An Introduction to Practical Implementation of Bayesian Inference on Linear Regression Problems

Kaspara Skovli Gåsvær
University of Oslo - Department of Physics

(Dated: June 21, 2021)

I. INTRODUCTION

It is one thing to obtaining new knowledge and another being able to apply said knowledge to solving a problem. Being taught relations and facts purely based on theory and passive learning strategies can often limit the amount of information that is really available. Practically implementing knowledge can help identify issues, gaps in knowledge and force the learning individual to understand the subject at a deeper level. The paper *An introduction to Bayesian Linear Regression* (Gåsvær 2021) gives an introduction to the theoretical background on Bayesian statistics and in this paper we wish to take it one step further. We will focus on the practical implementations of the concepts an methods introduced in the theoretical paper with the goal of gaining more intuition and see for our self the implication of model selection and parameter estimation. We will begin by recapping some of the core concepts as well as a bit of new theory before introducing the reader to programming libraries and packages we will be using. By implementing various models for Bayesian linear regression we will look at performance on a polynomial dataset and sinusoidal dataset, both with Gaussian added noise. We will experiment with different basis functions as well as look into parameter optimization. Lastly we will compare our model to a more established model from Scikit-Learn (Pedregosa *et al.* 2011) and as well as taking a look at PyMC3's HMC-NUTS sampler (Hoffman and Gelman).

The code used to produce the results in this report can be found in our GitHub repository.

II. THEORY & METHOD

A. Bayesian Statistics

The theoretical background about Bayesian statistics needed to follow the results and discussion in this report is covered in *An introduction to Bayesian Linear Regression* (Gåsvær 2021). Here we present a short recap of the core elements needed for performing Bayesian Linear Regression, but if at any point you think this paper is moving to fast take a look at the referenced paper.

1. Prior & Likelihood Function

In this report we will make use of a special type of prior, namely a *Conjugate prior*. Specifically we will use a a zero-mean isotropic Gaussian governed by a single precision parameter α (Bishop 2006, Ch.3.3.1)

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0), \tag{1}$$

where

$$\begin{aligned} \mathbf{m}_0 &= \mathbf{0} \\ \mathbf{S}_0^{-1} &= \alpha^{-1}\mathbf{I}, \end{aligned} \tag{2}$$

We will be using a Gaussian distribution for our likelihood function as well, defining it as

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \tag{3}$$

We will at all times during this report be dealing with a dataset of input variables $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = \{t_1, ..., t_N\}$. We assume that all data points are drawn independently so that $P(A \cap B) = P(A)P(B)$ and we can rewrite the likelihood function as

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1}), \tag{4}$$

where $\mathbf{w}$ are the weights of our model and ϕ are the basis functions. We will be implementing two types of basis functions, namely *Polynomial* and *Gaussian*. Polynomial basis functions are defined as

$$\phi_i(x) = x^i, \tag{5}$$

and Gaussian as

$$\phi_i(x) = \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma^2}\right\}. \tag{6}$$

Following Bayes theorem the posterior distribution is found as the product of the prior and likelihood function

$$p(\mathbf{w}|\mathbf{t}) = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N), \tag{7}$$

where

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^{\mathrm{T}} \mathbf{t}$$
$$\mathbf{S}_N^{-1} = \mathbf{S}_0 + \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}, \tag{8}$$

are the mean and covariance of the posterior. More on the implications of using conjugate priors, basis functions and derivation of the likelihood function and posterior can be found in *An introduction to Bayesian Linear Regression* (Gåsvær 2021).

B. Monte Carlo

When dealing with Bayesian inference problems of higher dimensional one can quickly run into trouble calculating the posterior distribution. As the marginal likelihood function is defined as an integral which very often is too computationally expensive to compute exactly. We therefore need to rely on sampling algorithms such as Markov chain Monte Carlo (MCMC). Instead of having to calculate the posterior, for which we need the normalization factor in the denominator of Bayes theorem, we can draw samples from the probability distribution defined by only the prior and the likelihood function. With enough samples we can estimate statistics such as the variance or mean of the posterior, and gain insight about our parameters. We will provide a short introduction to Markov processes and Monte Carlo sampling, but for the eager reader as this is not the main topic of this report we recommend looking at other sources for more insight.

A Markov process is a process which if presented with a the state i of a system generates a new state j. The Markov process must follow the principles of ergodicity and detailed, as well as the fact that it always has to put the system in a new state which includes the state $j = 1$. The principle of detailed balance states that as long as we have reached equilibrium the probability of a state moving from i to j should be equal to the probability of it moving from j to i, while the principle of ergodicity states that with a long enough chain one should be able to reach all states j from state i. The Markov process generates these new states using a transitional probability, which tells us the probability of the system moving from state i to j. Having defined our Markov Chain we can randomly initiate sequences of states from which to sample from. There are especially two things to keep in mind when sampling from a Markov chain. The first is that successive states in a chain are heavily correlated and we should therefore be careful not to sample states close to each other in a chain. The second is to not sample from the beginning of the chain as we have to let the process equilibrate. In other words, we wait to ensure that we get samples that are as close as possible to true samples from the intentional distribution.

