $Z(\mathcal{D})$ is a constant, we can minimize the loss by by setting $q(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})$.

Of course, we can use other kinds of loss, not just log likelihoods. This results in a framework known as **generalized Bayesian inference** [BHW16; KJD19; KJD21]. See Section 14.1.3 for more discussion.

6.7.1.2 Optimization of BLR using natural gradient descent

In general, we cannot compute the exact posterior $q(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})$, so we seek an approximation. We will assume that $q(\boldsymbol{\theta})$ is an exponential family distibution, such as a multivariate Gaussian, where the mean represents the standard point estimate of $\boldsymbol{\theta}$ (as in ERM), and the covariance represents our uncertainty (as in Bayes). Hence q can be written as follows:

$$q(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) \exp[\boldsymbol{\lambda}^\mathsf{T}\mathcal{T}(\boldsymbol{\theta}) - A(\boldsymbol{\lambda})] \tag{6.177}$$

where $\boldsymbol{\lambda}$ are the natural parameters, $\mathcal{T}(\boldsymbol{\theta})$ are the sufficient statistics, $A(\boldsymbol{\lambda})$ is the log partition function, and $h(\boldsymbol{\theta})$ is the base measure, which is usually a constant. The BLR loss becomes

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q_{\boldsymbol{\lambda}}(\boldsymbol{\theta})}\left[\overline{\ell}(\boldsymbol{\theta})\right] - \mathbb{H}(q_{\boldsymbol{\lambda}}(\boldsymbol{\theta})) \tag{6.178}$$

We can optimize this using natural gradient descent (Section 6.4). The update becomes

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \eta_t \tilde{\nabla}_{\boldsymbol{\lambda}}\left[\mathbb{E}_{q_{\boldsymbol{\lambda}_t}(\boldsymbol{\theta})}\left[\overline{\ell}(\boldsymbol{\theta})\right] - \mathbb{H}(q_{\boldsymbol{\lambda}_t})\right] \tag{6.179}$$

where $\tilde{\nabla}_{\boldsymbol{\lambda}}$ denotes the natural gradient. We discuss how to compute these natural gradients in Section 6.4.5. In particular, we can convert it to regular gradients wrt the moment parameters $\boldsymbol{\mu}_t = \boldsymbol{\mu}(\boldsymbol{\lambda}_t)$. This gives

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \eta_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}_t}(\boldsymbol{\theta})}\left[\overline{\ell}(\boldsymbol{\theta})\right] + \eta_t \nabla_{\boldsymbol{\mu}}\, \mathbb{H}(q_{\boldsymbol{\mu}_t}) \tag{6.180}$$

From Equation (6.84) we have

$$\nabla_{\boldsymbol{\mu}} \mathbb{H}(q) = -\boldsymbol{\lambda} - \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\log h(\boldsymbol{\theta})\right] \tag{6.181}$$

Hence the update becomes

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \eta_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}_t}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] - \eta_t \boldsymbol{\lambda}_t - \eta_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\log h(\boldsymbol{\theta})\right] \tag{6.182}$$

$$= (1 - \eta_t)\boldsymbol{\lambda}_t - \eta_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta}) + \log h(\boldsymbol{\theta})\right] \tag{6.183}$$

For distributions q with constant base measure $h(\boldsymbol{\theta})$, this simplifies to

$$\boldsymbol{\lambda}_{t+1} = (1 - \eta_t)\boldsymbol{\lambda}_t - \eta_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] \tag{6.184}$$

Hence at the fixed point we have

$$\boldsymbol{\lambda}_* = (1 - \eta)\boldsymbol{\lambda}_* - \eta \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] \tag{6.185}$$

$$\boldsymbol{\lambda}_* = \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[-\overline{\ell}(\boldsymbol{\theta})\right] = \tilde{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}(\boldsymbol{\theta})} \left[-\overline{\ell}(\boldsymbol{\theta})\right] \tag{6.186}$$

6.7.1.3 Conjugate variational inference

In Section 7.3 we showed how to do exact inference in conjugate models. We can derive Equation (7.11) from the BLR by using the fixed point condition in Equation (6.186) to write

$$\boldsymbol{\lambda}_* = \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_*} \left[-\overline{\ell}(\boldsymbol{\theta})\right] = \boldsymbol{\lambda}_0 + \sum_{i=1}^{N} \underbrace{\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q*} \left[\log p(\boldsymbol{y}_i|\boldsymbol{\theta})\right]}_{\tilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i)} \tag{6.187}$$

where $\tilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i)$ are the sufficient statistics for the i'th likelihood term.

