Interpretable Machine Learning

Munir Eberhardt Hiabu

March 12, 2024

Acknowledgements

I would like to thank Rasmus Vester Munkner for drafting the first version of these lecture notes based on the initial lecture I gave in 2023. These lecture notes are still in an early stage and possibly have many errors/typos. If you find any of such, please feel free to send them to me: mh@math.ku.dk.

Chapter 1

Introduction

Machine learning models have consistently demonstrated their ability to deliver good predictions across various applications. Yet, their complexity obscures the understanding of the relationships between input variables and outcomes. One objective of this lecture is to delve into the constraints of interpretability. Contrary to common belief, there isn't necessarily an inherent trade-off between accuracy and interpretability. In fact, a learning algorithm can achieve both interpretability and accuracy simultaneously. Before we come to this, we will first start explaining what a learning algorithm is and thereafter go from simple learning algorithms to more complex algorithms. By doing so, we will see that the options to improve performance of an estimator are options that also make an estimator more interpretable. In the second part of the lecture notes we will look at post hoc explanations of black box models.

1.1 Supervised learning

We start by defining the supervised learning problem.

Definition 1.1.1 (Supervised learning)**.** Supervised learning is a field in machine learning that works with labeled data, i.e. data consisting of a set of features X and a response Y. The goal is to learn a function that maps a given input to an output. For the purpose of these notes, data for this problem consists of n i.i.d. realisations $\mathcal{D}_n = (X_i, Y_i)_{i=1,\dots,n}$ called training data with $(X_i, Y_i) \overset{D}{=} (X, Y)$.

It is worth noting that the i.i.d. assumption of the training data can be a strong assumption. In many cases there is a time or space component and the distribution of X could change with that This is also known as distributional shift. Another problem could be that future values of X depend on past values of X (think of a time series or stochastic process) or that X is related to some location. An even more severe problem arises if the relationship between X and Y changes over time and/or space. We will not investigate those cases further in this lecture. In general, data can take many forms.

Example 1. Here are some examples.

- X = handwritten text, Y= words and/or numbers
  - E.g. automatic reading of the postcode from a mail envelope
- X = picture, Y= a chosen set of categories
  - E.g. cancer detection from a CT scan
- X = computer text, Y= a chosen set of categories
  - E.g. spam detection in emails
- X = spread sheet, Y= real valued
  - E.g. insurance pricing
- X = spread sheet, Y= a chosen set of categories
  - E.g. fraud detection

In this course, we will focus on tabular data, i.e. X = spread sheet, with X taking values on a subset of

$\mathbb{R}^p$. The response Y will either be real-valued (regression problem) or categorical (classification problem).

Example 2. Imagine that $X = (X^1, X^2)$ denotes hours studied for an exam and hours slept the night before the exam. The variable Y is binary denoting whether the exam is passed ($Y = 1$) or failed ($Y = 0$). Here, $X^1, X^2 \in \mathbb{R}_+$ and $Y \in \{0, 1\}$ (hence a classification problem). Data from this problem with $n = 4$ data points could be

$$(X_i, Y_i)_{i=1,2,3,4} = (20, 8, 1), (50, 3, 0), (50, 3, 0), (30, 5, 1)$$

One obvious reason to solve a supervised problem (i.e. learn the relationship between X and Y) is to be able to *make predictions*. Consider the problem from Example 2. We could have a fifth data point with a missing Y-value, i.e. we know how much a student studied and how much they slept in the night before the exam but we do not know whether the student has passed the exam. We can use the data of Example 2 to learn the relationship between X and Y in form of a function $\hat{m}(x) : \mathbb{R}^p \mapsto \{0, 1\}$. Afterwards, we can use $\hat{m}$ to make a prediction for the fifth data point with missing Y-value. Prediction does not need to be the (only) reason why someone wants to solve a supervised learning problem. In many cases, the interest is understanding the relationship between X and Y. In this case it will be essential to be able to "understand" what $\hat{m}$ is doing – with the implicit assumption that $\hat{m}$ is a good description of the real relationship between X and Y. In many machine learning problem this is not an easy task, and leads to the field now known as intepretable machine learning. Even if prediction is the goal intepretability can be useful for two reason. The first reason is being able to explain an employed prediction model to stakeholders. Examples here are that a bank should be able to explain why it did or did not grant a credit. An insurance company should be able to explain why it is charging a certain premium to a customer that is different to the premium of an other customer. The second reason is that intepretability can also be important to be able to judge performance of the predictions. To see this, consider the following example

Example 3. Assume the following relationships

$$\begin{aligned} X^1 &= N(0, 1) \\ X^2 &= 2X^1 + N(0, 0.2^2) \\ Y &= 3X^1 + 2X^2 + N(0, 0.5^2) \end{aligned}$$

Given $n = 100$ training samples, we train two models to learn the relationship between $X = (X^1, X^2)$ and Y. The first model is a Linear Model and the second model is XGBoost. Both models will be covered in later sections and their exact definition is not so important here. They each have produced functions $\hat{m}$ that can be used for prediction. In Figure 1.1, we can see the performance of the predictions in different settings measured via empirical mean squared error (MSE). The first row plots actual Y vs predicted Y on the training data. We observe that XGBoost is making better predictions here. Next in the second row, we check the performance on new data (New Data1), which is generated exactly the same way as the training data but has not been used to learn $\hat{m}$. We observe that now the Linear Model performs a bit better. This indicates that the Linear Model generalizes better, i.e., it performs better on new, unseen data. In the third row, the situation is more severe. Here while the relationship between X and Y is the same as in the training data, in the new data (New Data2), the relationship between X_1 and X_2 has changed – they are now uncorrelated:

$$\begin{aligned} X^1 &= N(0, 1) \\ X^2 &= 2N(0, 1 + N(0, 0.2^2) \\ Y &= 3X^1 + 2X^2 + N(0, 0.5^2) \end{aligned}$$

We observe that the performance XGBoost has massively deteriorated, while Linear Model is untouched by the change of the distribution of X. We may conclude: While two predictors can have similar performance on unseen *iid* test data (second row), they can still be very different functions and this difference can have a huge impact on different environments (third row). A way to detect the difference between two predictors (e.g. $\hat{m}_{\texttt{Linear Model}}, \hat{m}_{\texttt{XGBoost}}$) is to be able to visualize/interpret them.

To be able to discuss the performance of different predictions, we make the following definitions.

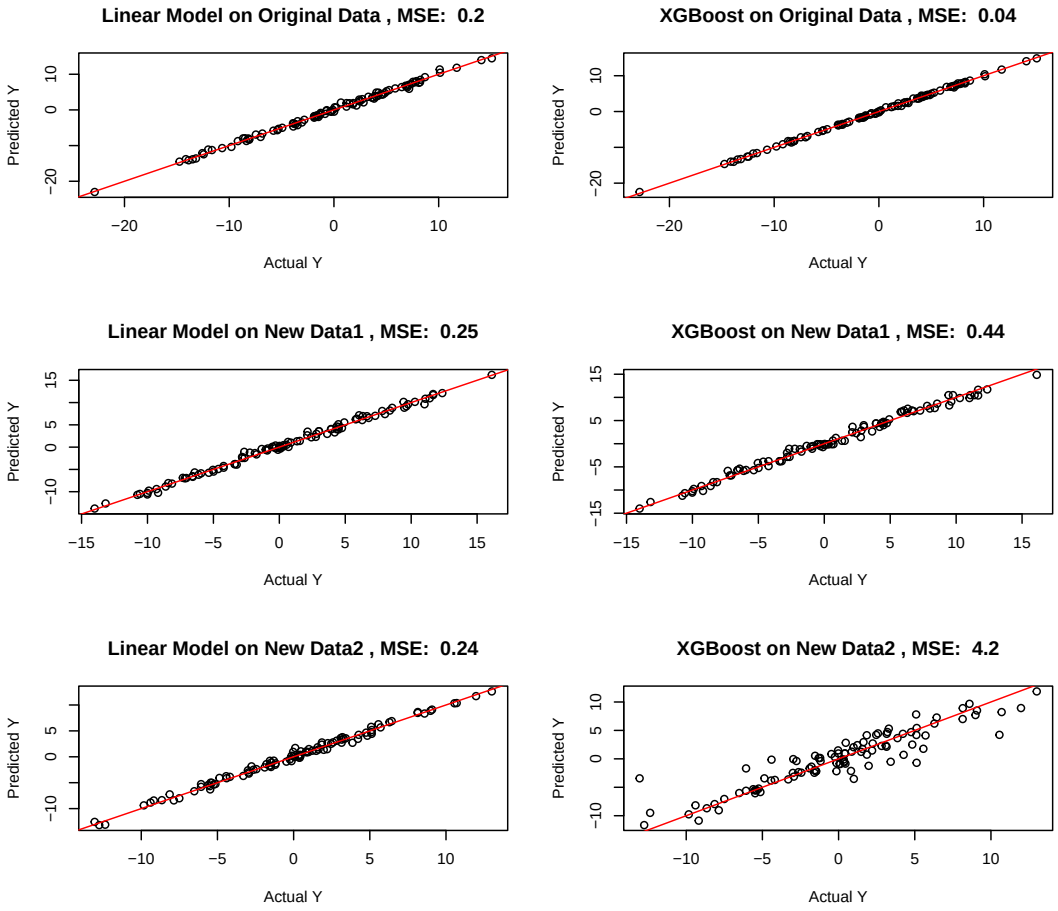

Figure 1.1: Comparison of prediction from a Linear Model and XGBoost. The first row shows performance on the training data. The second row shows performance on unseen test data that has the same distribution as the training data. The third row shows performance on unseen test data where the distributions of the predictors has changed: While having the same marginal distribution as before the predictors are now uncorrelated to each other.

Definition 1.1.2. Denote by $\mathcal{X}$ the support of X and by $\mathcal{Y}$ the support of Y. We say:

i) A decision function is a deterministic function $m : \mathcal{X} \mapsto \mathcal{Y}$.

ii) A function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+ = [0, \infty)$ is a loss function.

iii) The risk of a decision function m with respect to a loss function L is $r(m) = \mathbb{E}[L(Y, m(X))]$.

The idea of the decision function is that $m(X)$ describes our best guess of what Y should be given that we know X. We would like this guess to be good, but to know what that means, we need to be able to distinguish good guesses from bad guesses. This is the purpose of the loss function L. The quantity $L(a, b)$ describes how well b does as a guess for the value a where the better the guess, the smaller the value of $L(a, b)$. In other words, the quantity $L(Y, m(X))$ describes how badly $m(X)$ does as a guess for the value of Y. And thus the risk $r(m) = \mathbb{E}[L(Y, m(X))]$ describes how bad m will be on average at guessing Y from X.

Example 4. Consider a supervised learning problem where X is a number a user types into a calculator and Y is the number the calculator returns to the user when it is asked to multiply the given number by 2. Clearly, $Y = 2X$ if the calculator works correctly, so we would suppose the optimal decision rule for this problem is $m^*(x) = 2x$. However, nothing is stopping us from considering alternative decision rules such as $m_1(x) = x^2$ or $m_2(x) = 0$. To judge the performance of these different decision rules, we may define that we care about the loss computed by the loss function $L(y_1, y_2) = (y_1 - y_2)^2$. This is called the squared loss, but other examples of loss functions are:

- Absolute loss: $L(y_1, y_2) = |y_1 - y_2|$
- Poisson deviance: $L(y_1, y_2) = 2(y_1 \log\left(\frac{y_1}{y_2}\right) - y_1 + y_2)$
- Binary loss: $L(y_1, y_2) = \mathbb{1}(y_1 \neq y_2)$

Remark. While the expected loss may be a nice measure of prediction quality, one could also consider other risk measures over the distribution of $L(Y, m(X))$, e.g. the 95%-quantilel.

Since we are interested in manufacturing good predictions, it is useful to introduce a notion of an optimal decision function.

Definition 1.1.3. Given a function space $\mathcal{G} \subseteq \{m \mid m : \mathcal{X} \mapsto \mathcal{Y}\}$, we say that a decision function $m^* \in \mathcal{G}$ is the Bayes rule (regression) or Bayes classifier (classification) if

$$r(m^*) = \inf\{r(m) \mid m \in \mathcal{G}\}$$

We say that the quantity $\inf\{r(m) \mid m \in \mathcal{G}\}$ is the Bayes risk. Note that the latter, as an infimum over a real set bounded from below, always exists, even if the Bayes rule does not.

Note that even if by definition m^* is the function in $\mathcal{G}$ which is the best at guessing Y from X, it does not mean that $Y = m^*(X)$. Within the supervised learning problem framework, we don't assume that there must exists a deterministic function $m : \mathcal{X} \mapsto \mathcal{Y}$ such that $Y = m(X)$. Usually it is not possible to determine the value of Y precisely from X.

Example 5. Consider $X_1, X_2, Y \in \mathbb{R}$ stochastic variables with $X_1 \perp\!\!\!\perp X_2$ and $Y = X_1 + X_2$. If we consider the supervised learning problem consisting of (X_1, Y). Clearly we are missing the information about X_2 in this problem and thus cannot hope to predict the value of Y perfectly from X_1 alone. On a technical level, if $Y = m(X_1)$ for a measurable function m then $X_2 = Y - X_1 = m(X_1) - X_1$ would be $\sigma(X_1)$-measurable. By standard probabilistic arguments this can only be true if X_2 is almost surely constant. Hence, if X_2 is not constant, then at best $Y = m(X_1) +$ noise

Since we have to live with the fact that we may not be able to predict Y perfectly, it is often not very relevant to consider the risk of a decision function itself, but rather how the risk of the decision function compares to the best possible risk we could attain.

Definition 1.1.4. For some supervised learning problem restricted to the function space $\mathcal{G}$ and some decision function $\tilde{m} \in \mathcal{G}$, we say that

$$r(\tilde{m}) - \inf\{r(m) \mid m \in \mathcal{G}\}$$

is the excess risk of $\tilde{m}$. If there exists a minimizer m^* of the risk for the problem, then

$$r(\tilde{m}) - r(m^*)$$

is the excess risk.

1.2 Algorithms

While for a particular problem and a particular decision function we do not have to care how it was chosen in order to see how well it performs, it seems reasonable to suggest that we are more interested in finding robust ways of choosing optimal decision functions in general rather than evaluating the performance of some cherry picked decision function. Hence we will also care about the performance of the procedure leading us to find the decision function. Whatever this procedure looks like, it will obviously depend on the data we observe, formally:

Definition 1.2.1 (Algorithm)**.** An algorithm is a function $\hat{m} : \mathcal{D}_n \mapsto \{m \mid m : \mathcal{X} \mapsto \mathcal{Y}\}$ which takes the data set from a supervised learning problem as its input and returns a decision function. This decision function is denoted $\hat{m}[\mathcal{D}]$, or short $\hat{m}_n$. The decision function evaluated at some $x \in \mathcal{X}$ is denoted $\hat{m}_n(x)$.

To discuss the performance of an algorithms we make the following definitions.

Definition 1.2.2 (Conditional and unconditional risk)**.** The conditional risk of an algorithm with respect to the loss function L is

$$R(\hat{m}_n) = \mathbb{E}[L(Y, \hat{m}_n(X)) \mid (X_1, Y_1), \dots, (X_n, Y_n)].$$
(conditional risk or conditional generalization error)

The (unconditional) risk of the algorithm is

$$r(\hat{m}_n) = \mathbb{E}[R(\hat{m}_n)] = \mathbb{E}[L(Y, \hat{m}_n(X))].$$
((unconditional) risk or (unconditional) generalization error)

Note that the conditional risk depends on the data while the unconditional risk is deterministic since it additionally averages over all possible data sets. The two measures of risk for an algorithm measure different things. The conditional risk R of the algorithm measures how well the algorithm is expected to perform with the information (the data set $\mathcal{D}_n$) that is available to it. Contrary to this, the unconditional risk r measures how well the algorithm is expected to do before we know what data it has available to it. Since $\mathcal{D}_n$ is assumed stochastic, one could imagine a case where $\hat{m}_n$ is generally good in learning the relationship between X and Y but is performing bad on the data $\mathcal{D}_n$ at hand. One may be interested in the gunconditional risk when discussing theoretical results about the best estimator, while the conditional risk is probably the more interesting object in concrete applications with a given data-set.
Just like the risk of a decision function as given in definition 1.1.2, the risk of an algorithm is usually also not computable, since we don't know the distributions of $\mathcal{D}_n$, X and Y. We define the following empirical quantities.

Definition 1.2.3 (Empirical risk and empirical risk minimizer)**.** Given training data $\mathcal{D}_n$, and a loss function L, we call

$$\hat{R}_n(m) := \sum_i L(Y_i, m(X_i)) \qquad \text{(empirical risk)}$$

empirical risk. Given an additional function class $\mathcal{G}$,

$$\operatorname*{argmin}_{m\in\mathcal{G}} \hat{R}_n(m) = \operatorname*{argmin}_{m\in\mathcal{G}} \sum_i L(Y_i, m(X_i)) \qquad \text{(empirical risk minimzer or standard learner)}$$

is called empirical risk minimzer.

For larger function classes $\mathcal{G}$ the empirical risk minimizer might not be unique and possibly too noisy. One possible way of circumventing the problem is adding a penalty term $J_\lambda : \mathcal{G} \mapsto \mathbb{R}_{\geq 0}$, that penalizes the complexity of m. Here, the parameter λ controls the strength of the penalisation.

Definition 1.2.4 (Penalized empirical risk)**.** Given training data $\mathcal{D}_n$, a loss function L. and a penality term J_λ, we call

$$\hat{R}_{n,\lambda}(m) := \sum_i L(Y_i, m(X_i)) + J_\lambda(m) \qquad \text{(penalized empirical risk)}$$

penalized empirical risk. Given an additional function class $\mathcal{G}$,

$$\operatorname*{argmin}_{m\in\mathcal{G}} \hat{R}_{n,\lambda} := \operatorname*{argmin}_{m\in\mathcal{G}} \sum_i L(Y_i, m(X_i)) + J_\lambda(m) \qquad \text{(penalized empirical risk minimzer)}$$

is called penalized empirical risk minimzer.

Under convexity conditions on J_λ and $\hat{R}_n$ one can show that given data $\mathcal{D}_n$ and a function space $\mathcal{G}$, there exists a constant $\eta(\mathcal{D}_n, \lambda)$ such that for $\mathcal{G}_\eta\{m \in \mathcal{G} | J_\lambda(m) \leq \eta\}$, we have

$$\operatorname*{argmin}_{m\in\mathcal{G}} \hat{R}_{n,\lambda} = \operatorname*{argmin}_{m\in\mathcal{G}_\eta} \hat{R}_n.$$

Example 6 (Penalty terms)**.** Some penalty terms often used are

- $J_\lambda(m) = \lambda \int m''(x)\mathrm{d}x$,
- $J_\lambda(m) = \lambda \int |m(x)|\mathrm{d}x$,
- $J_\lambda(m) = \lambda \int (m(x))^2 \mathrm{d}x$.

In these examples the penalty parameter usually takes some positive value, $\lambda > 0$.

We previously discussed that it may not be possible to get a perfect decision function and therefore we introduced the excess risk as a more relevant object to consider. We can do the same for the risk of an algorithm.

Definition 1.2.5. Assume that the Bayes risk m^* exists. The conditional excess risk of an algorithm is

$$R_e(\hat{m}_n) = R(\hat{m}_n) - r(m^*). \qquad \text{(conditional excess risk)}$$

The (unconditional) excess risk of an algorithm as

$$r_e(\hat{m}_n) = r(\hat{m}_n) - r(m^*). \qquad \text{(unconditional excess risk)}$$

In general, independent of the data $\mathcal{D}_n$, it follows directly from the definition of the Bayes risk that both $R_e, r_e \geq 0$. Assume that the Bayes rule is the minimizer of some function class $\tilde{\mathcal{G}}$, but we have chosen that our estimator $\hat{m}_n$ is the empirical risk minimizer over some smaller function class $\mathcal{G}$. The conditional excess risk can be decomposed into

$$R(\hat{m}_n) - r(m^*) = \underbrace{\left[R(\hat{m}_n) - \inf_{m\in\mathcal{G}} r(m)\right]}_{\text{estimation error}} + \underbrace{\left[\inf_{m\in\mathcal{G}} r(m) - r(m^*)\right]}_{\text{inductive bias (or approximation error)}}.$$

Here, the heuristic is that a large $\mathcal{G}$ usually means small inductive bias but large estimation error and

vice versa. For example if $\mathcal{G}$ is the space of linear functions we can expect a relatively small estimation error but the inductive bias can be arbitrarily large if the Bayes risk m^* is non-linear.
The following result provides an upper bound for the excess risk, even if this is a rather weak upper bound.