C. Scikit-Learn & PyMC3

Scikit-Learn is a machine learning library developed for programming using Python (Pedregosa *et al.* 2011). It offers a large selection of algorithms for regression, classification, model selection and much more, and in this paper we will be making use of the package `linear_model.BayesianRidge()` as a comparison to our own model. The model performs estimations by maximizing the log marginal likelihood iteratively, so we will do the same using our model and compare the weights sampled by each of them.

PyMC3 is an open source probabilistic programming framework with functionalities including MAP estimation [1] and Markov chain Monte Carlo sampling. We wanted to investigate how this type of sampling technique would perform using a linear fit, more specifically we will be implementing a method using PyMC3s Hamiltonian Monte Carlo No-U-Turns (HMC-NUTS) sampler which has shown great convergence rates compared to other established methods of sampling (Hoffman and Gelman). Only looking at the MAP estimation has its limitations as it is not accompanied by an estimate of its uncertainty which often makes sampling from the posterior a better choice.

We set up normal prior distributions for our weight and the precision parameter β and defined the expected value of the outcome as $\mu = w_0 + w_1\mathbf{x}$, where $\mathbf{x}$ is the array containing the input values. We then defined the likelihood function as a normal distribution with mean $= \mu$ and standard deviation $= \beta^{-1}$ along with passing $\mathbf{t}$, the target values, as observed data. After having initiated our model we made use of the PyMC3 functions `find_MAP()` and `sample()` to make estimations of our parameters.

D. Datasets

We sampled x values between $(-1, 1)$ randomly from a uniform distribution. We produced one dataset by making corresponding y values

$$y_j = \left[\sum_{i=0}^{p} c_i x_j \right] + \epsilon_j, \tag{9}$$

and one as

$$y_j = \sin(2\pi x_j) + \epsilon_j, \tag{10}$$

where c are coefficients which can be chosen by preference, p is the degree of the polynomial and ϵ is noise

[1] Maximum a posteriori estimation is covered in this papers theoretical "parent" paper *An Introduction to Bayesian Linear Regression* (Gåsvær 2021)

generated from a normal distribution with mean $= \mu$ and standard deviation $= \sigma$, which can be adjusted as desired.

E. Structure

We began implementing a Gaussian conjugate prior and a function for updating the posterior using the method described in *P.R&M.L* (Bishop 2006, Ch 3.3.1). We start by assuming that the precision parameters α and β can be passed to our program. We make sure our program is able to take arbitrary basis functions and move on to producing a function to evaluation the log marginal likelihood as described in *P.R&M.L* (Bishop 2006, Ch 3.5.1). Finally we expand our program to be able to approximate the precision parameters α and β in an iterative manner, that's to say we are able to set initial values and run updates until satisfactory results are reached. Lastly we made use of both Scikit-Learns (Pedregosa *et al.* 2011) module for Bayesian Ridge regression and PyMC3s (Salvatier *et al.* 2016) NUTS-sampler to validate and compare our results.

III. RESULTS

A. General information

Unless explicitly specified all plots were created using precision parameters $\alpha = 5 \times 10^{-3}$ and $\beta = 1/\sigma^2$, where $\sigma = 0.2$ is the standard deviation of all datasets used in this report. For all results using a linear fit the coefficients used to create the dataset were $c_0 = -0.3, c_1 = 0.5$, ergo the golden values we wish the weights of our model to converge towards.

B. Linear Fit

Using polynomial basis functions of degree $p = 1$, we performed Bayesian Inference on data created using eq.9. In figure 1 we have plotted the prior/posterior probability distribution of the weights as well as the data space with posterior samples with 1σ prediction intervals for $N = 0, 2, 10, 100$ data points.

C. Gaussian Basis Functions

Changing our basis functions to Gaussian basis functions, we try to predict a fit using the sinusoidal dataset (still with Gaussian noise) from equation 10. We implement 9 different basis functions with the same standard deviation σ but different means μ sampled randomly from a uniform distribution between $(-1, 1)$. The results are shown in figure 2

D. Polynomial Basis Functions for Sinusoidal Data

Using the same dataset as for the Gaussian basis functions, we now switch to polynomial basis functions. We tried fitting the data using polynomials of degree $0-9$ and the resulting predicted line with prediction interval can be found in figure 3. We then calculated the log marginal likelihood for the models using the expression derived in the theoretical predecessor of this paper (Gåsvær 2021). The resulting plot is shown in figure 4. As one can observe from the figure, models of degree < 5 performs poorly compared to those of degree $>= 5$, which seemingly is a good match with figure 3.

E. Empirical Bayes

We approximated the hyperparameters α and β by setting initial values and following the iterative procedure described in *P.R&M.L* (Bishop 2006, Ch 3.5.2). The iterative process is illustrated in figure 5.

