The Clean Thesis Style

Ricardo Langner

August 26, 2015

Version: My First Draft

Clean Thesis Style University

CleanThesis

Department of Clean Thesis Style
Institut for Clean Thesis Dev
Clean Thesis Group (CTG)

Documentation

The Clean Thesis Style

Ricardo Langner

1. Reviewer	Jane Doe Department of Clean Thesis Style Clean Thesis Style University
2. Reviewer	John Doe Department of Clean Thesis Style Clean Thesis Style University
Supervisors	Jane Doe and John Smith

August 26, 2015

Ricardo Langner
The Clean Thesis Style
Documentation, August 26, 2015
Reviewers: Jane Doe and John Doe
Supervisors: Jane Doe and John Smith

Clean Thesis Style University
Clean Thesis Group (CTG)
Institut for Clean Thesis Dev
Department of Clean Thesis Style
Street address
Postal Code and City

Contents

1 A Bayesian Quasi-Randomized neural network, and its application to the optimization of black box functions

1.1 Introduction

In this chapter, we present a Bayesian Quasi-Random Vector Functional Link (BQRVFL) neural network model, with one hidden layer. It is a hybrid penalized regression/neural networks model on an augmented data set. In this hybrid regression model, we assume that the regression parameters are governed by a prior distribution, and that the hidden layer's nodes are (quasi-)randomized. As a prior distribution for the regression parameters, we will use a multivariate Gaussian distribution. And for the simulation of the nodes in the hidden layer, we will use a Sobol sequence.

To the best of our knowledge, randomized neural networks were introduced by [**schmidt1992feedforward**], and the Random Vector Functional Link neural networks (RVFL) were introduced by [**pao1994learning**]. RVFL networks are *multilayer feedforward* neural networks, in which there is a *direct link* between the predictors and the output variable, aiming at capturing the linear relationships. And in addition to the *direct link*, there are new features, the hidden nodes (the dataset is augmented).

The RVFL networks have been successfully applied to solving different types of classification and regression problems; see for example [**dehuri2010comprehensive**]. Here, we will use RVFL networks with only one hidden layer.

With the BQRVFL model presented here, it is possible to obtain predictions from a nonlinear model (actually, a combination of a linear and a nonlinear model), along with confidence intervals around the model's predictions. The choice of this relatively simple Gaussian prior should not, however, prevent the user from checking the confidence intervals around the predictions.

There are other types of models, which are also capable of achieving this purpose - obtain predictions from a nonlinear model, along with confidence intervals around the model's predictions - like Gaussian process regression models (cite Rasmussen et al. (2006)) and random forests models with confidence intervals (cite Wager et al. (2014)). Depending on the problem at hand, and the desire from the user for a higher accuracy and/or interpretability, one of these alternative models could also be considered.

In section 1.2, we present the BQRFVL model, with one hidden layer. Formulas for the estimation of its parameters are given, along with formulas for the calculation of confidence intervals around the predictions. The model's predictions on a validation set, and on simulated data from (cite Sapp (2014)) are also presented.

In section 1.3, the model is used as a workhorse for Bayesian optimization (cite Mockus et al. (1978)). This type of optimization methods are useful for finding minima or maxima of black box functions, whose evaluations are expensive, and gradients are not necessarily available in a closed form. It has been shown to be very effective on challenging optimization functions (cite Jones (1998) and Snoek et al. (2015)).

The idea of Bayesian optimization is to optimize an alternative, cheaper function called the acquisition function, rather than the main, expensive objective function. For doing this, the uncertainty around the predictions of an alternative machine learning model - called the surrogate model - is used in a way that will be described in more details in section 1.3.

The surrogate model's posterior distribution tries to approximate the objective function in the best way. And this posterior distribution is enhanced, as more points of the objective function are evaluated. Gaussian process regression models are often used are surrogate models (cite Snoek et al. (2015) for example). Here, the surrogate model will be the BQRVFL model presented in 1.2.

We apply Bayesian optimization, using the BQRVFL model, to the minimization of the cross-validation error (produced on discount curve forecasting) of four machine learning models. One of these models is based on an RVFL applied to multivariate time series forecast (cite Moudiki et al. (2018)). The second one applies Kernel Regularized Least Squares (cite chapter) (KRLS hereafter) directly to the curves' observation dates and time to maturities. The two remaining models are based on the popular DNS framework, and are a DNS-KRLS and a DNS-RVFL (more details here or refer to another section).