Lemma 1.2.6. Let $m^* = \operatorname{argmin}_{m\in\mathcal{G}} r(m)$ be the Bayes rule, $\hat{m}_n = \operatorname{argmin}_{m\in\mathcal{G}} \hat{R}_n(m)$ and $\hat{m}_{n,\lambda} = \operatorname{argmin}_{m\in\mathcal{G}} \hat{R}_{n,\lambda}(m)$. Then,

$$r(\hat{m}_n) - r(m^*) \leq 2 \sup_{m\in\mathcal{G}} |\hat{R}_n(m) - r(m)|,$$
$$r(\hat{m}_{n,\lambda}) - r(m^*) \leq 2 \sup_{m\in\mathcal{G}} |\hat{R}_n(m) - r(m)| + J_\lambda(m^*) - J_\lambda\left(\hat{m}_n\right).$$

Proof. We have that

$$\begin{aligned} r(\hat{m}_n) - r(m^*) &= r(\hat{m}_n) - \hat{R}_n(\hat{m}_n) + \hat{R}_n(\hat{m}_n) - \hat{R}_n(m^*) + \hat{R}_n(m^*) - r(m^*) \\ &\leq (r(\hat{m}_n) - \hat{R}_n(\hat{m}_n)) + 0 + (\hat{R}_n(m^*) - r(m^*)) \\ &\leq 2 \sup_{m\in\mathcal{G}} |\hat{R}_n(m) - r(m)| \end{aligned}$$

Where we used that $\hat{R}_n(\hat{m}_n) - \hat{R}_n(m^*) \leq 0$ due to the assumption that $\hat{m}_n$ minimizes the empirical risk. Similarly

$$\begin{aligned} r(\hat{m}_{n,\lambda}) - r(m^*) &= r(\hat{m}_{n,\lambda}) - \hat{R}_{n,\lambda}(\hat{m}_{n,\lambda}) + \hat{R}_{n,\lambda}(\hat{m}_{n,\lambda}) - \hat{R}_{n,\lambda}(m^*) + \hat{R}_{n,\lambda}(m^*) - r(m^*) \\ &\leq (r(\hat{m}_{n,\lambda}) - \hat{R}_{n,\lambda}(\hat{m}_{n,\lambda})) + 0 + (\hat{R}_{n,\lambda}(m^*) - r(m^*)) \\ &= 2 \sup_{m\in\mathcal{G}} |\hat{R}_n(m) - r(m)| + J_\lambda(m^*) - J_\lambda\left(\hat{m}_n\right) \end{aligned}$$

□

The significance of this result is that if for a specific loss function and function class, one can obtain a uniform bound for the approximation of the true risk by the empirical risk, $|\hat{R}_n(m) - R(m)|$, then this can be used to derive a bound for the excess risk.

Chapter 2

Benchmarking

To summarise on the vocabulary of the previous sections, solving a supervised learning problem corresponds to solving the minimization problem

$$\operatorname*{argmin}_{m \in \mathcal{G}} r(m).$$

It is not at all obvious, how one is to solve a problem of this type, but that is why we propose clever algorithms $\hat{m} : \mathcal{D}_n \mapsto \hat{m}_n$ in the hopes that $\hat{m}_n \approx \operatorname{argmin}_{m \in \mathcal{G}} r(m)$. While having a good strategy of finding a sensible $\hat{m}$ is very relevant, we defer this discussion to later sections.

The more pressing issue is the following. Assume that we have data $\mathcal{D}_n$ and algorithms $\hat{m}^1, \dots, \hat{m}^J$ at our disposal such that each algorithm $j = 1, \dots, J$ proposes a solution $\hat{m}_n^j$ to the supervised learning problem. The question we wish to consider is how we should compare the performance of the algorithms' solutions. This comparison called a benchmark.

2.1 Training and testing

The obvious way to perform the benchmark would be to simply compute $R(\hat{m}_n^j)$ or $j = 1, \dots, J$, since by definition, the best decision function would minimize the risk. The issue we face is that the risk $R(\hat{m}_n^j)$ is not computable since $R(\hat{m}^j) = \mathbb{E}_{X,Y}[L(Y, \hat{m}_n^j(X))]$, and we do not know the distribution of (X, Y) in order to be able to calculate the expectation. If we did know the distribution of (X, Y), we would have no reason to solve the supervised learning problem. As we have already seen in the last section, while we don't know the distribution of (X, Y), we can compute the empirical analog to the theoretical risk by using the data $\mathcal{D}_n$. By the law of large numbers we have

$$\hat{R}_n(m) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, m(X_i)) \overset{a.s.}{\to} \mathbb{E}[L(Y, m(X))] = r(m).$$

Does that mean that if we want to decide between two algorithms $\hat{m}_1, \hat{m}_2$ we can simply pick the one with smaller empirical risk? The answer is unfortunately no. For the law of large numbers to be valid, we need that the sequence $(L(Y, \hat{m}_n^j(X_i)))_{i=1,\dots,n}$ is an i.i.d. sequence. However, the decision functions $\hat{m}_n^j$ depends on the data: $\hat{m}_n^j = \hat{m}^j((X_1, Y_1), \dots, (X_n, Y_n))$, such that the sequence $(L(Y, \hat{m}_n^j((X_1, Y_1), \dots, (X_n, Y_n))(X_i)))_{i=1,\dots,n}$ is in fact composed of dependent entries.

To solve this problem, a standard technique is to use a train-test split. The idea is that if the entire data set available is $D_n = (X_i, Y_i)_{i=1,\dots,n_1+n_2}$, we designate $D_1 = (X_i, Y_i)_{i=1,\dots,n_1}$ as the data which we use to calibrate the estimator $\hat{m}_{n_1}^j$. The remaining data $D_2 = (X_{i+n_1}, Y_{i+n_1})_{i=1,\dots,n_2} = (\tilde{X}_i, \tilde{Y}_i)_{i=1,\dots,n_2}$ is then used to calculate/estimate the empirical loss. In other words, the estimator we use for the risk $R(\hat{m}_{n_1}^j)$ is

$$\frac{1}{n_2} \sum_{i=1}^{n_2} L(\tilde{Y}_i, \hat{m}_{n_1}^j(\tilde{X}_i)) \overset{a.s.}{\to} \mathbb{E}_{X,Y}[L(Y, \hat{m}_{n_1}^j(X))] = R(\hat{m}_{n_1}^j), \quad n_2 \to \infty.$$

Where the convergence follows by the law of large numbers, since $(L(\tilde{Y}_i, \hat{m}^j_{n_1}(\tilde{X}_i)))_{i=1,\dots,n_2}$ is an i.i.d. sequence conditional on the train data $D_1 = ((X_1, Y_1), \dots, (X_{n_1}, Y_{n_1}))$.
Comparison between the different algorithms can thus be based on the estimate of the risk derived from a train-test split. We summarize the procedure in the following box.

Train/Test Benchmarking

START: We are given a data set comprising of observations $\mathcal{D}_n = (X_1, Y_1), \dots, (X_n, Y_n)$ and J algorithms that propose solutions to the supervised learning problem $\hat{m}^j_n, j = 1, \dots, J$.

1. Shuffle the data $\mathcal{D}_n$, resulting in a data set $\mathcal{D}'_n$. Split the data set into two portions $\mathcal{D}'_n = (D_1, D_2)$.
2. For $j = 1, \dots, J$:
   (a) Calculate $\hat{m}^j_{n_1} = \hat{m}(D_1)$.
   (b) Evaluate the empirical risk $\hat{R}_{n_2}(\hat{m}^j_{n_1}) = \frac{1}{n_2} \sum_{i=1}^{n_2} L(\tilde{Y}_i, \hat{m}^j_{n_1}(\tilde{X}_i))$.
3. Rank the different algorithms based on $(\hat{R}_{n_2}(\hat{m}^j_{n_1}))_{j=1,\dots,J}$.

2.2 Cross-validtion

While the idea of a train-test split is useful, it involves some arbitrary splitting of the data set $\mathcal{D}_n$ into the training data set D_1 and the testing data set D_2. When comparing different algorithms, we should obviously use the same train-test split for every algorithm j for a fair comparison (as is being done in step 2). But even then the comparison may not be fair, since some algorithms may do better for that particular split. To make this point more clear we can consider the following example.

Example 7. Consider the following two algorithms

$$\hat{m}^1(\mathcal{D}_n)(x) = 0$$
$$\hat{m}^2(\mathcal{D}_n)(x) = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

The algorithm $\hat{m}^1$ is rather dumb - It always produces a decision function which is 0 everywhere. The algorithm $\hat{m}^2$ is a bit more reasonable. While it also produces a constant decision function, the value of the decision function is the average response. However it also does not make use $X_1, \dots, X_n$. In general we expect $\hat{m}_2$ to be a better algorithm than $\hat{m}_2$.

Let us consider data (let's assume we don't have x-values), $\mathcal{D}_n = (Y_1, Y_2, Y_3, Y_4) = (1, 1, 1, 0)$ and assume that we care about a squared loss, which means that algorithm $\hat{m}^2$ is the theoretically optimal algorithm (which we will see in some later section, by virtue of lemma 3.0.1). If we split the data such that $3/4 = 75\%$ of observations are used for training and $1/4 = 25\%$ of observations are used for testing, we obtain the four possible splits

$$D^4_1 = (Y_1, Y_2, Y_3), \quad D^4_2 = Y_4$$
$$D^3_1 = (Y_1, Y_2, Y_4), \quad D^3_2 = Y_3$$
$$D^2_1 = (Y_1, Y_3, Y_4), \quad D^2_2 = Y_2$$
$$D^1_1 = (Y_2, Y_3, Y_4), \quad D^1_2 = Y_1$$

Computing the error on the test set for each algorithm will yield

$$\hat{R}_{n_2}(\hat{m}^1(D_1^4)) = (Y_4 - 0)^2 = 0, \quad \hat{R}_{n_2}(\hat{m}^2(D_1^4)) = ((Y_4 - 1)^2 = 1$$
$$\hat{R}_{n_2}(\hat{m}^1(D_1^3)) = (Y_3 - 0)^2 = 1, \quad \hat{R}_{n_2}(\hat{m}^2(D_1^3)) = \left(Y_3 - \frac{2}{3}\right)^2 = \frac{1}{9}$$
$$\hat{R}_{n_2}(\hat{m}^1(D_1^2)) = (Y_4 - 0)^2 = 1, \quad \hat{R}_{n_2}(\hat{m}^2(D_1^3)) = \left(Y_2 - \frac{2}{3}\right)^2 = \frac{1}{9}$$
$$\hat{R}_{n_2}(\hat{m}^1(D_1^1)) = (Y_4 - 0)^2 = 1, \quad \hat{R}_{n_2}(\hat{m}^2(D_1^3)) = \left(Y_1 - \frac{2}{3}\right)^2 = \frac{1}{9}$$

The point of this example is that if we through bad luck decided to use $\mathcal{D}' = (D_1^4, D_2^4)$ as our train-test split, it would appear that algorithm $\hat{m}^1$ is better than algorithm $\hat{m}^2$. But this is only the case for that particular split - In general, algorithm $\hat{m}^2$ does better than algorithm $\hat{m}^1$.

The example illustrates that it is possible, especially for small or imbalanced data sets, that the splitting procedure has an effect on which algorithm is perceived to be the best. If we care about the general performance of each algorithm in consideration and want to pick the best one (e.g. for use in production, or for interpretability), a natural solution to get rid of the variance introduced by the arbitrary splitting would be to perform multiple splits, compute the empirical risk on the test set for each split and average the results.

A very slight modification of this procedure is the technique known as K-fold cross validation. This works by splitting the data in K folds (= disjunct parts)), usually of equal size. Going over each fold in turn, it is used as a test set, while the remaining folds are used together as a training set. This yields a value for the empirical risk for each algorithm on each fold, all of which are then aggregated into the measure $CV(\hat{m}^j)$, which corresponds to the average empirical risk over the different testing sets.

K-fold Cross Validation Benchmarking

START: We are given a data set comprising of observations $\mathcal{D}_n = (X_1, Y_1), \dots, (X_n, Y_n)$ and J algorithms that propose solutions to the supervised learning problem $\hat{m}_n^j, j = 1, \dots, J$. Furthermore we have chosen a natural number $K \geq 2$ describing the number of folds.

1. Shuffle the data $\mathcal{D}_n$, resulting in a data set $\mathcal{D}'_n$. Split the shuffled data set into K disjunct portions of equal size $\mathcal{D}'_n = (D_1, D_2, \dots, D_K)$.
2. For $k = 1, \dots, K$, set $D_{-k} = (D_1, \dots, D_{k-1}, D_{k+1}, \dots, D_K)$ and:
   (a) For $j = 1, \dots, J$:
      i. Calculate $\hat{m}^j_{n_{-k}} = \hat{m}^j(D_{-k})$. Here n_{-k} is the sample size of D_{-k}.
      ii. Evaluate the empirical risk $\hat{R}_{n_k}(\hat{m}^j_{n_{-k}}) = \frac{1}{n_k} \sum_{i=1}^{n_k} L(Y_i^k, \hat{m}^j_{n_{-k}}(X_i^k))$. where $D_k = ((X_1^k, Y_1^k), \dots, (X_{n_k}^k, Y_{n_k}^k))$.
3. For $j = 1, \dots, J$, compute the average empirical risk

$$CV(\hat{m}^j) := \frac{1}{K} \sum_{k=1}^{K} \hat{R}_{n_k}(\hat{m}^j_{-k})$$

4. Rank the different algorithms based on $(CV(\hat{m}^j))_{j=1,\dots,J}$.

The advantage of cross-validation compared to a simple train-test split can be illustrated by considering the following example.

Example 8. Continuing example 7, the 4-fold cross validation error would be the average of the four

expressions for the empirical risk for each algorithm, e.g.

$$CV(\hat{m}^1) = \frac{0+1+1+1}{4} = \frac{3}{4}$$
$$CV(\hat{m}^2) = \frac{1+\frac{1}{9}+\frac{1}{9}+\frac{1}{9}}{4} = \frac{1}{3}$$

Thus the cross validation errors reflect more adequately that algorithm $\hat{m}^2$ is better than algorithm $\hat{m}^1$ in a general sense.

It is worth noting that K-fold cross-validation is computationally expensive, since we have to fit the algorithms we are interested in for each fold separately, thus multiplying the workload by a factor of K, compared to a simple train-test split of the same size. Lastly, note that even the K-fold cross-validation score is random, though less noisy than the score from a simple test-train split criterion. While often not practical due to the computational time, noise can be reduced by performing K-fold cross-validation multiple times and averaging the obtained results. If done often enough, the results will not change anymore. But the resulting score is of course still conditional on the fixed data set $\mathcal{D}_n$ and fold-size K. In other words independent of whether K-fold cross-validation is performed once or multiple times the final result will just be an approximation of $R(\hat{m}_n)$. In particular, the the cross-validation estimate of the risk is biased because the sample sizes used for training in the cross-validation procedure (n_{-k}) is smaller than the final training size (n), used by $\hat{m}_n$. The cross-validation estimate of the risk is also noisy because the test sets are not of infinite size. A not so trivial (and partly open) question is whether the cross validation risk is actually estimating conditional risk $R(\hat{m}_n)$ or the unconditional risk $r(\hat{m}_n)$. The difficulty in determining the exact properties of cross validation arises because the K summands in step 3 are correlated for $K \geq 3$.The interested reader is referred to Bates et al. (2023) for a recent contribution towards answering this question.

2.3 Nested Cross Validation

In the previous subsection, we covered the case of comparing the performance of J algorithms $\hat{m}^1, \ldots, \hat{m}^J$. However, the proposed cross-validation procedure will in general only work well for moderately large J. That is because the larger J, the more optimistic is the estimated risk of the best performing algorithm, say $\hat{m}^{j^*}$. The problem here is similar to the problem known as p-hacking in multiple hypothesis testing: If we try out enough algorithms, the best scoring algorithm may have that score not because it is best in general (i.e. for a yet unseen test set sampled from (X, Y)), but because it is by chance performing well for the data sets $D_1, \ldots D_K$ used for evaluation when calculating the cross-validation scores. A more honest estimate of the risk of $\hat{m}^{j^*}$ can be obtained by evaluating it on a separate test set D_{K+1} that is different from $D_1, \ldots D_K$. Since $D_1, \ldots D_K$ cover already the whole data set, a test set D_{K+1} should be kept away before starting the cross-validation procedure. To derive a more stable estimate of the best performing algorithm the whole procedure can be repeated multiple times (i.e. put a test set away, perform cross-validation on the remaining data, pick the best performing algorithm and evaluate its performance on the held back test set). If different test sets are chosen as folds, we have performed what is known as nested cross-validation.

$K - L$-fold Nested Cross Validation

START: We are given a data set comprising of observations $\mathcal{D}_n = (X_1, Y_1), \dots, (X_n, Y_n)$ and J algorithms that propose solutions to the supervised learning problem $\hat{m}_n^j, j = 1, \dots, J$. Furthermore we have chosen a natural number $K \geq 2$ describing the number of outer folds and a natural number $L \geq 2$ describing the number of inner folds.

1. Shuffle the data $\mathcal{D}_n$, resulting in a data set $\mathcal{D}'_n$. Split the shuffled data set into K disjunct portions of equal size $\mathcal{D}'_n = (D_1, D_2, \dots, D_K)$.
2. For $k = 1, \dots, K$, set $D_{-k} = (D_1, \dots, D_{k-1}, D_k, \dots, D_K)$ and:
   (a) For $j = 1, \dots, J$:
      i. Perform L-fold cross-validation on D_{-k} for algorithm $\hat{m}^j$ and calculate $CV(\hat{m}^j)$.
   (b) Let $j_k^* = \text{argmin}_j CV(\hat{m}^j)$
   (c) Evaluate the empirical risk $\hat{R}_{n_k}(\hat{m}_{n_{-k}}^{j_k^*}) = \frac{1}{n_k} \sum_{i=1}^{n_k} L(Y_i^k, \hat{m}_{n_{-k}}^{j_k^*}(X_i^k))$. where $D_k = ((X_1^k, Y_1^k), \dots, (X_{n_k}^k, Y_{n_k}^k))$.
3. An estimate of risk to expect when an algorithm is picked from $\hat{m}_n^j, j = 1, \dots, J$ via L-fold cross-validation is

$$\frac{1}{K} \sum_{k=1}^{K} \hat{R}_{n_k}(\hat{m}_{-k}^{j_k^*})$$

2.4 Hyperparameter tuning

In this section, we briefly treat the issue of benchmarking for hyperparameters. For a more thorough treatment or as supplementary material, the interested reader is referred to Bischl et al. (2023).

One main application of nested cross-validation is when one wants to decide which family of machine learning algorithms to pick. Assume we want to decide whether to employ machine learning algorithm $\hat{m}^1$ or $\hat{m}^2$. Usually, a machine learning algorithm will depend on hyperparameters, sometimes also called tuning parameters. Hence, $\hat{m}^1$ and $\hat{m}^2$ are actually two families of algorithms: $\hat{m}^1 = \hat{m}^{1,\lambda_1}$, $\hat{m}^2 = \hat{m}^{2,\lambda_2}$, $\lambda_1 \in \Lambda_1, \lambda_2 \in \Lambda_2$. If the optimal hyper parameters for $\hat{m}^1$ and $\hat{m}_2$ are each chosen via L-fold cross validation, we can use $K - L$ fold nested cross validation to compare the expected risk between $\hat{m}^1$ and $\hat{m}_2$ and employ the algorithm with the smaller expected risk. But there is a catch. In practice, Λ_1, Λ_2 will often be high dimensional containing a mix of discrete and continuous entries. Hence, we can not simply try out all possible values.
The only solution is to use a method for searching the space of hyperparameters Λ to arrive at an supposedly optimal λ^* without going through every $\lambda \in \Lambda$. Two simple solution are:

- Grid search - Choose some parameter values in a somewhat systematic way, e.g, if $= [0,20] \times \{1,2\}$, once could try $(\lambda_1, \lambda_2) \in \{0, 5, 10, 15, 20\} \times \{1, 2\}$.
- Random search - (e.g. pick 200 parameters uniformly random from the parameter space)

In practice, one will often rely on expert knowledge and experience for selecting the hyperparameter search space. There is also a number of advanced optimization techniques (i.e. techniques that aim to find the minimzer of a function (here: cross validated empirical risk) without the requirement of knowing the analytical form of the function to optimize, see Bischl et al. (2023).

Chapter 3

Linear Models

In this chapter, we will study supervised learning problems with $(X, Y) \in \mathbb{R}^{p+1}$. We assume that we care about a squared loss, that is

$$L(y_1, y_2) = (y_1 - y_2)^2$$

This class of problems is particularly nice behaved.

Lemma 3.0.1. Assume that $Y \in L_2(\Omega, \mathcal{F}, P_{Y|X})$ and let $L(y_1, y_2) = (y_1 - y_2)^2$, the Bayes rule is

$$m^* = \text{argmin}_m r(m) = \Big\{ x \mapsto E[Y \mid X = x] \Big\}$$

Furthermore, we will (unless stated otherwise) make the assumption that $m^* : \mathbb{R}^p \to \mathbb{R}$ satisfies

$$m^*(x) = x^T \beta^*$$

where $\beta^* \in \mathbb{R}^p$. Here, an intercept β_0^* is left out for notational convenience, but it would not change any of the following results. By assuming $E[Y|X = x]$ to be linear we only consider estimation error and ignore inductive bias/approximation error. An alternative would have been to consider $\beta^* = \Sigma^{-1}\mathbb{E}[YX]$, $\Sigma = \mathbb{E}[XX^T]$, i.e., the best linear predictor without assuming that $\mathbb{E}[Y|X]$ is linear. Continuing with our linear assumption, we assume that we are given training data $\mathcal{D}_n = (X_i, Y_i)_{i=1,\dots,n}$. We have

$$Y_i = X_i^T \beta^* + \varepsilon_i,$$

with $\varepsilon_i = Y_i - m^*(X_i)$. An immediate implication of the assumptions made is that

$$\mathbb{E}[\varepsilon_i \mid X_i] = \mathbb{E}[Y_i - m^*(X_i) \mid X_i] = E[Y_i \mid X_i] - m^*(X_i) = 0.$$

We can also link the coefficients characterizing m^* to the moments of (X_i, Y_i) in the following sense.

