Project 1: Bayesian linear regression with ARD priors

Morten Blørstad

Abstract

Overfitting significantly impacts machine learning, especially with linear models encountering irrelevant features or complex basis functions. Traditional approaches involve training multiple models of varying complexities and selecting the one that best fits the data. Standard linear regression applies equal regularization across all features, potentially leading to underfitting by overly penalizing important features. Automatic Relevance Determination (ARD) addresses this by applying a flexible prior that zeroes out irrelevant features without penalizing relevant ones. This project demonstrates the effectiveness of ARD regression, utilizing a Gibbs sampler, in comparison to conventional regression models on simulated and real datasets. The project is part of the Probabilistic Machine Learning course at UiB.

1 Introduction

The project is about *Automatic Relevance Detection* (ARD), which is a Bayesian technique used in machine learning (ML) to automatically identify and eliminate irrelevant features from the data. This method addresses a fundamental challenge in ML, namely overfitting. Lasso/L1 or Ridge/L2 regularization are popular method that addresses the overfitting issue by promoting simpler models. However, they use a constant regularization factor which can lead to underfitting since it treats each feature equally. ARD addresses this by assigning individual data-dependent hyperpriors to each feature that controls their relevance, effectively shrining irrelevant features to zero whilst not shrinking relevant ones. This should lead to a simpler and more balanced model.

In this project, I demonstrate the effectiveness of ARD in a simulated study and on real data. The performance of the ARD is compared to ”standard” Bayesian Linear Regression, Linear Regression, Lasso Regression, and Ridge Regression. ARD is superior in simulated study being the best-performing model on all simulated datasets. ARD performs well in real data study but is outperformed by the Lasso Regression. Overall, this project illustrates how individual feature regularization can lead to better models.

The remainder of this project report is organized as follows: Section 2 presents the background on Linear model, ARD, and Gibbs sampling. Section 3 derives the full conditional distribution for the Gibbs sampler and describes the implementation of the Gibbs sampler. Section 4 entails the experiments, including data used, experimental setup, and results. Section 5 presents limitations and possible improvements and concludes.

2 Background

2.1 Linear Regression

A regression task of learning a function/model f that maps input $\mathbf{x} \in \mathbb{R}^d$ to the corresponding target variable $y \in \mathbb{R}$. In linear regression, the mapping is a linear combination of the input features, expressed as follows:

$$y = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon,$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight/parameter vector containing the coefficients that correspond to each feature in $\mathbf{x}$. The ϵ term is the error between y and $f(\mathbf{x})$, often assumed to be normally distributed with mean zero and variance σ_ϵ^2, $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon^2)$. To learn a non-linear function of the input $\mathbf{x}$ one can apply a set of *basis functions* $\boldsymbol{\phi}(\cdot)$,

$$y = f(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + \epsilon,$$

where $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_d)^T$.

2.2 Bayesian Linear Regression

Bayesian linear regression is a Bayesian approach to linear regression which integrates a prior probability distribution, over the model parameters $\mathbf{w}$, typically Gaussian $P(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$. This leads to the negative log posterior distribution

$$-\log P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta) = \frac{\beta}{2}(y - \mathbf{w}^T\mathbf{x})^T(y - \mathbf{w}^T\mathbf{x}) + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{constant},$$

which is equivalent to squared error loss with L2/Ridge regularization. Here α and β can be hyperparameters or hyperpriors (as we will see later on), representing the precision of the weights and noise, respectively. These hyperpriors are often assumed gamma-distributed (due to conjugate prior), $P(\cdot) = \Gamma(\cdot|k, \theta)$, where k and θ are hyperparameters controlling the shape and scale of the distribution. The updated belief, the posterior distribution $P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta)$, is obtained from the Bayes theorem and by completing the square:

$$P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w),$$

where $\boldsymbol{\Sigma}_w = (\beta\mathbf{x}^T\mathbf{x} + \alpha\mathbf{I})^{-1}$ and $\boldsymbol{\mu}_w = \beta\boldsymbol{\Sigma}_w)\mathbf{x}^T y$.

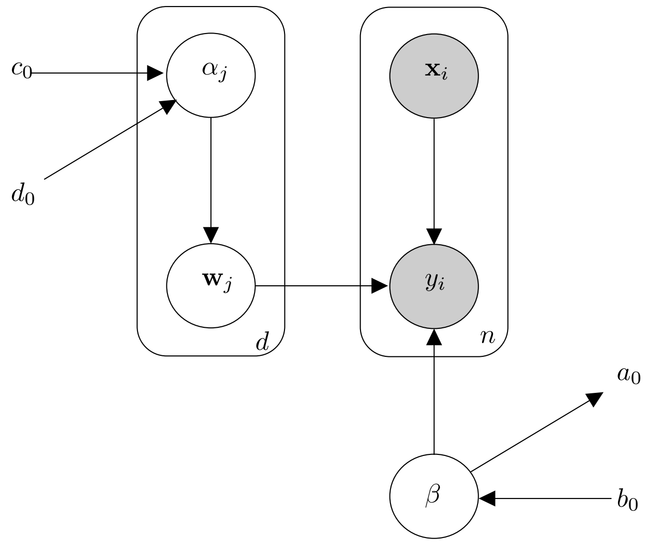

Figure 1: Plate diagram of the regression model with ARD prior.

2.3 Automatic Relevance Determination (ARD)

Automatic Relevance Determination (ARD) extends the Bayesian linear regression framework by identifying and eliminating irrelevant features from the input vector $\mathbf{x}$. It works by assigning a precision hyperprior α_j to each weight w_j. This allows the model to determine the relevance of each feature individually, shrinking irrelevant features to zero, resulting in a sparser model less prone to overfitting.

The model expresses the prior over the weights as a product of independent Gaussians:

$$P(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{d} \mathcal{N}(w_j|0, \alpha_j^{-1}),$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ represents the precision of each weight. Larger values of α_j correspond to a stronger belief that the weight w_j should be close to zero.

The $\boldsymbol{\alpha}$ are typically assumed to be Gamma distributed, due to conjugate relation to Gaussian distribution:

$$P(\alpha_j) = \Gamma(\alpha_j|k, \theta),$$

where k and θ are hyperparameters of the Gamma distribution that control its shape and scale.

2.4 Gibbs Sampling

Gibbs sampling (Geman and Geman, 1984) is a Markov chain Monte Carlo (MCMC) algorithm for sampling from a specified multivariate probability distribution when the joint distribution is not known explicitly or is difficult to sample from directly, but sampling from the conditional distribution is feasible.

