# *revrand*: Technical Report

Daniel Steinberg, Louis Tiao
Data61 | CSIRO
email: {firstname.lastname}@data61.csiro.au

---

**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. $\theta$,

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

where $\eta_k$ 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,

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

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,

$$\text{Likelihood:} \quad \prod_{n=1}^{N} p(y_n|\theta)\,, \tag{6}$$

$$\text{prior:} \quad p(\theta|\alpha)\,, \tag{7}$$

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

$$\underset{\alpha}{\operatorname{argmin}} - \log \int \prod_{n=1}^{N} p(y_n|\theta)\, p(\theta|\alpha)\, d\theta. \tag{8}$$

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),

$$\begin{aligned}\log p(\mathbf{y}|\alpha) &= \log \int \prod_{n=1}^{N} p(y_n|\theta)\, p(\theta|\alpha)\, d\theta \\ &= \log \int \frac{\prod_n p(y_n|\theta)\, p(\theta|\alpha)}{q(\theta)} q(\theta)\, d\theta \\ &\geq \int q(\theta) \log \left[ \frac{\prod_n p(y_n|\theta)\, p(\theta|\alpha)}{q(\theta)} \right] d\theta \end{aligned} \tag{9}$$

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

$$\mathcal{L} = \sum_{n=1}^{N} \langle \log p(y_n|\theta) \rangle_q - \mathrm{KL}[q(\theta) \,\|\, p(\theta|\alpha)]\,, \tag{10}$$

which takes the form of Equation (4), and so we can weight the Kullback-Leibler term like the constant term, $g(\cdot)$, from Equation (5) if we use stochastic gradients optimisation. 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,

$$\text{Likelihood:} \quad \prod_{n=1}^{N} \mathcal{N}\left(y_n \middle| \boldsymbol{\phi}_n^\top \mathbf{w}, \sigma^2\right), \tag{11}$$

$$\text{prior:} \quad \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D)\,, \tag{12}$$

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

- Optimise $\sigma^2, \lambda$ and $\theta$ w.r.t. log-marginal likelihood,

$$\log p\big(\mathbf{y}\big|\sigma^2, \lambda, \theta\big) = \log \mathcal{N}\left(\mathbf{y}\middle|\mathbf{0}, \sigma^2 \mathbf{I}_N + \lambda \boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right), \tag{13}$$

where $\mathbf{\Phi} \in \mathbb{R}^{N \times D}$ 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,

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

- Use the predictive distribution

$$\begin{aligned}p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) &= \int \mathcal{N}\left(y^* \middle| \boldsymbol{\phi}^{*\top}\mathbf{w}, \sigma^2\right) \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{C})\, d\mathbf{w}, \\ &= \mathcal{N}\left(y^* \middle| \boldsymbol{\phi}^{*\top}\mathbf{m}, \sigma^2 + \boldsymbol{\phi}^{*\top}\mathbf{C}\boldsymbol{\phi}^*\right) \end{aligned} \tag{14}$$

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

$$\mathbb{E}[y^*] = \boldsymbol{\phi}^{*\top}\mathbf{m}, \tag{15}$$

$$\mathbb{V}[y^*] = \sigma^2 + \boldsymbol{\phi}^{*\top}\mathbf{C}\boldsymbol{\phi}^{*\top}. \tag{16}$$

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,

$$\mathcal{L} = \sum_{n=1}^{N} \left\langle \log \mathcal{N}\left(y_n \middle| \boldsymbol{\phi}_n^\top \mathbf{w}, \sigma^2\right) \right\rangle_q - \mathrm{KL}[\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{C}) \,\|\, \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D)]\,. \tag{17}$$

More specifically,

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

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 $\mathcal{O}\left(D^3\right)$ 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,

$$\text{Likelihood:} \quad \prod_{n=1}^{N} p\left(y_n \middle| g\left(\boldsymbol{\phi}_n^\top \mathbf{w}\right), \gamma\right), \tag{18}$$

$$\text{prior:} \quad \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda \mathbf{I}_D), \tag{19}$$

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

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. We use a modification of the nonparametric variational algorithm presented in [1] for approximating the posterior and log marginal likelihood. The posterior takes the form

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}, \mathbf{X}) &\approx q(\mathbf{w}), \\ &= \frac{1}{K} \prod_{k=1}^{K} \mathcal{N}(\mathbf{w}|\mathbf{m}_k, \boldsymbol{\Psi}_k), \end{aligned} \tag{20}$$

i.e. a mixture of $K$ diagonal Gaussians, $\boldsymbol{\Psi}_k = \text{diag}\left([\Psi_{k,1}, \ldots, \Psi_{k,D}]^\top\right)$. This is a very flexible form for the approximate posterior, and still admits a closed-form lower bound to the log-marginal likelihood. Furthermore this has the nice property that our algorithm no longer has a $\mathcal{O}\left(D^3\right)$ cost associated with the number of features. Gershman, Hoffman, and Blei [1] make one more

