Chapter 4

Feature Selection and Model Regularization

The data may contain features that are either redundant or irrelevant, and their removal may have no or only small effect on model accuracy. The reduction of feature space may also help us avoid overfitting. By selecting the most informative features, we may reduce running times and computational complexity and increase the interpretability of results due to the inference of simpler models. Three main approaches to feature selection use filter, wrapper, and embedded methods. In the filter approach, we select the most informative features before modeling. Wrapper methods select features according to the observed performance of inferred models and treat the modeling technique as a black box. With embedded methods, we refer to modeling techniques, which include feature selection within a model inference procedure. In this chapter, we will dive into filter and wrapper approaches, and for embedded methods focus on model regularization.

4.1 Relation to Dimensionality Reduction

Dimensionality reduction is an essential part of quantitative data analysis whose aim is to reduce the dimensions of the data considered in the inference of the model. A positive effect of dimensionality reduction is a decreased model complexity and shortened inference time. Another potential benefit is increased interpretability due to the inference of simpler models. The central premise of dimensionality reduction is that this procedure will have little or no effect on the accuracy of the model.

Dimensionality reduction is effective if the input data includes redundant or irrelevant features, or if we can use new features to replace a subset of original features so that to encapsulates all their information. The three most common families of approaches for di-

mensionality reduction are:

- *Feature transformation* that embeds the data into a lower-dimensional space, replacing original features with a new set that retains as much of information as possible. Approaches of this kind include principal component analysis and deep autoencoders, some of which we will cover in later chapters.

- *Feature selection*, also known as feature subset selection, variable selection, or attribute selection. This approach removes the dimensions (*e.g.* columns) from the input data and results in a reduced data set for model inference.

- *Regularization*, where we are constraining the solution space while doing optimization. Here, we add adding the regularisation terms to which an optimization algorithm must adhere to when minimizing the loss function, apart from having to minimize the error between the true y and the predicted $\hat{y}$. In lasso regularization, for instance, optimization is instructed to find model parameters so that to minimize their absolute sum. This type of regularization may lead to some of the parameters be equal to zero, effectively imposing zero weight to corresponding features, essentially canceling them out from the model. Hence, we can also regard regularization as a feature selection, where the inference method per se includes the feature selection procedure.

4.2 Feature Selection

Feature selection is an optimization problem. The search space is the set of all possible subsets of features, that is, the power set, with 2^n possible solutions. We are trying to find the best solution under some utility and constraints. An example of utility could be the accuracy of the model when inferred from the reduced feature set, and we can express the constraint through a maximal number of features. Viewing feature selection as an optimization problem leads to the following properties of the procedure:

- Because we have a discrete search space, we can, in general, not find the optimal solution, unless we perform a global search and evaluate all 2^n solutions.

- Unless the number of original features n is very small, examining all 2^n solutions is infeasible.

- In practice, the best we can do is to use heuristic search methods with a good inductive bias. This approach tends to work well because we can impose assumptions that are more likely to hold on the data we encounter.

- Any search method that operates on discrete search spaces can also be applied to feature selection.

Filter Methods

Filter methods (**Guyon2003**) perform feature selection before the inference of the model. They rely on a feature scoring function that assigns the score to a feature according to how useful the feature could be in the model. Notice that the scoring is performed before and independently of the model. Features are scored and then ranked, and usually, a top k features are selected, where k is a user-defined parameter of the procedure. Alternatively, feature scores could be compared to their null-distribution that, in practice, could be obtained through feature scoring on a randomly permuted data set. In such cases, k is replaced with user-defined probability p that a particular (or higher) feature score could be obtained and randomly-permuted data.