In Gibbs, you need to be able to compute the full conditional distributions which often rely on the use of conjugate priors. The sampling consists of simulating a multivariate distribution by iterative sampling from the full conditional distribution of one variable given all the other variables. In more detail, the Gibbs sampling procedure involves updating the value of each variable with a value drawn from that variable's distribution conditioned on the current values of the remaining variables and then doing the same for the remaining variables (Pattern Recognition and Machine Learning book, Ch. 11 p.542).

3 Deriving the formulas and implementation

3.1 Derivation of the Full Conditional Distribution

To use Gibbs sampling for ARD linear regression I need the full conditional distribution for

- the posterior $P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta)$,
- the weight prior $P(\alpha|y, \mathbf{x}, \mathbf{w}, \beta)$,
- the noise prior $P(\beta|y, \mathbf{x}, \mathbf{w}, \alpha)$,

where $\alpha = (\alpha_1, \dots, \alpha_d)$. This section shows the derivation of the full conditional distributions.

The posterior distribution $P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta)$. Applying Bayes' theorem one gets:

$$\begin{aligned} P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta) &\propto P(y|\mathbf{x}, \mathbf{w}, \beta)P(\mathbf{w}|\alpha) \\ &\propto \exp\left\{-\frac{\beta}{2}(y - \mathbf{x}\mathbf{w})^T(y - \mathbf{x}\mathbf{w})\right\} + \exp\left\{-\frac{1}{2}\sum_{j=1}^{d}\alpha_j w_j^2\right\} \\ &= \exp\left\{-\frac{\beta}{2}(y - \mathbf{x}\mathbf{w})^T(y - \mathbf{x}\mathbf{w}) - \frac{1}{2}\sum_{j=1}^{d}\alpha_j w_j^2\right\} \end{aligned}$$

Rearrange to complete the square:

$$\begin{aligned} &= \beta(y - \mathbf{x}\mathbf{w})^T(y - \mathbf{x}\mathbf{w}) + \alpha\mathbf{w}^T\mathbf{w} \\ &= \beta(y^Ty - 2\mathbf{w}^T\mathbf{x}^Ty + \mathbf{w}^T\mathbf{x}^T\mathbf{x}\mathbf{w}) + \alpha\mathbf{w}^T\mathbf{w} \\ &= \beta y^Ty - 2\mathbf{w}^T(\beta\mathbf{x}^Ty) + \mathbf{w}^T(\beta\mathbf{x}^T\mathbf{x} + \alpha\mathbf{I})\mathbf{w} \end{aligned}$$

Match expressions

$$\boxed{(\mathbf{w} - \mu_\mathbf{w})^T\mathbf{\Sigma}^{-1}(\mathbf{w} - \mu_\mathbf{w}) = \mathbf{w}^T\mathbf{\Sigma}^{-1}\mathbf{w} - 2\mathbf{w}^T\mathbf{\Sigma}^{-1}\mu_\mathbf{w} + \text{constant}},$$

which corresponds to $\mathbf{\Sigma}^{-1} = \beta\mathbf{x}^T\mathbf{x} + \alpha\mathbf{I}$ and $\mu_\mathbf{w} = \beta\mathbf{\Sigma}\mathbf{x}^Ty$. Thus, the posterior distribution $P(\mathbf{w}|y, \mathbf{x}, \alpha, \beta)$ is a multivariate Gaussian with mean $\mu_\mathbf{w}$ and precision $\mathbf{\Sigma}^{-1}$, $\mathcal{N}(\mu_\mathbf{w}, \mathbf{\Sigma}^{-1})$.

The weight prior $P(\alpha|y, \mathbf{x}, \mathbf{w}, \beta)$.

Applying Bayes' theorem one gets:

$$\begin{aligned}
P(\alpha_j|w_j) &\propto P(w_j|\alpha_j)P(\alpha_j|c_0,d_0) \\
&\propto \sqrt{\alpha_j}\exp\left\{-\frac{\alpha_j}{2}w_j^2\right\}\alpha_j^{c_0-1}\exp\{-\alpha_j d_0\} \\
&\propto \alpha_j^{\frac{1}{2}}\exp\left\{-\frac{\alpha_j}{2}w_j^2\right\}\alpha_j^{c_0-1}\exp\{-\alpha_j d_0\} \\
&= \alpha_j^{c_0+\frac{1}{2}-1}\exp\left\{-\alpha_j(d_0+\frac{1}{2}w_j^2)\right\}.
\end{aligned}$$

This is recognized as a Gamma distribution with form:

$$\alpha_j \sim \Gamma(c,d),$$

where $c = c_0 + \frac{1}{2}$ and $d = d_0 + \frac{1}{2}w_j^2$.

The noise prior $P(\beta|y, \mathbf{x}, \mathbf{w}, \alpha)$.

Applying Bayes' theorem and dropping variable not in question one gets:

$$\begin{aligned}
P(\beta|y,\mathbf{x},\mathbf{w},\alpha) &\propto P(y|\mathbf{x},\mathbf{w},\beta)P(\beta|a_0,b_0) \\
&\propto \prod_{i=1}^{n}\left[\sqrt{\beta}\right]\exp\left\{-\frac{\beta}{2}(y-\mathbf{xw})^T(y-\mathbf{xw})\right\}\beta^{a_0-1}\exp\{-\beta b_0\} \\
&\propto \beta^{\frac{n}{2}}\exp\left\{-\frac{\beta}{2}(y-\mathbf{xw})^T(y-\mathbf{xw})\right\}\beta^{a_0-1}\exp\{-\beta b_0\} \\
&= \beta^{a_0+\frac{n}{2}-1}\exp\left\{-\beta(b_0+\frac{1}{2}(y-\mathbf{xw})^T(y-\mathbf{xw}))\right\}.
\end{aligned}$$

This is recognized as a Gamma distribution with the form:

$$\beta \sim \Gamma(a,b),$$

where $a = a_0 + \frac{n}{2}$ and $b = b_0 + \frac{1}{2}(y - \mathbf{xw})^T(y - \mathbf{xw})$.

3.2 Implemetation

I have implemented the ARD Bayesian Regression using Gibbs sampling. I have organized the code as a class `ARDRegression()` following the interface of Sklearn's `BaseEstimator` and `RegressorMixin`. With the `ARDRegression().fit()` function one training the model using the Gibbs sampling described in Algorithm 1. Model predictions are made by sampling from the prediction distribution after burn-in. In Algorithm 1, we iteratively swap between sample w from conditional distribution of w given the current values of β and α, sample β from conditional distribution of β given the current values of w and α, and sample α from conditional distribution of α given the current values of w and β.