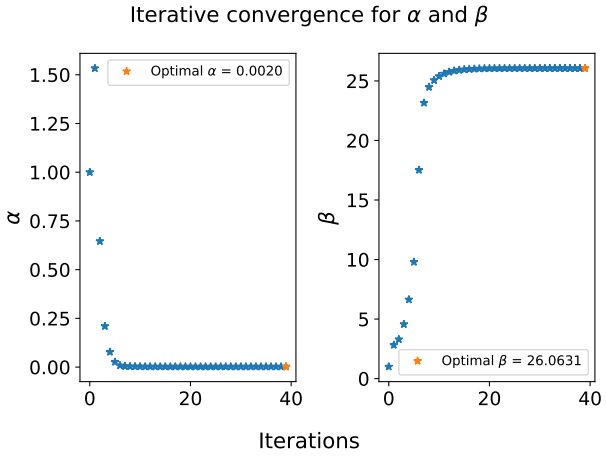

FIG. 5. Values for α and β updated through an iterative scheme until convergence defined as $\Delta\alpha, \Delta\beta < 10^{-5}$.

Using the optimal hyperparameters found, we updated our model and sampled weights to make predictions. We then used the package `linear_model.BayesianRidge()` from Scikit-Learn to compare our weights with. We trained the model and fitted it to the same dataset. The resulting prediction of both our model and the SKL model is shown in figure 6.

The weights sampled from the posterior distribution of our model an the weights from the model generated by Scikit-Learn are presented in table I.

F. Sampling with NUTS

As a final experiment we tested PYMC3s Hamiltonian Monte Carlo No-U-Turns (HMC-NUTS) on the linear