Scoring functions depend on the type of machine learning problem and type of the scored feature. For instance, for unsupervised learning, we may disregard features with near-constant values by selecting features with the highest deviance, that is, with the highest ratio between variance and the mean. Scoring functions for classification or regression most often consider the correlation between a predictor and the dependent variable. One of the most popular empirical estimates is the mutual information between the i-th predictor and the target y (**Guyon2003**) :

$$I(i) = \int_{x_i} \int_{y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} dx dy,$$

where $p(x_i)$ and $p(y)$ are the probability densities of predictor x_i and dependent variable y, and $p(x_i, y)$ is their joint density. These densities are all unknown and are hard to estimate from the data. The easiest of all is the case of nominal variables, where integral becomes a sum and where probabilities are then estimated from frequency counts:

$$I(i) = \sum_{x_i} \sum_{y} P(X = x_i, Y = y) \log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)}.$$

Notice that mutual information and all similar feature scoring techniques are univariate and assess the information held by the feature in the absence of the context of other features. The scoring function of this type would undervalue features that are in some interactions with other features and only combined with these provide information about the class. A typical example of such a combination is an exclusive disjunction, where participating features may provide no information about the class on their own, yet are information-rich when they are considered together with complementing argument. A field that studies the discovery and ranking of such features is called feature interaction analysis (**Jakulin2005**; **Anastassiou2007**). The approaches cited here rely on an exhaustive search for feature interactions, which are prohibitive in complexity even for reasonably-sized data sets. A bigger problem, though, is that estimates of feature interactions may report about highly interactive features simply by chance and due to an extremely high number of feature combinations

explored.

Interestingly, however, note that there are feature score estimators that take into consideration the contexts and are sensitive to feature interactions. The most prominent of these is Relief, an algorithm originally developed by **Kira1992**. The algorithm assumes that each feature in the data set has been scaled to the interval $[0, 1]$. Let $\boldsymbol{w}$ be a feature weight vector initialized to $\mathbf{0}$. The algorithm randomly draws a data instance $\boldsymbol{x}_i$ and updates the weight vector, such that:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - (x_i - \text{nearHit}(x_i))^2 + (x_i - \text{nearMiss}(x_i))^2$$

where $\text{nearHit}(x_i)$ is the closest same-class data instance to x_i, and $\text{nearMiss}(x_i)$ is the closest different-class data instance to x_i. Notice that the weight of any given feature decreases if it differs from that feature in nearby instances of the same class more than nearby instances of the other class, and increases in the reverse case. In a local neighborhood, the best features should distinguish between instances of the different classes and should be similar across instances of the same class. The score $\boldsymbol{w}$ is updated for m random draws, and the features with the highest scores are those that should be selected.

Kononenko1997 proposed several extensions and improvements of Relief. On the surface, Relief looks like a perfect feature scoring algorithm that can indeed cope with any hidden feature interactions. However, it relies on finding similar data instances and thus suffers from the same problem as any nearest neighbors approach. In other words, Relief would start failing in data sets with a higher number of features, which is precisely where feature interaction discovery would be of the highest value.

Wrapper Methods

Wrapper methods score feature sets according to the estimated accuracy or utility of the algorithm used for learning. The most common search approach is forward/backward stepwise selection (**Guyon2003**). Forward selection is an iterative procedure that starts with an empty set of features and adds one feature at a time, where the benefit of adding a feature is observed in raised accuracy of the inferred model when that feature is added to the feature set. Backward selection starts with a full set of features, and then eliminates one feature at the time, each time selecting feature with the smallest impact on the accuracy of the model. Notice, of course, that backward selection can actually increase the accuracy of the model, as we expect that there is an optimal feature subset with corresponding highest accuracy.

Forward and backward stepwise selection does not necessarily yield the same feature sets. Notice that in the presence of strong feature interactions (*e.g.* consider exclusive disjunction) forward selection would miss, including features that interact, while backward selection may leave interactive features in the selected set, provided that the underlying machine learning algorithm can detect and use the interactions.

There are other, more elaborate discrete space search procedures that could be used in combination with the wrapper approaches. Consider, for instance, local search algorithms, simulated annealing, or genetic algorithms.

Embedded Methods

Embedded methods refer to feature selection techniques that are part of the learning algorithm itself. A typical example of such a method is classification trees, where the inferred model often includes only a subset of most informative features. Perhaps more elaborate techniques in this respect are random forests, where the set of used features may be larger than those from a single tree and where out-of-bag examples can be used for feature scoring and hence ranking.

Below, we will consider a special approach to embedded feature selection that is based on regularization, a constrained optimization that jointly considers the accuracy of the inferred model and the magnitude of model parameters, and with it the complexity of the model.

