# Data Science 316 AF Project

# Regularisation: a crucial principle in statistical learning

Andre van der Merwe
24923273

Stellenbosch University

Date of Submission: 13 May 2024

# The importance of regularisation in all our models.

**What is regularisation?**
Regularisation is a technique we use in machine learning and statistical learning to prevent overfitting and to improve a model's ability to perform accurately on unseen or new data, also known as the generalisation of a model. Regularisation also reduces the amount of error or noise (variance) in a model's predictions.

**Regularisation in different models.**
A lot of the models we work with uses a form of regularisation in order to prevent overfitting, improve generalisation and reduce the variance of the model. It is very important to know which type of regularisation method a model uses, in order to interpret the parameters of the model and to prevent overfitting.

**The significance of Regularisation using a penalty term.**
Regularisation using a penalty or shrinkage term is implemented by adding a penalty term to the model's objective function (also known as the loss function or error function). The objective function of a model quantifies the error or difference between the model's predicted output and the true output.
The goal during the training of a model is to minimize this objective function. By adding a penalty term to the objective function, regularisation forces the coefficients (weights) of the model to be small. This then:

- reduces the complexity of the model
- prevents the model from overfitting
- reduces the variance of the model

**The impact of different strengths of the regularisation.**
The strength of the regularisation is controlled by the penalty term, which contains a hyperparameter. Since the penalty contains a hyperparameter, the person building a model, that uses regularisation, can choose the value of the hyperparameter used in the penalty term.
Choosing the appropriate value for the hyperparameter used in the penalty term is crucial, as it directly impacts the trade-off between model complexity and generalisation performance.

- A higher value for the hyperparameter results to:
  - Stronger regularisation.
  - A simpler model.
  - Models are less prone to overfitting.
- A lower value for the hyperparameter results to:
  - Weaker regularisation.
  - A more complex model.
  - Models may capture intricate patterns in the data.
  - Models are at higher risk of overfitting.

# Models and their regularisation methods.

## Regularisation in regression models.

There are three main types of regularisation methods we use in regression models (Logistic regression and regression splines), which are:

- Lasso Regression (L1 Regularisation)
- Ridge Regression (L2 Regularisation)
- Elastic Net Regression

All three of these regularisation methods adds a penalty term to the objective function in order to prevent overfitting and improve the generalisation of the model.

These three methods of regularisation are three of the most commonly used regularisation methods in statistical learning, therefore we will be investigating these methods much more thoroughly.

## Regularisation in smoothing splines and generalised additive models.

When we use regularisation in a **smoothing spline** model, we add a penalty term to the smoothing spline's loss function. In doing so, we ensure that some function g that makes RSS small is also smooth, by finding the function g that minimizes:

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2\, dt$$

where $\lambda$ is a non-negative tuning parameter that controls the shrinkage/regularisation and the function g that minimizes this equation is known as a smoothing spline. The first term ($\Sigma_{i=1}^{n}(y_i - g(x_i))^2$) is the loss function that encourages g to fit the data well and the second term ($\lambda \int g''(t)^2\, dt$) is the penalty term that penalizes the variability in g.

**Generalised additive models** replaces each linear component, in a linear regression model, with a (smooth) non-linear function. Thus, we replace each linear component with a smoothing spline function and regularisation in our generalised additive models is achieved by applying regularisation on each smoothing spline function as shown above.

## Regularisation in trees.

If we fit a tree model to our data without regularisation, it will most likely overfit the data, leading to poor results on unseen data. A smaller tree with fewer splits might lead a higher bias, but it would also lower the variance and perform much better on unseen data. We prevent overfitting by using a technique called **pruning**.
**Pruning** involves removing parts of the tree that leads to overfitting in order to find the subtree that leads to the lowest test error rate.
By using **cost complexity pruning (weakest link pruning)** we will only consider a sequence of trees indexed by a non-negative tuning parameter $\alpha$, instead of considering every possible subtree. The tree that minimises our *training RSS* + $\alpha|T|$, where $|T|$ is the number of terminal nodes, will be the subtree that has the lowest test error rate.

## Regularisation in bagging and random forests.

In our **bagging** models, we build a number of decision trees on bootstrapped training samples. **Random forests** provide an improvement over bagged trees, because a Random forest model decorrelates the trees, since at each split in the tree, the majority of the available predictors may not be considered.
Both of these models will use the same method of regularisation, since Random forests is an improved model of bagging. The two regularisation methods that these models use are **Out-of-Bag error estimation** and **Variable (Feature) importance measures**.

**Out-of-Bag error estimation** is our estimated error when we test our model on the observations that has not been used by the bootstrapping technique (out-of-bag samples) when fitting our model. By monitoring our Out-of-Bag error we can prevent overfitting, as it shows us how well our model performs on unseen data.
**Variable (Feature) importance measures** is a regularisation method where we measure the importance of each feature in the model by permuting the feature when we build our model, and then assessing how much the model's accuracy decreased or increased. This regularisation method identifies the most informative features and by removing these features, we reduce the risk of overfitting the model to noisy or irrelevant features.

## Regularisation in boosting and bayesian additive regression trees.

**Boosting** works similarly to bagging and random forests, except that each tree is grown using information from previously grown trees. The trees are thus grown sequentially and does not involve bootstrap sampling. Regularisation in boosting models are applied by way of slow learning, in order to generalise better to unseen data to prevent overfitting and to stabilise the learning process, making it less sensitive to noise in the training data. We consider three boosting models namely **AdaBoost.M1**, **Functional Gradient Descent (FGD)** and $L_2$**-boosting**.