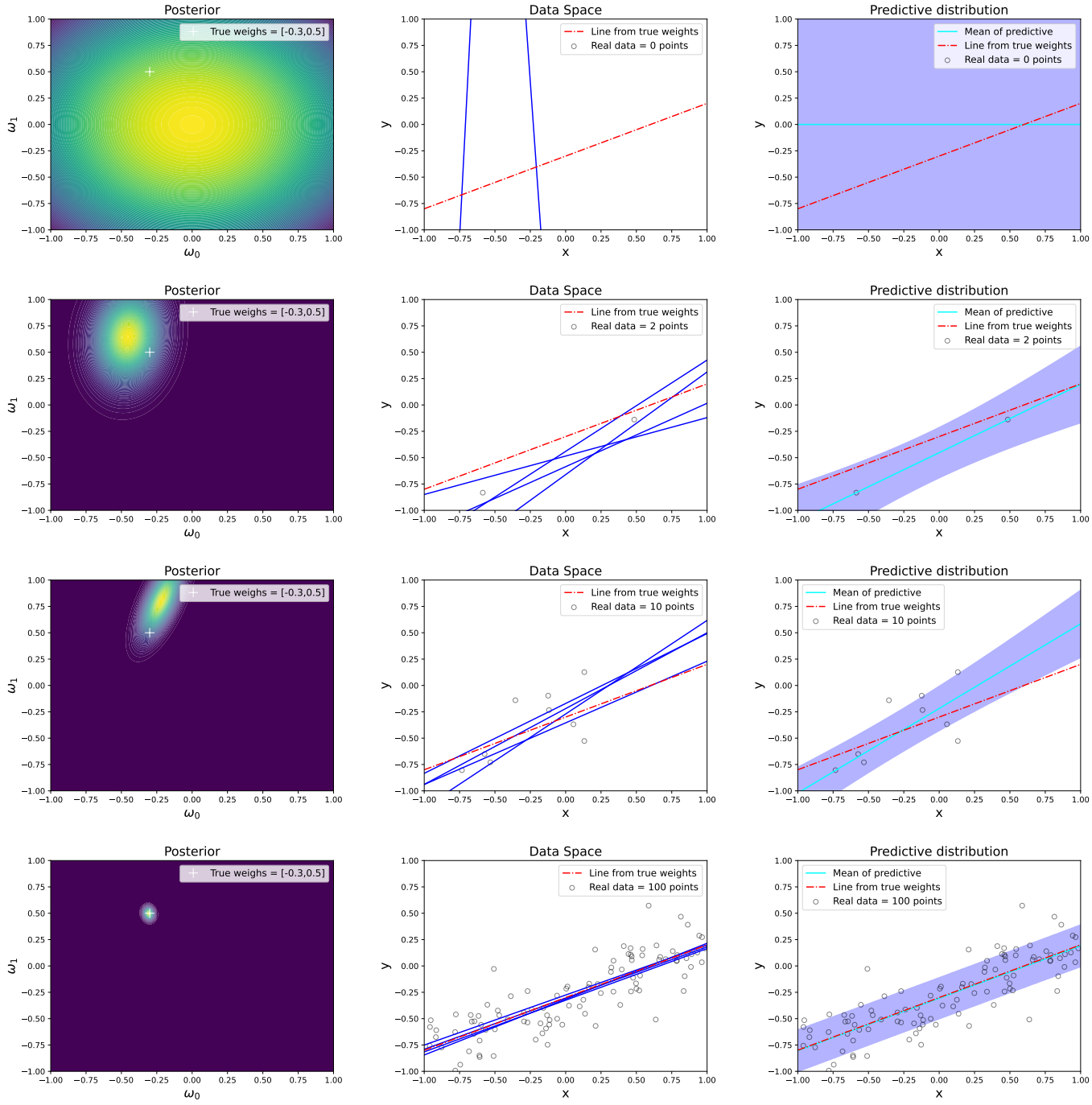

FIG. 1. Left column: Prior/Posterior of weights for $N = 0, 2, 10, 100$ data points. Middle column: Data space showing the real data points as well as line made from the real coefficients from the polynomial used to generate data. All other lines plotted are made using posterior samples of the weights. Right column: Showing 1σ prediction interval plotted together with the mean of the predictive distribution as well as real data points.

problem. We initiated priors as described under the *Theory & Method section* and the resulting posterior distributions after sampling together with corresponding plot of samples from the Markov chain are shown in figure 7. Note that the β estimated here is an approximation of the inverse standard deviation, not the inverse variance, of the likelihood function. The maximum a posteriori estimate found by the `find_MAP()` method were

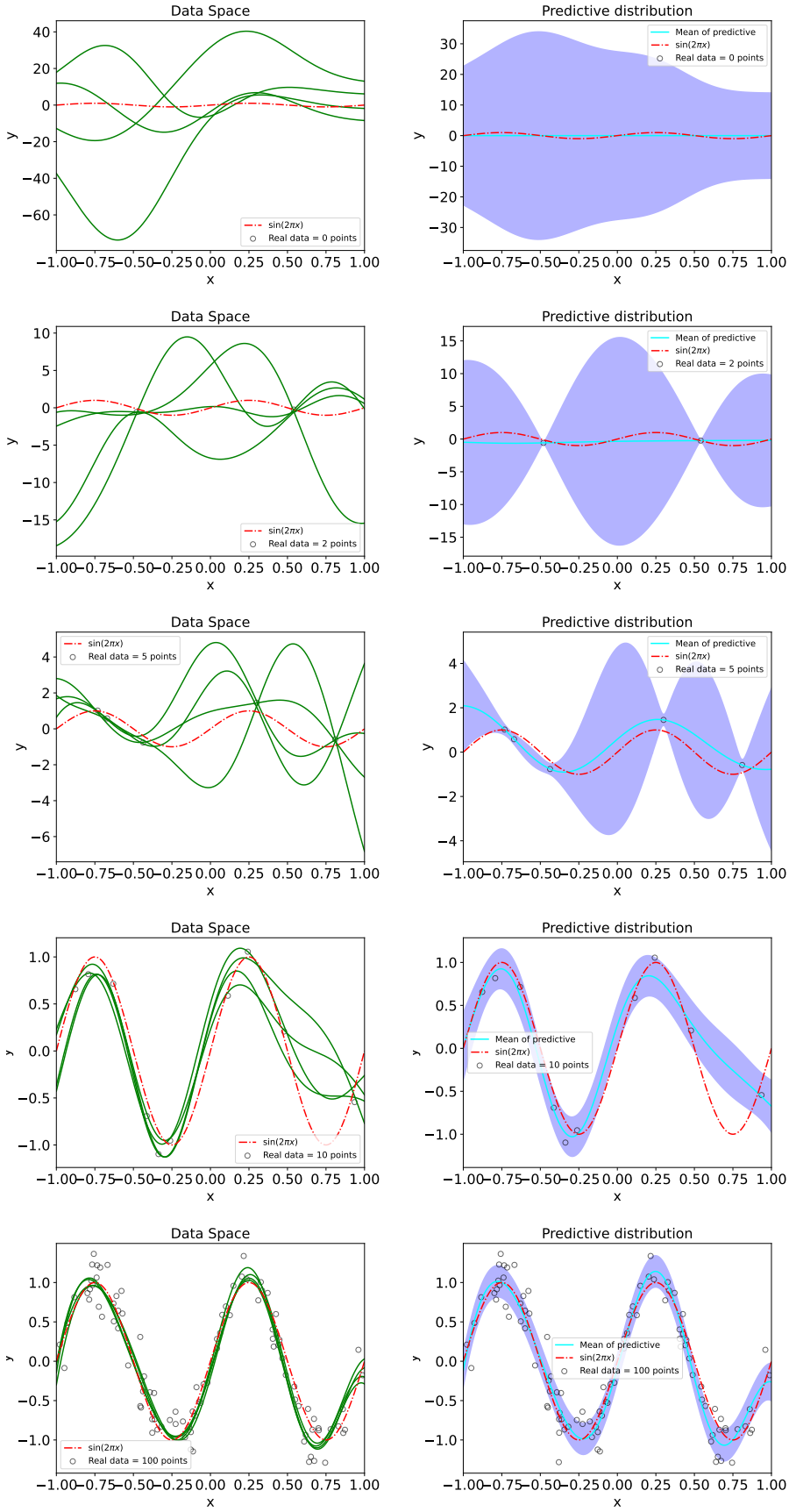

FIG. 2. Left column: Data space showing a line generated from the function $\sin(2\pi x)$, while all other lines are made using weights sampled from the posterior for $N = 0, 2, 5, 10, 100$ data points. Made using 9 Gaussian basis functions. Right column: Showing 1σ prediction interval plotted together with the mean of the predictive distribution as well as real data points.