approximation based on local second order Taylor expansions of the joint,

$$\langle \log p(\mathbf{y}, \mathbf{w}) \rangle_q \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{n=1}^{N} \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top \mathbf{m}_k\Big), \gamma\Big) + \log \mathcal{N}(\mathbf{m}_k | \mathbf{0}, \lambda \mathbf{I}_D) + \frac{1}{2} \mathrm{tr}(\boldsymbol{\Psi}_k \mathbf{H}_k), \quad (21)$$

to deal with non-conjugacy within their variational framework. Here $\mathbf{H}_k = \nabla^2_{\mathbf{w}} p(y, \mathbf{w})|_{\mathbf{w}=\mathbf{m}_k}$[1]. Unfortunately this approximation to the ELBO actually breaks the lower bound, but works well in practice. For more details on this inference scheme we direct the reader to [1]. The final objective function we optimise is,

$$\mathcal{L} \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{n=1}^{N} \log p\Big(y_n \Big| g\Big(\boldsymbol{\phi}_n^\top \mathbf{m}_k\Big), \gamma\Big) + \log \mathcal{N}(\mathbf{m}_k | \mathbf{0}, \lambda \mathbf{I}_D) + \frac{1}{2} \mathrm{tr}(\boldsymbol{\Psi}_k \mathbf{H}_k) - \log \left( \sum_{j=1}^{K} \mathcal{N}(\mathbf{m}_k | \mathbf{m}_j, \boldsymbol{\Psi}_k + \boldsymbol{\Psi}_j) \right) - \log K. \quad (22)$$

While this is an approximation to Equation (10), it retains the property that it easily factorises over the data ($n$), and so can be optimised using stochastic gradients as described in §1.1.

In *revrand* we actually optimise Equation (22) with respect to the hyperparameters, $\theta, \gamma$ *and* the parameters $\mathbf{m}_k, \boldsymbol{\Psi}_k \; \forall k$ simultaneously. This is unlike the optimisation scheme in [1], since it requires 3rd derivatives of the likelihood in Equation (18) w.r.t. $\mathbf{w}$. However, we find that providing these derivatives is simple (for our model) compared to the more complex optimisation scheme presented in [1], which is not as amenable to large datasets as stochastic gradients when hyperparameter optimisation is involved.

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,

$$p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) \approx \int p\Big(\mathbf{y} \Big| g\Big(\boldsymbol{\phi}^{*\top} \mathbf{w}\Big), \gamma\Big) \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{w} | \mathbf{m}_k, \boldsymbol{\Psi}_k) \, d\mathbf{w}. \quad (23)$$

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

[1] We only need the diagonals of this Hessian matrix.

(using Monte-Carlo integration) the predictive expectation,

$$\begin{aligned}\mathbb{E}[y^*] &\approx \int \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\mathbf{w}|\mathbf{m}_k, \mathbf{\Psi}_k)\int y^* p\Big(y^*\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\, dy^* d\mathbf{w} \\ &= \int \mathbb{E}\Big[p\Big(y^*\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\Big]\,\frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\mathbf{w}|\mathbf{m}_k, \mathbf{\Psi}_k)\, d\mathbf{w}. \end{aligned} \tag{24}$$

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 with true 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, $P(\cdot)$,

$$\begin{aligned}&P(y^* \leq \alpha|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) \\ &\quad\approx \int \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\mathbf{w}|\mathbf{m}_k, \mathbf{\Psi}_k)\int_{-\infty}^{\alpha} p\Big(y^*\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\, dy^* d\mathbf{w} \\ &\quad= \int P\Big(y^* \leq \alpha\Big|g\Big(\boldsymbol{\phi}^{*\top}\mathbf{w}\Big), \gamma\Big)\,\frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\mathbf{w}|\mathbf{m}_k, \mathbf{\Psi}_k)\, d\mathbf{w}. \end{aligned} \tag{25}$$

Once we have obtained sufficient samples we can obtain quantiles, $\alpha$, for some chosen level of probability, $p$, using root finding techniques. Specifically, we solve the following for $\alpha$,

$$P(y^* \leq \alpha|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) - p = 0. \tag{26}$$

## 1.4 Large Scale Gaussian Process Approximation

TODO: Show kernel approximations (from notebook) TODO: Some mention of these work with the SLM and GLM.

Highlight how we can use proper optimisation to learn kernel parameters *without* weight re-sampling (unlike what they say in "The Mondrian Kernel" paper). I.e. extend what Edwin wrote in the E/UKS paper in section 3 to show how we learn length-scales (this should be obvious from the A la Carte paper though).

## 2 Feature Composition Framework

## 3 Experiments

## References

[1] S. Gershman, M. Hoffman, and D. Blei. "Nonparametric variational inference". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Ed. by J. Langford and J. Pineau. ICML '12. Edinburgh, Scotland, GB: Omnipress, 2012, pp. 663–670. ISBN: 978-1-4503-1285-1.

[2] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, Massachusetts, 2006.