For models where the joint distribution over the latents factorizes (using a graphical model), we can further decompose this update into a series of local terms. This gives rise to the variational message passing scheme discussed in Section 10.2.7.

6.7.1.4 Partially conjugate variational inference

In Supplementary Section 10.3.1, we discuss CVI, which performs variational inference for partially conjugate models, using gradient updates for the non-conjugate parts, and exact Bayesian inference for the conjugate parts.

6.7.2 Deriving optimization algorithms from BLR

In this section we show how to derive several different optimization algorithms from BLR. Recall that in BLR, instead of directly minimizing the loss

$$\overline{\ell}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) + R(\boldsymbol{\theta}) \tag{6.188}$$

we will instead minimize

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\lambda})} \left[\overline{\ell}(\boldsymbol{\theta})\right] - \mathbb{H}(q(\boldsymbol{\theta}|\boldsymbol{\lambda})) \tag{6.189}$$

Below we show that different approximations to this objective recover a variety of different optimization methods that are used in the literature. (We discuss such algorithms in more detail in Chapter 6.)

6.7.2.1 Gradient descent

In this section, we show how to derive gradient descent as a special case of BLR. We use as our approximate posterior $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \mathbf{I})$. In this case the natural and moment parameters are equal, $\boldsymbol{\mu} = \boldsymbol{\lambda} = \boldsymbol{m}$. The base measure satisfies the following (from Equation (2.180)):

$$2 \log h(\boldsymbol{\theta}) = -D \log(2\pi) - \boldsymbol{\theta}^\mathsf{T} \boldsymbol{\theta} \tag{6.190}$$

Hence

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_q \left[\log h(\boldsymbol{\theta})\right] = \nabla_{\boldsymbol{\mu}}(-D \log(2\pi) - \boldsymbol{\mu}^\mathsf{T} \boldsymbol{\mu} - D) = -\boldsymbol{\mu} = -\boldsymbol{\lambda} = -\boldsymbol{m} \tag{6.191}$$

Thus from Equation (6.183) the BLR update becomes

$$\boldsymbol{m}_{t+1} = (1 - \eta_t)\boldsymbol{m}_t + \eta_t \boldsymbol{m}_t - \eta_t \nabla_{\boldsymbol{m}} \mathbb{E}_{q_{\boldsymbol{m}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] \tag{6.192}$$

We can remove the expectation using the first order delta method (Section 6.5.5):

$$\nabla_{\boldsymbol{m}} \mathbb{E}_{q_{\boldsymbol{m}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] \approx \nabla_{\boldsymbol{\theta}} \overline{\ell}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}} \tag{6.193}$$

Putting these together gives the gradient descent update:

$$\boldsymbol{m}_{t+1} = \boldsymbol{m}_t - \eta_t \nabla_{\boldsymbol{\theta}} \overline{\ell}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}_t} \tag{6.194}$$

6.7.2.2 Newton's method

In this section, we show how to derive Newton's second order optimization method as a special case of BLR, as first shown in [Kha+18].