A Rough Summary on Feature Selection Techniques

Of the three approaches to feature selection stated above, wrapper methods are the most general and may yield the best results, but are computationally intensive and often infeasible even with simple brute force forward or backward search. This is especially the case with data domains, which include tens or hundreds of thousands of features that are common in areas like genomics, text, sound, and image mining. Filter methods typically work one-feature-at-a-time and are faster, but they might provide suboptimal results because of decoupling the selection from actual learning. Embedded methods are the best of both worlds, but they require adaptation of the algorithm that implements them.

We already mentioned other approaches to dimensionality reduction that, instead of feature selection, rely on the inference of a new set of (latent) features. Examples of such feature transformation techniques include matrix factorization, principal component analysis, and deep autoencoders. In comparison with these techniques, please note that:

- feature selection has an advantage over feature transformation as it keeps the original features, which helps with interpretability,
- feature selection is related to explanation/interpretability also through methods that assess variable importance, that is, provides a ranked set of features which can be scrutinized by the domain experts,
- as computational power grows and data sets get larger, filter methods are used less and less and give way to models whose inference relies on optimization and gradient-based search of parameter space.

4.3 Regularization

Let us start with considering linear regression, where $y_i = \beta^\intercal x_i + \epsilon_i$ and where we assume that the error term is distributed normally, so that $\epsilon_i \sim_{\text{iid}} N(0, \sigma^2)$. The data, conveniently, includes a constant column, such that $x_0 = 1$, consequently using β_0 as an intercept, a constant term in linear combination. Linear regression aims to find β^* that minimizes residual sum of squares,

$$\begin{aligned}\text{RSS}(\beta) &= \sum_{i=1}^{n} (y_i - f(x_i))^2 \\ &= \sum_{i=1}^{n} (y_i - \beta^T x_i)^2\end{aligned}$$

so that

$$\beta_{\text{OLS}} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2.$$

Notice that this criteria function actually stems from the maximum likelihood estimation, where parameters β are estimated by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable.

Linear regression also has a closed form solution (**ESL**). Let X denote $n \times (1 + p)$ matrix with n training data instances described with p features. The first column of the matrix is a unit vector. Residual sum of squares can then be expressed as:

$$\text{RSS}(\beta) = (y - X\beta)^\intercal (y - X\beta)$$

Differentiating with respect to β we obtain

$$\begin{aligned}\frac{\partial \text{RSS}}{\partial \beta} &= -2X^\intercal (y - X\beta) \\ \frac{\partial^2 \text{RSS}}{\partial^2 \beta} &= -2X^\intercal X\end{aligned}$$

Setting the first derivative to zero and assuming that X is nonsingular and hence $X^\intercal X$ is positive definite, we obtain

$$\begin{aligned}X^\intercal (y - X\beta_{\text{OLS}}) &= 0 \\ \beta_{\text{OLS}} &= (X^\intercal X)^{-1} X^\intercal y\end{aligned}$$

The main problem that we will address with regularization are cases with few observations in the data set that are described with comparably many features. In other words, cases where the training data matrix X has relatively few rows compared with columns. Here, we are likely to overfit the data, that is, develop a complex model that includes many features

and fits training data well but performs poorly in prediction on new data. If we have more columns than rows, we will have colinearity, so $\boldsymbol{X}^\intercal\boldsymbol{X}$ will not be invertible, and if we want to solve it by minimizing the sum of squares, we will have an infinite number of equivalent solutions.

We will look at regularization from a few different perspectives. The first and the most commonly used one is addressing the overfit by penalizing deviations of model coefficients from zero. For linear regression, if the value of a coefficient is 0, we regard that the corresponding feature is not used in the model. A common approach to suppress the magnitude of model coefficients is to use a quadratic penalty term, leading to the modified optimization problem:

$$\boldsymbol{\beta}_{L2} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (\boldsymbol{y}_i - \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)^2 + \lambda \sum_{i=1}^{k} \boldsymbol{\beta}_i^2 \right),$$

where $\lambda \geq 0$ is the regularization parameter or regularization weight. Notice that λ is an additional parameter of the optimization problem whose value must either be set manually or determined through some optimization procedure that can involve estimation of accuracy of the resulting model by cross-validation or using a validation data set. There are two extreme regularization cases: if $\lambda = 0$, we get non-regularized regression; if $\lambda = \infty$, the regularization penalty is so high that the optimal solution is to select $\boldsymbol{\beta}_i = 0$ for all $i \geq 1$. Note that the intercept, $\boldsymbol{\beta}_0$, is not regularized and, in this case, becomes equal to the mean value of the outcome variable. The optimal value of λ lies somewhere between these two extremes, and penalizes the coefficients just enough to prevent overfitting, but not too much to interfere with the learning, that is, not obfuscating the likelihood term.