4 Experiments

This section describes the experimental setup and presents results with discussions. First, I provide a data description, then I describe the experimental setup. Finally, I present the result with a discussion. Each part alternates between the experiments on simulated and real data.

4.1 Data

This section describes the data used in this project, both simulated and real data. A dataset $\mathcal{D}$ of size N consists of an feature matrix $\mathbf{x} \in \mathbb{R}^D$ and a target variable $y \in \mathbb{R}$, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$.

Algorithm 1 ARD Regression with Gibbs Sampling

```
1: Input: Data $\mathcal{D} = \{\mathbf{x}, y\}$, Hyperparameters $(a_0, b_0, c_0, d_0)$, Iterations $n_{iter}$, Initial $\beta$
2: Initialize $\mathbf{w} \leftarrow \mathbf{0}$, $\boldsymbol{\alpha} \leftarrow \mathbf{1}$
3: for $i \leftarrow 1$ to $n_{iter}$ do
4:     $\Sigma_w^{-1} \leftarrow \beta \mathbf{x}^T \mathbf{x} + \text{diag}(\boldsymbol{\alpha})$
5:     $\Sigma_w \leftarrow (\Sigma_w^{-1})^{-1}$
6:     $\mu_w \leftarrow \beta \Sigma_w \mathbf{x}^T y$
7:     $\mathbf{w} \leftarrow$ Sample from $\mathcal{N}(\mu_w, \Sigma_w)$
8:     $a \leftarrow a_0 + \frac{n}{2}$
9:     $\hat{\mathbf{y}} \leftarrow \mathbf{x}\mathbf{w}^T$
10:    $b \leftarrow b_0 + \frac{1}{2}(y - \hat{\mathbf{y}})^T(y - \hat{\mathbf{y}})$
11:    $\beta \leftarrow$ Sample from $\Gamma(a, \frac{1}{b})$
12:    for $j \leftarrow 1$ to $D$ do
13:        $c \leftarrow c_0 + \frac{1}{2}$
14:        $d \leftarrow d_0 + \frac{1}{2} w_j^2$
15:        $\alpha_j \leftarrow$ Sample from $\Gamma(c, \frac{1}{d})$
16:    end for
17:    Store samples $\mathbf{w}$, $\beta$, $\boldsymbol{\alpha}$
18: end for
19: return Posterior samples of $\mathbf{w}$, $\beta$, $\boldsymbol{\alpha}$, and log probabilities
```

4.1.1 Simulated Data

I perform a simulated experiment to verify whether the ARD priors can mitigate overfitting. The simulated experiment uses five generated datasets varying $\mathbf{x}_i$ and $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $f(\mathbf{x}_i) = \text{bias} + \sum_{j=1}^{D} w_j \mathbf{x}_{i,j}$ and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is Gaussian noise. I use different functions f, noise levels $\sigma^2_{epsilon}$, sample sizes N, number of features D and number of relevant features R to get diversified datasets. Different features $\mathbf{x}_j$ are generated from a uniform distribution, $\mathcal{U}(-10, 10)$. Different functions are generated by sampling their weights w_j from a uniform distribution, $\mathcal{U}(-20, 20)$. The bias term is set to 0 for each dataset. More details can be found in Table 1.

Name	samples (N)	features (D)	relevant features (R)	noise σ_ϵ^2
`S1`	50	2	1	20
`S2`	50	10	8	5
`S3`	100	25	5	10
`S4`	100	50	5	25

Table 1: Simulated datasets

- `S1` is a simpler dataset with only two features. The objective of this dataset is to see how `ARD` perform in a simpler case. The dataset also makes it easier to plot functions from the posterior distribution.
- `S2` have less noise and few irrelevant features. The objective of this dataset is to see how `ARD` perform on a dataset where there is less risk of overfitting on (small noise and few irrelevant features)
- `S3` has many features where 25% are irrelevant. The objective of this dataset is to see how `ARD` perform on a dataset it in theory should be good at.
- `S4`, is similar `S3`, only with more features where 50% are irrelevant and more noise. The objective of this dataset is to see how `ARD` perform on a hard dataset which may also be difficult for `ARD`.

4.1.2 Real Dataset

An experiment with a real dataset was conducted to determine if the findings from simulated experiments are applicable in a real-world context. Here, I use the `Lung_cancer_small` dataset which is a smaller dataset of the lung cancer genomic data from the Chemores Cohort Study[1]. The `Lung_cancer_small` dataset consists of 123 samples with

[1]More details can be found here: https://rdrr.io/cran/PRIMsrc/man/Real.2-data.html.

99 features (age, type, KRAS status, EGFR status, P53 status + 94 miRNA expression levels) to predict patients' "Disease-Free Survival Time".

Note that the dataset also contains a variable "delta" which indicates which patients were still alive when the study ended. This variable is excluded as it is used for censored regression.

4.2 Experimental Setup

4.2.1 Simulated Data

Models: We compare the "standard" Bayesian Linear Regression (`BRL`), Bayesian Linear Regression with ARD prior (`ARD`), standard Linear Regression (`LR`), Ridge Regression (`Ridge`), and Lasso Regression (`Lasso`).

Hyperparameters: The `BRL` and `ARD` used $(a_0 = 0.1, b_0 = 0.1)$. For the additional hyperparameter the `ARD` use $(c_0 = 0.1, d_0 = 0.01)$. For `Ridge` and `Lasso` α is a hyperparameter which is set to 0.1. The impact of the priors is larger for smaller sample sizes for the Bayesian model. Therefore I acknowledge that the choice of (a_0, b_0) and (c_0, b_0) hyperparameters ideally should be tuned. This is more important for smaller sample sizes and less important for larger sample sizes. Similarly, α should be tuned for `Ridge` and `Lasso`.

Performance metric: The posterior of the weights is based on Gaussian likelihood (assumes $\mathcal{N}(y_i|\mathbf{x}_i, \sigma_\epsilon^2)$). Given that the squared error is based on the Gaussian likelihood assumption, I use the mean squared error (MSE) to evaluate the models.

Pipeline: For each of the simulated datasets, we simulated N training samples (see Table 1) and 1000 validation and test samples. Training data is used to train the models, validation data are used for model selection and test data is used to give an unbiased performance estimate.

For the Bayesian models, the Gibbs sampler uses 10,000 iterations to learn the parameters using the training data. The burn-in period is set to half of Gibbs samples, as recommended in Exercise 9. Visualization of the trace plots and the marginal distributions (excluding the burn-in period) of parameters is used to assess convergence and mixing. Predictive distributions are evaluated by verifying that the actual target variable proportions within the 50% and 95% credible intervals are as expected. Finally, I check the parameter estimates and their uncertainty for the Bayesian models and their estimates are compared to `LR`, `Ridge`, and `Lasso`.