1.2 Description of the model

1.2.1 Estimation of the parameters and confidence intervals

We consider a response variable $y \in \mathbb{R}^n$ that has to be explained as a function of p predictors stored in a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$. The data set $\mathbf{X}$ is augmented by a set of new predictors:

$$g\left(\mathbf{X}W\right)$$

where g is an activation function, as in of neural networks models, and $W \in \mathbb{R}^{p \times L}$ are the nodes in the hidden layer, obtained from a deterministic sobol sequence of quasi-random numbers (cite Niderreiter (1992) and Boyle et Tan (1997)). L is the number of nodes in the hidden layer.

The parameters W are used to combine the p columns of $\mathbf{X}$ into L new variables, hence taking into account the potential nonlinear relationships between the response y and the predictors $\mathbf{X}$ when passed through the activation function g. Having obtained a new set of predictors stored in columns in a matrix $\mathbf{Z} = [\mathbf{X} \;\; g\left(\mathbf{X}W\right)]$, we apply a Bayesian linear regression model, to explain y as a function $\mathbf{Z}\beta$ of the new $p + L$ predictors. We assume that y could be explained as:

$$y = \mathbf{Z}\beta + \epsilon$$

where $\beta \sim \mathcal{N}(0_{\mathbb{R}^{p+L}}, I_{p+L})$ are the parameters of the model that have to be estimated, and $\epsilon \sim \mathcal{N}(0_{\mathbb{R}^n}, \sigma^2 I_n)$ is the error term. It is possible to show, in a Bayesian linear regression setting (cite Rasmussen et al. (2006) for example, for details) that for new observations arriving in the model and stored in matrix $\mathbf{Z}_*$, we have the following properties for the associated predictions:

$$y* \sim \mathcal{N}(\mathbf{Z}_* \mu_{\beta|y,\mathbf{Z}}, \; \mathbf{Z}_* \Sigma_{\beta|y,\mathbf{Z}} \mathbf{Z}_*^T + \sigma^2 I_n)$$

with

$$\mu_{\beta|y,\mathbf{Z}} = \mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T + \sigma^2 I_n\right)^{-1} y$$

and

$$\Sigma_{\beta|y,\mathbf{Z}} = I_n - \mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T + \sigma^2 I_n\right)^{-1} \mathbf{Z}$$

By using the Woodbury identity (cite Gene H. Golub and Charles F. van Loan. Matrix Computations and cite Max Welling The Kalman filter, Lecture Note) for $\mathbf{P}$ and $\mathbf{R}$ positive definite

$$\left(\mathbf{P}^{-1} + \mathbf{B}^T \mathbf{R}^{-1} \mathbf{B}\right)^{-1} \mathbf{B}^T \mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T \left(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R}\right)^{-1}$$

and by noting $\sigma^2 = \lambda$, this is equivalent to having the same result for y^*, but with:

$$\mu_{\beta|y,\mathbf{Z}} = \left(\mathbf{Z}^T\mathbf{Z} + \lambda I_p\right)^{-1}\mathbf{Z}^T y$$

and

$$\Sigma_{\beta|y,\mathbf{Z}} = I_p - \left(\mathbf{Z}^T\mathbf{Z} + \lambda I_p\right)^{-1}\mathbf{Z}^T\mathbf{Z}$$

And depending on the situations in which $n << p$, or $p << n$, we can use one of the two previous expressions. The one resulting in the lowest number operations. In the sequel of the chapter, we will use a ReLU activation function, which is equal to

$$g : x \mapsto max(x, 0)$$

In the case where g is derivable (for example when $g : x \mapsto x$ or $g : x \mapsto sigmoid(x) = \frac{1}{1+exp(-x)}$ or $g : x \mapsto tanh(x)$), it is possible to obtain a sensitivity (at the first order) of the response to a change in the covariates. Indeed, in this case, we have for a fixed (tied to the observations) $i \in \{1, \ldots, n\}$:

$$y_i = \sum_{j=1}^{p} \alpha_j X_i^{(j)} + \sum_{k=1}^{L} \gamma_k g(X_i^T W^{(k)})$$

Where $W^{(k)}$ is the k^{th} column of W among L. And for a fixed (tied to the covariates) integer $j_0 \in \{1, \ldots, p\}$, and $u := X_i^T W^{(k)}$, we have the following sensitivity of y_i:

$$\begin{aligned}\frac{\partial y_i}{\partial X_i^{(j_0)}} &= \alpha_{j_0} + \sum_{k=1}^{L} \gamma_k \frac{\partial g}{\partial u}(u) \frac{\partial u}{\partial X_i^{(j_0)}}(u) \\ &= \alpha_{j_0} + \sum_{k=1}^{L} \gamma_k W_{j_0}^{(k)} \frac{\partial g}{\partial u}(u)\end{aligned}$$

For $g : x \mapsto x$ for example, this leads to:

$$\frac{\partial y_i}{\partial X_i^{(j_0)}} = \alpha_{j_0} + \sum_{k=1}^{L} \gamma_k W_{j_0}^{(k)}$$

1.2.2 Examples on data from (cite Sapp (2014))

In order to present the performances of the model described in the previous section, we use four simulated randomized data sets, from (cite Sapp (2014)), $SLC14_1$ and $SLC14_2$ with 2 different seeds each. These datasets are available in `R` package `caret`.