The quadratic (L2 norm) penalty is not the only one we can use. We will later discuss the other commonly used penalty, the absolute or L1 norm penalty. And we can here note that another, potentially useful penalty uses the L0 norm (counting), which penalizes for the number of features selected, that is, the number of non-zero $\boldsymbol{\beta}_i$. But first, we will explore how L2 penalty term transforms the initial optimization problem of finding the maximum of the likelihood.

Closed-Form Solution for L2 Regularization

Similar to how we derived the above closed-form solution to the least squares problem we can also derive a closed-form solution to the penalized regression. We want to minimize $\left(\sum_{i=1}^{n}(\boldsymbol{\beta}^T\boldsymbol{x}_i - \boldsymbol{y}_i)^2 + \lambda \sum_{i=1}^{k} \boldsymbol{\beta}_i^2\right)$ or, in matrix shorthand $||\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$. Again, we find the extreme the usual way by differentiating and checking where the gradient is 0:

$$\frac{d}{d\boldsymbol{\beta}} \left(||\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2 \right) = 2(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})^\intercal \boldsymbol{X} + 2\lambda\boldsymbol{\beta} = 2\boldsymbol{\beta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X} - 2\boldsymbol{y}^\intercal \boldsymbol{X} + 2\lambda\boldsymbol{\beta}^\intercal$$

Note that if we differentiate again, we get $2\boldsymbol{X}^\intercal\boldsymbol{X} + 2\lambda\boldsymbol{I}$. This is always positive definite

for $\lambda > 0$, so we have a minimum. This is in contrast with non-penalized regression, where we rely on the additional assumption that $\boldsymbol{X}$ has full rank, making $2\boldsymbol{X}^\intercal\boldsymbol{X}$ positive definite by itself.

Finally, the extreme is where the gradient is zero, so that $2\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\beta}_{L2} - 2\boldsymbol{X}^\intercal\boldsymbol{y} + 2\lambda\boldsymbol{\beta}_{L2} = (2\boldsymbol{X}^\intercal\boldsymbol{X} + 2\lambda\boldsymbol{I})\boldsymbol{\beta}_{L2} - 2\boldsymbol{X}^\intercal\boldsymbol{y} = 0$ or $(\boldsymbol{X}^\intercal\boldsymbol{X} + \lambda\boldsymbol{I})\boldsymbol{\beta}_{L2} = \boldsymbol{X}^\intercal\boldsymbol{y}$, which leads to

$$\boldsymbol{\beta}_{L2} = (\boldsymbol{X}^\intercal\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y}$$

. Note that the term is invertible for the reasons discussed above. In essence, we make $\boldsymbol{X}^\intercal\boldsymbol{X}$ invertible by adding at least a tiny number to its diagonal elements, making the resulting matrix invertible even if it was not invertible by itself. Besides constraining the solution space, L2 regularization solves the problem of non-invertibility that we can encounter when using plain linear regression.

Equivalence of Penalized and Constrained Forms

L2 regularization, as stated above, can be viewed as penalized optimization, where we deal with the objective and a penalty:

$$\text{minimize}_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{\beta}^\intercal\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \right\}.$$

This optimization problem can be formulated as an alternative way that provides additional insight into what regularization does geometrically:

$$\text{minimize}_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{\beta}^\intercal\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 \right\}, \text{subject to } \|\boldsymbol{\beta}\|_2^2 \leq c,$$

where $c \geq 0$ is some constant. Now we show that these two are indeed equivalent. We will use $\boldsymbol{\beta}_P$ to denote the solution to the penalized form and $\boldsymbol{\beta}_C$ the solution of the constrained form. First, we show that for any $\boldsymbol{X}$, $\boldsymbol{y}$, and every c there exists a constant λ that does not depend on $\boldsymbol{X}$ and $\boldsymbol{y}$ such that the solutions of the two problems are the same, that is, $\boldsymbol{\beta}_C = \boldsymbol{\beta}_P$. In other words, we will show that every constraint formulation of the problem has an equivalent penalized formulation.