Lemma 3.0.2. It holds for $j = 1, \dots, p$ that if $Var(X_{1j}) > 0$

$$\beta_j^* = \frac{Cov(X_{ij}, Y - \sum_{k \neq j} X_{ik}\beta_k^*)}{Var(X_{ij})}$$

In particular, if the components of X_i are uncorrelated, we have

$$\beta_j^* = \frac{Cov(X_{ij}, Y_i)}{Var(X_{ij})}.$$

Proof. The proof immediately follows from the representation $Y = X^T\beta^* + \varepsilon$. We have that

$$Cov(X_{ij}, Y_i - \sum_{k \neq j} X_{ik}\beta_k^*) = Cov(X_{ij}, X_{ij}\beta_j^* + \varepsilon_i) = \beta_j^* Var(X_{ij}) + Cov(X_{ij}, \varepsilon_i)$$

Observing that

$$Cov(X_{ij}, \varepsilon_i) = \mathbb{E}[X_{ij}\varepsilon_i] - \mathbb{E}[X_{ij}]\mathbb{E}[\varepsilon_i] = \mathbb{E}[X_{ij}\mathbb{E}[\varepsilon_i \mid X_i]] - \mathbb{E}[X_{ij}]\mathbb{E}[\mathbb{E}[\varepsilon_i \mid X_i]] = 0 - 0 = 0$$

We get the expression for β_j^* by diving through with $Var(X_j)$. □

We now characterize the Bayes risk and the excess risk of a linear estimator m. Let $\varepsilon = Y - X^T\beta^*$.

Lemma 3.0.3. It holds that

$$r(m^*) = Var(\varepsilon) := \sigma^2.$$

and for $m(x) = x^T\beta$ we have

$$r(m) - r(m^*) = ||\Sigma^{1/2}(\beta - \beta^*)||_2^2,$$

where $\Sigma = \mathbb{E}[XX^T]$.

Proof. We have

$$r(m^*) = \mathbb{E}[L(Y, m^*(X))] = \mathbb{E}[(Y - m^*(X))^2] = \mathbb{E}[\varepsilon^2] = Var(\varepsilon).$$

Here, we used $\mathbb{E}[\varepsilon] = 0$.

For the excess risk, we have

$$\begin{aligned}
r(m) - r(m^*) &= \mathbb{E}[L(Y, m(X))] - \mathbb{E}[L(Y, m^*(X))] \\
&= \mathbb{E}[(Y - m(X))^2 - (Y - m^*(X))^2] \\
&= \mathbb{E}[(m^*(X) - m(X))(2Y - m(X) - m^*(X))] \\
&= \mathbb{E}[(X^T(\beta^* - \beta)(2\varepsilon + X^T(\beta^* - \beta))] \\
&= \mathbb{E}[(X^T(\beta^* - \beta)\mathbb{E}[2\varepsilon \mid X]] + \mathbb{E}[X^T(\beta^* - \beta) \cdot X^T(\beta^* - \beta)] \\
&= 0 + E[(\beta^* - \beta)^T XX^T(\beta^* - \beta)] \\
&= (\beta^* - \beta)^T\Sigma(\beta^* - \beta) \\
&= ||\Sigma^{1/2}(\beta - \beta^*)||_2^2
\end{aligned}$$

□

Next, we derive a result for the conditional excess risk.

Lemma 3.0.4. Let $\hat{m}_n : x \mapsto x^T\hat{\beta}$ be some estimator for m^*. Then

$$R(\hat{m}) - r(m^*) = ||\Sigma^{1/2}(\hat{\beta} - \beta^*)||_2^2$$

Proof. Note that m^* is deterministic and $Y \perp\!\!\!\perp \mathcal{D}_n$, hence $E[(Y - m^*(X))^2] = E[(Y - m^*(X))^2 \mid \mathcal{D}_n]$. We have that

$$\begin{aligned}
R(\hat{m}) - r(m^*) &= \mathbb{E}[L(Y, \hat{m}(X)) \mid \mathcal{D}_n)] - E[L(Y, m^*(X))] \\
&= \mathbb{E}[(Y - \hat{m}(X))^2 - (Y - m^*(X)) \mid \mathcal{D}_n] \\
&= \mathbb{E}[(m^*(X) - \hat{m}(X))(2Y - \hat{m}(X) - m^*(X)) \mid \mathcal{D}_n] \\
&= \mathbb{E}[X^T(\beta^* - \hat{\beta}) \cdot (X^T(\beta^* - \hat{\beta}) + 2\varepsilon) \mid \mathcal{D}_n] \\
&= \mathbb{E}[(\beta^* - \hat{\beta})^T XX^T(\beta^* - \hat{\beta}) + 2\varepsilon X^T(\beta^* - \hat{\beta}) \mid \mathcal{D}_n].
\end{aligned}$$

For the second term, note $\mathbb{E}[\varepsilon \mid X] = 0$ and $\varepsilon \perp\!\!\!\perp \mathcal{D}_n$, hence

$$\begin{aligned}\mathbb{E}[2\varepsilon X^T(\beta^* - \hat{\beta}) \mid \mathcal{D}_n] &= \mathbb{E}[\mathbb{E}[2\varepsilon X^T(\beta^* - \hat{\beta}) \mid X, D] \mid \mathcal{D}_n] \\ &= \mathbb{E}[2\mathbb{E}[\varepsilon \mid X, D]X^T(\beta^* - \hat{\beta}) \mid \mathcal{D}_n] \\ &= \mathbb{E}[2\mathbb{E}[\varepsilon \mid X]X^T(\beta^* - \hat{\beta}) \mid \mathcal{D}_n] \\ &= 0.\end{aligned}$$

For the first term, we have that $X \perp\!\!\!\perp \mathcal{D}_n$ and thus $\mathbb{E}[XX^T \mid \mathcal{D}_n] = \mathbb{E}[XX^T] = \Sigma$. Hence

$$\begin{aligned}\mathbb{E}[(\beta^* - \hat{\beta})^T XX^T(\beta^* - \hat{\beta}) \mid \mathcal{D}_n] &= (\beta^* - \hat{\beta})^T E[XX^T \mid \mathcal{D}_n](\beta^* - \hat{\beta}) \\ &= (\beta^* - \hat{\beta})^T \Sigma(\beta^* - \hat{\beta}) \\ &= ||\Sigma^{1/2}(\hat{\beta} - \beta^*)||_2^2\end{aligned}$$

□

3.1 Ordinary Least Squares

Since there is a lot of linear structure in the model, it is helpful to alternatively represent the data in matrix-form as

$$\mathbf{X} = \begin{pmatrix} X_{11} & \cdots & X_{1p} \\ \vdots & \ddots & \vdots \\ X_{n1} & \cdots & X_{np} \end{pmatrix} \quad , \quad \mathbf{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} \quad , \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Note that each row of the data represent a different observation. For notational convenience, we will assume that $X_1 = 1$ and $Y = X\beta^* + \varepsilon$, since this representation is equivalent to a representation with an explicit intercept. Let $m(x) = x^T\beta$, we have

$$\hat{R}_n(m) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - m(X_i))^2 = \frac{1}{n}||\mathbf{Y} - \mathbf{X}\beta||_2^2.$$

The problem of minimizing the sum of squares $||\mathbf{Y} - \mathbf{X}\beta||_2^2$ is well-studied in introductory statistics, and we have the following result

Lemma 3.1.1. Let

$$\hat{\beta}_n^{LS} = \text{argmin}_{\beta \in \mathbb{R}^p}\left(\frac{1}{n}||\mathbf{Y} - \mathbf{X}\beta||_2^2\right).$$

Then the set of normal equations hold

$$(\mathbf{X}^T\mathbf{X})\hat{\beta}_n^{LS} = \mathbf{X}^T\mathbf{Y}.$$

And if $\mathbf{X}^T\mathbf{X}$ is invertible, then there exists a unique solution given by

$$\hat{\beta}_n^{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}.$$

Proof. See proof for theorem 3.2.2 with $\lambda = 0$. □

This result is very useful, because it gives an explicit solution to the minimization problem we wish to solve. However, we don't yet know how well the estimator $\hat{\beta}_n^{LS}$ does at predicting Y from X, so for that we need to know the excess risk. The following theorem tells us that the excess risk (or at least a version thereof) is $O_p(n^{-1})$.

Theorem 3.1.2. If $\mathbf{X}^T\mathbf{X}$ is invertible, then

$$\mathbb{E}[R(\hat{m}_n^{LS}) \mid \mathbf{X}] - r(m^*) = \frac{\sigma^2}{n} \cdot \mathrm{tr}(\Sigma\hat{\Sigma}^{-1})$$

Where $\Sigma = \mathbb{E}[XX^T]$ and $\hat{\Sigma}_n = \frac{1}{n}\mathbf{X}^T\mathbf{X}$.

Proof. Note that

$$\hat{\beta}_n^{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\beta^* + \varepsilon) = \beta^* + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\varepsilon$$

Hence

$$\begin{aligned} R(\hat{m}_n^{LS}) - r(m^*) &\overset{(3.0.4)}{=} ||\Sigma^{1/2}(\hat{\beta}_n^{LS} - \beta^*)||_2^2 \\ &= ||\Sigma^{1/2}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\varepsilon||_2^2 \\ &:= ||A\varepsilon||_2^2 \end{aligned}$$

Using that the trace of a matrix is invariant under cyclic permutations (e.g. $tr(ABC) = tr(BCA)$), we have that

$$||A\varepsilon||_2^2 = tr(||A\varepsilon||_2^2) = tr(\varepsilon^T A^T A\varepsilon) = tr(A\varepsilon\varepsilon^T A^T)$$

Following through, we have that

$$\begin{aligned} \mathbb{E}[R(\hat{m}_n^{LS}) \mid X] - r(m^*) &= \mathbb{E}[||A\varepsilon||_2^2 \mid \mathbf{X}] \\ &= \mathbb{E}[tr(A\varepsilon\varepsilon^T A^T) \mid \mathbf{X}] \\ &= tr(A\mathbb{E}[\varepsilon\varepsilon^T \mid \mathbf{X}]A^T) \\ &= tr(A \cdot \sigma^2 I_{n\times n} \cdot A^T) \\ &= \sigma^2 tr(AA^T) \\ &= \sigma^2 tr\Big(\Sigma^{1/2}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}(\Sigma^{1/2})^T\Big) \\ &= \sigma^2 tr\Big((\Sigma^{1/2})^T\Sigma^{1/2}(\mathbf{X}^T\mathbf{X})^{-1}\Big) \\ &= \sigma^2 tr\Big(\Sigma\frac{1}{n}\left(\frac{1}{n}\mathbf{X}^T\mathbf{X}\right)^{-1}\Big) \\ &= \frac{\sigma^2}{n} tr(\Sigma\hat{\Sigma}_n^{-1}) \end{aligned}$$

Where we have used that $\mathbf{X}^T\mathbf{X}$ and thus also $(\mathbf{X}^T\mathbf{X})^{-1}$ are symmetric. Note that we also used the cyclic invariance property of the trace in the above. □

Remark. If $\hat{\Sigma}_n \approx \Sigma$, then

$$\frac{\sigma^2}{n} tr(\Sigma\hat{\Sigma}_n^{-1}) \approx \frac{\sigma^2}{n} tr(I_{p\times p}) = \frac{\sigma^2 p}{n}$$

This approximation does not take into account the variation of $\mathbf{X}$. Due to the inverse, it is not easily possible to derive an upper bound for the expectation of $\hat{\Sigma}_n$ Therefore, we may only obtain a result for the excess Bayes risk which holds with high probability and under additional assumptions (which could be relaxed but would lead to much more complicated proofs). The takeaway from this is that the excess risk for the least squares method, for fixed dimension p, is approximately $O_p(n^{-1})$.

3.2 Ridge Regression

In the ordinary least squares problem, we need to assume that $\mathbf{X}^T\mathbf{X}$ has full rank. If we have a large set of features ($p \gg n$), this will most likely not be the case, and even if it is, the variance of the estimation

will potentially be too large.

To attempt to resolve this issue, we can use penalization that reduces variance by adding some bias.. Note that this works by introducing an alternative empirical risk function to be minimized.

Definition 3.2.1 (Ridge regression)**.** Let $\lambda \geq 0$ and define the penalty function

$$J_\lambda(\beta) = \lambda ||\beta||_2^2$$

The ridge estimator is defined as

$$\hat{\beta}_{n,\lambda}^{ridge} = \text{argmin}_{\beta \in \mathbb{R}^p} \{\hat{R}_n(\beta) + J_\lambda(\beta)\}$$

And the corresponding decision function is

$$\hat{m}_{n,\lambda}^{ridge}(x) = x^T \hat{\beta}_{n,\lambda}^{ridge}$$

It turns out that the minimization problem defining the estimator $\hat{\beta}_{n,\lambda}^{ridge}$ can be solved explicitly.

Theorem 3.2.2. Let $\lambda > 0$. Then

$$\hat{\beta}_{n,\lambda}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\mathbf{Y}$$

Proof. Note that the version of the empirical risk that is to be minimized is

$$\hat{R}_{n,\lambda}^{ridge}(\beta) = \hat{R}_n(\beta) + J_\lambda(\beta) = \frac{1}{n}||\mathbf{Y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_2^2 = \frac{1}{n}(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + \lambda\beta^T\beta.$$

Differentiating with respect to β yields

$$\begin{aligned} D_\beta \hat{R}_{n,\lambda}^{ridge}(\beta) &= \frac{1}{n}(-\mathbf{X})^T(\mathbf{Y} - \mathbf{X}\beta) + \frac{1}{n}(\mathbf{Y} - \mathbf{X}\beta)^T(-\mathbf{X}) + 2\lambda\beta \\ &= \frac{1}{n}\left(-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\beta\right) + \frac{1}{n}\left(-\mathbf{Y}^T\mathbf{X} + \beta^T\mathbf{X}^T\mathbf{X}\right) + 2\lambda\beta \\ &= 2\frac{1}{n}(-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\beta) + 2\lambda\beta \\ &= \frac{2}{n}\left((\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})\beta - \mathbf{X}^T\mathbf{Y}\right). \end{aligned}$$

Setting the expression equal to the 0-vector and rearranging yields

$$(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})\beta = \mathbf{X}^T\mathbf{Y}.$$

Since $\lambda n I_{p\times p}$ is positive definite for $\lambda > 0$, the matrix $(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})$ is invertible, hence we have that the unique stationary point of $\hat{R}_{n,\lambda}^{ridge}(\beta)$ is $\hat{\beta}_{n,\lambda}^{ridge}$ as given in the theorem statement. Note that we obtain

$$D_\beta^2 \hat{R}_{n,\lambda}^{ridge}(\beta) = \mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p}$$

Which is positive definite and hence any stationary point is a minimum, which proves the theorem. □

The intuition is that the matrix $\mathbf{X}^T\mathbf{X}$ is 'made invertible' by adding some multiple of the identity matrix $\lambda n I_{p\times p}$.

Remark (Lagrangian duality)**.** The name 'ridge regression' stems from the fact that one can prove the minimization problem equivalent to

$$\min_{\beta \in \mathbb{R}^p} \hat{R}_n(\beta) \quad s.t. \quad ||\beta||_2 \leq t$$

for some suitable $t = t(\lambda) > 0$.

Out next result characterizes the excess risk for the ridge estimator.

Theorem 3.2.3. Under the linear model, the excess risk for the ridge estimator is

$$\mathbb{E}[R(\hat{m}_{n,\lambda}^{ridge}) \mid \mathbf{X}] - r(m^*) = \frac{\sigma^2}{n} \cdot tr\left(\Sigma\hat{\Sigma}_\lambda^{-1}\hat{\Sigma}\hat{\Sigma}_\lambda^{-1}\right) + \lambda^2||\Sigma^{1/2}\hat{\Sigma}_\lambda^{-1}\beta^*||_2^2$$

Where $\Sigma = \mathbb{E}[XX^T], \hat{\Sigma} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$ and $\hat{\Sigma}_\lambda^{-1} = (\hat{\Sigma} + \lambda I_{p\times p})^{-1}$.

Proof. Note initially that

$$\begin{aligned}
\hat{\beta}_{n,\lambda}^{ridge} - \beta^* &= (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\mathbf{Y} - \beta^* \\
&= (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T(\mathbf{X}\beta^* + \varepsilon) - \beta^* \\
&= ((\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\mathbf{X} - I_{p\times p})\beta^* + (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\varepsilon.
\end{aligned}$$

To handle the first term, notice that

$$\begin{aligned}
I_{p\times p} &= (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p}) \\
&= (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\mathbf{X} + (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1} n\lambda I_{p\times p}.
\end{aligned}$$

Rearranging yields

$$\begin{aligned}
(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\mathbf{X} - I_{p\times p} &= -(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1} n\lambda I_{p\times p} \\
&= -\lambda n(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}.
\end{aligned}$$

Plugging this into the previous calculations, we obtain

$$\begin{aligned}
&((\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\mathbf{X} - I_{p\times p})\beta^* + (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\varepsilon \\
&\quad = -\lambda n(\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\beta^* + (\mathbf{X}^T\mathbf{X} + \lambda n I_{p\times p})^{-1}\mathbf{X}^T\varepsilon \\
&\quad = -\lambda(\hat{\Sigma} + \lambda I_{p\times p})^{-1}\beta^* + \frac{1}{n}(\hat{\Sigma} + \lambda I_{p\times p})^{-1}\mathbf{X}^T\varepsilon \\
&\quad = -\lambda\hat{\Sigma}_\lambda^{-1}\beta^* + \frac{1}{n}\hat{\Sigma}_\lambda^{-1}\mathbf{X}^T\varepsilon \\
&\quad = \hat{\Sigma}_\lambda^{-1}(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon).
\end{aligned}$$

Now, using lemma 3.0.4, we find that

$$\begin{aligned}
R(\hat{m}_{n,\lambda}^{ridge}) - r(m^*) &= ||\Sigma^{1/2}(\hat{\beta}_{n,\lambda}^{ridge} - \beta^*)||_2^2 \\
&= (\hat{\beta}_{n,\lambda}^{ridge} - \beta^*)^T\Sigma(\hat{\beta}_{n,\lambda}^{ridge} - \beta^*) \\
&= \left(\hat{\Sigma}_\lambda^{-1}(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon)\right)^T \Sigma \left(\hat{\Sigma}_\lambda^{-1}(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon)\right).
\end{aligned}$$

Before rearranging this, note that $\hat{\Sigma}_\lambda^{-1}$ is symmetric as it is the inverse of the sum of two symmetric matrices. Thus

$$\begin{aligned}
R(\hat{m}_{n,\lambda}^{ridge}) - r(m^*) &= \left(\hat{\Sigma}_\lambda^{-1}(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon)\right)^T \Sigma \left(\hat{\Sigma}_\lambda^{-1}(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon)\right) \\
&= \left(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon\right)^T \hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon\right) \\
&= tr\left(\left(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon\right)^T \hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon\right)\right) \\
&= tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon\right)\left(-\lambda\beta^* + \frac{1}{n}\mathbf{X}^T\varepsilon\right)^T\right) \\
&= tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(\lambda^2\beta^*\beta^{*T} - \frac{\lambda}{n}\left(\beta^*\varepsilon^T\mathbf{X} + \mathbf{X}^T\varepsilon\beta^{*T}\right) + \frac{1}{n^2}\mathbf{X}^T\varepsilon\varepsilon^T\mathbf{X}\right)\right).
\end{aligned}$$

Now, taking expectation and using $\varepsilon \perp\!\!\!\perp \mathbf{X}$, we get

$$\begin{aligned}
&E[R(\hat{m}^{ridge}_{n,\lambda}) \mid \mathbf{X}] - r(m^*) \\
&\quad = E[R(\hat{m}^{ridge}_{n,\lambda}) - r(m^*) \mid \mathbf{X}] \\
&\quad = E\left[tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(\lambda^2\beta^*\beta^{*T} - \frac{\lambda}{n}\left(\beta^*\varepsilon^T\mathbf{X} + \mathbf{X}^T\varepsilon\beta^{*T}\right) + \frac{1}{n^2}\mathbf{X}^T\varepsilon\varepsilon^T\mathbf{X}\right)\right) \mid \mathbf{X}\right] \\
&\quad = tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}E\left[\lambda^2\beta^*\beta^{*T} - \frac{\lambda}{n}\left(\beta^*\varepsilon^T\mathbf{X} + \mathbf{X}^T\varepsilon\beta^{*T}\right) + \frac{1}{n^2}\mathbf{X}^T\varepsilon\varepsilon^T\mathbf{X} \mid \mathbf{X}\right]\right) \\
&\quad = tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(\lambda^2\beta^*\beta^{*T} + \frac{1}{n^2}\mathbf{X}^TE\left[\varepsilon\varepsilon^T \mid \mathbf{X}\right]\mathbf{X}\right)\right) \\
&\quad = tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(\lambda^2\beta^*\beta^{*T} + \frac{\sigma^2}{n^2}\mathbf{X}^TI_{n\times n}\mathbf{X}\right)\right) \\
&\quad = tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\left(\lambda^2\beta^*\beta^{*T} + \frac{\sigma^2}{n}\hat{\Sigma}\right)\right) \\
&\quad = \lambda^2 tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\beta^*\beta^{*T}\right) + \frac{\sigma^2}{n}tr\left(\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\hat{\Sigma}\right) \\
&\quad = \lambda^2 tr\left(\beta^{*T}\hat{\Sigma}_\lambda^{-1}\Sigma\hat{\Sigma}_\lambda^{-1}\beta^*\right) + \frac{\sigma^2}{n}tr\left(\Sigma\hat{\Sigma}_\lambda^{-1}\hat{\Sigma}\hat{\Sigma}_\lambda^{-1}\right) \\
&\quad = \lambda^2||\Sigma^{1/2}\hat{\Sigma}_\lambda^{-1}\beta^*||_2^2 + \frac{\sigma^2}{n}tr\left(\Sigma\hat{\Sigma}_\lambda^{-1}\hat{\Sigma}\hat{\Sigma}_\lambda^{-1}\right).
\end{aligned}$$

Which was the desired result. □

While the result of the previous theorem does provide an expression for the excess risk, it is hardly something enlightening. However, there are some special cases where we can get a meaningful interpretation.