Suppose we assume $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \mathbf{S}^{-1})$. The natural parameters are

$$\boldsymbol{\lambda}^{(1)} = \mathbf{S}\boldsymbol{m}, \; \boldsymbol{\lambda}^{(2)} = -\frac{1}{2}\mathbf{S} \tag{6.195}$$

The mean (moment) parameters are

$$\boldsymbol{\mu}^{(1)} = \boldsymbol{m}, \; \boldsymbol{\mu}^{(2)} = \mathbf{S}^{-1} + \boldsymbol{m}\boldsymbol{m}^\mathsf{T} \tag{6.196}$$

Since the base measure is constant (see Equation (2.193)), from Equation (6.184) we have

$$\mathbf{S}_{t+1}\boldsymbol{m}_{t+1} = (1 - \eta_t)\mathbf{S}_t \boldsymbol{m}_t - \eta_t \nabla_{\boldsymbol{\mu}^{(1)}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] \tag{6.197}$$

$$\mathbf{S}_{t+1} = (1 - \eta_t)\mathbf{S}_t + 2\eta_t \nabla_{\boldsymbol{\mu}^{(2)}} \mathbb{E}_{q_{\boldsymbol{\mu}}(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] \tag{6.198}$$

In Section 6.4.5.1 we show that

$$\nabla_{\boldsymbol{\mu}^{(1)}} \mathbb{E}_{q(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] = \mathbb{E}_{q(\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \overline{\ell}(\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})} \left[\nabla^2_{\boldsymbol{\theta}} \overline{\ell}(\boldsymbol{\theta})\right] \boldsymbol{m} \tag{6.199}$$

$$\nabla_{\boldsymbol{\mu}^{(2)}} \mathbb{E}_{q(\boldsymbol{\theta})} \left[\overline{\ell}(\boldsymbol{\theta})\right] = \frac{1}{2} \mathbb{E}_{q(\boldsymbol{\theta})} \left[\nabla^2_{\boldsymbol{\theta}} \overline{\ell}(\boldsymbol{\theta})\right] \tag{6.200}$$

Hence the update for the precision matrix becomes

$$\mathbf{S}_{t+1} = (1 - \eta_t)\mathbf{S}_t + \eta_t \mathbb{E}_{q_t} \left[\nabla^2_{\boldsymbol{\theta}} \overline{\ell}(\boldsymbol{\theta})\right] \tag{6.201}$$

For the precision weighted mean, we have

$$\mathbf{S}_{t+1}\boldsymbol{m}_{t+1} = (1-\eta_t)\mathbf{S}_t\boldsymbol{m}_t - \eta_t\mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] + \eta_t\mathbb{E}_{q_t}\left[\nabla^2_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right]\boldsymbol{m}_t \tag{6.202}$$

$$= \mathbf{S}_{t+1}\boldsymbol{m}_t - \eta_t\mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] \tag{6.203}$$

Hence

$$\boldsymbol{m}_{t+1} = \boldsymbol{m}_t - \eta_t\mathbf{S}^{-1}_{t+1}\mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] \tag{6.204}$$

We can recover Newton's method in three steps. First set the learning rate to $\eta_t = 1$, based on an assumption that the objective is convex. Second, treat the iterate as $\boldsymbol{m}_t = \boldsymbol{\theta}_t$. Third, apply the delta method to get

$$\mathbf{S}_{t+1} = \mathbb{E}_{q_t}\left[\nabla^2_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] \approx \nabla^2_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}_t} \tag{6.205}$$

and

$$\mathbb{E}_q\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] \approx \nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}_t} \tag{6.206}$$

This gives Newton's update:

$$\boldsymbol{m}_{t+1} = \boldsymbol{m}_t - [\nabla^2_{\boldsymbol{m}}\bar{\ell}(\boldsymbol{m}_t)]^{-1}[\nabla_{\boldsymbol{m}}\bar{\ell}(\boldsymbol{m}_t)] \tag{6.207}$$

6.7.2.3 Variational online Gauss-Newton

In this section, we describe the **Variational Online Gauss-Newton** or **VOGN** method of [Kha+18]. This is an approximate second order optimization method that can be derived from the BLR in several steps, as we show below.