We already know the solution to the penalized formulation (we derived it above):

$$\boldsymbol{\beta}_P = (\boldsymbol{X}^\intercal\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y}$$

Now we write the Lagrangian of the constrained formulation:

$$L(\boldsymbol{\beta}, \mu) = \|\boldsymbol{\beta}^\intercal\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 + \mu(\|\boldsymbol{\beta}\|_2^2 - c)$$

According to Karush–Kuhn–Tucker (KKT), we have the following conditions to guarantee an optimal solution, which are in this case sufficient, because we have a convex problem and

continuously differentiable constraints:

$$\frac{d}{d\boldsymbol{\beta}}L(\boldsymbol{\beta},\mu) = 0 \tag{4.1}$$

$$\mu \geq 0 \tag{4.2}$$

$$\mu(\|\boldsymbol{\beta}\|_2^2 - c) = 0 \tag{4.3}$$

Observe that, for the first of the above conditions, the left hand side is the same as the gradient of the penalized form, just using μ instead of λ.

Now assume that $\boldsymbol{\beta}_P$ solves the penalized formulation for a given λ. Setting $\mu = \lambda$, $\boldsymbol{\beta} = \boldsymbol{\beta}_P$, and $c = \|\boldsymbol{\beta}_P\|_2^2$ satisfies all three KKT conditions, so there exists for every λ a c such that the solutions to the two problems are the same. Conversely, if $\boldsymbol{\beta}_C, \mu$ solves the constrained formulation for a given c, then $\boldsymbol{\beta}_C$ solves the penalized formulation at $\lambda = \mu$. So, there exists for every c a λ such that the solutions to the two problems are the same. Thus the formulations are equivalent.

In essence, we have shown that penalizing the solution with the quadratic norm is equivalent to putting a hypersphere constraint on the solution. This equivalence of constrained and penalized forms applies in general to p-norms (**ESL**).

L1 regularization

The optimization problem of L1 regularization, also known as Lasso regression, is:

$$\boldsymbol{\beta}_{L1} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (\boldsymbol{\beta}^\intercal \boldsymbol{x}_i - \boldsymbol{y}_i)^2 + \lambda \sum_{i=1}^{k} |\boldsymbol{\beta}_i| \right)$$

or, in vector notation:

$$\boldsymbol{\beta}_{L1} = \arg\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{\beta}^\intercal \boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right).$$

With the result from the previous section on comparison of constrained and penalized formulation, it is not too big a cheat if we immediately say that L1 is equivalent to a 'diamond' constraint. However, the proof is not so obvious. This also allows for the geometric discussion of why Lasso regression tends to set coefficients to zero, while L2 regularization usually infers small but non-zero values of coefficients.

Bayesian Interpretation of Regularization

Regularization is sometimes referred to as *a bet on sparsity*. That is, we are making an apriori assumption that not all (or even not most) of the input variables are relevant predictors. As soon as we introduce prior information, it should not come as a great surprise that regularization has a very elegant Bayesian interpretation.