4.2.2 Real Data

Models, **Hyperparameters**, and **Performance metric** are the same as in the simulated case (Section 4.2.1).

Pipeline: We divide the dataset into training and test data (80/20). We train and evaluate the models using 5-fold cross-validation on the training set and select the model with the lowest MSE validation score. The test set is used to give an unbiased performance estimate

For the Bayesian models, the Gibbs sampler uses 10,000 iterations to learn the parameters using the training data with a burn-in period set to half. Convergence and mixing are assessed using trace plots and marginal distributions of parameters. Parameter estimates and their uncertainty for the Bayesian models are plotted and compared to Parameter estimate plots for `LR`, `Ridge`, and `Lasso`.

4.3 Experimental Results

4.3.1 Simulated Data

4.3.1.1 Model Selection

	S1		S2		S3		S4	
Model	Training	Validation	Training	Validation	Training	Validation	Training	Validation
ARD	383.59	**405.14**	17.12	**30.03**	73.90	**112.49**	537.37	**635.50**
BLR	372.64	416.88	17.12	32.51	66.38	127.70	444.72	782.93
LR	372.90	412.64	16.85	31.12	66.22	127.58	444.51	777.80
Lasso	372.90	412.59	16.85	31.20	66.23	127.00	444.53	774.67
Ridge	372.90	412.65	16.85	31.18	66.22	127.58	444.51	777.79

Table 2: Comparative performance of models on simulated datasets S1 to S4.

From the validation scores in Table 2, the ARD model performed the best and was selected on all four datasets. The ARD model's unbiased performance estimate on unseen data is 432.83, 29.10, 131.29, and 800.15 on S1 to S4, respectively. Interestingly, BRL performs the worst of the models on all the datasets.

When comparing training and Validation MSE for the four datasets, all models have larger MSE on the validation than the training set, which may indicate overfitting. This might suggest that hyperparameters that impact regularization could be adjusted to favour even simpler models. The difference between training and validation scores is smallest for ARD model, followed by Lasso.

This simulated study showcases that having individual regularization strength for each model parameter can be beneficial as ARD performs the best across all the simulated datasets.

4.3.1.2 Analysis of the Bayesian Models

Figure 2 shows the trace plots for the log unnormalized log probabilities over the Gibbs sampling iterations for the four simulated datasets for ARD and BRL. Based on the figure, there is no clear trend (samples do not seem to be correlated) and seems to be stationary (fluctuating around a mode). This suggests convergence and good mixing.

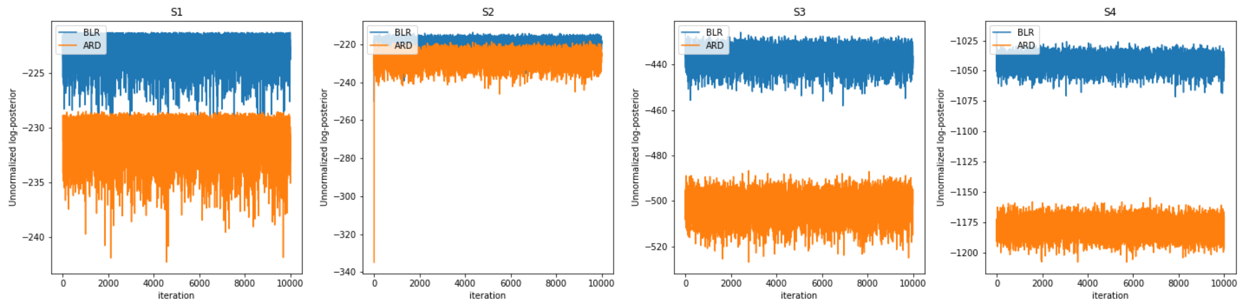

Figure 2: Trace plots of log unnormalized log probabilities of ARD and BRL on simulated datasets.

Figure 3 and 4 show the trace plots and marginal distributions of 2 randomly selected weights from $\mathbf{w}$ and β for the four simulated datasets, respectively. Similar to the trace plots for the log unnormalized log probabilities, the trace plots for $\mathbf{w}$ and β show no clear trends (samples do not seem to be correlated) and seem to be stationary (fluctuating around a mode). This further suggests convergence and good mixing. The histograms of the marginal distributions of the weights and β show unimodal (single peak) and symmetric around the mean. This suggests well-behaved models.

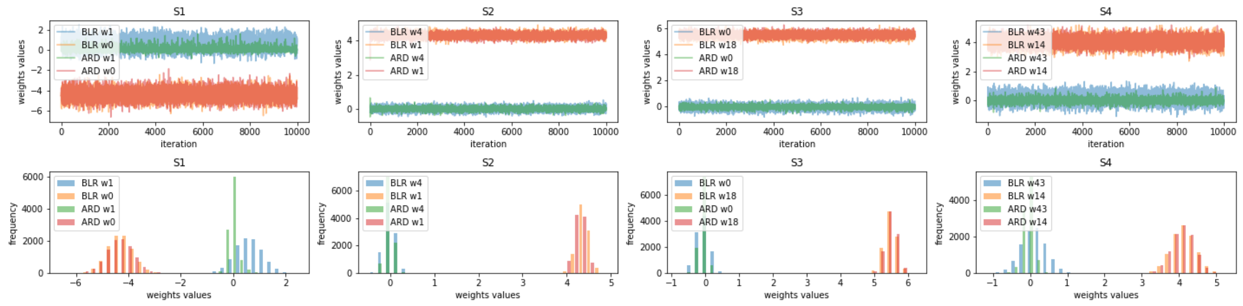

Figure 3: Trace plots 2 randomly selected $\mathbf{w}$, and β addition to histograms of $\mathbf{w}$ and β of ARD and BRL on simulated datasets.

Figure 5 shows the trace plots and marginal distributions of αs corresponding to the weights in Figure 3 for ARD on the four simulated datasets. Similar to the other trace plots, the trace plots for α show no clear trends and seem to be stationary, further suggesting convergence and good mixing. The histograms of the marginal distributions of the αs show unimodal and look very similar to a Gamma distribution. This suggests a well-behaved model.

The trace plots and marginal distributions strongly suggest that the Gibbs sampler of ARD and BLR has converged.

