revrand: Technical Report

Daniel Steinberg DANIEL.STEINBERG@DATA61.CSIRO.AU
Louis Tiao LOUIS.TIAO@DATA61.CSIRO.AU
Alistair Reid ALISTAIR.REID@DATA61.CSIRO.AU
Lachlan McCalman LACHLAN.MCCALMAN@DATA61.CSIRO.AU
Simon O'Callaghan SIMON.OCALLAGHAN@DATA61.CSIRO.AU
DATA61, CSIRO
Sydney, Australia

Abstract

This is a technical report on the *revrand* software library. This library implements various Bayesian linear models (Bayesian linear regression), approximate Gaussian processes and generalised linear models. These algorithms have been implemented such that they can be used for large-scale inference by using stochastic gradients. All of the algorithms in *revrand* use a unified feature composition framework that allows for easy concatenation and selective application of regression basis functions.

Contents

1. Core Algorithms

1.1 Stochastic Gradients and Variational Objective Functions

Stochastic Gradients is now a ubiquitous method for optimisation when a whole dataset does not fit in memory, or when optimisation has to be distributed amongst many computational nodes.

When an objective function factorises over data,

$$f(\mathbf{X}, \theta) = \sum_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x}, \theta)\,, \tag{1}$$

a regular gradient descent would perform the following iterations to minimise the function w.r.t. θ,

$$\theta_k := \theta_{k-1} - \eta_k \sum_{\mathbf{x} \in \mathbf{X}} \nabla_\theta f(\mathbf{x}_n, \theta)|_{\theta=\theta_{k-1}}, \tag{2}$$

where η_k is the learning rate (step size) at iteration k. Stochastic gradients proposes the following update,

$$\theta_k := \theta_{k-1} - \eta_k \sum_{\mathbf{x} \in \mathbf{B}} \nabla_\theta f(\mathbf{x}, \theta)|_{\theta=\theta_{k-1}}, \tag{3}$$

where $\mathbf{B} \subset \mathbf{X}$ is a mini-batch of the original dataset, where $|\mathbf{B}| \ll |\mathbf{X}|$.

Unfortunately some objective functions to not entirely decompose over the data, i.e.

$$f(\mathbf{X}, \theta) = \sum_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x}, \theta) + g(\theta)\,. \tag{4}$$

Let $M = |\mathbf{B}|$ and $N = |\mathbf{X}|$, then we divide the contribution of the constant term amongst the mini-batches in stochastic gradients,

$$\theta_k := \theta_{k-1} - \eta_k \sum_{\mathbf{x} \in \mathbf{B}} \nabla_\theta f(\mathbf{x}, \theta)|_{\theta=\theta_{k-1}} - \frac{M}{N} \eta_k \nabla_\theta g(\theta)|_{\theta=\theta_{k-1}}. \tag{5}$$

or, equivalently, boost the contribution of the mini-batch,

$$\theta_k := \theta_{k-1} - \frac{N}{M} \eta_k \sum_{\mathbf{x} \in \mathbf{B}} \nabla_\theta f(\mathbf{x}, \theta)|_{\theta=\theta_{k-1}} - \eta_k \nabla_\theta g(\theta)|_{\theta=\theta_{k-1}}. \tag{6}$$

This is particularly relevant for variational inference where the evidence lower bound objective has a component independent of the data. For example, lets consider the model,

$$\text{Likelihood:} \quad \prod_{n=1}^{N} p(y_n|\theta)\,, \tag{7}$$

$$\text{prior:} \quad p(\theta|\alpha)\,, \tag{8}$$

where we want to learn the values of the hyper-parameters, α. Minimising negative log-marginal likelihood is a good objective in this instance, since we don't care about the value(s) of θ,

$$\underset{\alpha}{\operatorname{argmin}} - \log \int \prod_{n=1}^{N} p(y_n|\theta)\, p(\theta|\alpha)\, d\theta. \tag{9}$$