Corollary 3.2.4. Assume that $\Sigma = \hat{\Sigma}$ and let $\Sigma = UDU^T$ be the spectral decomposition of Σ with U orthogonal and $D = \text{diag}(s_1, \dots, s_p)$. Then

$$E[R(\hat{m}^{ridge}_{n,\lambda}) \mid \mathbf{X}] - r(m^*) = \frac{\sigma^2}{n}\sum_{j=1}^{p}\left(\frac{s_j}{\lambda + s_j}\right)^2 + \lambda^2\sum_{j=1}^{p}\frac{s_j(\beta^*_j)^2}{(s_j+\lambda)^2}.$$

Proof. We have that

$$\begin{aligned}
\Sigma\hat{\Sigma}_\lambda^{-1} = \Sigma(\Sigma + \lambda I)^{-1} &= UDU^T(UDU^T + \lambda UU^T)^{-1} \\
&= UDU^T(U(D+\lambda I)U^T)^{-1} \\
&\overset{(\dagger)}{=} UDU^TU(D+\lambda I)^{-1}U^T \\
&= UD(D+\lambda I)^{-1}U^T \\
&= UD\left(\underset{j=1,\dots,p}{\text{Diag}}\,(s_j+\lambda)\right)^{-1}U^T \\
&= U\left(\underset{j=1,\dots,p}{\text{Diag}}\left(\frac{s_j}{s_j+\lambda}\right)\right)U^T
\end{aligned}$$

Where (†) relies of the formula $(AB)^{-1} = B^{-1}A^{-1}$ and the fact that $U^{-1} = U^T$. This implies that

$$\begin{aligned}
tr(\Sigma(\Sigma+\lambda I)^{-1}\Sigma(\Sigma+\lambda I)^{-1}) &= tr\left(U\left(\underset{j=1,\dots,p}{\text{Diag}}\left(\frac{s_j}{s_j+\lambda}\right)\right)U^TU\left(\underset{j=1,\dots,p}{\text{Diag}}\left(\frac{s_i}{s_j+\lambda}\right)\right)U^T\right) \\
&= tr\left(\left(\underset{j=1,\dots,p}{\text{Diag}}\left(\frac{s_j}{s_j+\lambda}\right)\right)^2\right) \\
&= \sum_{j=1}^{p}\left(\frac{s_j}{\lambda+s_j}\right)^2
\end{aligned}$$

For the other term we are to simplify, notice that.

$$\begin{aligned}||\Sigma^{1/2}\hat{\Sigma}_\lambda^{-1}\beta^*||_2^2 &= \beta^{*T}(\Sigma+\lambda I)^{-1}\Sigma(\Sigma+\lambda I)^{-1}\beta^* \\ &\overset{(\dagger)}{=} \beta^{*T}U(D+\lambda I)^{-1}U^TUDU^TU(D+\lambda I)^{-1}U^T\beta^* \\ &= \beta^{*T}D(D+\lambda I)^{-2}\beta^* \\ &= \sum_{j=1}^{p}\frac{s_j(\beta_j^*)^2}{(s_j+\lambda)^2}\end{aligned}$$

Combining the results, we get that

$$E[R(\hat{m}_{n,\lambda}^{ridge}) \mid \mathbf{X}] - r(m^*) = \frac{\sigma^2}{n}\sum_{j=1}^{p}\left(\frac{s_j}{\lambda+s_j}\right)^2 + \lambda^2\sum_{j=1}^{p}\frac{s_i(\beta_j^*)^2}{(s_j+\lambda)^2}$$

□

Remark. Even if the result of the corollary is not as simple as we could have hoped, setting $\beta_i = b$ and $s_i = s$ for all $i = 1, \dots, p$ yields

$$E[R(\hat{m}_{n,\lambda}^{ridge}) \mid \mathbf{X}] - r(m^*) = \frac{\sigma^2 p}{n}\frac{s^2}{(\lambda+s)^2} + \lambda^2 p\frac{sb^2}{(s+\lambda)^2}$$

Since we are free to choose the penalization constant λ, we can choose it in such a way that the above is minimized. Setting $\lambda = \frac{\sigma^2}{nb^2}$ we get

$$E[R(\hat{m}_{n,\lambda}^{ridge}) \mid \mathbf{X}] - r(m^*) = \frac{\sigma^2 p}{n}\cdot\frac{b^2 s}{\frac{\sigma^2}{n}+b^2 s} \leq \frac{\sigma^2 p}{n}$$

Which provides is an upper bound for the conditional excess risk. Since this upper bound is potentially smaller than the expression for the conditional excess risk found for the ordinary least squares regression in theorem 3.1.2, this indicates that there are cases where the ridge regression estimator may actually perform better than the ordinary least squares estimator.

3.3 LASSO Regression

This section will give a brief overview on LASSO regression. The theoretical results in this section are taken from Hastie et al. (2015); Wainwright (2019) and the interested reader is referred there for additional results and discussions.

As we saw in the previous section, ridge regression was a potential improvement for ordinary least squares when the effects (β_i's) were all similar. But we may also be in the situation where we have some covariates that are irrelevant to the supervised learning problem, i.e. $\beta_j^* \approx 0$ for some j. In this case, the most obvious choice to penalize β would be of the form $J_\lambda(\beta) = \lambda\|\beta\|_0 = \lambda\#\{j = 1, \dots, p : \beta_j \neq 0\}$. That is, one would simply penalize the number of non-zero entries of β. However, this leads to an NP-hard optimization problems whose solutions are not accessible in practice. One therefore often uses a different norm which has similar properties but leads to a convex optimization problem. LASSO regression is an attempt to improve upon ordinary least squares for this situation. This section will give a brief overview on some theoretical results for LASSO.

Definition 3.3.1. Let $\lambda > 0$ and

$$J_\lambda(\beta) = \lambda||\beta||_1 = \lambda\sum_{i=1}^{p}|\beta_i|.$$

Then the LASSO estimator for β^* is

$$\hat{\beta}_{n,\lambda}^{LASSO} = \text{argmin}_{\beta\in\mathbb{R}^p}\left(\hat{R}_n(\beta) + J_\lambda(\beta)\right).$$

And the corresponding decision function is

$$\hat{m}_{n,\lambda}^{LASSO}(x) = x^T \hat{\beta}_{n,\lambda}^{LASSO}.$$

The abbreviation 'LASSO' stands for '**L**east **A**bsolute **S**hrinkage and **S**election **O**perator'. In general, there exists no closed-form solution for the estimate $\hat{\beta}_{n,\lambda}^{LASSO}$

Remark (Lagrangian duality)**.** Similarly to ridge regression, LASSO regression can also be formulated as a constrained minimization problem

$$\min_{\beta \in \mathbb{R}^p} \hat{R}_n(\beta) \quad s.t. \quad ||\beta||_1 \leq \eta,$$

for some $\eta > 0$. The difference is that the constraint for LASSO is with respect to the 1-norm.

Before stating some results on LASSO regression, we will formalize the concept of 'relevant covariates'.

Definition 3.3.2. Let the relevance set $S : \mathbb{R}^P \to \mathcal{P}(\{1, \dots, p\})$ be defined as $S(\beta) = \{j \in \{1, \dots, p\} \mid \beta_j \neq 0\}$. Furthermore, for any $\nu \in \mathbb{R}^p$ and $A \subseteq \{1, \dots, p\}$, ν_A is the vector ν truncated to 0 at the entries not present in A:

$$\nu_A := (\nu \mathbb{1}_{\{} j \in A\})_{j=1,\dots,p}.$$

If $p \ll n$, then $\hat{\Sigma}_n$ would usually be invertible and the smallest eigenvalue (Rayleigh quotient) would satisfy

$$\lambda_{\min}(\hat{\Sigma}_n) := \inf_{v \in \mathbb{R}^p} \frac{v^T \hat{\Sigma}_n v}{\|v\|_2^2} > 0.$$

Then $\hat{\Sigma}_n$ would be one-to-one and the linear system of equations $\hat{\Sigma}_n \beta = \frac{1}{n} \mathbf{X}^T \mathbf{Y}$ would lead to the (unique) least squares estimator. But for $p \gg n$, one has $\lambda_{\min}(\hat{\Sigma}_n) = 0$. In particular there is no unique minimizer for the not penalized least squares loss. However when employing LASSO, we are usually only interested in estimators $\hat{\beta}$ whith non-zero entries at the components $S(\beta^*)$. This means that in principle we only need injectivity of $\hat{\Sigma}_n$ on the set

$$\tilde{C} = \{\beta \in \mathbb{R}^p : S(\beta) = S(\beta^*)\} = \left\{\beta \in \mathbb{R}^p : \left\|\beta_{S(\beta^*)^c}\right\|_1 = 0\right\},$$

or equivalently,

$$\inf_{v \in \tilde{C}} \frac{v^T \hat{\Sigma}_n v}{\|v\|_2^2} = \inf_{v \in \tilde{C}} \frac{v^T \hat{\Sigma}_n v}{\left\|v_{S(\beta^*)}\right\|_2^2} > 0.$$

Definition 3.3.3 (Restricted eigenvalue property (REP))**.** Define

$$C(\alpha) := \left\{\beta \in \mathbb{R}^p : \left\|\beta_{S(\beta^*)^c}\right\|_1 \leq \alpha \left\|\beta_{S(\beta^*)}\right\|_1\right\}.$$

We say that the restricted eigenvalue property (REP) is satisfied with (γ, α) if for all $\nu \in C(\alpha)$,

$$\frac{v^T \hat{\Sigma}_n v}{\left\|v_{S(\beta^*)}\right\|_2^2} = \frac{n^{-1} \|\mathbf{X} v\|_2^2}{\left\|v_{S(\beta^*)}\right\|_2^2} > \gamma.$$

Using the REP property we will now be able to derive a bound for $n^{-1}\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2$. Note that this corresponds to a type of conditional excess risk where the algorithm is only evaluated on the training data:

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[L(Y, \hat{m}(X_i)) | \mathcal{D}_n] = n^{-1}\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2 + \sigma^2.$$

Theorem 3.3.4. Assume that $\lambda \geq 4\left\|\frac{\mathbf{X}^T\varepsilon}{n}\right\|_\infty$. If β^* is supported on a subset $S(\beta^*)$ of cardinality s, and the design matrix satisfies the REP property with $(\gamma, 3)$, then any solution of the LASSO loss, $\hat{\beta} = \hat{\beta}^{LASSO}_{n,\lambda}$, satisfies

$$\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \frac{9s\lambda^2}{4\gamma}.$$

Proof. Since $\hat{\beta}$ minimizes the LASSO criterion, we have

$$\frac{1}{n}\|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda_n \left\|\hat{\beta}\right\|_1 \leq \frac{1}{n}\|\varepsilon\|_2^2 + \lambda_n \left\|\beta^*\right\|_1.$$

Using this inequality together with

$$\|\mathbf{Y} - \mathbf{X}\beta\|_2^2 = \|\mathbf{X}(\beta^* - \hat{\beta}) + \varepsilon\|_2^2 = \|\mathbf{X}(\beta^* - \hat{\beta})\|_2^2 + \|\varepsilon\|_2^2 + 2\varepsilon^T\mathbf{X}(\beta^* - \hat{\beta}),$$

and re-arranging, we get

$$0 \leq \frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \frac{2\varepsilon^T\mathbf{X}(\beta^* - \hat{\beta})}{n} + \lambda_n(\left\|\beta^*\right\|_1 - \left\|\hat{\beta}\right\|_1).$$

In the next step we will use that $\hat{\beta}$ only has s non-zero components, implying

$$\left\|\beta^*\right\|_1 - \left\|\hat{\beta}\right\|_1 = \left\|\beta^*_S\right\|_1 - \left\|\beta^*_S + (\hat{\beta} - \beta^*)_S\right\|_1 - \left\|(\hat{\beta} - \beta^*)_{S^c}\right\|_1.$$

We conclude

$$\begin{aligned}
\frac{2\varepsilon^T\mathbf{X}(\beta^* - \hat{\beta})}{n} + \lambda(\left\|\beta^*\right\|_1 - \left\|\hat{\beta}\right\|_1) &= \frac{2\varepsilon^T\mathbf{X}(\beta^* - \hat{\beta})}{n} + \lambda(\left\|\beta^*_S\right\|_1 - \left\|\beta^*_S + (\hat{\beta} - \beta^*)_S\right\|_1 - \left\|(\hat{\beta} - \beta^*)_{S^c}\right\|_1) \\
&\leq 2\left\|\frac{\mathbf{X}^T\varepsilon}{n}\right\|_\infty \left\|\beta^* - \hat{\beta}\right\|_1 + \lambda(\left\| (\hat{\beta} - \beta^*)_S\right\|_1 - \left\|(\hat{\beta} - \beta^*)_{S^c}\right\|_1) \\
&\leq \lambda(\frac{3}{2}\left\|(\hat{\beta} - \beta^*)_S\right\|_1 - \frac{1}{2}\left\|(\hat{\beta} - \beta^*)_{S^c}\right\|_1) \\
&\leq \frac{3}{2}\lambda(\left\|(\hat{\beta} - \beta^*)_S\right\|_1 \\
&\leq \frac{3}{2}\lambda\sqrt{s}(\left\|(\hat{\beta} - \beta^*)_S\right\|_2,
\end{aligned}$$

where in the first inequality we applied Hölder's inequality for the first summand and the triangle inequality for the second summand; in the second inequality we used that by assumption $\lambda \geq 4\left\|\frac{\mathbf{X}^T\varepsilon}{n}\right\|_\infty$. Note that knowing that the third line above is greater zero, we have that $\hat{\beta} - \beta^*$ is in the set $C(3)$ as given in the REP definition. Hence, we can use that the REP property with $(\gamma, 3)$ holds, meaning

$$(\left\|(\hat{\beta} - \beta^*)_S\right\|_2 \leq \frac{1}{\sqrt{\gamma}}\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2}{\sqrt{n}}.$$

Putting things together we deduce

$$\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \frac{3}{2}\lambda\sqrt{s}\frac{1}{\sqrt{\gamma}}\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2}{\sqrt{n}}$$

which is equivalent to

$$\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \frac{9}{4}\lambda^2 s\frac{1}{\gamma}$$

□

We collect some remarks about this result.

Remark. In the case of a fixed design, i.e. $\mathbf{X}$ deterministic, and Gaussian noise with $Var(\varepsilon) = \sigma^2$, one can show via the Gaussian tail bound that for any $\tau > 2$,

$$\mathbb{P}\left[\frac{\|\mathbf{X}\varepsilon\|_\infty}{n} \geq \sigma\sqrt{\frac{\tau \log p}{n}}\right] \leq 2e^{-\frac{1}{2}(\tau-2)\log p}.$$

Hence $\lambda = \sqrt{2}\sigma\sqrt{\frac{\tau \log p}{n}}$, is a valid choice with high probability leading to an upper bound of the prediction error

$$\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \sigma^2 s\tau\frac{\log p}{4\gamma n}.$$

The interpretation of this is that the LASSO estimator behaves like the ordinary least squares estimator in a model which had s features instead of p features. Since s was the number of relevant (non-zero β^*) features, the reduction from p to s may represent a large improvement for the excess risk if $p \gg s$.

Note that the factor $\log(p)$ can be though of as 'payment' for this dimensional reduction. But this will usually be a rather small factor if p is very large.

The assumptions of normality and fixed design can be relaxed, which will lead to a similar statement that still ends up with a $\log(p)$-term.

Corollary 3.3.5. Assume that $\lambda \geq 4\left\|\frac{\mathbf{X}^T\varepsilon}{n}\right\|_\infty$. If β^* is supported on a subset S of cardinality s, and the design matrix satisfies the REP property with $(\gamma, 3)$, then any solution of the LASSO loss, $\hat{\beta} = \hat{\beta}^{LASSO}_{n,\lambda}$, satisfies

$$\left\|\hat{\beta} - \beta^*\right\|_2 \leq \frac{3}{2\gamma}\sqrt{s}\lambda$$

Proof. To prove this results, most steps have already been derived in the proof of Theorem 3.3.4, which has the same assumptions. In particular, we have already established that

$$\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \lambda(\frac{3}{2}\left\|(\hat{\beta} - \beta^*)_S\right\|_1 - \frac{1}{2}\left\|(\hat{\beta} - \beta^*)_{S^C}\right\|_1),$$

and by Hölder's inequality this implies

$$\frac{\left\|\mathbf{X}\left(\hat{\beta} - \beta^*\right)\right\|_2^2}{n} \leq \frac{3}{2}\lambda\sqrt{s}(\left\|(\hat{\beta} - \beta^*)\right\|_2.$$

We also know that $\hat{\beta} - \beta^*$ is in the set $C(3)$, hence applying the REP property on the left hand side, we get

$$\gamma(\left\|(\hat{\beta} - \beta^*)_S\right\|_2^2 \leq \frac{3}{2}\lambda\sqrt{s}(\left\|(\hat{\beta} - \beta^*)\right\|_2.$$

□

How realistic is the REP property? Theorem 3.3.4 and Corollary 3.3.5 are based on the REP property. In pracrice it will be difficult to verify this condition since it it based on the location of the non-zero entries of β^*. In the following we will look at one concrete example (without proof) where we can say that the REP property will be satisfied with high probability.

Theorem 3.3.6. Consider a random matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, in which each row $X_i \in \mathbb{R}^p$ is drawn i.i.d. from

a $\mathcal{N}(0, \hat{\Sigma}_n)$ distribution. Then there are universal positive constants $c_1 < 1 < c_2$ such that

$$\frac{\|\mathbf{X}\theta\|_2^2}{n} \geq c_1 \|\sqrt{\hat{\Sigma}_n}\theta\|_2^2 - c_2 \rho^2(\hat{\Sigma}_n) \frac{\log p}{n} \|\theta\|_1^2 \quad \text{for all } \theta \in \mathbb{R}^p$$

with probability at least $1 - \frac{e^{-n/32}}{1-e^{-n/32}}$. In particular, if the minimal eigenvalue satisfies $\lambda_{\min}(\hat{\Sigma}_n) > 0$, then with the same probability, the REP holds with parameter $\gamma = \frac{c_1}{2}\lambda_{\min}(\hat{\Sigma}_n)$ over $C(\alpha)$, uniformly for all subsets S of cardinality at most $|S| \leq \frac{c_1}{2c_2} \frac{\lambda_{\min}(\hat{\Sigma})}{\rho^2(\hat{\Sigma})}(1+\alpha)^{-2} \frac{n}{\log p}$, where $\rho^2(\hat{\Sigma}_n)$ is the maximum variance of Σ.

So far, we have only focused on prediction and estimation error. Another interpretability related question could be: Are the LASSO estimated nonzero entries of the regression parameter in the same positions as the true regression vector, i.e. do we have $S(\hat{\beta}_{n,\lambda}^{LASSO}) = S(\beta^*)$? This property is also known as variable selection consistency or sparsistency. Note that it is possible for an estimation or prediction error to be quite small even if $\hat{\beta}$ and β^* have different non-zero entries. In particular, $\hat{\beta}$ could have very small entries where β^* is zero. To derive a result about the sparsistency of LASSO, we need the following definition.