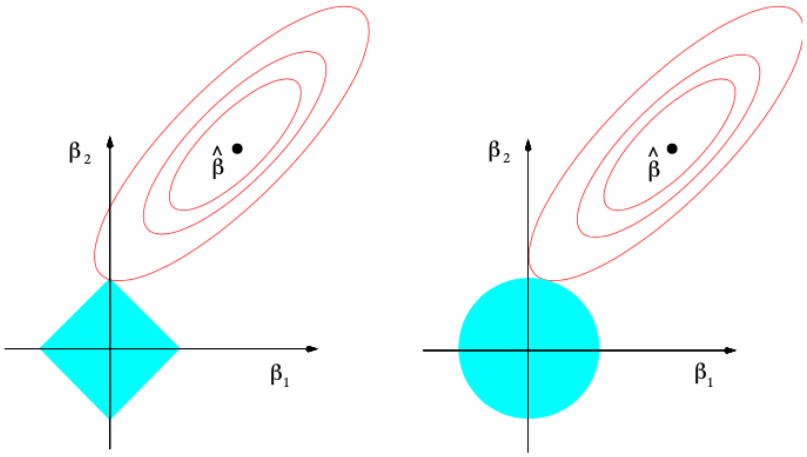

Figure 4.1: Geometric presentation of the optimization problem for the lasso (L1, left) and ridge (L2, right) regression. Shown are conturs of the penalty (least squares error), and the constrain regions $|\beta_1| + |\beta_2| \leq c$ and $\beta_1^2 + \beta_2^2 \leq t^2$. The sharp corners of the constraint region of the lasso yield sparse solutions. In high dimensions, sparsity arises from corners and edges of the lasso's constraint region (from **Tibshirani2014**).

To see this, we go back to the optimization goal of ordinary least squares regression from the beginning of the chapter:

$$\boldsymbol{\beta}_{OLS} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (\boldsymbol{\beta}^\intercal \boldsymbol{x}_i - \boldsymbol{y}_i)^2,$$

and recalling that this minimization is equivalent to maximizing the normal (Gaussian) likelihood assumed by the linear regression model,

$$L(\boldsymbol{\beta}; ...) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(\boldsymbol{\beta}^\intercal \boldsymbol{x}_i - \boldsymbol{y}_i)^2}{2\sigma^2})$$

Indeed, maximizing the log-likelihood, we obtain

$$\ell(\boldsymbol{\beta}; ...) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\boldsymbol{\beta}^\intercal \boldsymbol{x}_i - \boldsymbol{y}_i)^2$$

Note that the maximum w.r.t. the $\boldsymbol{\beta}$ does not depend on σ^2, only on mimimizing the sum of squares.

With what do we have to multiply the likelihood to produce the extra term $-\lambda \sum_{i=1}^{k} \boldsymbol{\beta}_i^2$ in the log-likelihood and therefore get the negative of this term which appears in the minimization problem of L2 regression?

The answer should be obvious: $\prod_{i=1}^{k} \exp(-\frac{\boldsymbol{\beta}_i^2}{1/\lambda})$. In terms of $\boldsymbol{\beta}_i$ this is proportional to the product of normal likelihoods $\boldsymbol{\beta}_i \sim_{iid} N(0, \frac{1}{\sqrt{\lambda}})$. So, if we look at this in terms of the Bayesian

(log)posterior $\log p(\boldsymbol{\beta}|\boldsymbol{y},\boldsymbol{x}) \propto \log p(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{x}) + \log p(\boldsymbol{\beta})$, we see that if we place a normal prior on the coefficients, we get a posterior maximum that corresponds to the L2 regularized MLE solution. In other words, the Bayesian interpretation of L2 regression is that we express a prior opinion that coefficients are normally distributed around 0 with some variance that is a function of $\sigma^2 = \frac{1}{\sqrt{\lambda}}$. Higher variance implies lower λ (less regularization), while lower variance implies higher λ (more regularization).

Similarly we can find the analogue to L1 regularization - the absolute penalty in log-space corresponds to the Laplace distribution (pdf of Laplace is $p(\boldsymbol{x}) = \frac{1}{2b}\exp(-\frac{|\boldsymbol{x}-\mu|}{b})$).

Note that the Bayesian approach to regularization lends itself to an alternative way of inferring λ. Instead of using a fixed λ or choosing the best λ via CV or similar procedure, we can instead treat λ as a parameter, put a hyper-prior on it, and infer it simultaneously with the coefficients.

Also, note that regularization (adding a penalty term to the likelihood) is in statistical circles more often referred to as *penalized likelihood*.

Final Remarks

Regularization is not limited to linear regression. Although it might lead to more complex optimization/sampling problems, the basic principles apply to all models that have coefficients that we can penalize (all linear models, SVM, and other kernel methods).

We typically do not regularize the constant (intercept) coefficient. Either that, or we demean the data and not use an intercept at all when regularizing. Regularizing it does not make sense because it should fit the mean of the data, and that is typically not 0, and we have no reason to have a prior opinion that it is related to the other coefficients.

A particular form of regularization, called elastic net regularization, combines lasso and ridge penalties with an additional weight parameter. This regularization, however, introduces another meta-parameter (this mixing between L1 and L2 penalties) that needs to be inferred from the data.