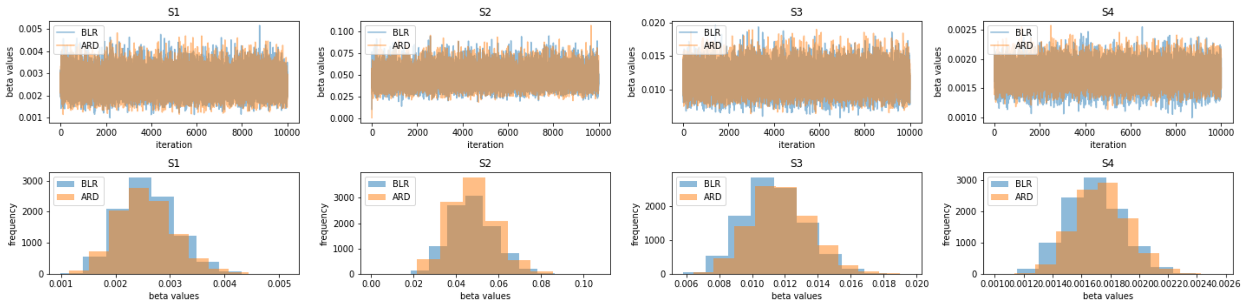
Figure 4: Trace plot and histogram of the α of the ARD model on simulated datasets.

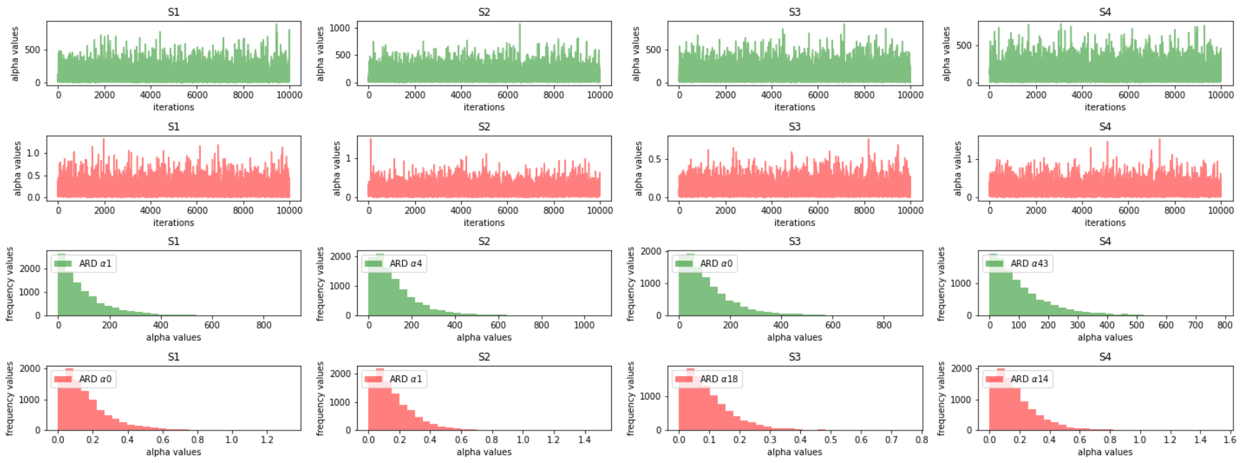
Figure 5: Trace plot and histogram of the α of the ARD model trained on `S1`.

Table 3 shows the proportion of true target observations that are within the 50% and 95% credible interval of the predictive distribution for training and validation set across the simulated data. Here one expects the result to be around 50% and 95% for the 50% and 95% credible intervals, which they are. This suggests that the predictive distributions are well-calibrated, providing credible intervals that accurately reflect the underlying uncertainty in the model's predictions. Figure 6 visualizes 95% credible interval for the training data on `S1` for `ARD` from Table 3. It plots the mean predictions with a 95% credible interval from the predictive distribution for `ARD` model for training data of `S1`.

	S1		S2		S3		S4	
				Training				
Model	50% CI	95% CI	50% CI	95% CI	50% CI	95% CI	50% CI	95% CI
ARD	50.0	96.0	50.0	94.0	53.0	95.0	51.5	96.0
BLR	52.0	96.0	50.0	94.0	53.0	95.0	51.0	96.0
				Validation				
ARD	46.2	92.7	49.6	95.0	50.0	94.4	51.2	95.4
BLR	46.7	93.5	49.6	94.9	49.8	94.4	50.4	95.4

Table 3: Present the proportion of target variable (in percentage) that lies within the credible intervals (50% and 95%) for the training and validation sets for the simulated datasets S1 to S4.

Figure 7 plots 10 functions from the posterior distribution using a thinning factor of 10 to break the correlation. The functions make sense as they seem to fit the data reasonably by capturing the underlying data pattern.

Figure 8 shows the parameters for the different models on the simulated datasets. From Table 2 we found `ARD` to be the best-performing model for all datasets. This can be explained by `ARD` being better to set weights for irrelevant features to zero as Figure 8 shows. This is especially highlighted for `S1`, `S2` and `S4`. `ARD` ability to ignore irrelevant

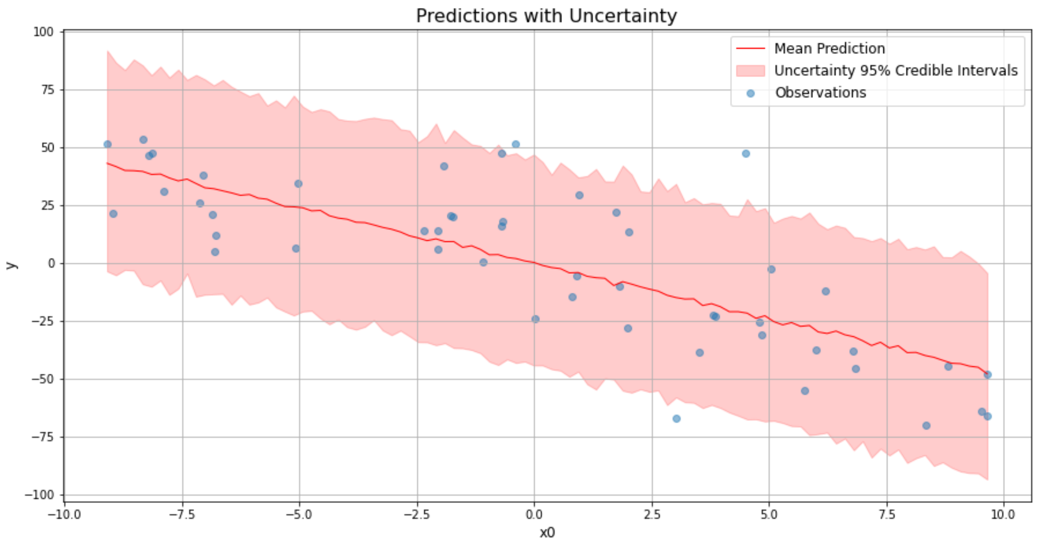

Figure 6: Plots the mean predictions from the predictive distributions with 95% credible interval of the ARD model trained on `S1`.