**AdaBoost.M1** makes use of sequential learning to regularise the learning process. Each subsequent learner focuses on the mistakes made by the previous ones that allows the model to iteratively improve the model's performance by correcting the errors made by earlier weak learners. AdaBoost.M1 also makes use of a slow learning rate in order to regularise the model.
**Functional Gradient Descent (FGD)** is a framework that extends gradient descent methods to function spaces. We add a shrinkage parameter $\lambda$ (a small value) in order to control the rate at which boosting learns. This shrinkage parameter slows the learning process down even further, allowing more and different shaped trees to attack the residuals and minimise the loss function.
$L_2$**-boosting** uses the FGD framework above and adds another tuning parameter that indicates the number of splits in each tree, which controls the complexity of the boosted ensemble.

**Bayesian additive regression trees (BART)** is an ensemble method that constructs each new tree randomly, as in bagging and RF, and each tree tries to capture information not yet accounted for by the current model, as in bagging. Regularisation is obtained by slow learning, limiting the size of the tree and by trying to improve the fit of the current partial residuals by slightly modifying the tree obtained in the previous iteration.

## Regularisation in optimal separating hyperplanes and SVCs and SVMs.

**Optimal separating hyperplanes** are models that aims to find the hyperplane that best separates the data points of different classes. These models are used for binary classification tasks. The smallest distance from each training observation to a separating hyperplane is known as the *margin*. Regularisation is thus obtained by maximising the margin.

**Support vector classifiers (SVC)** is an improved model of the optimal separates hyperplanes models, as it allows some observations to be on the wrong side of the margin and on the wrong side of the hyperplane (misclassified). The SVC model is an improvement, as optimal separating hyperplanes may overfit due to noisy data, and they are typically impossible to fit. Regularisation of the support vector classifiers is obtained by maximising the margin and by adding a regularisation parameter ($C$), which determines the amount of observations that can violate the margin.

- A large $C$ leads to fewer classification errors and a small margin, which leads to overfitting.
- A small $C$ leads to a large margin and allowing more classification errors.

Thus, it is really important to find an optimal value of $C$ which isn't too large or too small. How we find an optimal value of our hyperparameter is discussed in a later section.

**Support vector machines** is a support vector classifier making use of kernel functions in order to obtain a non-linear function in order to fit the model to data with high dimensions. Since our support vector machine model is a support vector classifier in a high dimensions, regularisation of our model is obtained by regularising our support vector classifiers and then applying kernel functions to our model.

## Regularisation in Cox's proportional hazards model.

The **Cox proportional hazards model** is an important multivariable model when the outcome is the length of time to reach a discrete event. We add a penalty term to negative log partial likelihood (objective function) of our model in order to minimise this objective function with respect to $\underline{\boldsymbol{\beta}}$:

$$-ln\left(\prod_{i:\delta_i=1}\frac{exp(\sum_{j=1}^{p}\beta_j x_{ij})}{\sum_{i':y_{i'}\geq y_i} exp(\sum_{j=1}^{p}\beta_j x_{i'j})}\right)+\lambda P(\underline{\boldsymbol{\beta}})$$

Where $\lambda$ is a non-negative hyperparameter and $P(\underline{\boldsymbol{\beta}})$ is either:

- $\sum_{j=1}^{p}\beta_j^2$ (ridge penalised Cox model)
- $\sum_{j=1}^{p}|\beta_j|$ (lasso penalised Cox model)

By adding these penalty terms, we reduce the variance of each coefficient ($\beta_j$), we prevent overfitting and our model generalises better to unseen data. Thus, by applying regularisation to our Cox model, we build a much more reliable model for survival analysis.

## Regularisation in K-Means clustering.

Regularisation in our unsupervised clustering models is also very important as it:

- Helps control the complexity of the models
- Prevents overfitting
- Improves the models generalisation
- Stabilises clustering results.

In a **k-means clustering** model, we form clusters by minimising the with-in cluster variation. We apply regularisation to our k-means clustering model by adding a penalty term to this function in order to avoid excessively large clusters. Our objective function that we minimise would then be:

$$\frac{1}{n}\sum_{k=1}^{K}\left(\sum_{i\epsilon C_k}\left\|\underline{x}_i-\underline{\mu}_k\right\|_2^2\right)+\lambda P(\underline{\boldsymbol{\mu}})$$

where $\lambda$ is a non-negative hyperparameter, n is the number of observations ,$\{C_1,...,C_K\}$ is a collection of K disjoint sets of cluster indices satisfying $\bigcup_{k=1}^{K} C_k = \{1,...,n\}$ and $P(\underline{\boldsymbol{\mu}})$ is the penalty term that depends on the cluster centers $\underline{\boldsymbol{\mu}}$.
Possible forms our penalty term can be is:

- $P(\underline{\boldsymbol{\mu}}) = \sum_{j=1}^{p} abs\left(\left\|\underline{\boldsymbol{\mu}}_{\cdot,j}\right\|\right)$ {Lasso regression}
- $P(\underline{\boldsymbol{\mu}}) = \sum_{j=1}^{p}\left\|\underline{\boldsymbol{\mu}}_{\cdot,j}\right\|^2$ {Ridge regression}

Where $\underline{\boldsymbol{\mu}}_{\cdot,j}$ denotes the $j^{th}$ column of the matrix of cluster centers $\underline{\boldsymbol{\mu}}$.

# Tuning a hyperparameter