First, we use a diagonal Gaussian approximation to the posterior, $q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_t, \mathbf{S}_t^{-1})$, where $\mathbf{S}_t = \mathrm{diag}(\boldsymbol{s}_t)$ is a vector of precisions. Following Section 6.7.2.2, we get the following updates:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t\frac{1}{\boldsymbol{s}_{t+1}} \odot \mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] \tag{6.208}$$

$$\boldsymbol{s}_{t+1} = (1-\eta_t)\boldsymbol{s}_t + \eta_t\mathbb{E}_{q_t}\left[\mathrm{diag}(\nabla^2_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}))\right] \tag{6.209}$$

where $\odot$ is elementwise multiplication, and the division by $\boldsymbol{s}_{t+1}$ is also elementwise.

Second, we use the delta approximation to replace expectations by plugging in the mean. Third we use a minibatch approximation to the gradient and diagonal Hessian:

$$\hat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}) = \frac{N}{M}\sum_{i\in\mathcal{M}}\nabla_{\boldsymbol{\theta}}\ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \nabla_{\boldsymbol{\theta}}R(\boldsymbol{\theta}) \tag{6.210}$$

$$\hat{\nabla}^2_{\theta_j}\bar{\ell}(\boldsymbol{\theta}) = \frac{N}{M}\sum_{i\in\mathcal{M}}\nabla^2_{\theta_j}\ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \nabla^2_{\theta_j}R(\boldsymbol{\theta}) \tag{6.211}$$

where M is the minibatch size.

For some non-convex problems, such as DNNs, the Hessian may be not be positive definite, so we can get better results using a Gauss-Newton approximation, based on the squared gradients instead of the Hessian:

$$\hat{\nabla}^2_{\theta_j} \bar{\ell}(\boldsymbol{\theta}) \approx \frac{N}{M} \sum_{i \in \mathcal{M}} [\nabla_{\theta_j} \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))]^2 + \nabla^2_{\theta_j} R(\boldsymbol{\theta}) \tag{6.212}$$

This is also faster to compute.

Putting all this together gives rise to the **Online Gauss-Newton** or **OGN** method of [Osa+19a].

If we drop the delta approximation, and work with expectations. we get the **Variational Online Gauss-Newton** or **VOGN** method of [Kha+18]. We can approximate the expectations by sampling. In particular, VOGN uses the following **weight perturbation** method

$$\mathbb{E}_{q_t}\left[\hat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta})\right] \approx \hat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}_t) \tag{6.213}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{s}_t))$. It also also possible to approximate the Fisher information matrix directly; this results in the **Variational Online Generalized Gauss-Newton** or **VOGGN** method of [Osa+19a].

6.7.2.4 Adaptive learning rate SGD

In this section, we show how to derive an update rule which is very similar to the **RMSprop** [Hin14] method, which is widely used in deep learning. The approach we take is similar to that VOGN in Section 6.7.2.3. We use the same diagonal Gaussian approximation, $q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_t, \mathbf{S}_t^{-1})$, where $\mathbf{S}_t = \text{diag}(\boldsymbol{s}_t)$ is a vector of precisions. We then use the delta method to eliminate expectations:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{1}{\boldsymbol{s}_{t+1}} \odot \nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \tag{6.214}$$

$$\boldsymbol{s}_{t+1} = (1 - \eta_t)\boldsymbol{s}_t + \eta_t \text{diag}(\nabla^2_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t)) \tag{6.215}$$

where $\odot$ is elementwise multiplication. If we allow for different learning rates we get

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \frac{1}{\boldsymbol{s}_{t+1}} \odot \nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \tag{6.216}$$

$$\boldsymbol{s}_{t+1} = (1 - \beta_t)\boldsymbol{s}_t + \beta_t \text{diag}(\hat{\nabla}^2_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t)) \tag{6.217}$$

Now suppose we replace the diagonal Hessian approximation with the sum of the squares per-sample gradients:

$$\text{diag}(\nabla^2_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t)) \approx \hat{\nabla} \bar{\ell}(\boldsymbol{\theta}_t) \odot \hat{\nabla} \bar{\ell}(\boldsymbol{\theta}_t) \tag{6.218}$$

If we also change some scaling factors we can get the RMSprop updates:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{1}{\sqrt{\boldsymbol{v}_{t+1}} + c\mathbf{1}} \odot \hat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \tag{6.219}$$

$$\boldsymbol{v}_{t+1} = (1 - \beta)\boldsymbol{v}_t + \beta[\hat{\nabla} \bar{\ell}(\boldsymbol{\theta}_t) \odot \hat{\nabla} \bar{\ell}(\boldsymbol{\theta}_t)] \tag{6.220}$$

This allows us to use standard deep learning optimizers to get a Gaussian approximation to the posterior for the parameters [Osa+19a].