$SLC14_1$ consists of n rows, where n is a function parameter, and 20 columns. Its construction starts with the simulation of a $n \times 20$ matrix `dat`, containing random Gaussian numbers, with mean equal to zero and variance equal to 9. Each row of `dat` contains:

$$\mathbf{x} = (x_1, x_2, \ldots, x_{20})$$

and the following transformation is applied to obtain each column of $SLC14_1$

$$\begin{aligned}\mathbf{x} &\mapsto x_1 + sin(x_2) + log(|x_3|) + x_4^2 + x_5 \times x_6 + I(x_7 \times x_8 \times x_9 < 0) + I(x_{10} > 0)\\ &+ x_{11} \times I(x_{11} > 0) + \sqrt{(|x_{12}|)} + cos(x_{13}) + 2 \times x_{14} + |x_{15}|\\ &+ I(x_{16} < -1) + x_{17} \times I(x_{17} < -1) - 2 \times x_{18} - x_{19} \times x_{20} + \epsilon\end{aligned}$$

where $\epsilon \sim \mathcal{N}(0, 9)$. The same type of idea is applied for obtaining $SLC14_2$. The construction of the dataset begins with a matrix `dat` having n rows and 200 columns, and the distribution of the random numbers within `dat`, is a Gaussian distribution with mean equal to 0, and variance equal to 16. And the transformation applied to the columns of `dat` is:

$$\mathbf{x} \mapsto -1 + log(|x_1|) + \ldots + log(|x_{200}|) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 25)$.

For each seed, `123` and `456`, and $n = 250$, we separate the datasets $SLC14_1$ and $SLC14_2$ into a training/testing test, and a validation set. The training set contains 70% of the whole data (175 observations), and the test set contains the remaining 30% of the data (75 observations).

A 5-fold cross-validation repeated 10 times is then applied to the training set, in order to choose *optimal* hyperparameters for the BQRVL model. And to finish, the BQRVFL model with these *optimal* hyperparameters is applied to predict values on the validation set. The following figures, 1.1, 1.2, 1.3, and 1.4 present the validation sets' observations (black line), along with the out-of-sample predictions on the validation set (blue line), along with 80% and 95% confidence intervals around the predictions.

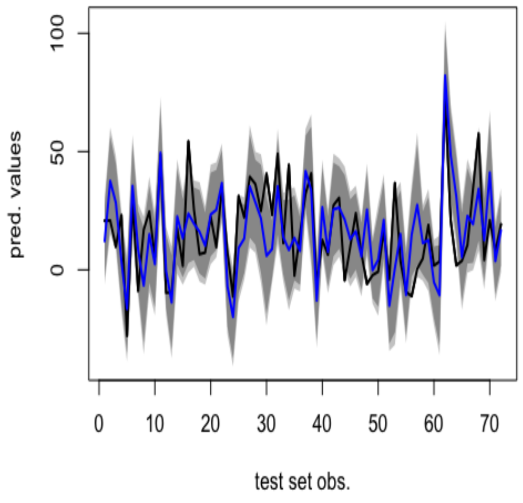

Fig. 1.1: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.1 (with `seed = 123`)

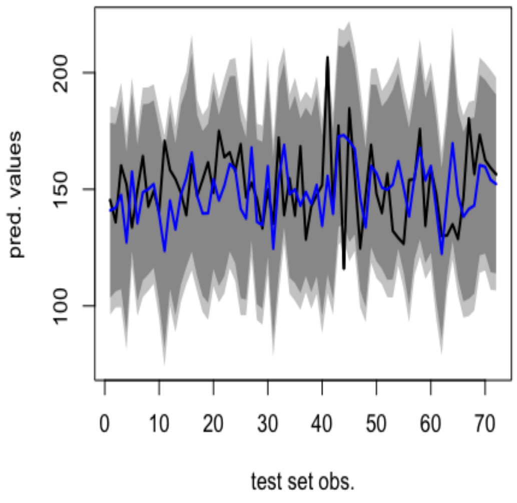

Fig. 1.2: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.2 (with `seed = 123`)