There are two problems with this objective however, (1) it may not factor over data and (2) the integral may be intractable, for instance, if the prior and likelihood are not conjugate. In variational inference we use Jensen's inequality to lower-bound log-marginal likelihood with a tractable objective function called the evidence lower bound (ELBO),

$$\begin{aligned} \log p(\mathbf{y}|\alpha) &= \log \int \prod_{n=1}^{N} p(y_n|\theta)\, p(\theta|\alpha)\, d\theta \\ &= \log \int \frac{\prod_n p(y_n|\theta)\, p(\theta|\alpha)}{q(\theta)} q(\theta)\, d\theta \\ &\geq \int q(\theta) \log \left[\frac{\prod_n p(y_n|\theta)\, p(\theta|\alpha)}{q(\theta)} \right] d\theta \end{aligned} \tag{10}$$

where $q(\theta)$ is an approximation of $p(\theta|\alpha)$ that makes inference easier. This can be re-written as,

$$\mathcal{L} = \sum_{n=1}^{N} \left\langle \log p(y_n|\theta) \right\rangle_q - \mathrm{KL}[q(\theta) \| p(\theta|\alpha)]\,, \tag{11}$$

which takes the form of Equation (4), and so if we use stochastic gradients optimisation we can weight the Kullback-Leibler term like the constant term, $g(\cdot)$, from Equation (5), or boost the expected log likelihood term like in Equation (6). Furthermore, if $q(\theta) = p(\theta|\alpha)$ then the lower bound is tight, and this will be equivalent to optimising log-marginal likelihood.

1.2 Bayesian Linear Regression

The first machine learning algorithm in revrand is a simple Bayesian linear regressor of the following form,

$$\text{Likelihood:} \quad \prod_{n=1}^{N} \mathcal{N}\left(y_n \middle| \boldsymbol{\phi}_n^\top \mathbf{w}, \sigma^2\right), \tag{12}$$

$$\text{prior:} \quad \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D), \tag{13}$$

where $\boldsymbol{\phi}_n := \phi(\mathbf{x}_n, \theta)$ is a feature, or basis, function that $\phi\mathbb{R}^d \rightarrow \mathbb{R}^D$. This is the same algorithm described in Rasmussen and Williams (2006, Chapter 2). We then:

- Optimise σ^2, λ and θ w.r.t. log-marginal likelihood,

$$\log p\left(\mathbf{y}\middle|\sigma^2, \lambda, \theta\right) = \log \mathcal{N}\left(\mathbf{y}\middle|\mathbf{0}, \sigma^2 \mathbf{I}_N + \lambda \boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right), \tag{14}$$

  where $\boldsymbol{\Phi} \in \mathbb{R}^{N \times D}$ is the concatenation of all the features, $\boldsymbol{\phi}_n$. Note this results in the covariance of the log-marginal likelihood being $N \times N$, though we can use the Woodbury identity to simplify the corresponding matrix inversion.

- Solve analytically for the posterior over weights, $\mathbf{w}|\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \mathbf{C})$ given the above hyperparameters, where,

$$\mathbf{C} = \left[\lambda \mathbf{I}_D + \frac{1}{\sigma^2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right]^{-1},$$
$$\mathbf{m} = \frac{1}{\sigma^2} \mathbf{C} \boldsymbol{\Phi}^\top \mathbf{y}.$$

- Use the predictive distribution

$$\begin{aligned} p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) &= \int \mathcal{N}\left(y^* \middle| \boldsymbol{\phi}^{*\top} \mathbf{w}, \sigma^2\right) \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{C})\, d\mathbf{w}, \\ &= \mathcal{N}\left(y^* \middle| \boldsymbol{\phi}^{*\top} \mathbf{m}, \sigma^2 + \boldsymbol{\phi}^{*\top} \mathbf{C} \boldsymbol{\phi}^*\right) \end{aligned} \tag{15}$$

  for query inputs, $\mathbf{x}^*$. This gives us the useful expectations,

$$\mathbb{E}[y^*] = \boldsymbol{\phi}^{*\top} \mathbf{m}, \tag{16}$$

$$\mathbb{V}[y^*] = \sigma^2 + \boldsymbol{\phi}^{*\top} \mathbf{C} \boldsymbol{\phi}^{*\top}. \tag{17}$$