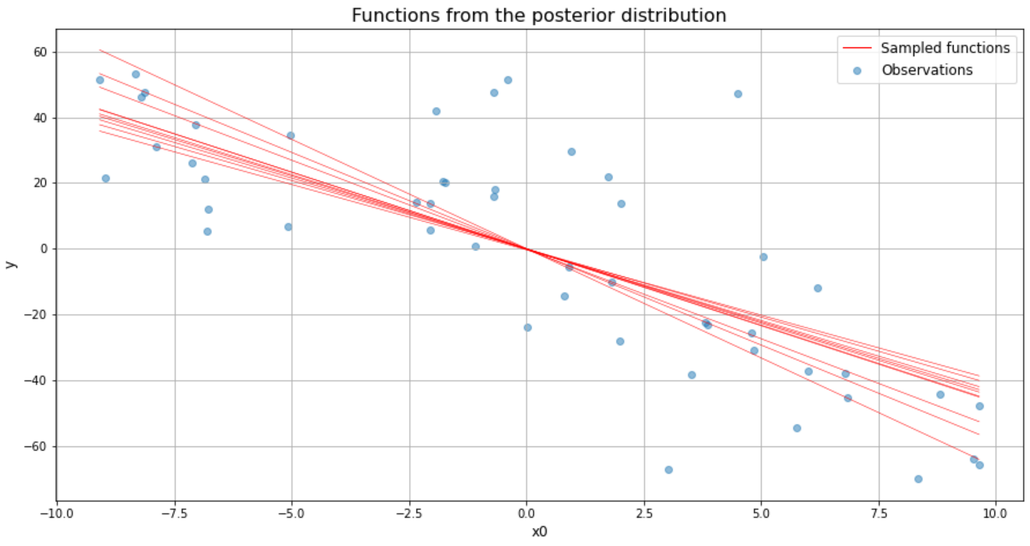

Figure 7: Plots 10 functions from the posterior distribution of the ARD model trained on `S1`.

features results in better performance, which, again, suggests that having individual regularization strength for each model parameter can be beneficial.

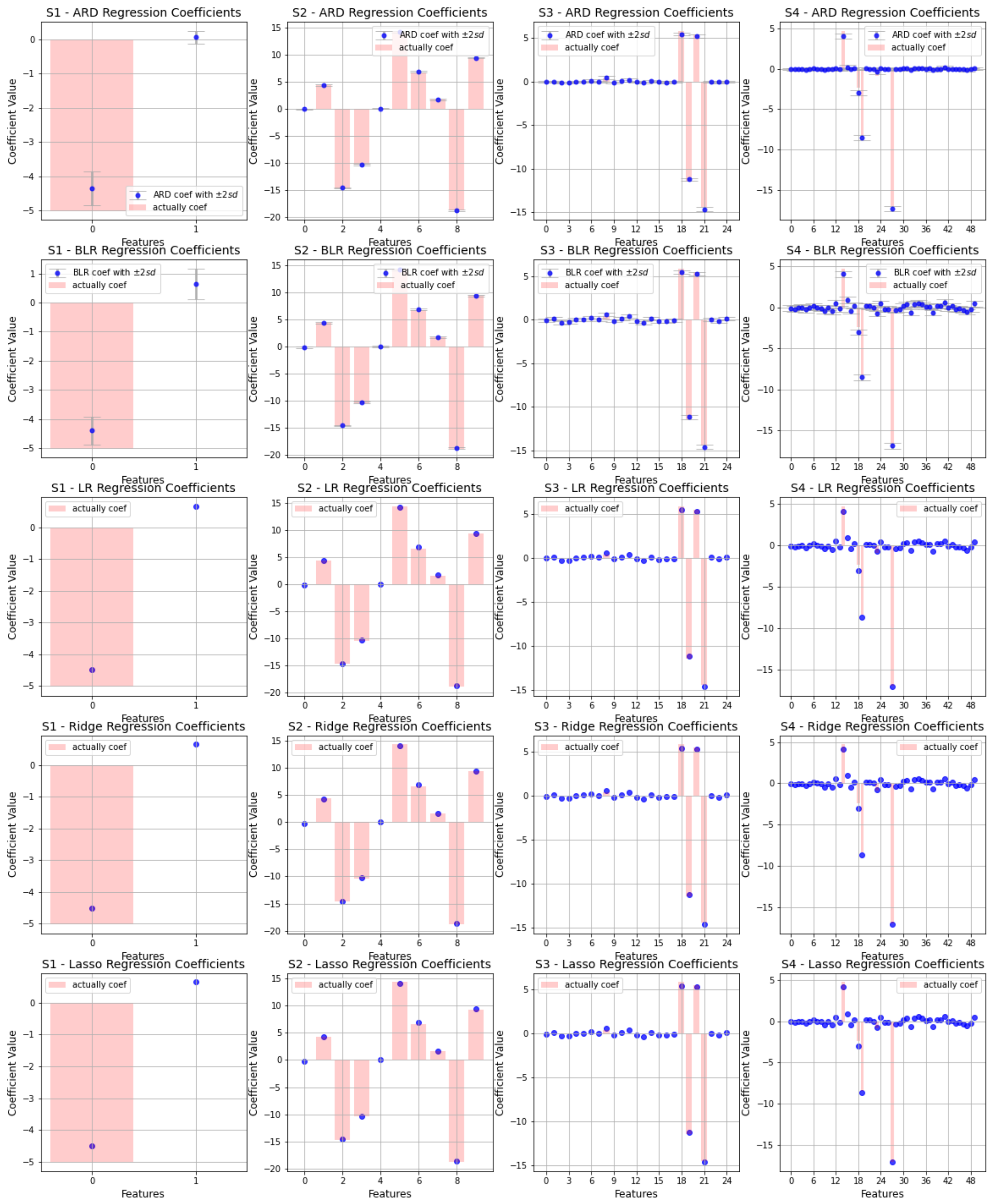

Figure 8: Plots the model weighs for the different models on the simulated datasets. Bayesian models show the mean weights with uncertainty (± 2 standard deviations).

4.3.2 Real Data

4.3.2.1 Model Selection

The best model was Lasso with a validation score of 5.89 from cross-validation. Based on the test data, the Lasso model's unbiased estimate of performance on unseen data is 5.01.