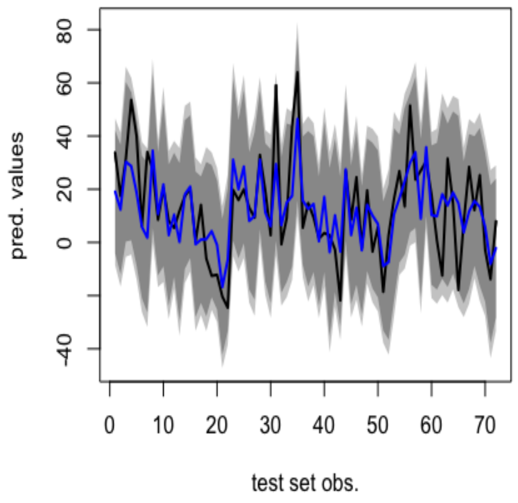

Fig. 1.3: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.1 (with `seed = 456`)

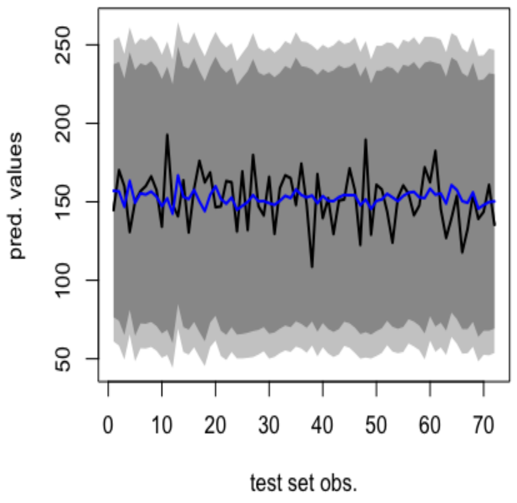

Fig. 1.4: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.2 (with `seed = 456`)

We observe that the confidence intervals often contain the validation set value, with a few exceptions. The following table 1.1 contain the *optimal* (based on a grid search) parameters found for each dataset, with a ReLU activation function for the BQRVFL, and the proportions of points of the validation set contained into the 80%, 95% confidence intervals:

Tab. 1.1

Dataset	Number of nodes	λ	Prop at 95%	Prop at 80%
$SLC14_1$ - `seed = 123`	975	1.53	86.11%	77.77%
$SLC14_1$ - `seed = 456`	-1.047	-1.024	0%	0%
$SLC14_2$ - `seed = 123`	825	20	97.22%	94.44%
$SLC14_2$ - `seed = 456`	-1.047	-1.027	-1.027	-1.027

1.3 Bayesian optimization of black box functions

1.3.1 Description of the method

The optimization problem is about finding $\mathbf{x}^* \in \mathcal{C} \subseteq \mathbb{R}^k$, so that:

$$\mathbf{x}^* = ArgMin_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x})$$

f is the objective function, whose evaluations are expensive to calculate, and gradients are not necessarily available in closed-form. Methods based on gradient descent, or requiring to evaluate f several times will hence be inefficient for carrying out this task.

Bayesian optimization (cite Mockus et al. (1978) and Jones (2001)) will minimize an alternative, cheaper function called the acquisition function, instead of minimizing f directly.

The acquisition function is based on the uncertainty around the predictions of a surrogate machine learning model, trying to approximate f. Here, we consider only two types of acquisition functions. We denote by $\widetilde{f}(\mathbf{x}, \theta)$, the prediction obtained with the surrogate model (whose distribution depends on θ) on a point $\mathbf{x} \in \mathcal{C}$:

- The **Upper Confidence Bound** (UCB) acquisition function, (actually a Lower Confidence Bound in the case of minimization). The idea is to minimize:

  $$a_{UCB}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x})$$

  κ is a tuning parameter, designed to balance between exploration and exploitation (more details on this). With $\kappa = 1.96$ for example, this looks like the lower confidence bound of $\widetilde{f}(\mathbf{x}, \theta)$, at a risk level of 5%, when the surrogate model is $\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$.

- The **Expected Improvement** (EI) acquisition function. If f^* is the current minimum value found after a few evaluations of f, we would like to maximize the expected improvement of the surrogate model's predictions over f^* :

$$a_{EI}(\mathbf{x}) = \mathbb{E}\left[max(\widetilde{f}(\mathbf{x}, \theta) - f^*, 0)\right]$$

The whole optimization process that we use is iterative and is described below, with the total number of iterations denoted as nb_{iter}:

1. We start with nb_{init} points randomly sampled in $\mathcal{C}$, for which some evaluations of f have been obtained. With these points, we construct a training data set for the surrogate model:

$$\mathcal{D} = \{(\mathbf{x}_1, f\,(\mathbf{x}_1))\,, \dots, (\mathbf{x}_{nb_{init}}, f(\mathbf{x}_{nb_{init}}))\}$$

2. We train the surrogate model on $\mathcal{D}$, with a fixed parameter θ for the distribution, and search for the next point to be evaluated for f, as:

$$\mathbf{x}_{next} = ArgMax_{\mathbf{x} \in \mathcal{C}}\, a_{EI}(\mathbf{x})$$

or

$$\mathbf{x}_{next} = ArgMin_{\mathbf{x} \in \mathcal{C}}\, a_{UCB}(\mathbf{x})$$

If $\mathbf{x}_{next}$ is already found in $\mathcal{D}$, then $\mathbf{x}_{next}$ is picked randomly in $\mathcal{C}$.