Definition 3.3.7 (mutual incoherence)**.** The design matrix $\mathbf{X}$ satisfies the mutual incoherence condition with parameter γ if for $S = S(\beta^*)$ if

$$\max_{j \in S^{\mathbb{C}}} \left\| \left(\mathbf{X}_S^T \mathbf{X}_S\right)^{-1} \mathbf{X}_S^T \mathbf{x}_j \right\|_1 \leq 1 - \gamma$$

To give a heuristic for the mutual incoherence condition, note that $(\mathbf{X}_S^T\mathbf{X}_S)^{-1}\mathbf{X}_S^T\mathbf{x}_j$ is the least squares regression parameter when predicting $\mathbf{x}_j$ given data $\mathbf{X}_S$. Hence, if the columns of X_S are orthogonal to all $\mathbf{x}_j$, $j \in S^{\mathbb{C}}$, then all parameters are zero and $\gamma = 1$. However orthogonality is not possible in the high dimensional case, $p >> n$. The mutual incoherence condition provides a bound on the dependency.

Theorem 3.3.8. Suppose that the design matrix $\mathbf{X}$ is normalized (i.e. columns with empirical mean zero and variance one) and that it satisfies the mutual incoherence condition with parameter $\gamma > 0$. Additionally, assume that the REP condition is satisfied with $(K, 0)$. Then there are constants c_1, c_2 such that with probability greater than $1 - c_1 e^{-c_2 n \lambda^2}$, the LASSO estimator $\hat{\beta} = \hat{\beta}_{n,\lambda}^{LASSO}$ has the following properties:

(a) Uniqueness: The optimal solution $\hat{\beta}$ is unique.

(b) No false inclusion: The unique optimal solution has its support contained within the true support

$$S(\hat{\beta}) \subseteq S(\beta^*).$$

(c) ℓ_∞-bounds: The error $\hat{\beta} - \beta^*$ satisfies the ℓ_∞ bound

$$\left\|\hat{\beta}_S - \beta_S^*\right\|_\infty \leq \underbrace{\lambda \left[\frac{4\sigma}{\sqrt{s}} + \left\|\hat{\Sigma}^{-1}\right\|_\infty\right]}_{B(\lambda,\sigma;\mathbf{X})}$$

(d) No false exclusion: The LASSO solution includes all indices $j \in S(\beta^*)$ such that $\left|\beta_j^*\right| > B(\lambda, \sigma; \mathbf{X})$, and hence is sparsistent as long as $\min_{j \in S} \left|\beta_j^*\right| > B(\lambda, \sigma; \mathbf{X})$.

3.4 Elastic Net

Having discussed both ridge regression and LASSO, we can conclude that they aim for different things. Ridge regression aims to reduce variance of the estimator by adding some bias, thus potentially lowering the risk, while LASSO attempts to reduce the dimensionality of the problem by eliminating irrelevant covariates. Note that it may not be clear, what method does best in practice, and hence it could be suggested to mix the methods according to some hyperparameter. This is exactly an elastic net estimator

does.

Definition 3.4.1 (Elastic net)**.** For $\lambda > 0$ and $\alpha \in (0, 1)$ we define the penalty function

$$J_{\lambda,\alpha}^{\text{Elastic.Net}}(\beta) = \lambda(\alpha||\beta||_1 + (1-\alpha)||\beta||_2^2).$$

The corresponding estimator of β^* is

$$\hat{\beta}_{n,\lambda,\alpha}^{\text{Elastic.Net}} = \text{argmin}_{\beta\in\mathbb{R}^p}\left(\hat{R}_n(\beta) + J_{\lambda,\alpha}^{\text{Elastic.Net}}(\beta)\right).$$

When it comes to picking values for α, λ, we refer to the discussion of section 2 for general considerations regarding benchmarking and hyperparameters.

Chapter 4

Nonparametric estimators

In the previous chapter, we discussed the linear model, but the assumption that $Y = X^T\beta^* + \varepsilon$ is rather restrictive, since Y could depend on X in many other ways. In this chapter, we thus consider methods that do not assume a linear relationship between X and Y. We will still operate with the squared loss $L(y_1, y_2) = (y_1 - y_2)^2$ and hence the Bayes rule is $m^*(x) = \mathbb{E}[Y \mid X = x]$.

4.1 Linear Smoothers

Initially, we will consider algorithms based on so-called linear smoothers.

Definition 4.1.1. We say that an estimator $\hat{m}_n$ is a linear smoother if

$$\hat{m}_n(x) = w^T(x)\mathbf{Y}, \qquad w : \mathbb{R}^p \to \mathbb{R}^n$$

with w possibly dependent on $\mathbf{X}$.

The motivation behind a linear smother is that given some new observation X_{New}, the linear smoother will compute Y_{New} as a weighted average of $\mathbf{Y}$. The weight a single Y_i receives should be large if X_i is similar to X_{New} and small if it is very different. This is encoded in $w(X)$ as a weight vector. Our main example of a linear smoother is the k-nearest-neighbors estimator.

Definition 4.1.2. The k-nearest-neighbors estimator is

$$\hat{m}_n^{knn}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} Y_i = \sum_i \frac{1}{k} \mathbb{1}\{i \in \mathcal{N}_k(x)\} Y_i$$

where $\mathcal{N}_k(x) \subseteq \{1, \dots, n\}$ is the set of the k indices of the observations that are 'closest' to x in the sense of some metric on $\mathbb{R}^p$.

Below, we provide a result on a variant of the excess risk for the k-nearest-neighbors estimator.

Theorem 4.1.3. Assume that $p \geq 3$ and $E[Y \mid X = x] = m^*(x) \in \mathcal{G}_L = \{m : \mathbb{R}^p \to \mathbb{R} \mid m \text{ is } L\text{-Lipschitz continuous}\}$ for some $L > 0$. That is

$$\forall x_1, x_2 \in \mathbb{R}^p : |m^*(x_1) - m^*(x_2)| < L||x_1 - x_2||_2.$$

We also assume that $Var(Y \mid X = x) = \sigma^2(x) \leq \sigma^2$ for all $x \in \mathbb{R}^p$ and some $\sigma^2 \geq 0$. Then

$$\mathbb{E}[(\hat{m}_n^{knn}(X), m^*(X))^2] \leq (cL)^2 \left(\frac{k}{n}\right)^{2/p} + \frac{\sigma^2}{k}$$

Proof. We have that

$$\begin{aligned}\mathbb{E}[(\hat{m}_n^{knn}(X) - m^*(X))^2] &= \mathbb{E}[(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] + \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X))^2] \\ &= \mathbb{E}\Big[(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}])^2 + (E[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X))^2 \\ &\quad + 2(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}])(\mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X))\Big]\end{aligned}$$

Note that

$$\begin{aligned}&\mathbb{E}\Big[\Big(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}])(\mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X)\Big)\Big] \\ &\quad = \mathbb{E}\Big[E\Big[\Big(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}]\Big)\Big(\mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X)\Big) \mid X, \mathbf{X}\Big]\Big] \\ &\quad = \mathbb{E}\Big[E\Big[\Big(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}]\Big) \mid X, \mathbf{X}\Big]\Big(\mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X)\Big)\Big] \\ &\quad = \mathbb{E}\Big[0 \cdot \Big(E[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X)\Big)\Big] \\ &\quad = 0\end{aligned}$$

Thus

$$\mathbb{E}[L(\hat{m}_n^{knn}(X), m^*(X))] = \mathbb{E}\Big[(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}])^2 + (\mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X))^2\Big]$$

For the first term

$$\begin{aligned}&\mathbb{E}\Big[(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}])^2\Big] \\ &\quad = \mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n Y_i \mathbb{1}\{i \in \mathcal{N}_k(X)\} - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^n Y_i \mathbb{1}\{i \in \mathcal{N}_k(X)\} \mid X, \mathbf{X}\right]\right)^2\right] \\ &\quad = \mathbb{E}\left[\frac{1}{k^2}\Big(\sum_{i=1}^n \Big(Y_i - \mathbb{E}[Y_i \mid X, \mathbf{X}]\Big)\mathbb{1}\{i \in \mathcal{N}_k(X)\}\Big)^2\right] \\ &\quad = \mathbb{E}\left[\frac{1}{k^2}\Big(\sum_{i=1}^n \sum_{j=1}^n \Big(Y_i - \mathbb{E}[Y_i \mid X, \mathbf{X}]\Big)\Big(Y_j - \mathbb{E}[Y_j \mid X, \mathbf{X}]\Big)\mathbb{1}\{i \in \mathcal{N}_k(X)\}\mathbb{1}\{j \in \mathcal{N}_k(X)\}\Big)\right] \\ &\quad = \frac{1}{k^2}\sum_{i=1}^n \sum_{j=1}^n E\Big[E\Big[\Big(Y_i - \mathbb{E}[Y_i \mid \mathbf{X}]\Big)\Big(Y_j - \mathbb{E}[Y_j \mid \mathbf{X}]\Big) \mid \mathbf{X}\Big]\mathbb{1}\{i \in \mathcal{N}_k(X)\}\mathbb{1}\{j \in \mathcal{N}_k(X)\}\Big]\end{aligned}$$

Note that

$$\mathbb{E}\Big[\Big(Y_i - \mathbb{E}[Y_i \mid \mathbf{X}]\Big)\Big(Y_j - \mathbb{E}[Y_j \mid \mathbf{X}]\Big) \mid \mathbf{X}\Big] = Cov(Y_i, Y_j \mid \mathbf{X}) = \begin{cases} Var[Y_i \mid \mathbf{X}] & i = j \\ 0 & i \neq j \end{cases}$$

Since $(X_1, Y_1), \ldots, (X_n, Y_n)$ are independent and hence also $(Y_i \mid \mathbf{X}) = (Y_i \mid X_i) \perp\!\!\!\perp (Y_j \mid X_j) = (Y_j \mid \mathbf{X})$. Inserting this, we obtain

$$\begin{aligned}&\mathbb{E}\Big[(\hat{m}_n^{knn}(X) - \mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}])^2\Big] \\ &\quad = \frac{1}{k^2}\sum_{i=1}^n E\Big[Var[Y_i \mid \mathbf{X}]\mathbb{1}\{i \in \mathcal{N}_k(X)\}\Big] \\ &\quad \leq \frac{1}{k^2}\sum_{i=1}^n E\Big[\sigma^2 \mathbb{1}\{i \in \mathcal{N}_k(X)\}\Big] \\ &\quad = \frac{\sigma^2}{k}\end{aligned}$$

Since $\mathbb{E}[\sum_{i=1}^n \mathbb{1}\{i \in \mathcal{N}_k(X)\}] = \mathbb{E}[k] = k$.

For the second term, we have that

$$\begin{aligned}
\mathbb{E}\left[\left(\mathbb{E}[\hat{m}_n^{knn}(X) \mid X, \mathbf{X}] - m^*(X)\right)^2\right] &= \mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n \mathbb{E}[Y_i \mid X, \mathbf{X}]\mathbb{1}\{i \in \mathcal{N}_k(X)\} - m^*(X)\right)^2\right] \\
&= \mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n \Big(\mathbb{E}[Y_i \mid X_i] - m^*(X)\Big)\mathbb{1}\{i \in \mathcal{N}_k(X)\}\right)^2\right] \\
&= \mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n \Big(m^*(X_i) - m^*(X)\Big)\mathbb{1}i \in \mathcal{N}_k(X)\right)^2\right] \\
&\leq \mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n L||X_i - X||_2 \mathbb{1}\{i \in \mathcal{N}_k(X)\}\right)^2\right] \\
&= L^2\mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n ||X_i - X||_2 \mathbb{1}\{i \in \mathcal{N}_k(X)\}\right)^2\right]
\end{aligned}$$

It can be proven[1], that this expression satisfies

$$\begin{aligned}
\mathbb{E}\left[\left(\mathbb{E}[\hat{m}_n^{knn}(X) \mid \mathbf{X}] - m^*(X)\right)^2\right] &= L^2\mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^n ||X_i - X||_2 \mathbb{1}\{i \in \mathcal{N}_k(X)\}\right)^2\right] \\
&\leq L^2 \cdot c^2 \left(\frac{k}{n}\right)^{2/p}
\end{aligned}$$

for some constant $c > 0$. Hence combining the estimates, we obtain

$$\mathbb{E}[(\hat{m}_n^{knn}(X) - m^*(X))^2] \leq c^2 L^2 \left(\frac{k}{n}\right)^{2/p} + \frac{\sigma^2}{k}$$

□

Remark. It may seem reasonable to choose $k = n^q$ for some $0 < q < 1$. Ignoring constants, this would mean that the upper bound becomes

$$\left(\frac{n^q}{n}\right)^{2/p} + \frac{1}{n^q} = n^{\frac{2(q-1)}{p}} + n^{-q}$$

To obtain an optimal rate of convergence for $n \to \infty$, we should force each term to have the same exponent. This leads to the following equation

$$\begin{aligned}
\frac{2(q-1)}{p} &= -q \Leftrightarrow \\
2q + pq &= 2 \Leftrightarrow \\
q &= \frac{2}{2+p}
\end{aligned}$$

Hence the optimal choice of k is $k = n^{2/(2+p)}$. In particular, for $k = O_p(n^{2/(2+p)})$, we get

$$\mathbb{E}[(\hat{m}^{knn}(x) - m^*(x))^2] = O_p(n^{-2/(2+p)}).$$

Remark. Note that by construction the decision function m^* is optimal for predicting Y from X. The theorem we have just stated shows that the excess risk for the k-nearest neighbors estimator under some regularity conditions is bounded by $O_p(n^{-2/(2+p)})$. This means that even for a small number of features, the risk decreases much slower than $O_p(n^{-1})$, which was the rate of decay of the risk we found for the parametric (linear model) case. For a growing number of features, the bound deteriorates exponentially.

[1] Chap. 6.3,Györfi et al. (2002)

The k-nearest neighbors estimator has the potential disadvantage that $\hat{m}_n^{knn}(x)$ is not continuous. An alternative to this estimator is the class of linear smoothers based on kernel functions.

Definition 4.1.4. Let $K : [0,\infty) \to [0,\infty)$, $h > 0$ and let $x \mapsto ||x||$ denote some norm on $\mathbb{R}^p$. Define

$$w_i(x) = \frac{K\left(\frac{||X_i - x||}{h}\right)}{\sum_{j=1}^n K\left(\frac{||X_j - x||}{h}\right)}$$

The corresponding estimator

$$\hat{m}_n^{ks}(x) = w^T(x)\mathbf{Y}$$

is called the kernel-smoothing estimator (corresponding to $K, h, ||\cdot||$).

The idea of a kernel-smoothing estimator is that the function $i \mapsto K\left(\frac{||X_i - X_{New}||}{h}\right)$ should represent the measure of similarity between Y_{New} and Y_i based on the distance $||X_i - X_{New}||$. Here $h > 0$ is some hyperparameter which can optionally be tuned. A popular choice, mostly for computational reasons, is $K\left(\frac{||x - X_i||}{h}\right) = \prod_j k(x - X_{ij})$. The function $k : \mathbb{R} \to \mathbb{R}$ is usually a symmetric density function.

Remark. It is possible to show that, under conditions similar to the ones we had for the k-nearest neighbors algorithm, the kernel-smoothing estimator will have excess risk of order $O_p(n^{-\frac{2}{2+p}})$. Hence the kernel-smoothing does not improve performance, but may lead to a more stable estimator due to continuity. One can show that under the assumption that m^* is twice continuously differentiable, the rate for both methods can be improved to $n^{-4/(4+p)}$. But this rate is still exponentially decreasing in p. Furthermore, it has been shown that no method can do better under the given assumptions.

4.2 Curse of Dimensionality

The results we have for the linear smoothing estimators are somewhat discouraging in the sense that the bounds we obtain on the excess risk decrease much too slowly to 0. These results can be thought of a symptoms of what is called the curse of dimensionality - or in other words - $\mathbb{R}^p$ grows fast for $p \to \infty$.

Example 9. To elaborate on the notion of $\mathbb{R}^p$ 'growing fast', consider the following problem. We imagine we are in a supervised learning problem with $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}$. We will assume that the features $X_i \sim Unif(0,1)$ and hence $X \in [0,1]^p$. Furthermore we assume that these are independent.

Say we would like to be able to estimate m^* for some region, say $A = [0.5, 0.6]^p \subseteq [0,1]^p$. How many observations can we expect to have available for this? The probability for each observation $X_1, X_2, ... \in [0,1]^p$ to fall into the region A is

$$P(X_i \in A) = \int_A dP^X = \mu_{\text{Lebesgue}}(A) = \prod_{i=1}^p (0.6 - 0.5) = \frac{1}{10^p}$$

Hence for n independent observations, we could expect $\frac{n}{10^p}$ observations to fall into A. Note that even for moderately large p, this number is very small, meaning that for most regions like A, we would have very few or no observations within the region to estimate the behaviour of m^* there.

The problem that the example highlights is that if we want to predict the Y_{New}-part of some (X_{New}, Y_{New}) based on X_{New}, then we need to have seen previous observations that are similar to X_{New}. And if we only consider similarity based on some measure of distance, we may never have seen any observations that are similar to X_{New} if p is large. To resolve this issue, there are two approaches one could take.

The first one is to assume sparsity - That is, reduce the dimension of the problem. We may have observations $X_{New} = (X_{New,1}, \dots, X_{New,100})$, but if we only believe $X_{New,1}$ is relevant for the value of Y_{New}, then we can discard the additional information from $X_{New,2}, \dots, X_{New,100}$. If we think of our data as $X_1, X_2, X_3, \dots, X_n$, none of these may be similar to X_{New} in $\mathbb{R}^{100}$, but after discarding the 99 irrelevant

dimensions, we may have that some of the observed data is similar to X_{New} and hence we can use the data to predict Y_{New}.

As a further comment on sparsity - It is most easily though of in the case where some features are irrelevant, but it could also be that only certain combinations of features are meaningful.

Example 10. Let $X_1 = \{\text{Minutes in the bath}\}, X_2 = \{\text{Minutes spent on breakfast}\}, X_3 = \{\text{Minutes spent traveling}\}$ and $Y = \{\text{Minutes from waking up to attending work}\}$. Let's assume

$$Y = X_1 + X_2 + X_3 + \varepsilon.$$

Hence if we want to predict Y from X, we cannot discard any of $(X_1, X_2, X_3) \in \mathbb{R}^3$, but rather we can combine these into a feature $(X_1 + X_2 + X_3) \in \mathbb{R}^1$, which contains all the relevant information for predicting Y.

The second solution to the curse of dimensionality is to assume a certain structure for m^*. One example of such a structure could be the linear model. Assuming structure, we limit the potential for modelling interaction between the different features, but if we believe that there should be no such interaction, this may be very reasonable.

An interesting observation is that the options we have to improve performance of an estimator are options that also make the estimator more interpretable.

4.3 Additive Models

In this chapter, we will consider a class of models that is more flexible than the linear model, while still retaining some of the structure that makes the linear model feasible.

Definition 4.3.1 (additive model)**.** Consider the supervised learning problem with

$$Y = m_0 + \sum_{j=1}^{p} m_j^*(X_j) + \varepsilon$$

Where $m_0 \in \mathbb{R}$ and the functions $m_j^* : \mathbb{R} \to \mathbb{R}$ are measurable. We call this type of supervised learning problem an additive model.

Note that the linear model is included in an additive model with $m_j^*(x_j) = x_j\beta_j^*$. The goal in the linear model was to infer the β^*'s, since these determine the relation between X and Y. In an additive model, the m_j^*'s (functions) are the objects we care to determine.

In Stone (1985) it has been shown that the components m_k, if twice continuously differentiable, can be estimated with one-dimensional non-parametric rate of $n^{-4/4+1}$ The components m_j are usually estimated via the so called backfitting algorithm Hastie (2017). Before we come to that, we first look at the one dimensional problem, which can be used as an ingredient in the backfitting algorithm.

4.3.1 Splines

Generally, the additive model with no restrictions on m_j^* is a to broad class to work with, hence we will instead focus our attention on the smaller class of supervised learning problems with $m_j^* \in \mathcal{D}([a,b]) = \{m : [a,b] \to \mathbb{R} \mid m'' \text{ exists}\}$ (hence all the functions that are twice differentiable within some bounded support).

There exists no finite basis $(g_1, \ldots, g_\ell)$ for the space $\mathcal{D}([a,b])$, hence we cannot simply reduce the problem of estimating m_j^* into estimating the coefficients that would represent m_j^* with respect to this basis, e.g.

in the form

$$m_j^*(x) = \sum_{k=1}^{\ell} \beta_k^* \cdot g_k(x)$$

However, it will turn out that for any finite data set, the (suitably penalized) empirical risk will be minimized within the class $\mathcal{D}([a,b])$ by a so-called natural, cubic spline (to be defined below). The advantage of these splines is that they can be represented with a finite functional basis, hence the problem of estimating the best such spline for the given data essentially becomes a problem in the linear model, which we have good methods for solving.