It is actually easier to use the ELBO form with stochastic gradients for learning the parameters of this algorithm, rather than log-marginal likelihood recast using the Woodbury identity. This is because it is plainly in the same form as Equation (4), though it would give the same result as log-marginal likelihood, the "approximate" posterior is the same form as the true posterior, i.e. $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{C})$. The ELBO for this model is,

$$\mathcal{L} = \sum_{n=1}^{N} \left\langle \log \mathcal{N}\left(y_n \middle| \boldsymbol{\phi}_n^\top \mathbf{w}, \sigma^2\right) \right\rangle_q - \mathrm{KL}[\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{C}) \| \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D)] \,. \tag{18}$$

More specifically,

$$\begin{aligned} \left\langle \log \mathcal{N}\left(y_n \middle| \boldsymbol{\phi}_n^\top \mathbf{w}, \sigma^2\right) \right\rangle_q =& \log \mathcal{N}\left(y_n \middle| \boldsymbol{\phi}_n^\top \mathbf{m}, \sigma^2\right) - \frac{1}{2\sigma^2} \mathrm{tr}\left(\boldsymbol{\phi}_n^\top \boldsymbol{\phi}_n \mathbf{C}\right), \\ \mathrm{KL}[\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{C}) \| \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D)] =& \frac{1}{2\lambda}\left[\mathrm{tr}(\mathbf{C}) + \mathbf{m}^\top \mathbf{m}\right] - \frac{1}{2}\log|\mathbf{C}| \\ & + \frac{D}{2}\left(\log \lambda - 1\right). \end{aligned}$$

We have not implemented a stochastic gradient version of this algorithm since it still requires a determinant involving a $D \times D$ matrix, and so is $\mathcal{O}(D^3)$ in complexity, per iteration. This is true even if we optimise the posterior covariance directly (or a triangular parameterisation). The GLM presented in the next section circumvents this issue, and is more suited to really large N and D problems.

1.3 Bayesian Generalised Linear Models

The algorithm of primary interest in *revrand* is the Bayesian generalised linear model. The general form of the model implemented by this algorithm is,

$$\text{Likelihood:} \quad \prod_{n=1}^{N} p\left(y_n \middle| g\left(\boldsymbol{\phi}_n^\top \mathbf{w}\right), \gamma\right), \tag{19}$$

$$\text{prior:} \quad \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D), \tag{20}$$

for an arbitrary univariate likelihood, $p(\cdot)$, with an appropriate transformation (inverse link) function, $g(\cdot)$, and parameter(s), γ.

Naturally, both calculating the exact posterior over the weights, $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$, and the log-marginal likelihood, $p(\mathbf{y})$, for hyperparameter learning are intractable since we may have a non-conjugate relationship between the likelihood and prior. Therefore we must resort to approximating the true posterior and the log-marginal likelihood.

Firstly, we approximate the true posterior over weights with a mixture of K diagonal Gaussians,

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}, \mathbf{X}) &\approx q(\mathbf{w}), \\ &= \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k), \end{aligned} \tag{21}$$

where $\boldsymbol{\Psi}_k = \mathrm{diag}\big([\Psi_{k,1}, \ldots, \Psi_{k,D}]^\top\big)$, which is inspired from similar approximations made in Gershman et al. (2012); Nguyen and Bonilla (2014). This is a very flexible form for the approximate posterior, and has the nice property that our algorithm no longer has a $\mathcal{O}\big(D^3\big)$ cost associated with the number of features.