3. $\mathcal{D}$ is enriched with $(\mathbf{x}_{next}, f(\mathbf{x}_{next}))$ and we return to point 2., until the nb_{iter} budget is reached.

Step 2. is done with a gradient based optimization method (cite nlminb PORT routine).

For the BQRVFL, we have to choose θ, which, in this specific case, are two parameters: the number of nodes in the hidden layer, and the regularization parameter σ^2. Ths is done by using Generalized Cross Validation (GCV). More details.

We assume that the Random Forest predictions are Gaussian (you should check this further in the paper of Wager (2014)!). The BQRVFL's posterior distribution is also Gaussian, $\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$. In this case, the acquisition function with Expected Improvement can be re-written as:

$$a_{EI}(\mathbf{x}) = \sigma(\mathbf{x}) \left(\mu(\mathbf{x})\Phi\left(\gamma(\mathbf{x})\right) + \Phi^{'}\left(\gamma(\mathbf{x})\right)\right)$$

Where Φ is the probability distribution function of a $\mathcal{N}(0,1)$, and $\Phi^{'}$ is its density function.

This, is to illustrate the model's behaviour on a simple example. A further tuning of the Random Forest hyperparameters, or of the parameter κ, or a different choice for nb_{init}, or activation function for the BQRVFL, would of course lead to different results. For 100 seeds, and a fixed number of iterations nb_{iter}, we rerun the procedure described before, 100 times. For the BQRVFL, nb_{iter} goes from 5 to 100 with steps of 5, and for the Random Forest with confidence intervals, $nb_{iter} \in \{5, 50, 100\}$.

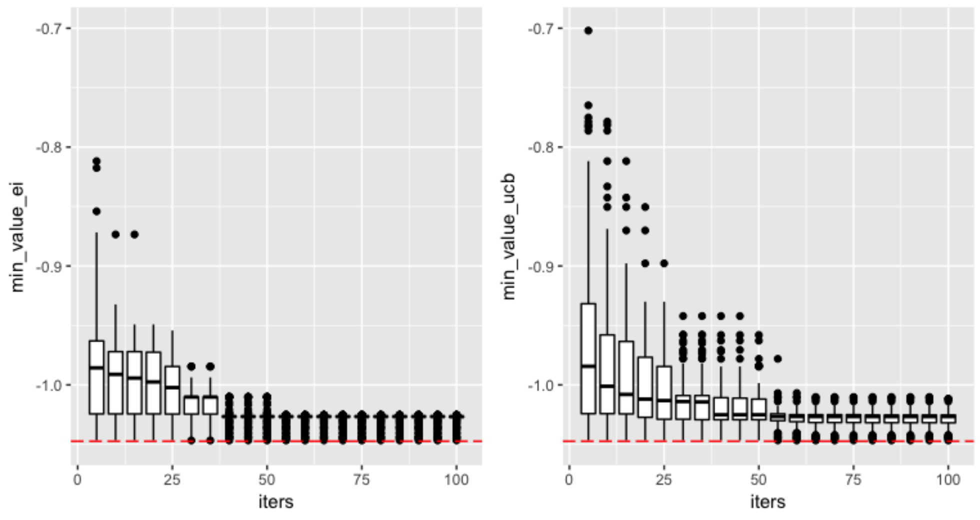

Fig. 1.5: Minimum value for the rescaled Branin function, as a function of the number of iterations with BQRVFL. **Left:** With Expected Improvement (EI) **Right:** With Upper Confidence Bound (UCB) - BQRVFL

Tab. 1.2: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Expected Improvement) with BQRVFL

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.024	-0.986	-0.986	-0.963	-0.812	0.046
50	-1.047	-1.027	-1.027	-1.027	-1.027	-1.010	**0.008**
100	-1.047	-1.027	-1.027	-1.029	-1.027	-1.025	**0.006**

Tab. 1.3: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Upper Confidence Bound) with BQRVFL

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.024	-0.984	-0.963	-0.932	-0.702	0.079
50	-1.047	-1.029	-1.025	-1.020	-1.012	-0.958	0.017
100	-1.047	-1.032	-1.027	-1.028	-1.025	-1.012	0.008