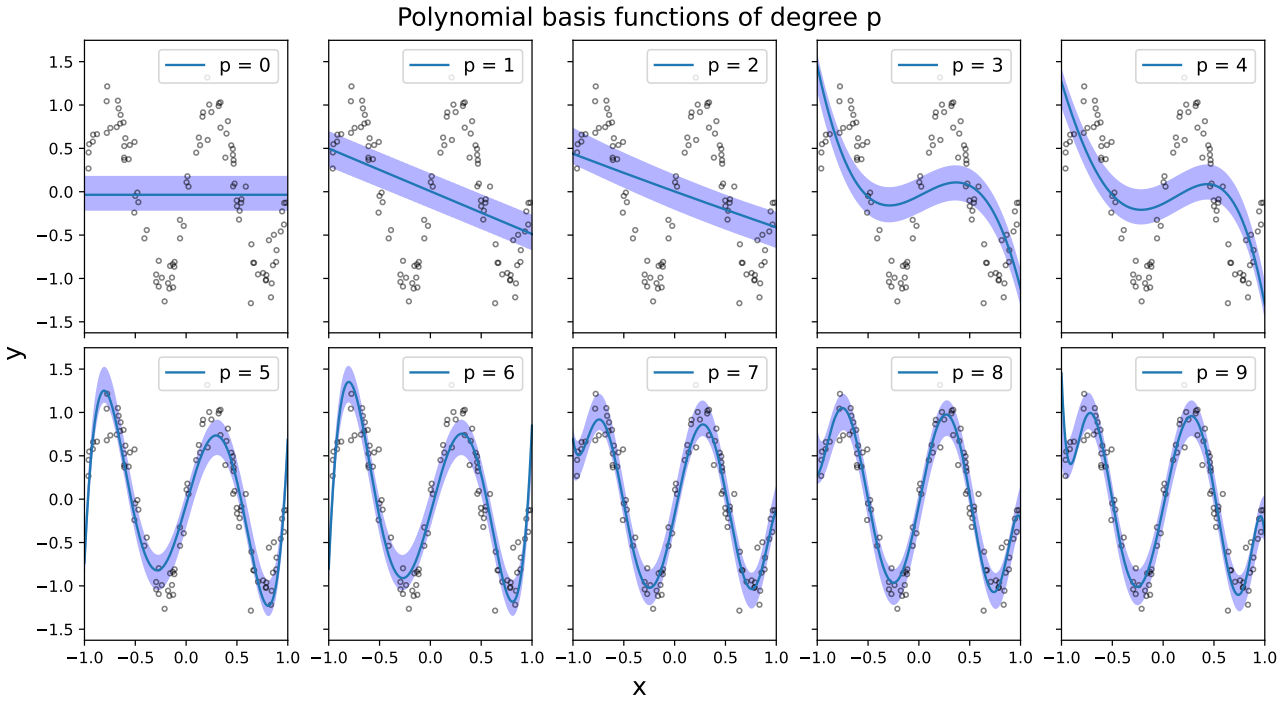

FIG. 3. Predicted line with prediction interval for polynomials of degrees 0–9. Scattered points are the real data points.

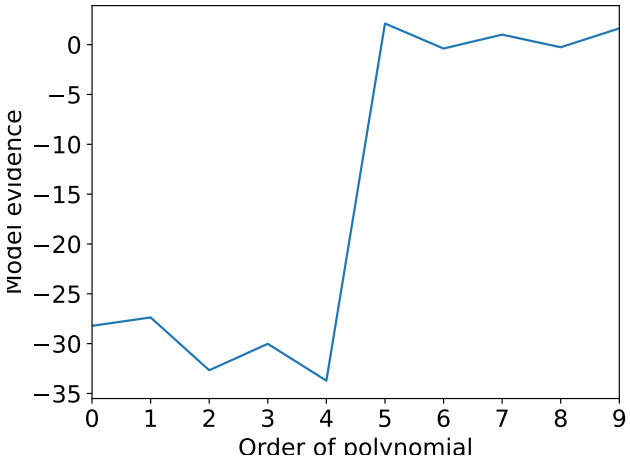

FIG. 4. Log marginal likelihood vs. degree of polynomial.