Then we approximate the log marginal likelihood using auto-encoding variational Bayes (Kingma and Welling, 2014). The exact lower bound on log marginal likelihood is,

$$\mathcal{L} = \sum_{n=1}^{N} \left\langle \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top \mathbf{w}\Big), \gamma\Big)\right\rangle_q - \mathrm{KL}\big[\tfrac{1}{K}\textstyle\sum_k \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k) \big\| \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D)\big]\,. \tag{22}$$

This can be expanded,

$$\mathcal{L} = \frac{1}{K}\sum_{k=1}^{K}\sum_{n=1}^{N} \left\langle \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top \mathbf{w}\Big), \gamma\Big)\right\rangle_{q_k} + \frac{1}{K}\sum_{k=1}^{K} \langle \log \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda\mathbf{I}_D)\rangle_{q_k} + \mathbb{H}\left[\tfrac{1}{K}\textstyle\sum_k \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k)\right], \tag{23}$$

but unfortunately there are two intractable integrals here, the expected log likelihood, and the entropy of the Gaussian mixture. We can use the lower bound on the entropy term also used in Gershman et al. (2012); Nguyen and Bonilla (2014),

$$\mathbb{H}\left[\tfrac{1}{K}\textstyle\sum_k \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k)\right] \geq -\frac{1}{K}\sum_{k=1}^{K} \log \sum_{j=1}^{K} \frac{1}{K}\mathcal{N}(\mathbf{m}_k|\mathbf{m}_j, \boldsymbol{\Psi}_k + \boldsymbol{\Psi}_j)\,. \tag{24}$$

We can then use the reparameterisation trick in auto-encoding variational Bayes to sample the expected log likelihood,

$$\sum_{n=1}^{N} \left\langle \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top \mathbf{w}\Big), \gamma\Big)\right\rangle_{q_k} \approx \frac{1}{L}\sum_{l=1}^{L}\sum_{n=1}^{N} \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top f_k\Big(\mathbf{m}_k, \boldsymbol{\Psi}_k, \boldsymbol{\epsilon}^{(l)}\Big)\Big), \gamma\Big) \tag{25}$$

where,

$$f_k\Big(\mathbf{m}_k, \boldsymbol{\Psi}_k, \boldsymbol{\epsilon}^{(l)}\Big) = \mathbf{m}_k + \sqrt{\boldsymbol{\Psi}_k} \odot \boldsymbol{\epsilon}^{(l)}, \qquad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)\,. \tag{26}$$

Here $\odot$ is the element-wise product. We can also use this trick to compute approximate derivatives, $\frac{\partial}{\partial\alpha}\langle \log p(y|\alpha)\rangle_{q(\alpha)} \approx \frac{1}{L}\sum_{l=1}^{L} \frac{\partial}{\partial\alpha} \log p\big(y\big|f\big(\alpha, \boldsymbol{\epsilon}^{(l)}\big)\big)$, which simplifies the implementation greatly! The final auto-encoding variational Bayes objective for our GLM is,

$$\mathcal{L} \approx \frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{L}\sum_{l=1}^{L}\sum_{n=1}^{N} \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top f_k^{(l)}\Big), \gamma\Big) + \log \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda\mathbf{I}_D) - \frac{1}{2\lambda}\mathrm{tr}(\boldsymbol{\Psi}_k) - \log \sum_{j=1}^{K} \frac{1}{K}\mathcal{N}(\mathbf{m}_k|\mathbf{m}_j, \boldsymbol{\Psi}_k + \boldsymbol{\Psi}_j)\right]. \tag{27}$$

We can straight forwardly use this objective with in a stochastic gradients setting using with the tactic in Equations (5) or (6).

TODO: Gradients in appendix?

The most simple and accurate method for approximating the predictive distribution, $p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*)$ is to Monte-Carlo sample the integral,

$$p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) \approx \int p\Big(\mathbf{y}\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big) \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k)\, d\mathbf{w}. \tag{28}$$

However, this integral is not particularly useful unless we wish to evaluate known $\mathbf{y}^*$ under the model. For prediction, it is more useful to compute (using Monte-Carlo integration) the predictive expectation,

$$\begin{aligned}\mathbb{E}[y^*] &\approx \int \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k) \int y^* p\Big(y^*\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\, dy^* d\mathbf{w} \\ &= \int \mathbb{E}\Big[p\Big(y^*\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\Big] \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k)\, d\mathbf{w}. \end{aligned} \tag{29}$$