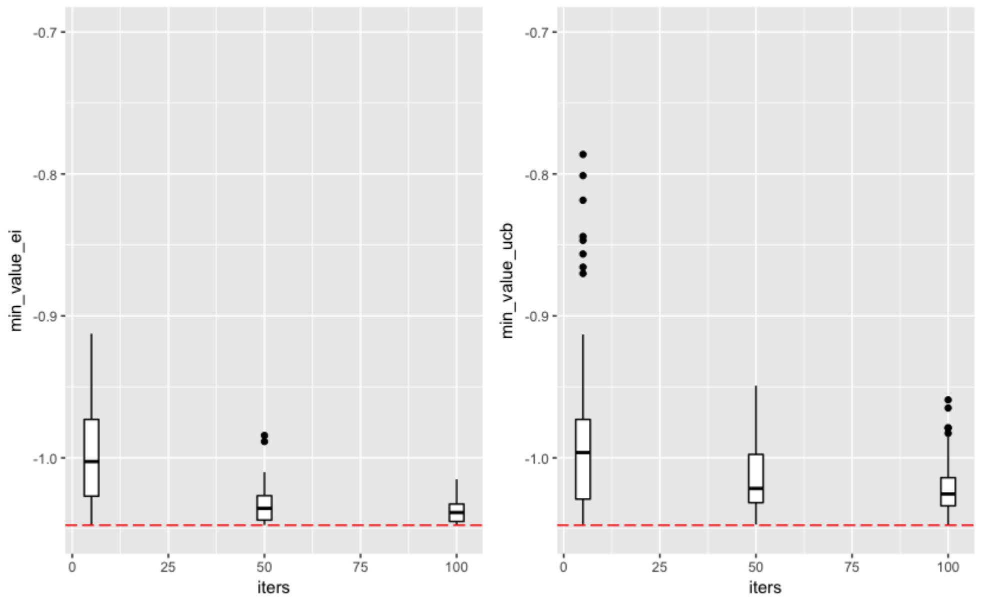

Fig. 1.6: Minimum value for the rescaled Branin function, as a function of the number of iterations. **Left:** With Expected Improvement (EI) **Right:** With Upper Confidence Bound (UCB) - Random Forest

Tab. 1.4: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Expected Improvement) with Random Forest

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.027	**-1.003**	**-0.999**	-0.973	-0.912	**0.033**
50	-1.047	-1.044	**-1.036**	**-1.033**	-1.027	-0.984	0.012
100	-1.047	-1.045	**-1.039**	**-1.038**	-1.033	-1.015	0.008

Tab. 1.5: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Upper Confidence Bound) with Random Forest

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.029	-0.996	-0.989	-0.973	-0.786	0.055
50	-1.047	-1.032	-1.022	-1.015	-0.998	-0.949	0.023
100	-1.047	-1.034	-1.025	-1.022	-1.014	-0.959	0.019

Some pretty good results were obtained with the Random Forest for the mean and the median value. But generally with a higher variance on the minimum value obtained than BQRVFL.

Another interesting information obtained through these simulations is the number of times (over $nb_{iter} = 100$ iterations) $\mathbf{x}_{next}$ has been picked randomly in $\mathcal{C}$ at Step 2. (when $\mathbf{x}_{next}$ is already in $\mathcal{D}$).

In table 1.6 and figure 1.7, we observe that on this part, BQRVFL generally performs better. With BQRVFL and Expected Improvement acquisition function for example, the next point to be evaluated is never obtained by simulation of $\mathbf{x}_{next} \in \mathcal{C}$.

Tab. 1.6: Distribution of the number of times a simulation of $\mathbf{x}_{next} \in \mathcal{C}$ has been required at step 2.

Method	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
BQRVFL with EI	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
BQRVFL with UCB	0.00	0.00	1.00	1.96	3.00	13.00	2.22
RF with EI	0.00	2.00	3.00	3.66	5.00	9.00	1.98
RF with UCB	1.00	6.00	10.00	9.98	13.00	22.00	4.54