Our model vs. Scikit-Learn										
w	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
Our	-0.142	6.302	1.755	-37.574	-5.521	59.379	6.185	-32.155	-2.246	3.852
SKL	-0.127	6.000	1.670	-35.623	-5.163	54.710	5.136	-26.812	-1.308	1.357

TABLE I. Polynomial weights generated by our model and the Scikit-Learn BayesianRegression model.

$$\begin{aligned} w_0 &= -0.31 \\ w_1 &= 0.50 \\ \beta &= 5.08. \end{aligned} \tag{11}$$

IV. DISCUSSION

A. Linear Fit

We mentioned the requirements and advantages of sequential updating in the theoretical predecessor of this paper (Gåsvær 2021) which figure 1 is a good visualisation of. Studying the figure we clearly see that giving our model access to more data points very quickly sharpened the posterior distribution in the first column around the true weights. This fits well with our notion of how a posterior distribution using $N \to \infty$ data points would converge towards a sharp peak in the location of the true parameters. This is not a "true" sequential style of updating, but as the results are visualised in a sequential manner it proves some of the main points. In the second column we observe that as more data points are introduces, the lines created using weights sampled from the posterior align more and more with the line plotted from the true weights. In other words, as the posterior sharpens the difference between the weights sampled becomes smaller and smaller and more closely centred around the true values. In the third column we observe the same

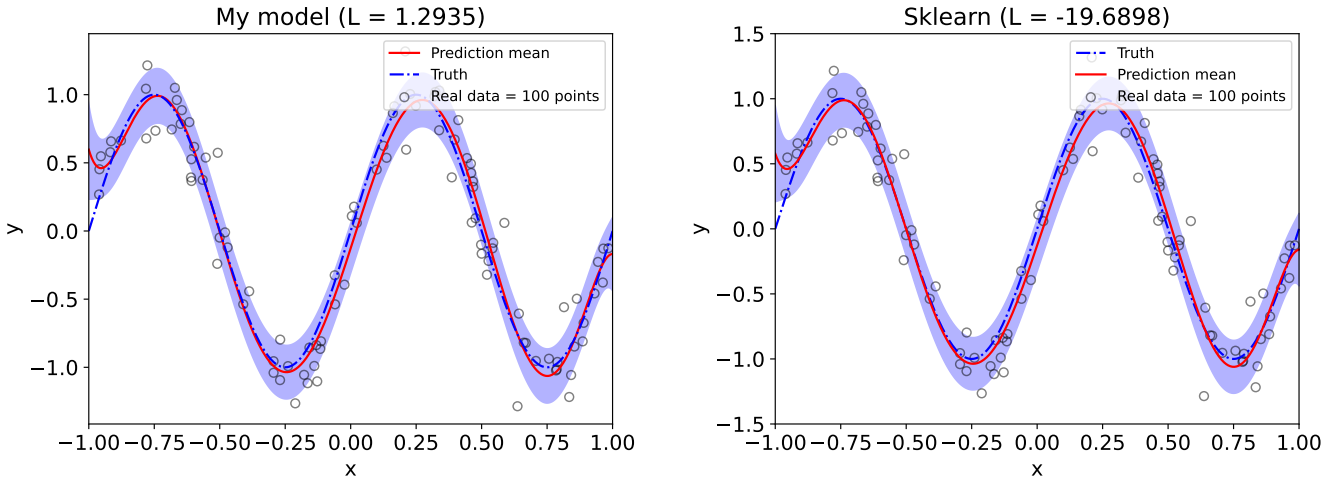

FIG. 6. Left: Data space showing a blue line generated from the function $\sin(2\pi x)$, while the red line is a prediction from the weights sampled from the posterior. Made using a polynomial basis function of degree 9. Also plotted is a 1σ prediction interval as well as real data points. Right: Training model and predicting using Scikit-Learn Bayesian Ridge Regression (Pedregosa *et al.* 2011). Data space showing a blue line generated from the function $\sin(2\pi x)$, while the red line is the mean of the predictive distribution.Also plotted is a 1σ prediction interval as well as real data points