From the validation scores for the `Lung_cancer` dataset in 4, the `Lasso` model has the best with a validation score of 5.89 and was selected. The `Lasso` model's unbiased performance estimate on unseen data is 5.01. The `ARD` model was the second-best model with a validation score of 7.98.

When comparing training and Validation scores, all models have much larger MSE on the validation than the training set, which strongly indicates overfitting. The difference between training and validation scores is smallest for `Lasso` model, followed by `ARD`. The real data study shows an example where having individual regularization strength does not result in the best model, providing a counter-example to the simulated study. Explanation to why `Lasso` is better than `ARD` on this dataset may be due to improper hyperparameters for `ARD`, resulting in worse priors with not enough data to make up for it.

Later in Section 4.3.2.3, I compare the model weights of `Lasso` and `ARD` to analyse feature relevance.

Model	Training (SD)	Validation (SD)
ARD	0.91 (0.15)	7.98 (1.57)
BLR	0.0 (0.0)	25.22 (3.93)
LR	0.0 (0.0)	25.95 (4.1)
Lasso	2.23 (0.1)	**5.89 (0.48)**
Ridge	0.01 (0.0)	23.22 (2.65)

Table 4: Comparative performance of models on the `Lung_cancer` dataset.

4.3.2.2 Analysis of the ARD Model

Figure 9 shows the trace plots for the log unnormalized log probabilities over the Gibbs sampling iterations for `ARD` on the `Lung_cancer` dataset. Based on the figure, there is no clear trend (samples do not seem to be correlated) and seems to be stationary (fluctuating around a mode). This suggests convergence and good mixing.

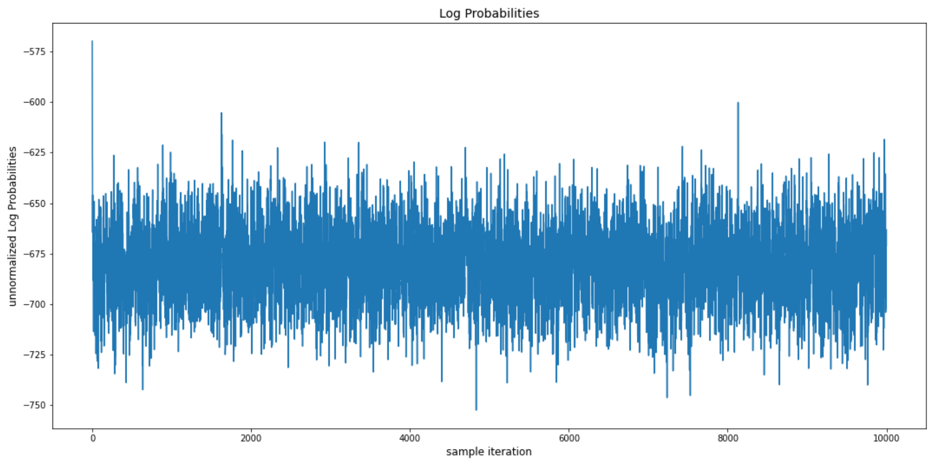

Figure 9: Trace plots of log unnormalized log probabilities of `ARD` on the `Lung_cancer` dataset.

Figure 10 shows the trace plots and marginal distributions of β, 2 selected weights from $\mathbf{w}$ and their corresponding αs. Similar to the trace plots for the log unnormalized log probabilities, there are no clear trends (samples do not seem to be correlated) and seem to be stationary (fluctuating around a mode). This further suggests convergence and good mixing. The histograms of the marginal distributions of the weights,α and β show unimodal (single peak). The weights and β look symmetric around the mean, whereas α looks very similar to a Gamma distribution. This suggests well-behaved models. The trace plots and marginal distributions strongly suggest that the Gibbs sampler of `ARD` has converged.

4.3.2.3 Feature Relevance

Figure 11 compares the weight of `ARD` and `Lasso`. From the figure, one sees that `ARD` allows for higher weights for the feature it deems relevant than the `Lasso`. Relevant features for `ARD` range from -1.1 to 1.6, whereas for

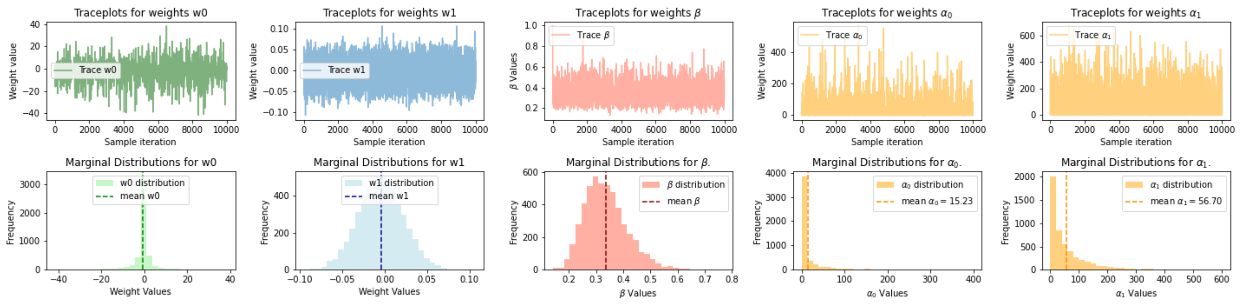

Figure 10: Trace plots and marginal distributions of β, 2 selected weights from $\mathbf{w}$ and their corresponding αs on the `Lung_cancer` dataset.

`Lasso` it is from -0.48 to 0.33. Based on weight values, `Lasso` suggests feature index 65 ($w_{65} = 0.33$) and 47 ($w_{48} = -0.48$) to be the most relevant features, whereas for `ARD` the two most important features are $w_{65} = 1.6$ and $w_{49} = -1.1$. Here the weights indices are corrected so they are comparable (`Lasso` excludes whereas `ARD` includes bias in Figure11).