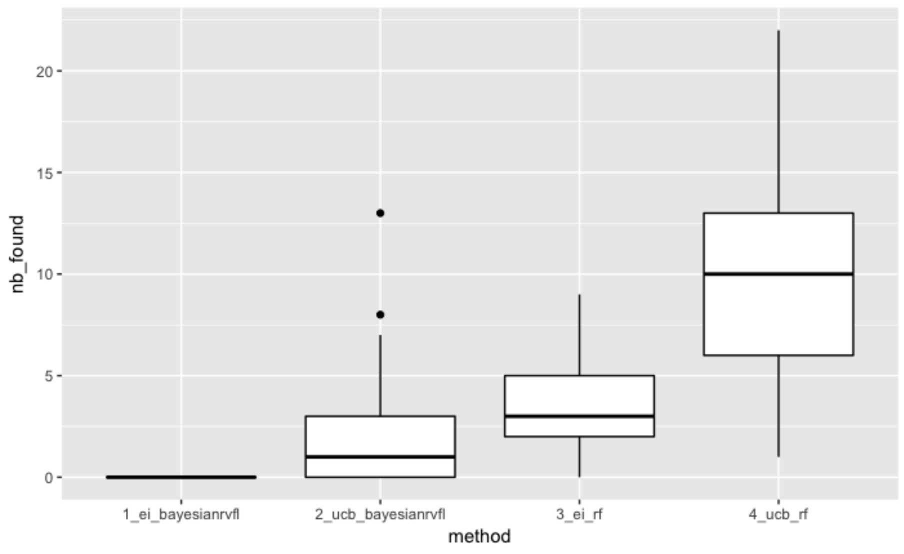

Fig. 1.7: Distribution of the number of times a simulation of $\mathbf{x}_{next} \in \mathcal{C}$ has been required at step 2.

A good place to apply Bayesian optimization, is to the choice of hyperparameters of machine learning algorithms. The cross-validation function, which is the objective function of the hyperparameters, is indeed expensive to train/predict over the cross-validation resamples.

1.3.2 Example of models based on Dynamic Nelson Siegel and Kernel Regularized Least Squares

Description of the models

In the previous chapters, we introduced the RVFL, the KRLS, and the DNS-KRLS models. Here, we compare all these models by using the Bayesian optimization algorithm described in this chapter.

Examples of minimization of the out out-of-sample RMSE

Tab. 1.7: out-of-sample RMSE for models based on the DNS framework

Horizon	NS-Gaussian	NS-Matérn 3/2	NS-Matérn 5/2	NS-RVFL
12	0.6584	0.6590	0.6652	**0.5483**
18	0.8356	0.8356	0.8356	**0.7078**
24	1.0130	1.0130	1.0130	**0.8564**
30	1.1607	1.1607	1.1607	**1.0289**
36	1.2863	1.2863	1.2863	**1.1405**

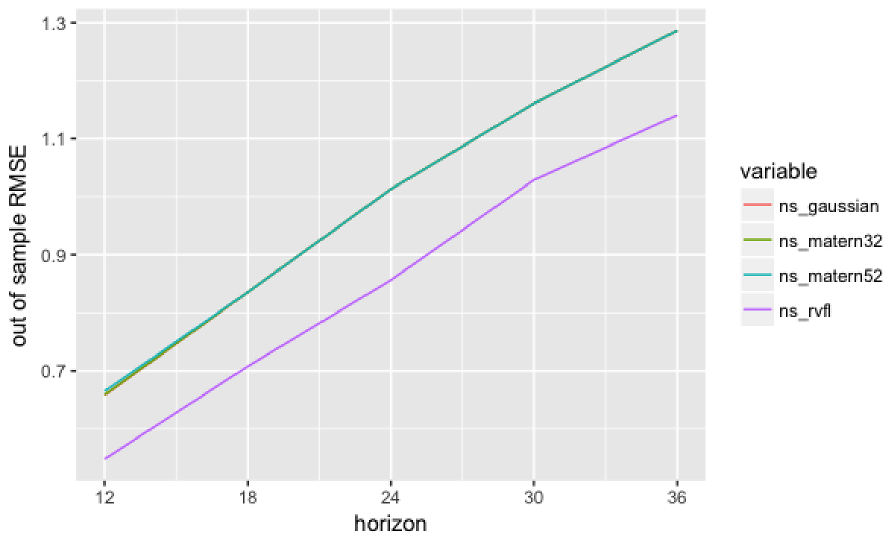

Fig. 1.8: out-of-sample RMSE for models based on the DNS framework

Tab. 1.8: average out-of-sample RMSE for data-driven KRLS models

Horizon	Gaussian	Matérn 3/2	Matérn 5/2	RVFL
12	0.6392	**0.5168**	0.5798	0.5540
18	0.7466	**0.6591**	0.7394	0.7032
24	0.8885	**0.7716**	0.8965	0.8191
30	1.0095	**0.8844**	1.0083	0.9535
36	1.1544	**1.0099**	1.1396	1.0646