Often we find $\mathbb{E}\big[p\big(y^*\big|g\big(\boldsymbol{\phi}^{*\top}\mathbf{w}\big), \gamma\big)\big] = g\big(\boldsymbol{\phi}^{*\top}\mathbf{w}\big)$, however this is only true with with right choice and usage of the activation function. Furthermore, it is useful to compute quantiles of the predictive density in order to ascertain the predictive uncertainty. We start by sampling the predictive cumulative density function, $P(\cdot)$,

$$\begin{aligned} &P(y^* \leq \alpha|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) \\ &\quad\approx \int \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k) \int_{-\infty}^{\alpha} p\Big(y^*\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\, dy^* d\mathbf{w} \\ &\quad= \int P\Big(y^* \leq \alpha\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big) \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k)\, d\mathbf{w}. \end{aligned} \tag{30}$$

Once we have obtained sufficient samples from the (mixture) posterior we can obtain quantiles, α, for some chosen level of probability, p, using root finding techniques. Specifically, we use root finding techniques to solve the following for α,

$$P(y^* \leq \alpha|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) - p = 0. \tag{31}$$

1.4 Large Scale Gaussian Process Approximation

TODO: re-write the following:

In *revrand* we approximate Gaussian Processes with our standard and generalised linear models by using random feature functions such as those of rah (2007; 2008). They use Bochner's theorem regarding the relationship between a kernel and the Fourier transform of a non-negative measure that (via Wiener-Khintchine's theorem) establishes the duality of the covariance function of a stationary process and its spectral density,

$$k(\boldsymbol{\tau}) = \int S(\mathbf{s}) e^{i\mathbf{s}^\top \boldsymbol{\tau}} d\mathbf{s}, \tag{32}$$

$$S(\mathbf{s}) = \int k(\boldsymbol{\tau}) e^{-i\mathbf{s}^\top \boldsymbol{\tau}} d\boldsymbol{\tau}. \tag{33}$$

rah's main insight (2007) is that we can approximate the kernel by constructing 'suitable' random features and Monte Carlo averaging over samples from $S(\mathbf{s})$,

$$k(\mathbf{x} - \mathbf{x}') = k(\boldsymbol{\tau}) \approx \frac{1}{D} \sum_{i=1}^{D} \phi_i(\mathbf{x})^\top \phi_i(\mathbf{x}') \,, \tag{34}$$

$\phi_i(\mathbf{x})$ corresponds to the ith sample from the feature map. An example of a feature vector construction in the above approximation is,

$$\begin{aligned} [\phi_i(\mathbf{x}) \,, \phi_{D+i}(\mathbf{x})] = \frac{1}{\sqrt{D}} \left[\cos(\mathbf{s}_i^T \mathbf{x}), \sin(\mathbf{s}_i^T \mathbf{x})\right] \,, \\ \text{with } \mathbf{s}_i \sim \mathcal{N}\left(\mathbf{s}_i \middle| \mathbf{0}, \sigma_\phi^2 \mathbf{I}_d\right) \,, \end{aligned} \tag{35}$$

for $i = 1, \ldots, D$, which in fact is a mapping into a $2D$-dimensional feature space. rah (2007) used the above feature map to approximate the commonly used (isotropic) squared exponential kernel, and showed that such an approximation converges in expectation to the true kernel.

TODO: table of kernels and sampling distributions currently in revrand (Laplace, Cauchy, RBF, Matern5/2, Matern3/2 etc)

TODO: FastFood

TODO: A la Carte spectral mixtures

2. Experiments

TODO: for now see the notebooks.

References

Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*. 2007.

Samuel Gershman, Matt Hoffman, and David Blei. Nonparametric variational inference. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, July 2012.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Trung V. Nguyen and Edwin V. Bonilla. Automated variational inference for gaussian process models. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NIPS)*. 2008.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, Massachusetts, 2006.