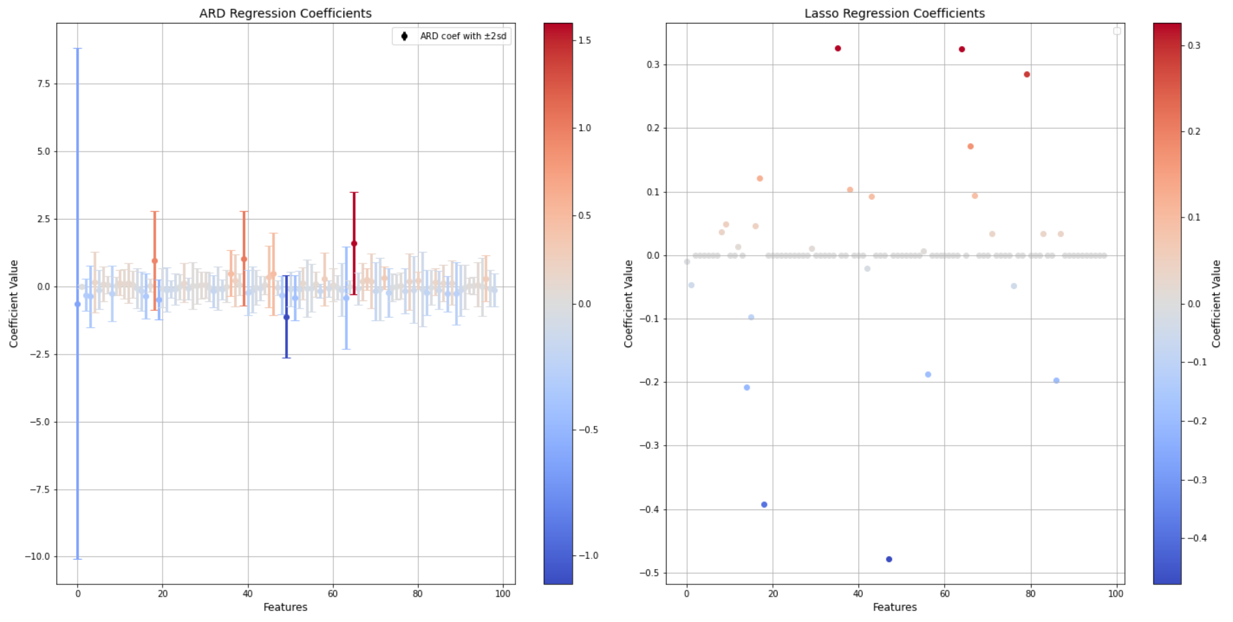

Figure 11: Plots the weights values of the `ARD` and `Lasso` on the `Lung_cancer` datasets. The `ARD` values also report uncertainty (± 2 standard deviations). The `ARD` includes bias as a weight (w_0), whereas `Lasso` does not. This mean w_1 of `ARD` is comparable to w_0 of `Lasso`.

5 Summary

In this, I demonstrate the effectiveness of `ARD` to handle irrelevant features. I compared `ARD` with `BLR`, `LR`, `Lasso` and `Ridge` on four simulated datasets and one real dataset. The simulated datasets were designed so that dataset `S3` and `S4` should be favourable for `ARD` and `Lasso`, where `S1` and `S2` are more fair for the other models. The experiments resulted in the `ARD` being the superior model on all four datasets, achieving the lowest evaluation MSE. I then visually assessed the convergence and mixing of the Gibbs sampler for `ARD` and `BRL`, which the plots indicated. The predictive distribution of `ARD` and `BRL` were examined by checking the proportion of true observations that are within the 50% and 95% credible intervals, showing the distribution where well-calibrated. A sanity check was conducted for `ARD` on `S1` by plotting 10 functions from the posterior distribution, which showed that functions captured the underlying data pattern. I examined and compared the weights of the different models, which showed that the `ARD` were better able to ignore (setting irrelevant weights to zero/close to zero). This explains why ARD was able to achieve better performance than the other models on the simulated, highlighting the benefit of individual regularization strength for the weights.

The ARD performs well on the real dataset as well, being able to set many irrelevant features close to zero. However, ARD seems to overfit having a much higher MSE score on the validation compared to the training set. The same goes for BRL, LR and Ridge, which has way higher validation loss than ARD. The best model on the real dataset is the Lasso. A possible explanation for Lasso performing better than ARD is improper hyperparameter settings resulting in worse priors with not enough data to make up for it. A hyperparameter search for the models is needed to confirm this explanation. A visual assessment of the Gibbs sampler for ARD is done, which indicates convergence and good mixing. A comparison of ARD and Lasso is made showing ARD allows for higher weights for the feature it deems relevant, than the Lasso. Relevant features for ARD range from -1.1 to 1.6, whereas for Lasso it is from -0.48 to 0.33. Lasso suggests feature index 65 ($w_{65} = 0.33$) and 47 ($w_{48} = -0.48$) to be the most relevant features, whereas for ARD the two most important features are $w_{65} = 1.6$ and $w_{49} = -1.1$.

Overall, this project illustrates the effectiveness of ARD in handling irrelevant features across four simulated and one real dataset.

5.1 Limitation and Possible Improvements

- I could have explored the effect of the hyperparameters on the on ARD and BRL. The hyperparameters impact the priors and with smaller sample sizes priors are more influential than with larger datasets.
- I could have done a hyperparameter search for ARD, BRL, Ridge and Lasso, which would most likely led to better-performing models.
- I could have explored more different datasets e.g. making non-linear functions through basis functions, larger sample sizes and all relevant features. This would provide better insight into when ARD works and does not.
- I could have implement a version of ARD that sets weights that are not statistically different from 0 to zero or use a threshold hyperparameter similar to Sklearn's ARD implementation.
- I could have created an example where Lasso underfit and ARD do not, which better illustrates how constant regularization strength can lead to underfitting, whereas individual weight regularization can mitigate this.

5.2 Q&A:

- How did the ARD model work on real data?
  - It handled the overfitting challenge better than BRL, LR and Ridge. On real dataset Lasso did the best.
- Do your results make sense?
  - Yes, all results seem to make sense. The results show examples where ARD approach is the best (simulated study) and when it is good but beaten by Lasso (real data study).
- What variables affect the survival time most?
  - According to ARD, the two most relevant features are feature 65 and 47. According to Lasso, the two most relevant features are feature 65 and 48.
- How does your Gibbs sampler work?
  - See Section 2.4 and Algorithm 1.
- What are the full conditional distributions of your Gibbs sampler and how did you derive them?
  - This is described and derived in Section 3.1.
- When does ARD work? When doesn't it work?
  - ARD worked well on all the simulated data as it performed the best, illustrating its benefit. Typically, ARD work well on datasets with many irrelevant features and when the prior assumptions are right/good.
  - ARD did well on the real dataset but not as well as Lasso. This might be due to improper hyperparameters. Typically, ARD might struggle if there is insufficient data or if the prior is wrong/bad. The hyperparameters impact the prior which can cause ARD to struggle. Sufficient data can mitigate the influence of poor prior. A limitation, which also goes for BRL, is that learning is computationally intensive compared to LR, Ridge, and Lasso.