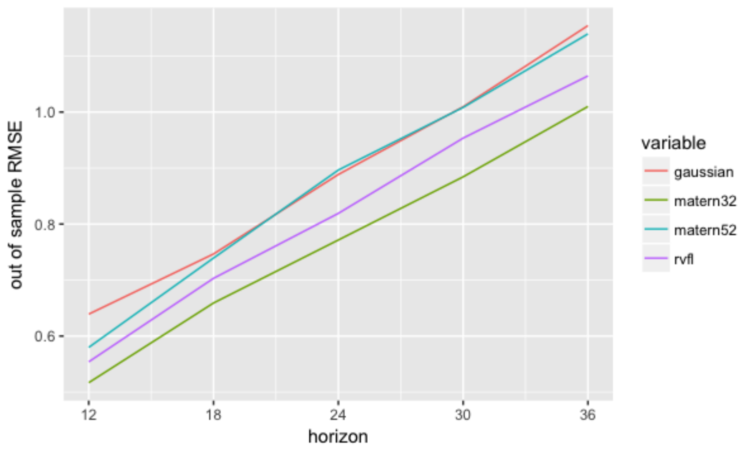

Fig. 1.9: average out-of-sample RMSE for data-driven models

Details of the distribution of errors are provided in the appendix (boxplots and summaries).

We also show how the Matérn 3/2 and the DNS-RVFL are performing over the time, for horizon = 12 and for horizon = 36, particularly during the credit/liquidity crunch.

1.3.3 Example of ensembles of RVFL

In this section, we compare the bagging and stacking of RVFL models. As in the previous section, the optimization of the out-of-sample error (the choice of model's hyperparameters) is made by using the BQRVFL as a workhorse for Bayesian optimization.

1.4 Conclusion

In this chapter, we presented a quasi-randomized functional link neural networks (BQRVFL) model, by linking it to a Bayesian linear regression framework, and obtained formulas for the confidence intervals around its predictions. The BQRVFL model that we constructed is then used as a workhorse for Bayesian optimization, and notably to find *optimal* hyperparameters for all the forecasting models described in the previous chapters.

In general, we observe that the *best* results are obtained in the DNS-KRLS framework by the RVFL (applied to the level, slope and curvature). And in the KRLS framework, the kernel Matérn 3/2 performs the best. This kernel also performs the best overall, in terms of out-of-sample RMSE. However, it is worth mentioning, as in previous chapters, that the cost of computing the kernel is a quadratic function of the number of input spot rates.

The following figure (1.10) illustrates this fact very well. It compares the KRLS and RVFL models timings on training and forecasting horizons equal to 12 months and 36 months, and respectively 15, 22 and 30 observed time to maturities. The RVFL timings remain relatively stable as the number of time to maturities increase, whereas the KRLS timings increase at a fast pace in the same situation.

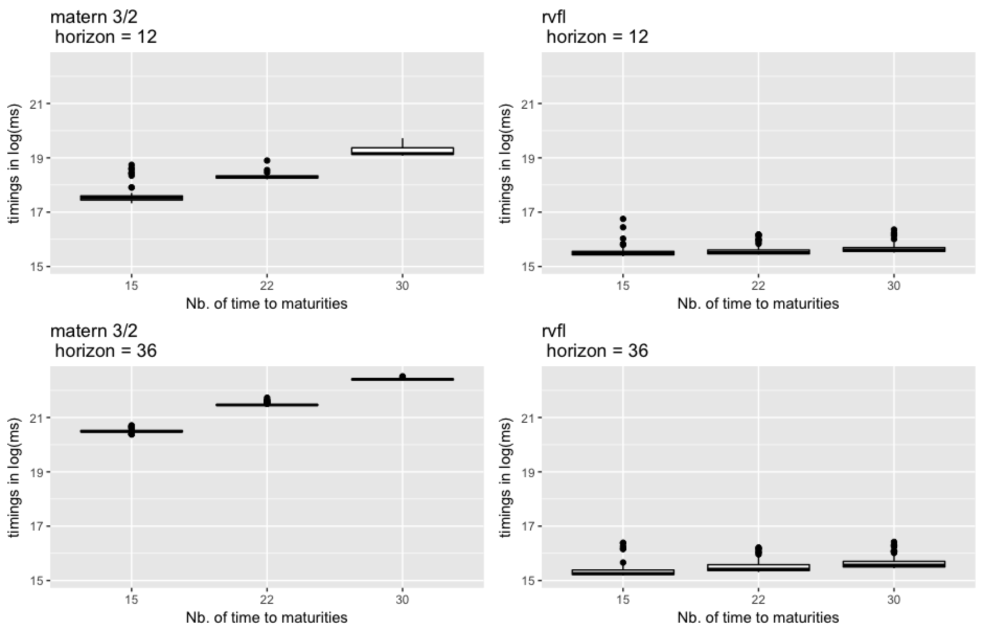

Fig. 1.10: timings KRLS vs timings RVFL

When it comes to ensemble learning, we compare bagging, boosting and stacking models. Both.

List of Figures

List of Tables

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

City, August 26, 2015

Ricardo Langner