It is also possible to derive the Adam optimizer [KB15] from BLR by adding a momentum term to RMSprop. See [KR21a; Ait18] for details.

6.7.3 Variational optimization

Consider an objective defined in terms of discrete variables. Such objectives are not differentiable and so are hard to optimize. One advantage of BLR is that it optimizes the parameters of a probability distribution, and such expected loss objectives are usually differentiable and smooth. This is called "**variational optimization**" [Bar17], since we are optimizing over a probability distribution.

For example, consider the case of a **binary neural network** where $\theta_d \in \{0, 1\}$ indicates if weight d is used or not, we can optimize over the parameters of a Bernoulli distribution, $q(\boldsymbol{\theta}|\boldsymbol{\lambda}) = \prod_{d=1}^{D} \text{Ber}(\theta_d|p_d)$, where $p_d \in [0, 1]$ and $\lambda_d = 1/2 \log(p_d/(1 - p_d)$ is the log odds. This is the basis of the **BayesBiNN** approach [MBK20].

If we ignore the entropy and regularizer term, we get the following simplified objective:

$$\mathcal{L}(\boldsymbol{\lambda}) = \int \bar{\ell}(\boldsymbol{\theta}) q(\boldsymbol{\theta}|\boldsymbol{\lambda}) d\boldsymbol{\theta} \tag{6.221}$$

This method has various names: **stochastic relaxation** [SB12; SB13; MMP13], **stochastic approximation** [HHC12; Hu+12], etc. It is closely related to **evolutionary strategies**, which we discuss in Section 6.9.6.

In the case of functions with continuous domains, we can use a Gaussian for $q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The resulting integral in Equation (6.221) can then sometimes be solved in closed form, as explained in [Mob16]. By starting with a broad variance, and gradually reducing it, we hope the method can avoid poor local optima, similar to simulated annealing (Section 12.9.1. However, we generally get better results by including the entropy term, because then we can automatically learn to adapt the variance. In addition, we can often work with natural gradients, which results in faster convergence.

6.8 Bayesian optimization

In this section, we discuss **Bayesian optimization** or **BayesOpt**, which is a model-based approach to black-box optimization, designed for the case where the objective function $f : \mathcal{X} \to \mathbb{R}$ is expensive to evaluate (e.g., if it requires running a simulation, or training and testing a particular neural net architecture).

Since the true function f is expensive to evaluate, we want to make as few function calls (i.e., make as few **queries** $\boldsymbol{x}$ to the **oracle** f) as possible. This suggests that we should build a **surrogate function** (also called a **response surface model**) based on the data collected so far, $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i) : i = 1 : n\}$, which we can use to decide which point to query next. There is an inherent tradeoff between picking the point $\boldsymbol{x}$ where we think $f(\boldsymbol{x})$ is large (we follow the convention in the literature and assume we are trying to maximize f), and picking points where we are uncertain about $f(\boldsymbol{x})$ but where observing the function value might help us improve the surrogate model. This is another instance of the exploration-exploitation dilemma.

In the special case where the domain we are optimizing over is finite, so $\mathcal{X} = \{1, \ldots, A\}$, the BayesOpt problem becomes similar to the **best arm identification** problem in the bandit literature (Section 34.4). An important difference is that in bandits, we care about the cost of every action we take, whereas in optimization, we usually only care about the cost of the final solution we find. In other words, in bandits, we want to minimize cumulative regret, whereas in optimization we want to minimize simple or final regret.