A spline is the function resulting from gluing a bunch of polynomials together in a way such that the polynomials agree about what value the spline should take at each knot (gluing point) and also what the derivatives should be there. More formally:

Definition 4.3.2 ((natural) splines)**.** Let $x_1 < \cdots < x_\ell$ and let $p_0, \dots, p_\ell$ be a collection of polynomials of degree less than or equal to k that satisfy

$$\forall h \in \{0, \dots, k-1\} \forall j \in \{1, \dots, \ell\} : \frac{d^h}{dx^h} p_{i-1}(x_j) = \frac{d^h}{dx^h} p_j(x_j)$$

Using the convention $x_0 = -\infty$ and $x_{\ell+1} = \infty$, we say that

$$g : \mathbb{R} \to \mathbb{R} \quad , \quad g(x) = \sum_{j=0}^{\ell} \mathbb{1}\{x_j \leq x < x_{j+1}\} \cdot p_j(x)$$

is a spline of degree k with knots in $x_1, \dots, x_\ell$.

We say that g is a natural spline if p_0 and p_ℓ have degree less than or equal to $\lfloor \frac{n}{2} \rfloor$.

Definition 4.3.3 (cubic spline)**.** A cubic spline is a spline of degree $n = 3$. Thus a natural, cubic spline is linear on $(-\infty, x_1]$ and on $[x_\ell, \infty)$.

It turns out that any spline of degree k with knots $x_1, \dots, x_\ell$ can be represented as a linear combination of $k + \ell + 1$ basis functions. We show this for a particular (simple) choice of basis, but note that the basis we give is not unique and neither the one that would typically be used in practice.

Lemma 4.3.4 (Truncated power basis)**.** Let m be a spline of degree k with knots $x_1, \dots, x_\ell$. Then there exists a unique vector $\theta \in \mathbb{R}^{k+\ell+1}$ such that

$$m(x) = \theta^T g(x)$$

Where $g(x) = (g_0(x), \dots, g_{k+\ell}(x))$ with

$$g_j(x) = \begin{cases} x^j & 0 \leq j \leq k \\ ((x - x_{j-k})^+)^k & k+1 \leq j \leq k+\ell \end{cases}$$

Proof. Let $m(x) = \sum_{j=1}^{\ell} \mathbb{1}\{x_{j-1} \leq x < x_j\} p_j(x)$ with $p_j(x) = \sum_{h=0}^{k} \alpha_h^{(j)} x^h$. We wish to show that there exists a unique $\theta \in \mathbb{R}^{k+\ell+1}$ such that $\hat{m}(x) = \theta^T g(x) = m(x)$ for all $x \in \mathbb{R}$.

Start by noting that we can alternatively write

$$m(x) = \sum_{j=0}^{\ell} \mathbb{1}\{x_j \leq x < x_{j+1}\} p_j(x) = p_0(x) + \sum_{j=1}^{\ell} \mathbb{1}\{x_j \leq x\} q_j(x)$$
$$q_j(x) = p_j(x) - p_{j-1}(x)$$

Note that since m is a spline of degree k, we have $\frac{d^h}{dx^h}p_j(x_j) = \frac{d^h}{dx^h}p_{j-1}(x_j)$. Which means that

$$\frac{d^h}{dx^h}q_j(x_j) = 0 \quad , \quad h = 0, \dots, k-1$$

Noting that q_j is a polynomial of degree k, it is equal to its Taylor expansion of order k around any point. We thus have

$$q_j(x) - \underbrace{q_j(x_j)}_{=0} = \sum_{h=0}^{k} \frac{\frac{d^h}{dx^h}q_j(x_j)}{h!}(x-x_j)^h = \frac{k!(\alpha_k^{(j)} - \alpha_k^{(j-1)})}{k!}(x-x_j)^k = (\alpha_k^{(j)} - \alpha_k^{(j-1)})(x-x_j)^k$$

Which shows that

$$\begin{aligned} m(x) &= p_0(x) + \sum_{j=1}^{\ell} \mathbb{1}\{x_j \leq x\}(\alpha_k^{(j)} - \alpha_k^{(j-1)})(x-x_j)^k = p_0(x) + \sum_{j=1}^{\ell}(\alpha_k^{(j)} - \alpha_k^{(j-1)})((x-x_j)^+)^k \\ &= \sum_{h=0}^{k} \alpha_h^{(0)} x^h + \sum_{j=1}^{\ell}(\alpha_k^{(j)} - \alpha_k^{(j-1)})((x-x_j)^+)^k \end{aligned}$$

To have $\hat{m}(x) = m(x)$, we now simply read off that $\theta_0 = \alpha_0^{(0)}, \dots, \theta_k = \alpha_k^{(0)}$ and $\theta_{k+j} = (\alpha_k^{(j)} - \alpha_k^{(j-1)})$ for $j = 1, \dots, \ell$. Hence we get both existence and uniqueness from this representation. □

Remark. Typically, we would use a basis called B-spline basis for actual computations, since it is faster to compute and more stable. It is also possible to represent the class of natural splines with both a truncated power basis or a B-spline basis, but the dimension of the basis will be $\ell + 1$ rather than $k + \ell + 1$ in that case.

Now, we provide the promised result - That a natural, cubic spline will act as the minimizer of the empirical risk within the class $\mathcal{D}([a,b])$.

Theorem 4.3.5. Assume that $p = 1$ and $P(X \in (a,b)) = 1$. Consider the minimization-problem

$$\min_{m \in \mathcal{D}([a,b])} \hat{R}_n(m) + \lambda \int_a^b m''(x)^2 dx$$

If there exists a solution $\hat{m}$ to this problem, then $\hat{m}$ is a natural, cubic spline.

Proof. Assume that $\hat{m}$ solves the minimization problem. Let m be a natural, cubic spline with knots in $X_1, \dots, X_n$ and such that $\hat{m}(X_i) = m(X_i)$ for $i = 1, \dots, n$. Define $h = \hat{m} - m$. If we can show $h = 0$, then $\hat{m} = m$ is a natural, cubic spline as desired.

Note that since m is a spline, it is almost-everywhere C^∞. Note also that since m was assumed to be a natural, cubic spline, it is linear on $(-\infty, X_{(1)}] \cup [X_{(n)}, \infty)$. Since a, b both lie in the interior of this set with probability 1, we have that $\frac{d^j}{dx^j}m(x) = 0$ for $j \geq 2$ and $x \in \{a, b\}$. Additionally, m has any fourth derivative equal to 0. Hence

$$\begin{aligned} \int_a^b m''(x)h''(x)dx &= [m''(x)h'(x)]_a^b - \int_a^b m'''(x)h'(x)dx = 0 - [m'''(x)h(x)]_a^b + \int_a^b m''''(x)h(x)dx \\ &= \int_a^b 0 \cdot h(x)dx = 0. \end{aligned}$$

Now, note that

$$\begin{aligned} \int_a^b \hat{m}''(x)^2 dx &= \int_a^b (m''(x) + h''(x))^2 dx = \int_a^b m''(x)^2 dx + \int_a^b h''(x)^2 dx + \int_a^b 2m''(x)h''(x)dx \\ &= \int_a^b m''(x)^2 dx + \int_a^b h''(x)^2 dx \\ &\geq \int_a^b m''(x)^2 dx \end{aligned}$$

Since we have $\hat{m}(X_i) = m(X_i)$ for each observation, we have that $\hat{R}_n(m) = \hat{R}_n(\hat{m})$ and thus by the assumption that $\hat{m}$ minimizes the penalized empirical risk we must have

$$\int_a^b \hat{m}''(x)^2 dx \leq \int_a^b m''(x)^2 dx.$$

Together, these inequalities imply that

$$\int_a^b \hat{m}''(x)^2 dx = \int_a^b m''(x)^2 dx$$

and thus

$$\int_a^b h''(x)^2 dx = 0$$

Hence we can conclude that $h'' = 0$ and thus that h is linear.

Since we have $h(X_1) = \hat{m}(X_1) - m(X_1) = 0 = \hat{m}(X_2) - m(X_2) = h(X_2)$, it follows that we must have $h = 0$, and thus we know that $\hat{m} = m$ is a natural, cubic spline as desired[2]. □

The takeaway from the theorem is that if we care about finding the minimizer for our penalized empirical risk over the space $\mathcal{D}([a, b])$, this minimizer will always be a natural, cubic spline. However, the spline we used for the proof had knots in each observation, which quickly becomes a lot of knots.

The practical solution to this issue is to use fewer knots, thus not allowing the spline to fit as flexibly to data. In more mathematical terms, we would thus only consider the minimization problem of the empirical risk over the space of splines with knots in a few, selected points. There are two issues that arise from this solution.

Firstly, we cannot know if the minimizer of the empirical risk within this class is also a minimizer of the empirical risk within $\mathcal{D}([a, b])$. While this may seem problematic, it is not as bad as one would think. Remembering what we truly care about is the risk, not the empirical risk. Hence, once can think of having less knots as additional regularization.
The second issue is related to actually selecting which points we use as knots. The number of knots and the location of the knots themselves can all be seen as hyperparameters to be chosen, but most often we will simply pick an appropriate number of quantiles for the training data as knots.

Example 11. Let us consider a one dimensional supervised learning problem, where we wish to use splines. We assume that

$$Y = m^*(X) + \varepsilon \quad , \quad X \sim \mathcal{N}(0, 1) \quad , \quad \varepsilon \sim \mathcal{N}(0, 1/4)$$

and $m^* : \mathbb{R} \to \mathbb{R}$ given by $m^*(x) = \sin(2\pi x)$. For this problem, our data will consist of samples $(X_1, Y_1), \ldots, (X_{1000}, Y_{1000})$.

At first, we will need to decide for the number of knots ℓ to use. For the sake of simplicity, we choose $\ell = 10$ and choose the knots at the $\frac{1}{11}, \ldots, \frac{10}{11}$-quantiles of the data. Having decided the knots, we produce a basis consisting of $\ell + 1$ (11) B-splines $g_1, \ldots, g_{11}$. Our estimator will be

$$\hat{m}(x) = \sum_{i=1}^{11} \hat{\beta}_i g_i(x)$$

And hence all we need to do is estimate the $\hat{\beta}_i$'s. This is done through ordinary least squares regression, corresponding to the supervised learning problem (X', Y') with

$$X' = (g_1(X), \ldots, g_{11}(X)) \quad , \quad Y' = Y$$

[2]Note that we assumed here that $X_1 \neq X_2$ to conclude $h = 0$. If we do not have two different observations, then the minimizer of the penalized loss is a constant function and the Theorem is trivially satisfied

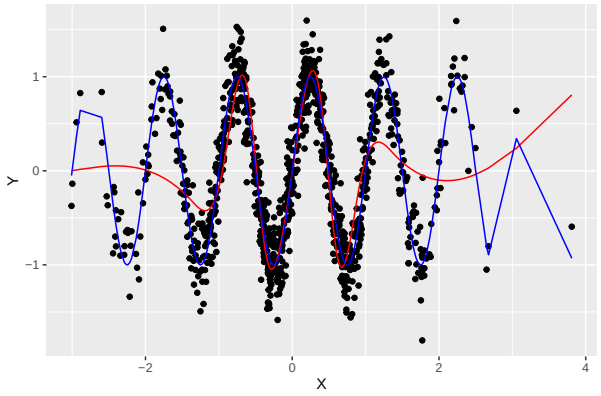

Figure 4.1: 1-dimensional regression problem solved via splines. The red line shows the estimated decision function $\hat{m}$ and the points are the observed data. The blue line shows the true function $m^*(x) = \sin(2\pi x)$ (or at least a linear interpolation of it).

The estimates we obtain for the β_i's are not meaningful. They represent the weights given to each basis spline. The meaningful object we get from this procedure is $\hat{m}(x)$, which we show in the plot below together with the observed data $(X_i, Y_i)_{i=1,\dots,1000}$.

We observe that the estimated spline decision function $\hat{m}$ is pretty good (in a qualitative sense) at approximating the true data-generating function m^*.

Remark. Before we end our treatment of splines, it should be mentioned that it can be shown that smoothing with splines is equivalent in some asymptotic sense to kernel smoother methods with varying bandwidthsSilverman (1984); Wang et al. (2013). Kernel smoothers can sometimes be easier to analyse theoretically, which has lead to more theory being developed for them. However, models based on splines are often easier to implement and more efficient, hence these are used more in practice.

4.3.2 Backfitting

For our treatment of splines, we have been assuming $p = 1$. Hence the type of additive models we have been working with are

$$Y = m^*(X) + \varepsilon \quad , \quad X \in \mathbb{R}$$

In the more general case, we instead have

$$Y = \sum_{j=1}^{p} m_j^*(X_j) + \varepsilon \quad , \quad X \in \mathbb{R}^p,$$

and thus we are to estimate each function m_j^* rather than just a single one.
If we assume an additive structure, then a naive approach would be to use the spline approach, as discussed in the previous section, for each coordinate separately. However, this will only lead to consistent estimators if the features are uncorrelated. In the following, we will present an alternative (the backfitting algorithm) that does also work when features are correlated. The idea is the following.Assume that by some luck, we know $(m_k^*)_{k\neq j}$. This means that we can compute the residuals $Y^{(j)}$ as

$$Y^{(j)} := Y - \sum_{k\neq j} m_k^*(X_k) = m_j^*(X_j) + \varepsilon$$

The advantage of rewriting the additive model in this way is that we can now consider the supervised learning problem corresponding $(X_j, Y^{(j)})$. This reduced problem has $p = 1$ and hence we can employ any one-dimensional nonparametric supervised learning model we prefer (e.g. smoothing splines, kernel

smoothing or k-nearest neighbors) to infer m_j^*.

Clearly, we don't actually know the functions $(m_k^*)_{k\neq j}$, but the above heuristic leads to the following formal algorithm:

Backfitting Algorithm

Goal: An estimator for $(m_j^*)_{j=0,\dots,p}$ under the constraint $\mathbb{E}[m_j^*(X_j)] = 0$ for $j = 1, \dots, p$.

a) Set $\hat{m}_0 = \frac{1}{n}\sum_{i=1}^n Y_i$ and set $Y_i^{(0)} = Y_i - \hat{m}_0$.

b) Initialise $\hat{m}_1^{(0)}, \dots, \hat{m}_p^{(0)}$ arbitrarily. Often $\hat{m}_j^{(0)} = 0$ for each $j = 1, \dots, p$.

c) For $r = 1, 2, \dots$ until convergence, do:

1) For $j = 1, \dots, p$ do:

I) For $i = 1, \dots, n$, calculate

$$Y_i^{(j)} = Y_i^{(0)} - \sum_{k=1}^{j-1} \hat{m}_k^{(r)}(X_{i,k}) - \sum_{k=j+1}^{p} \hat{m}_k^{(r-1)}(X_{i,k})$$

II) Set $\hat{m}_j^{(r)} = m_j((X_{k,j}, Y_k^{(j)})_{k=1,\dots,n})$, where m_j is the algorithm used for the j'th feature.

2) **Optional:** Ensure that the net contribution of the j'th feature is zero by setting

$$m_j^{(r)} \leftarrow m_j^{(r)} - \frac{1}{n}\sum_{i=1}^n \hat{m}_j^{(r)}(X_{i,j})$$

d) The resulting estimator is $\hat{m}_n(x) = \hat{m}_0 + \sum_{j=1}^p \hat{m}^{(r_{\text{Stop}})}(x_j)$.

A further note on Step (c2). Assuming the additive model

$$Y = \sum_{j=1}^{p} m_j^*(X_j) + \varepsilon,$$

we have that the functions $m_j^* : \mathbb{R} \to \mathbb{R}$ are only defined up to a additive constant. In other words, we cannot distinguish between $Y = m_1^*(X_1) + (m_2^*(X_2) + c)$ and $Y = (m_1^*(X_1) + c) + m_2^*(X_2)$. To circumvent this degeneracy, we note that in the algorithm we have chosen the identification $\mathbb{E}[m_j^*(X_i)] = 0$ for $j = 1, \dots, p$, $m_0^* = \mathbb{E}[Y]$. Lastly, note that in most cases Step (c2) can be omitted within the iteration and employed only once at the very end. This will lead to the same result but with improved computational time.

It has been shown backfitting via smoothing splines achieve optimal rate of $n^{-4/(4+1)}$. While this is still slower than the n^{-1} rate in the linear case, it is does only deteriorate linearly in p..

Chapter 5

Tree models

In this chapter, we consider models based on decision trees. Since these models are both used for regression and classification, we may sometimes be interested in the squared loss as in the previous chapters, but also at other times we may care about the classification error $L_{\text{Bin}}(y_1, y_2) = \mathbb{1}\{y_1 \neq y_2\}$.

Note that for both types of loss, the Bayes rule is known. For the squared loss, we gave the result in lemma 3.0.1. For the classification error, we provide the result with the following lemma.

Lemma 5.0.1. Consider a classification problem with a finite number of categories, $\#\mathcal{Y} < \infty$, with classification error as the loss. Then

$$\text{argmin}_{m \text{ measurable}} L_{\text{Bin}}(Y, m(X)) = \text{argmax}_{y \in \mathcal{Y}} P(Y = y \mid X = x)$$

Remark. The function

$$m(x) = \text{argmax}_{y \in \mathcal{Y}} P(Y = y \mid X = x)$$

is called the majority vote, because

$$P(Y = y \mid X = x) \approx \frac{\sum_{i=1}^{n} \mathbb{1}\{X_i = x, Y_i = y\}}{\sum_{i=1}^{n} \mathbb{1}\{X_i = x\}}$$

corresponds to the proportion of observations indicating that $Y = y$. And thus we can interpret $m(x)$ as picking out the $Y = y$ that was 'voted for the most'.

5.1 Tree Model Fundamentals

At this point, it would be reasonable to introduce a formal definition of a tree model. However, I believe it is more instructive to start by motivating the model. We will assume $p = 1$ for now, or in other words $X \in \mathbb{R}$ and furthermore that we are in a regression setting, e.g. we care about the squared loss.

Our objective is to describe the distribution $Y \mid X$. If $X \perp\!\!\!\perp Y$, then this would boil down to simply having $Y \mid X \overset{D}{=} Y$. Since we care about the squared loss, the Bayes rule for this case would be $m^*(x) = E[Y \mid X = x] = E[Y]$. Hence our best guess for Y is simply the mean.

Independence is is some sense the most boring case. To consider something more interesting, say that $Y \perp\!\!\!\perp X$ conditional on $X > 0$. In other words, the distribution of Y is something, say $Y \overset{D}{=} Y_{(0,\infty)}$ for $X > 0$ and something else, say $Y_{(-\infty,0]}$ for $X \leq 0$. In this case, the Bayes rule is

$$\begin{aligned} m^*(x) &= \mathbb{E}[Y \mid X = x] = \mathbb{E}[Y_{(0,\infty)} \mathbb{1}\{X > 0\} + Y_{(-\infty,0]} \mathbb{1}\{X \leq 0\} \mid X = x] \\ &= \mathbb{E}[Y_{(0,\infty)}] \mathbb{1}\{x > 0\} + \mathbb{E}[Y_{(-\infty,0]}] \mathbb{1}\{x \leq 0\} \end{aligned}$$

What we observe is that the best guess for Y depend on the value of x in such a way that if $x > 0$, we should guess at $\mathbb{E}[Y_{(0,\infty)}]$, while if $X \leq 0$, we should guess at $\mathbb{E}[Y_{(-\infty,0]}]$.

The obvious way to make Y even more dependent on X is to split either of the intervals $(-\infty, 0]$ or $[0, \infty)$, for the sake of the example, say we pick the positive part of the real axis, into two new intervals, say $(0, 1]$ and $(1, \infty)$. Then we could have $Y \perp\!\!\!\perp X$ conditional on which of the three intervals $(-\infty, 0], (0, 1], (1, \infty)$ we observe X in. The same calculations we had before shows that

$$m^*(x) = \mathbb{E}[Y_{(-\infty,0]}]\mathbb{1}x \in (-\infty, 0] + \mathbb{E}[Y_{(0,1]}]\mathbb{1}x \in (0, 1] + \mathbb{E}[Y_{(1,\infty)}]\mathbb{1}x \in (1, \infty)$$

From here, we could obviously add as many extra subdivisions/intervals as we like, but the point is that each subdivision leads to a new Bayes rule, which is constant for X in some interval. In terms of tree models, the system of intervals $\mathcal{A}$ is called the 'leaves' or 'terminal nodes' of the tree.

A tree model is essentially just a model producing an estimator that is consistent with the above subdivision scheme. We will formalize this in the following definitions.

Definition 5.1.1 (Tree)**.** Let $(\mathcal{X}, \mathcal{F})$ and $(\mathcal{Y}, \mathcal{E})$ be measurable spaces and consider the supervised learning problem corresponding to the product space $(\mathcal{X} \times \mathcal{Y}, \mathcal{F} \otimes \mathcal{E}, P)$ where P is some probability measure.
A tree model T consists of some finite system $\mathcal{B} \subseteq \mathcal{F}$ of $\mathcal{F}$-measurable sets and a function $f : \mathcal{B} \setminus \{\mathcal{X}\} \to \mathcal{B}$ that fulfill the following conditions:

T1) $\mathcal{X} \in \mathcal{B}$ (Root node)

T2) $\forall B_1, B_2 \in \mathcal{B} : B_1 \cap B_2 = \emptyset \vee B_1 \subseteq B_2 \vee B_2 \subseteq B_1$ (Nodes cannot partially overlap)

T3) $\forall B_1, B_2 \in \mathcal{B} : B_1 \subseteq B_2 \Rightarrow B_2 \setminus B_1 \in \mathcal{B}$ (Splitting a node always produces two nodes)]

T4)

$$\forall B \in \mathcal{B} \exists d_B \in \mathbb{N}_0 : f^{d_B}(B) := \underbrace{f \circ \cdots \circ f}_{d_B \text{ times}}(B) = \mathcal{X}$$

We say that d_B is the depth of the node B.