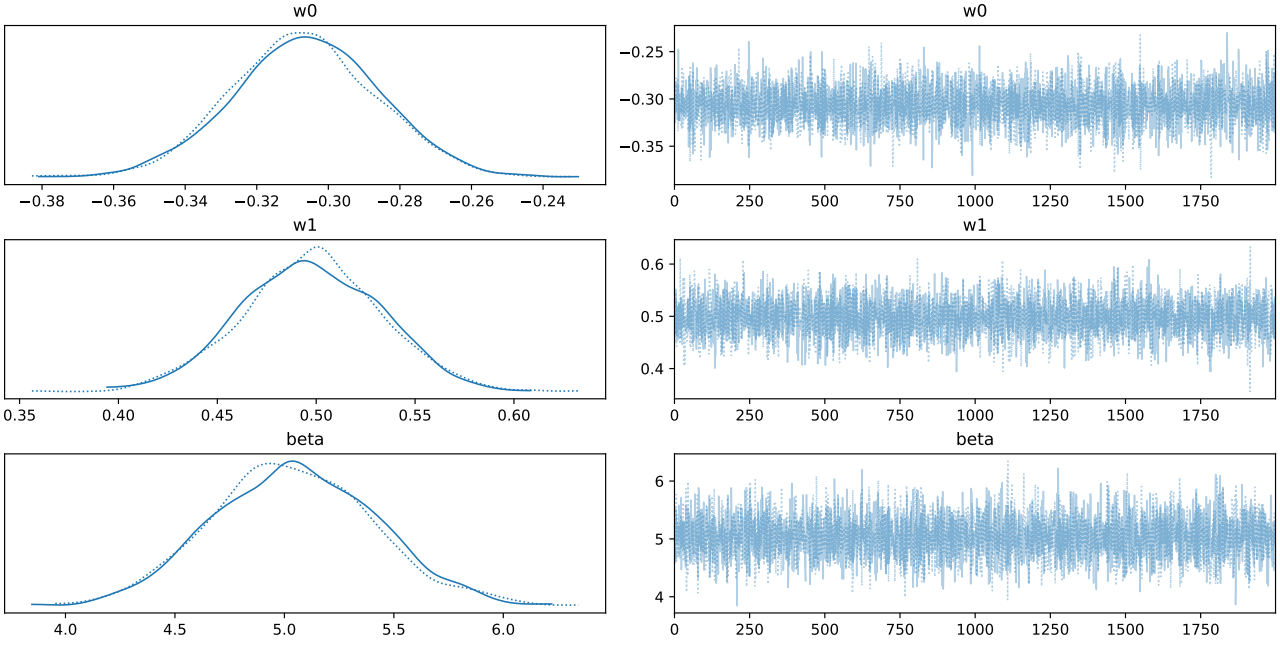

FIG. 7. Left: Smoothed histogram of the posterior distribution of each parameter. Right: Samples of the Markov chain plotted in sequential order

pattern. The predictive interval shrinks around the area of already observed data points to narrow the space we expect future data to lie.

We approximated the best precision parameters by trial and error. During this process we observed that typically larger choices of α resulted in poorer accuracy in the mean of the posterior, in other words it became sharper as more data was introduced but centred at the wrong values. As the precision matrix of a multivariate normal distribution is the inverse of its covariance matrix, it tells us something about the broadness or sharpness of the prior distribution. Choosing a large precision is equivalent to choosing a small variance in the prior which in turn signals more confidence in where we believe the true values lie. As we also discussed in the paper mentioned at the beginning of this section, when not having lots of quantifiable knowledge about the data one should choose a uninformative prior as it reduces the

dependence of the prior. Us then choosing a small value for α probably did reduced the dependency of the prior which worked better for this particular problem. For most datasets one does not know the true variance of the noise and one would have to test out different values for the precision parameter β. For this result we assumed it to be known but we still did some experimentation with different values just to validate our expectation of the true value being the one yielding the best results. Our assumptions were validated as we observed that the prior distribution grew less precise in accordance with β values further away from the true value.

B. Gaussian Basis Functions

As expected, looking at figure 2 we see that the addition of more data points greatly improves our models ability to fit the data. For $N = 100$ data points it looks like the basis functions have been able to pick up all the necessary information for approximating a sinusoidal function. Comparing the two columns make it very obvious how dependent the predictive intervals are of the posterior distribution. The intervals seem to encase and also mirror the extremities of the lines created using samples from the posterior. As a prediction interval is the space where we expect future data to fall and the posterior is the distribution from where we draw samples, the dependency makes perfect sense. Looking closer at the third plot in the second column we get a glimpse of one of the downfalls of using Gaussian basis functions. It appears that our model, based on the "slimness" of the predictive interval, is too confident in its prediction. This is often an issue as Gaussian basis functions are not well designed to make predictions outside of the area close to its centre. As explained in Ch.3.3.2 of *P.R&M.L* (Bishop 2006), in spaces far from the centres of the basis functions or for very large numbers of data points, the variance of the predictive distribution becomes solely dependent of the noise precision β.

C. Polynomial Basis Functions for Sinusoidal data

The polynomial degree can reasonably be viewed as both model selection, as a polynomial of degree 2 can be viewed as an entirely different model than one of degree 9, and parameter estimation as the "best" degree of the polynomial is a parameter which can be approximated. Our goal was to use the result shown in figure 3 to gain more insight about model selection. We observe that for low degrees of polynomial the model is unable to learn the necessary details of a sinusoidal function. As the degree of polynomial grows the model picks up more and by degree = 6 it looks like we have reached the ceiling of what can be improved in our model just by changing the degree. This is supported by the result shown in figure 4. Going off the principle of *Ockhams razor*, comparing figure 3 and 4 it can be argued that choosing a polynomial of degree 6 is favourable to one of degree 8 as they result in the same log marginal likelihood score and from what we can observe fit the data with the same degree of accuracy.

Now, having used both Gaussian basis functions and polynomial ones, we ask ourselves which one is the better fit? Since both do well by approximating our sinusoidal function for some chosen set of parameters it is not straight forward do declare one the winner. Again going off the principle of Ockhams razor, is the simpler model one with 9 Gaussians or a polynomial of degree 6? This is something one should explore further looking at both log marginal likelihood score and experimenting with for example sparseness in data.