The function $f : \mathcal{B} \to \mathcal{B}$ maps a node to its parent node and is a technical construct that we use to keep track of the tree structure. It is most easily understood with a drawing, see figure 5.1.

Definition 5.1.2. Given a tree model $T = (\mathcal{B}, f)$, we say that

$$\mathcal{A} = \mathcal{A}(T) = \{A \in \mathcal{B} \mid f^{-1}(\{A\}) = \emptyset\}$$

is the set of leaves (or terminal nodes) for the tree.

In Figure 5.1, the leaves are B_4, B_5, B_6, B_7.
Up to now, the formal definition of a tree model has not involved an estimator. To get an decision function, we should approximate either the conditional mean in each leaf in the case of a squared loss (in virtue of lemma 3.0.1) or approximate the majority vote in each leaf in the case that we consider the classification error as loss (in virtue of Lemma 5.0.1).

Corollary 5.1.3. Let T be a tree model and let $\mathcal{D}_n = (X_i, Y_i)_{i=1,\ldots,n}$ be i.i.d. data from the supervised learning problem in consideration. In the case of regression, the squared loss-minimizing decision function produced by the tree model is then

$$\hat{m}_n(x) = \sum_{A \in \mathcal{A}} \mathbb{1}x \in A \left(\frac{\sum_{i=1}^n \mathbb{1}\{X_i \in A\} Y_i}{\sum_{i=1}^n \mathbb{1}\{X_i \in A\}} \right).$$

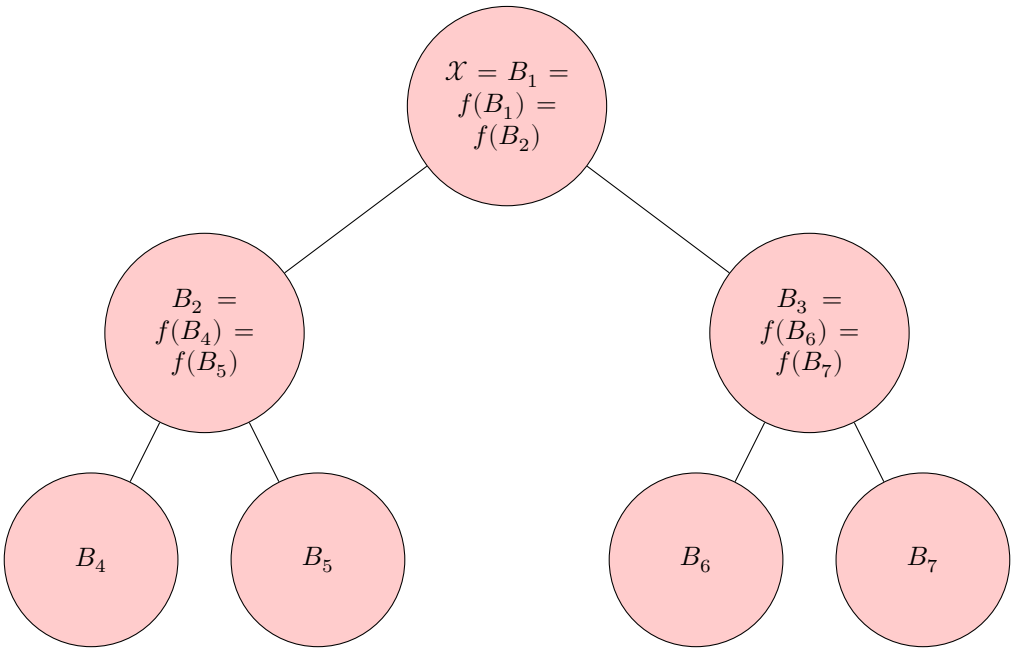

Figure 5.1: A visual representation of a tree model. The function value $f(B)$ is the set that was subdivided to produce B.

In the case of classification, the classification error minimizing decision function is

$$\hat{m}_n(x) = \sum_{A \in \mathcal{A}} \mathbb{1}\{x \in A\} \cdot \text{argmax}_{y \in \mathcal{Y}} \left(\sum_{i=1}^{n} \mathbb{1}\{X_i \in A\} \mathbb{1}\{Y_i = y\} \right).$$

While the estimators are somewhat abstractly when written up like this, they convey the idea that if we want to predict the value of y from x, we decide which 'leaf' $A \in \mathcal{A}$ that satisfies $x \in A$, and then the prediction for y will be some appropriate average of the Y_i's corresponding to the X_i's that ended up in that 'leaf'. For regression, the appropriate average is really just the ordinary average, while for classification it is the so-called majority vote, which corresponds to picking the $y \in \mathcal{Y}$ that coincide with the most Y_i's within the leaf.

5.2 Fitting Tree Models

We have introduced tree models in terms of their leaves, the system $\mathcal{A}$ of sets they assign distinct values to. Since we are clearly interested in producing tree models that are good at solving supervised learning problems, we should device a clever way to pick the leaves for our tree models.

In principle, we could have the leaves make up any disjoint partition of the feature space $\mathcal{X}$, but since there are either a lot of these partitions when $\mathcal{X}$ is finite or infinitely many partitions when $\mathcal{X}$ is infinite, we cannot simply try each one.

So for actually producing a reasonable tree, we will adhere to the CART algorithm. Before introducing this, we need the following shorthand.

Definition 5.2.1. Let $B \in \mathcal{B}$ be some node. We say that the loss Q associated with the node is

$$Q(B) = \sum_{i=1}^{n} \mathbb{1}\{X_i \in B\} L(Y_i, M(B))$$

$$M^{\text{Regression}}(B) = \frac{\sum_{i=1}^{n} \mathbb{1}\{X_i \in B\} Y_i}{\sum_{i=1}^{n} \mathbb{1}\{X_i \in B\}}$$

$$M^{\text{Classification}}(B) = \text{argmax}_{y \in \mathcal{Y}} \left(\sum_{i=1}^{n} \mathbb{1}\{X_i \in B\} \mathbb{1}\{Y_i = y\} \right)$$

The loss of a node is really just the contributions to the empirical loss from the observations that fall into that node.

Remark. Sometimes it can also be beneficial to pick a loss different from the one that we care about for the supervised learning problem itself. This is akin to the way that penalization may sometimes improve performance, even if we don't want to minimize a penalized loss in the end. In the binary classification case, common alternative losses are 'Brier score', 'Misclassification error', 'Gini index' and 'Entropy'. Define

$$\hat{p}_k = \frac{1}{|B|} \sum_{i: X_i \in B} \mathbb{1}(Y_i = k),$$

i.e., the proportion of class k observations in B.

- Brier score: $Q(B) = \sum_{i: X_i \in B} (Y_i - \overline{Y}_i(B))^2$ (=squared loss for $Y_i = 0, 1$)
- Gini index: $Q(B) = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k)$
- Entropy : $Q(B) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$

Now, we can introduce the CART algorithm, which is a greedy way of finding a tree model for a given data set.

CART Algorithm ($\mathbb{R}^p$)

Hyperparameter:

- One stopping criterion. Two possibilities are
  - $d_{\text{Max}} \in \mathbb{N}$ - An integer specifying the maximal depth of the decision tree: $d_A < d_{max}$?
  - $n_{\text{Stop}} \in \mathbb{N}$ - An integer specifying the number of observations that must fall into a leaf before the leaf may be split: $|A| > n_{\text{Stop}}$?

I) Start with $\mathcal{B} = \{\mathbb{R}^p\}$, $\mathcal{A} = \mathcal{B}$, $f(\mathbb{R}^p) = \mathbb{R}^p$.

II) For each $A \in \mathcal{A}$ that does not fulfill stopping criteria do:

i) For each $j = 1, \dots, p$, and every point $s \in \{X_{ij} | X_i \in A\}$ do:

a) Define $A^1(j, s) = \{x \in A \mid x_j \leq s\}$ and $A^2(j, s) = \{x \in A \mid x_j > s\}$.

b) Calculate $Q(A^1(j, s) + Q(A^2(j, s))$.

ii) Pick

$$(j^* s^*) = \text{argmin}_{\substack{j \in \{1, \dots, p\} \\ s \in \{X_{ij} | X_i \in A\}}} Q(A^1(j, s)) + Q(A^2(j, s))$$

and add $A^1(j^*, s^*)$ and $A^2(j^*, s^*)$ to $\mathcal{B}$ with $f(A^1(j^*, s^*)) = f(A^2(j^*, s^*)) = A$.

III) Repeat II) until stopping criterion is fulfilled for all leaves $A \in \mathcal{A}$

IV) The resulting tree model is now $T = (\mathcal{B}, f)$.

Let us explain the steps of the CART algorithm in some more detail. The algorithm starts with only the root node $\mathcal{B} = \{\mathbb{R}^p\}$. At this time, this node is terminal, hence we split it in step (II). The split works

by initially choosing some dimension (step (i)) and then splitting the node along the chosen dimension at a split point, see Figure 5.2. Each split is evaluated with respect to the loss of the resulting nodes. This is repeated for each dimension $j \in \{1, \dots, p\}$ and the dimension and split resulting in the largest improvement to the empirical loss is chosen (step ii). This will produce two new nodes A_1 and A_2, which are currently terminal, that is, $A_1, A_2 \in \mathcal{A}$. For each of these, we repeat step (II) given that they do not fulfill the stopping criterion. Each of these will then potentially be split and generate new terminal nodes that can potentially be split further. The procedure stops when all terminal nodes fulfill the stopping criterion.

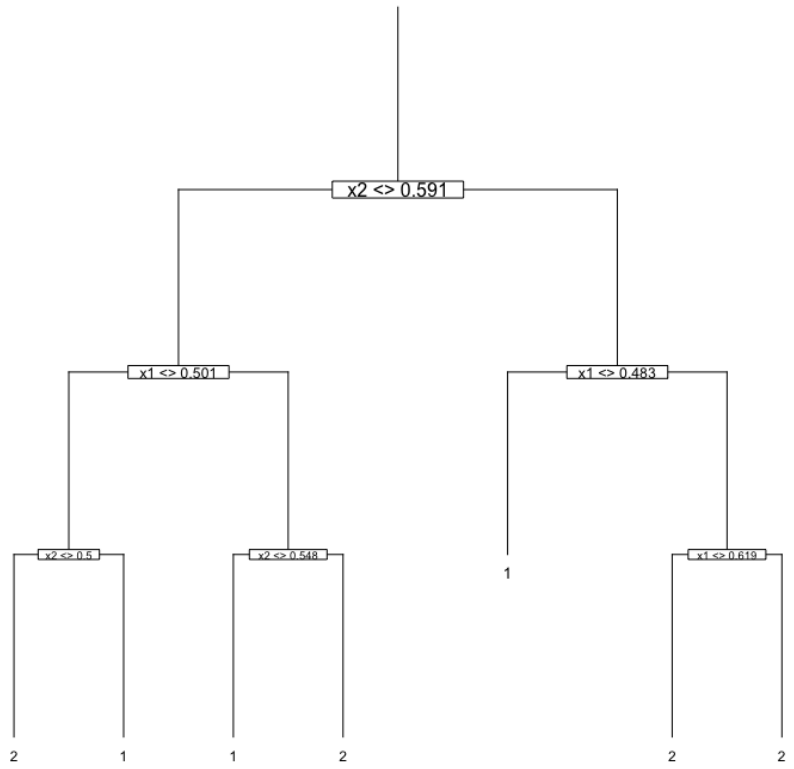

Figure 5.2: Illustration of decision function stemming from a binary tree

So are we happy with our tree model now? Not quite. Since we would like to be sure that we allow the algorithm the freedom to find good splits, we will usually set n_{Stop} and d_{Max} generously, thus letting the algorithm grow the tree very far. If we allowed for $d_{\text{Max}} = \infty$ or $n_{\text{Stop}} = 1$, the algorithm would terminate with

$$\mathcal{A} \approx \{\{X_1, \text{No other } X_i's\}, \dots, \{X_n, \text{No other } X_i's\}\}$$

and hence for any $x \in \mathbb{R}^p$, the decision function would just take the value (Y_i) of a nearly observation (X_i).

Very likely, the situation described above would lead to over-fitting to data, and hence the model would generalize poorly. To avoid this, we could tune the hyper parameters d_{Max} and n_{Stop} less generously, but the disadvantage of this is that the tree may miss out on some very advantageous splits.

Figure 5.3: An example of a feature distribution where no initial split leads to a notable improvement in performance. The second split can potentially lead to a large improvement in performance.

Example 12. Consider a supervised learning problem with

$$X_1, X_2 \sim Unif(0,1) \quad , \quad Y = \mathbb{1}\{X_1 > 0.5\}\mathbb{1}\{X_2 > 0.5\} + \mathbb{1}\{X_1 \leq 0.5\}\mathbb{1}\{X_2 \leq 0.5\}$$

We have sampled $n = 500$ points from this problem and shown the result in figure 5.3. It is apparent from the way the features are distributed that no matter how the first split is placed, it will not lead to a good separation of the points with $Y = 0$ from $Y = 1$. However, this does not mean that the following splits could not provide very good separation, in this case a second split along the other axis would probably do quite well.

The morale is that terminating the growth of a tree early may cause out on the model missing important splits that could have notably improved it. This would suggest that if we have enough computational resources available, we should rather grow the tree as much as possible and then remove any splits that ended up not doing very well.

Of course, the a tree model will always have a smaller empirical loss if we allow for more leaves. This is caused by the fact that any leaf can be split into two new leaves that each predict the same value as the original leaf. Since no predictions change, the empirical loss stays the same, and hence the model can never be worse if we add another leaf, but it may improve if setting the values of the two new leaves different from one another would allow for an improvement.

Therefore, to tell whether a split is good or bad, we need to introduce some form of penalization on the tree model that would encourage it to have fewer splits. This is done in the form of a penalty factor $\alpha|\mathcal{A}|$, which is some constant $\alpha > 0$ times the number of leaves in the tree.

To formalize the process of pruning a tree after growing it, we introduce the notion of sub-trees.

Definition 5.2.2. Consider a tree model $T = (\mathcal{B}, f)$. We say that a tree model $T' = (\mathcal{B}', f')$ is a sub-tree if $B' \subseteq \mathcal{B}$ and $f = f'$ on $\mathcal{B}'$. Note that we may abuse the notation and write $(\mathcal{B}', f)$ instead of $(\mathcal{B}', f')$.

In other words, T' is a sub-tree of T if deleting the nodes contained in $\mathcal{B} \setminus \mathcal{B}'$ from T would yield exactly T'. In terms of growing trees, this means that T is is grown in the same way as T', but T is possibly be grown further.

The point of introducing the notion of a sub-tree is that the problem of deleting the bad splits produced by the CART algorithm can now be phrased in terms of finding an optimal sub-tree.

Definition 5.2.3. Let $\hat{T}_n$ be a tree model. We say that the α-pruned sub-tree corresponding to $\hat{T}_n$ is

$$\hat{T}_{n,\alpha} := \text{argmin}_{\hat{T} \text{sub-tree of } \hat{T}_n} \left(\hat{R}_n(\hat{m}_{\hat{T}}) + \alpha |\mathcal{A}(\hat{T})| \right)$$

Where $\hat{m}_{\hat{T}}$ is the estimator corresponding to the tree $\hat{T}$ and $\mathcal{A}(\hat{T})$ is the set of leaves for the tree $\hat{T}$.

Note that it is not clear how α should be chosen - It is just some hyperparameter controlling the cost of splitting a leaf. The immediate thought one could have would be to try some different α's, compute the optimal sub-tree for each value, and use some form of cross-validation to decide which one is better. This could potentially be very computationally intensive, but luckily there is a better (more systematic) way to do this.

Since $\alpha > 0$ is a continuous hyperparameter, each $\alpha > 0$ will lead to some optimal sub-tree. However, since any tree model has finitely many leaves and hence finitely many sub-trees, there are only finitely many possible optimal sub-trees. Hence many values of $\alpha > 0$ will lead to the same optimal sub-tree, and thus it is sufficient to consider the optimal sub-trees themselves instead of the values for α that make them optimal.

The first issue we have to solve is that we should find each optimal sub-tree. This is done through the weakest-link algorithm.

Weakest-Link Algorithm

Input:

- $(\mathcal{B}_0, f)$ - A tree model that we wish to find an optimal sub-tree model of.

a) For $r = 0, \dots, \infty$ until $\mathcal{B}_r = \{\mathcal{X}\}$, do:

I) Go through each non-terminal node $B \in \mathcal{B}_r \setminus \mathcal{A}_r$:

i) Define $B_{\text{Child}} = \{A \in \mathcal{B}_r \mid \exists n \in \mathbb{N} : f^n(A) = B\}$,

$$g(B) = \frac{Q(B) - \sum_{A \in B_{\text{Child}} \cap \mathcal{A}_r} Q(A)}{|B_{\text{Child}} \cap \mathcal{A}_r| - 1}.$$

II) Set

$$\alpha_{r+1} = \min_{B \in \mathcal{B}_r \setminus \mathcal{A}_r} g(B)$$

.

III) Let

$$\mathcal{B}_{r+1} = \mathcal{B}_r \setminus \left(\bigcup_{\substack{B \in \mathcal{B}_r \\ g(B) = \alpha_{(r+1)}}} B_{\text{Child}} \right)$$

b) The sequence of optimal sub-trees is $(\mathcal{B}_0, f), (\mathcal{B}_1, f), (\mathcal{B}_2, f), \dots$

The weakest-link algorithm does, as the name suggest, eliminate the weakest node from the tree model in each iteration. Starting with penalty $\alpha_0 = 0$, the optimal sub-tree is the one we started with, since we have no penalization and thus no reason to prune any nodes from the tree.

In the first iteration, the algorithm goes through each node and identifies $g(B)$, which is the improvement-per-child ratio for that particular node. If we were to prune all children for some node B, the loss would increase by exactly $g(B) \cdot (|B_{\text{Child}} \cap \mathcal{A}_r| - 1)$. However, since the loss is penalized by α, we also decrease it by $\alpha \cdot (|B_{\text{Child}} \cap \mathcal{A}_r| - 1)$. With $\alpha = \alpha_0 = 0$, this is always a bad idea, but what the algorithm does in step (II) is to find the α_{i+1} such that it becomes exactly worth it to prune all children of the node with the poorest average performance. This pruning is carried out in step (III).

From here, the algorithm continues pruning one node (thus deleting all the children of that node) at a time. This results in the sequence of optimal sub-trees $(\mathcal{B}_0, f), (\mathcal{B}_1, f), \dots$ corresponding to the penalties $\alpha_0, \alpha_1, \dots$.

Having obtained a sequence of possible tree models that we know are optimal for some level of penalization, we should decide which tree model we would actually like to use in the end. While there may be multiple ways to do this, a possible approach is based on cross validation[1].

[1]The following algorithm is based on cross-validation section of the 'Introduction to rpart' vignette for the `R`-package `rpart`.

Cross Validation CART + WL Algorithm

a) Use the CART algorithm to produce an initial, fully grown tree model $(\mathcal{B}_0, f)$.
b) Use the Weakest-link algorithm to obtain a sequence $(\mathcal{B}_0, f), \dots, (\mathcal{B}_N, f)$ of pruned models with corresponding penalization constants $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_N < \alpha_{N+1} = \infty$.
c) For $j = 0, \dots, N$, compute $\beta_j = \sqrt{\alpha_j \alpha_{j+1}}$.
d) Split the data into K folds $(\mathcal{D}_k, \mathcal{D}_{-k})_{k=1,\dots,K}$. For each fold, do:
   i) Fit an initial tree model $(\mathcal{B}_0^k, f^k)$ on $\mathcal{D}_{-j}$ using CART.
   ii) Prune the initial tree $(\mathcal{B}_0^k, f)$ for each $\beta_0, \dots, \beta_N$ to obtain tree models $(\mathcal{B}_0^k, f), \dots, (\mathcal{B}_N^k, f)$.
   iii) Compute the loss for each pruned tree model on the held out fold $\mathcal{D}_k$, e.g.

$$L_{k,j} = \frac{1}{|\mathcal{D}_k|} \sum_{(X,Y)\in\mathcal{D}_k} L(Y, \hat{m}^{\mathcal{B}_j^k}(X))$$

e) Compute the cross-validation errors

$$CV_j = \frac{1}{K} \sum_{k=1}^{K} L_{kj}$$

The final model is $(\mathcal{B}_{j^*}, f)$ where $j^* = \text{argmin}_{j=0,\dots,N} CV_j$.

5.2.1 Categorical Features

While categorical features are omni-present in many applications there are often ignored in many methodological and theoretical developments. One obvious way to deal with them that may circumvent most problems is to dummy encode them:

Definition 5.2.4 (dummy encoding). Given a feature X that can take k different categorical values, dummy encoding entails transforming the feature into $k-1$ binary features where, given a reference l^*,

$$X^{(l)} = \mathbb{1}(X = l), \quad l = 1, \dots, l^* - 1, l^* + 1, \dots, k.$$

In particular the one categorical feature has been transformed into $k-1$ binary features.