D. Empirical Bayes and Comparison to Other Models

1. *Scikit-Learn*

In figure 6 we show a comparison between our model and the Scikit-Learn Bayesian Ridge Regression model using the sampled weights shown in table I. As can be seen from the table the weights produced by our model and the weights produced from Scikit-Learn Bayesian Regression model are consistent with each other. All corresponding weights are of the same magnitude and sign, which indicates that our model preforms well compared to more established models. At least it provides an indication of rightful implementation and supports the validity of our results. As expected from the comparable weights, both prediction and predictive intervals appear very similar for our model and the Scikit-Learn model. The marginal likelihood score for both models is harder to compare as they have different signs.

The log likelihood score is allowed to be positive. It is the sum of the log likelihood core of every observation, which can be both negative and positive depending on the size of the density. We are often dealing with receiving values smaller than one when sampling from a density which gives a negative when taking the natural logarithm. Lets imagine a normal density distribution centred around x with a very small standard deviation y. When sampling from an area very close to x, the sample can likely be a value larger than 1 which in turn gives us a positive log likelihood score. In other words when dealing with distributions with a small standard deviation positive log marginal likelihood scores can often be expected.

If we look at figure 5 see that both hyperparameters have seemingly converged after quite few iterations. This can be an indication of us having chosen a too strict tolerance for convergence and that it could be sufficient to chose a larger value in the future. The optimal value for α is quite small and as it is the inverse of the variance of the prior it indicates that the best fit for our problem is a

prior with large variance. This can be interpreted as using an uninformative prior rather than an informative one as we don't have enough quantifiable information about the data to form a good informative prior from. The optimal value of β, the inverse variance of the likelihood function, is approximated to roughly 26 which is not equal to the true inverse variance of the dataset $1/\sigma^2 = 25$, but we believe close enough as in terms of standard deviation in the data set they correspond to $\sigma = 0.196$ and $\sigma = 0.2$ respectively. As we found empirically when experimenting with the linear model, β values close to the true value of the inverse variance of the dataset gave more precise posterior distributions. While not being equivalent problems, it at least points to rightful implementation of the iterative procedure.

2. *PyMC3*

Looking at the plots in figure 7 we see that all distributions are more or less centred around the values of the coefficients and inverse standard deviation of the data set, which indicates that the model works very well. For the right hand figures of the sample trace we observe that the mean values are also practically identical to the true values. The advantage of using this model is that it gives you a lot of power using fairly simple implementation. Here we were able to find a posterior distribution over one of the hyperparameters β which tells us more about the parameter than a singe point estimate as well as being the more Bayesian way of performing parameter estimation. Comparing this to the MAP estimates presented in eq.11 the values are equally as accurate as the means of the distributions, but give us much less information about the parameters. The resulting posteriors from this kind of sampling can in turn be used as priors over the hyperparameters of another original model or in further work using PyMC3. Another thing to note is that, as mentioned earlier, for problems where the variables are of higher dimensions the calculation of the posterior can become extremely computationally expensive. That's to say it can become almost impossible to compute exactly, and we are dependent on methods such as Monte Carlo sampling to help us make approximations instead.

V. CONCLUSION

In this paper we have looked at different ways of using Bayesian inference to tackle linear regression problems. We saw the effects of having access to more data points on the precision of the posterior as well as the influence hyperparameters and choice of prior have on our models accuracy. This illustrated the impact choice of prior can have on our model, especially when having few data points. We saw that more than one type of basis functions can yield good results when looking at a specific set of data, but that determining which one is the better fit can be difficult even when having the principle of Ockhams razor in mind. We found that our model performed well compared to other established models which is an indication of rightful implementation as well as Bayesian inference being a method with high pay off. What we have covered in this report is, as the title states, an introduction to the practise. The results show that one can do a lot with a little and therefore we believe that Bayesian statistics and its applications can be both an intuitive starting point for someone going in to the field of statistics in the first place [2] and also highly motivating in terms of being rewarded with good results quite quickly. There is a lot more to be explored, for example comparing the log marginal likelihood scores of the Gaussian and polynomial basis functions on the sinusoidal data set to try to determine the best model, which we hope will be looked into if someone reproduces the results in this paper.

[1] K. S. Gåsvær, "An intoduction to bayesian linear regression," https://github.com/KasparaGaasvaer/Bayesian-ML/blob/main/Articles/An_Introduction_to_Bayesian_Linear_Regression.pdf (2021).

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).

[3] M. D. Hoffman and A. Gelman, "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo," http://www.stat.columbia.edu/~gelman/research/unpublished/nuts.pdf.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer-Verlag, Berlin, Heidelberg, 2006).

[5] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, PeerJ Computer Science **2**, e55 (2016).

[2] More on this in the paper *An Introduction to Bayesian Linear Regression* (Gåsvær 2021)