A dummy encoded very can usually be treated the same way as a numerical variable. While dummy encoding may work well in low cardinality cases, i.e. if the number of categories the feature has is small, problems may arise in the high cardinality case. Classical example of high cardinality categorical features are country or postcode where the number of categories can easily be in the hundreds or thousands. Dummy encoding in those cases would lead to an explosion in the dimension p of the feature space; probably leading to a poor performance of the algorithm.
In theory, the CART algorithm can be extended to deal with categorical features without any encoding. Instead of splitting a numerical feature X_j at every possible split point $s \in \{X_{ij} | X_i \in A\}$ of the current node A(cf. II(i) in the CART Algorithm $(\mathbb{R}^p)$), for a categorical feature X_j one can consider all sets $S \subseteq \text{supp}(X_j|A) = \{s : x \in A, x_j = s\}$ with $A^1(j, S)$ and $A^2(j, S)$ non-empty, where

$$A^1(j, S) = \{x | x \in A, x_j \in S\}, \quad A^2(j, S) = \{x | x \in A, x_j \notin S\},$$

leading to a partition $A = A^1(j, S) \cup A^2(j, S)$, $A^1(j, S) \cap A^2(j, S) = \emptyset$. The problem with naively considering all such sets S and picking the one that reduces the loss the most is that there are actually many sets S; to be precise $2^{k_A} - 1$, $k_A = |\text{supp}(X_j|A)|$. This number can be reduced slightly. Since we don't care about the ordering of the partition (S and $\text{supp}(X_j|A)| \setminus S$ lead to the same partitioning) the actual number of partitions we need to consider is $2^{k_A - 1} - 1$. This is a prohibitively large number for slightly large k_A.

We will now discuss that if one is looking for the subset S that minmizes $Q(A^1(j,S) + Q(A^2(j,S))$, and Q is the squared loss, then the minimizing S is a contiguous subset when the elements s in $\text{supp}(X_j|A)|$ are ordered according to their target encoding.

Definition 5.2.5 (target encoding)**.** Given a feature X that can take k different categorical values $x_1, \dots, x_k$, target encoding entails encoding category x_l, $l = 1, \dots, k$ as the conditional mean response minus the unconditional mean response:

$$x_l \leftarrow \mathbb{E}[Y|X = s_l] - \mathbb{E}[Y] =: \tilde{x}_l.$$

Theorem 5.2.6. Let the loss Q be measured via squared loss. or Gini index Let X_j be a target encoded categorical feature. Then,

$$S^* := \arg\min_{S \subseteq \text{supp}(X_j|A)} Q(A^1(j,S) + Q(A^2(j,S))$$

is a contiguous subset. i.e. there is a constant c such that

$$S^* = \{l \subseteq \text{supp}(X_j|A) : \tilde{x}_l < c\}.$$

In particular S^* is to be found among the $k_A - 1$ non-empty contiguous subsets.

Proof Sketch. We proof the theorem by contradiction. Assume S^* is not a contiguous subset. Then there are elements l_1, l_2, l_3 such that

$$\tilde{x}_{l_1} \leq \tilde{x}_{l_2} \leq \tilde{x}_{l_3}$$

and without loss of generality

$$x_{l_1}, x_{l_3} \in S^*, \quad x_{l_2} \in S^{*-} = \text{supp}(X_j|A) \setminus S^*$$

(otherwise the roles of S and S^- are switched). Define

$$\overline{S}^* = \frac{1}{|\{i : X_{ij} \in A^1(j,S^*)\}|} \sum_{i:X_{ij} \in A^1(j,S^*)} \tilde{X}_{ij}, \quad \overline{S}^{*-} = \frac{1}{|\{i : X_{ij} \in A^2(j,S^*)\}|} \sum_{i:X_{ij} \in A^2(j,S^*)} \tilde{X}_{ij},$$

It follows that one of the three statements must be true.

1. $|\tilde{x}_{l_1} - \overline{S}^*| \geq |\tilde{x}_{l_1} - \overline{S}^{*-}|$
2. $|\tilde{x}_{l_2} - \overline{S}^*| \geq |\tilde{x}_{l_2} - \overline{S}^{*-}|$
3. $|\tilde{x}_{l_3} - \overline{S}^{*-}| \geq |\tilde{x}_{l_3} - \overline{S}^*|$

Case 1. implies that the loss $Q(A^1(j,S^*) + Q(A^2(j,S^*))$ can be improved by moving x_{l_1} to S^{*-}. Similarly, in case 2. the loss can be improved by moving x_{l_2} to S^{*-} and in case 3. the loss can be improved by moving x_{l_3} to S^*. Hence, S^* cannot be the minimizer of $Q(A^1(j,S) + Q(A^2(j,S))$. □

The theorem implies that instead of searching through $2^{k_A - 1} - 1$ possible partitions for every node A when considering a categorical feature j as split candidate, one only needs to consider $k_A - 1 - 1$ partitions after target encoding. This heavily reduces the computational cost making that strategy computationally more feasible. Wright and König (2019) go one step further and propose to target encode categorical feature only once at the beginning and not at every node. They show in empirical studies that predictive power does not change much if doing so. The gain of this strategy is a further heavily reduced computational burden.

5.3 Bagging and Random Forests

Tree models are particularly advantageous in the sense that they are easily interpretable. Their representation as decision trees makes them very transparent. However, they suffer from high variance in the sense that slightly different data may lead to very different tree models. Heuristically, this is due to

the model's dependence on the location of the initial splits, which may be very affected by noise. For example, the initial split for the data in Figure 5.3 would be more or less random, since no split leads to a good separation of blue and red points.

A possible solution to this problem is the idea of **bagg**ing, which stands for '**B**ootstrap **Agg**regation'. The overall idea is summarised in Figure 5.4.

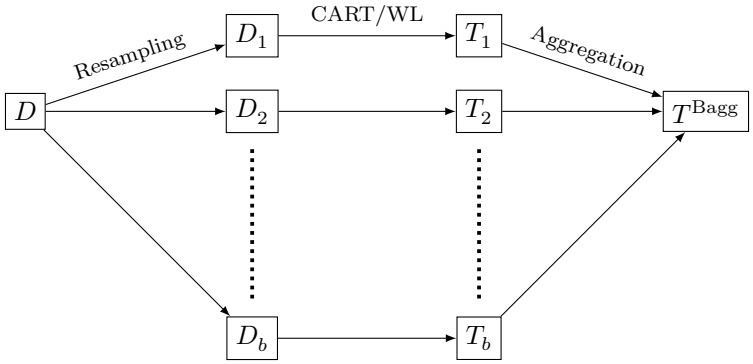

Figure 5.4: Bagging Procedure

Initially, the data $\mathcal{D}_n$ we have for the supervised learning process is used to produce b resampled data sets $D_1, \dots, D_k$. Usually, these would be of the same size as $\mathcal{D}_n$, obviously drawing with replacement.

Remark. If the resampled data sets are instead drawn without replacement and with a smaller number of observations compared to the original data set, this process is refered to as subsampling.

Each re-sampled data set is the used as the basis for fitting a tree model as discussed in section 5.2. Finally, the estimators produced by the b fitted tree models are then aggregated into a final estimator (depending on the kind of supervised learning problem):

$$\hat{m}_n^{\text{Bagg, Regr}}(x) = \frac{1}{b}\sum_{i=1}^{b} \hat{m}_n^{T_i}(x),$$

$$\hat{m}_n^{\text{Bagg, Classif}}(x) = \text{argmax} y \in \mathcal{Y} \left(\sum_{i=1}^{b} \mathbb{1}\{\hat{m}_n^{T_i}(x) = y\} \right).$$

While bagging does improve the issue of variability, it significantly worsens interpretability, since for large b it will depend on the decision of many decision trees.

Example 13. Without knowing anything about the subject in particular, animals are classified into species based on certain traits that make them similar. This classification roughly corresponds to a decision tree. If we used bagging with $b = 100$ for classifying species, this would correspond to having 100 biologists each come up with a set of classification rules for dividing animals into species. Whenever we would like to decide the species of an animal, those 100 biologists would then vote on species of the animal in question. Note that the 100 biologists are picked at random from the larger population of biologists in the world, corresponding to the resampled data sets being only a few of the possible data sets that could be resampled from the original data.

Our motivation for introducing bagging was that we wanted to have lower variance compared to that of a single tree estimator. Random forests are introduced as a further improvement over simple bagging in this regard. Let $(\hat{m}[\mathcal{D}_n, U_i])_{i=1,\dots,b}$ be a set of resampled tree algorithms based on data $\mathcal{D}_n$, where the U_i's are i.i.d. and describe the randomness introduced to the estimators apart from the one that arises from $\mathcal{D}_n$, e.g. the resampling in the bagging procedure (or the parameter selection that we will introduce shortly for random forests).

Assuming that $Var(\hat{m}[\mathcal{D}_n, U_i](X)) = \sigma^2$ and $Cov(\hat{m}[\mathcal{D}_n, U_i](X), \hat{m}[\mathcal{D}_n, U_j](X)) = \rho\sigma^2$ for $i \neq j$, we have that

$$\begin{aligned} Var\left(\frac{1}{b}\sum_{i=1}^{b}\hat{m}[\mathcal{D}_n, U_i](X)\right) &= \frac{1}{b^2}\sum_{i=1}^{b}\sum_{j=1}^{b} Cov(\hat{m}[\mathcal{D}_n D, U_i](X), \hat{m}[\mathcal{D}_n, U_j](X)) \\ &= \frac{1}{b^2}\left(n\sigma^2 + (n^2 - n)\rho\sigma^2\right) \\ &= \frac{\sigma^2 n}{b^2}(1 + (n-1)\rho) \end{aligned}$$

This is increasing in ρ, hence to reduce the variance of the final estimator, it would be beneficial to have little correlation between the different estimators. It is most likely not reasonable to expect a negative correlation, since the estimators are supposed to estimate the same thing and hence will agree for the most part, but if we can make them 'more independent', this would potentially reduce the variance.

The way we propose to do this is by modifying the first line in step II)i) in the CART algorithm.

CART Algorithm Modification (Random Forest)

i^*) Draw `mtry` numbers from $\{1, \dots, p\}$ and denote those $j_1, \dots, j_{\texttt{mtry}}$. For every $j_1, \dots, j_{\texttt{mtry}}$ and for every point $s \in \{X_{ij_l} | X_i \in A\}$ do:

The bagging estimator with this change implemented is called a random forest. Essentially, the modification to the CART algorithm ensures that the different tree models that are trained on bootstrapped data have to make different choices for which dimensions they split in regard to. Essentially, this means that the random-forest algorithm is less greedy, since it cannot freely chose the optimal dimension to split in.

Example 14. In terms of our earlier example of classifying species, using a random forest to clasify species would be like having an independent master-biologist supervising each of the 100 working biologists. Each time one of the 100 biologists would want to introduce a new rule for discerning between species, the supervising biologist would randomly tell the working biologist a set of `mtry` features that the working biologist would be allowed to use this rule. For example, if the possible features that could discern species are {Number of Legs, Maximum Speed, Amphibious (Y/N), Lifespan} and `mtry` $= 2$, the supervisor would randomly draw 2 features, say {Maximum Speed, Lifespan}. For each following rule, the supervisor would draw a potentially new set of two features from the available ones.

The supervisor would supervise each of the 100 working biologists, but they would all be supplied with different sets of features for their different rules, hence making their species classification more diverse/less correlated.

As a final note, random forests are known to perform quite well without particularly insightful tuning of hyperparameters. Good results are often obtained for `mtry` $= \lfloor\sqrt{p}\rfloor$, $d_{\max} = \infty$ and $n_{\text{stop}} = 1 + 4 \cdot \mathbb{1}\{\text{is.Regression}\}$. One should note that they also deal quite well with sparse setting, e.g. when there are few features that are important for the prediction.

Remark. Decision trees are not good with additive functions. To illustrate this, consider the following example. Let $m^*(x) = \sum_j^p \mathbb{1}(x_j \leq 0)$ for large p. For a perfect fit, a tree algorithm would have to grow a tree with depth p, where each leaf is the result of splitting once with respect to each covariate. Hence, we end up with 2^p leaves, which on average contain $n/(2^p)$ data points. In particular, the number of data points an estimated value is based on decreases exponentially. In the case of $2^p > n$, even if all splits are optimal the tree is not consistent.

5.4 Gradient Boosting

In this section, we will facilitate a discussion of gradient boosting. The idea underpinning this method is to build an estimator iteratively from so-called 'weak-learners'. We start by providing the following

definition.

Definition 5.4.1 (Forward Stagewise Additive Modeling)**.** The estimator $\hat{m}_n = \sum_{j=0}^{B} \hat{m}_{n,j}$ is called a forward stagewise additive modelling estimator corresponding to the function space $\mathcal{G}$ if

$$\hat{m}_{n,0} = \text{argmin}_{\eta \in \mathcal{G}} \hat{R}_n(\eta)$$

and for $b = 1, \dots, B$

$$\hat{m}_{n,b} = \text{argmin}_{\eta \in \mathcal{G}} \hat{R}_n \left(\sum_{j=0}^{b-1} \hat{m}_{n,j} + \eta \right)$$

Each component $\hat{m}_{n,j}$ is referred to as a weak learner. The reason for this is that we assume $\mathcal{G}$ to be a very restrictive class of functions, hence no function $\eta \in \mathcal{G}$ will be particularly good at approximating m^* by itself. However, the hope with forward stagewise additive modelling is that sums of these estimators may do a better job at this.

Remark. Forward stagewise additive modelling has to solve the inherent problem of calculating

$$\hat{m}_{n,b} = \text{argmin}_{\eta \in \mathcal{G}} \hat{R}_n \left(\sum_{j=0}^{b-1} \hat{m}_{n,j} + \eta \right)$$

The empirical loss $\hat{R}_n$ will depend on the loss function chosen for the problem, and for many loss functions, finding the minimizer (or something suitably close) is a hard problem. There are some notable cases, where this is easier:

- If we work with squared loss $L(y_1, y_2) = (y_1 - y_2)^2$, then

$$\begin{aligned} \hat{R}_n \left(\sum_{j=0}^{b-1} \hat{m}_{n,j} + \eta \right) &= \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \left(\sum_{j=0}^{b-1} \hat{m}_{n,j}(X_i) + \eta(X_i) \right) \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} \left(\left(Y_i - \sum_{j=0}^{b-1} \hat{m}_{n,j}(X_i) \right) - \eta(X_i) \right)^2 \\ &:= \frac{1}{n} \sum_{i=1}^{n} \left(Y_i^{(b-1)} - \eta(X_i) \right)^2 \end{aligned}$$

  Where $Y_i^{(b-1)} = Y_i - \sum_{j=0}^{b-1} \hat{m}_{n,j}(X_i)$ are the residuals after $b-1$ iterations. The reason this observation is useful is that the problem of finding the minimizing $\eta \in \mathcal{G}$ is the same for all $j = 0, \dots, B$, but rather than always calculating the empirical loss with respect to $Y_i = Y_i^{(0)}$, it will instead be with respect to the residuals $Y_i^{(0)}, Y_i^{(1)}, \dots, Y_i^{(B-1)}$.

  The same calculations apply for $L(y_1, y_2) = |y_1 - y_2| = \sqrt{(y_1 - y_2)}$ and similar transformations of the squared loss.

- If we work with $Y \in \{-1, 1\}$ (Classification problem) and the so-called exponential loss $L(y_1, y_2) = e^{-y_1 y_2}$, then

$$\begin{aligned} \hat{R}_n \left(\sum_{j=0}^{b-1} \hat{m}_{n,j} + \eta \right) &= \frac{1}{n} \sum_{i=1}^{n} e^{-Y_i \cdot \left(\sum_{j=0}^{b-1} \hat{m}_{n,j}(X_i) + \eta(X_i) \right)} \\ &= \frac{1}{n} \sum_{i=1}^{n} e^{-Y_i \sum_{j=0}^{b-1} \hat{m}_{n,j}(X_i)} e^{-Y_i \cdot \eta(X_i)} \\ &:= \frac{1}{n} \sum_{i=1}^{n} w_i^{(b-1)} \cdot L(Y_i, \eta(X_i)) \end{aligned}$$

Where $w_i^{(b-1)} = e^{-Y_i \sum_{j=0}^{b-1} \hat{m}_{n,j}(X_i)}$ are weights that can be computed after the $b-1$'th iteration of the minimization. Hence each minimization problem corresponds to minimizing an empirical weighted loss for the original target variable Y_i over function $\eta \in \mathcal{G}$. This algorithm is also known as AdaBoost.M1. Adaboost.M1 was introduced in Freund and Schapire (1997) and was only later Friedman et al. (2000) identified as Forward Stagewise Additive Modeling with exponential loss.

While we will not pursue a direct, forward stagewise additive modelling strategy, we introduce gradient boosting as an approximation to this.

Gradient Boosting Machines (Friedman (2001))

Hyperparameters:

$\mathcal{G}$ - Function space. Controls what weak learners are allowed. Should be restrictive.

η - Learning rate. Controls the rate of learning. Higher learning rates may lead to instability. We require $0 < \eta \leq 1$.

B - Number of iterations.

---

1) Set

$$\hat{m}_n^{(0)}(x) = \text{argmin}_{m \in \mathcal{G}} \hat{R}_n(m)$$

2) For $b = 1, \dots, B$, do:

(a) For each observation $i = 1, \dots, n$, do:

i. Calculate the (negative) gradients

$$g_{ib} = -\frac{\partial L(Y_i, y)}{\partial y}\Big|_{y=\hat{m}_n^{(b-1)}(X_i)}$$

(b) Chose $\tilde{m}_{n,b} \in \mathcal{G}$ by solving the minimization problem

$$(\xi_b, \tilde{m}_{n,b}) = \text{argmin} \xi \in \mathbb{R}, m \in \mathcal{G} \sum_{i=1}^{n} \Big(g_{ib} - \xi m(X_i)\Big)^2$$

(c) Chose $\alpha_b \in \mathbb{R}$ by solving the minimization problem

$$\alpha_b = \text{argmin}_{\alpha \in \mathbb{R}} \hat{R}_n(\hat{m}_n^{(b-1)}(X_i) + \alpha \tilde{m}_{n,b}(X_i))$$

(d) Define $\hat{m}_{n,b} = \alpha_b \tilde{m}_{n,b}$ and $\hat{m}_n^{(b)} = \hat{m}_n^{(b-1)} + \eta \hat{m}_{n,b}$.

Let us go through the steps of the algorithm one-by-one. Step (1) is an initialisation step, where we start by using the best minimizer $m \in \mathcal{G}$ of the empirical risk as our initial estimator. Remember, that we assume $\mathcal{G}$ to be a very restrictive class of functions, so the estimator will most likely not do a very good job at predicting Y from X by itself.

The algorithm then proceeds to step (2), where we repeat a sequence of steps where each repetition adds one further weak learner to the estimator. The idea is that each weak learner should be chosen such that it moves the estimator in the direction that minimizes the empirical loss. This direction is exactly the negative gradient $(g_{1b}, \dots, g_{nb})^T$, which we calculate in step (a).

Ideally, we would like to have our b'th weak learner to move the estimator in the direction of the gradient, but since the weak learner $\hat{m}_{n,b} \in \mathcal{G}$, we cannot exactly move in the direction of the gradient. So instead we need to find the function $m \in \mathcal{G}$ such that $(m(X_1), \dots, m(X_n))^T$ resembles the gradient as much as possible. This is what we do in step (b). The object we minimize to find the correct m is the euclidean

distance between the gradient and a optimally scaled version of m. If $\mathcal{G}$ is stable under scaling, e.g. $m \in \mathcal{G} \Rightarrow \xi \cdot m \in \mathcal{G}$, then adding the ξ to the problem is unnecessary.

Step (c) is also known as line search. We have decided on the form $\tilde{m}_{n,b}$ of the weak learner we wish to add to the estimator because it is closest to the right direction as indicated by the gradient. We know need to establish how far we want to go in that direction. This is controlled via the parameter α and is chosen such that the empirical risk is minimized.

For step (d), we decide on our final form of the weak learner $\hat{m}_{n,b} = \alpha_b \cdot \tilde{m}_{n,b}$ and add a η-scaled version of the weak learner to the previous estimator $\hat{m}_n^{(b-1)}$ to obtain our updated estimator $\hat{m}_n^{(b)}$. The scaling is performed here to ensure that the estimator does not change too much in one step, since this may lead to divergence for the estimator. The entire process is repeated B times step (2) to obtain our final gradient boosting estimator.

While the hyperparameters η and B are rather simple in the sense that they are just numbers with clearly defined roles in the boosting algorithm, the function space $\mathcal{G}$ is somewhat more complicated. It is common to use $\mathcal{G}_{d_{\text{Max}}} = \{\hat{m}_T \mid T \text{ is a tree of depth } d_{\text{Max}}\}$, which correspond roughly to piecewise constant functions that take up to $2^{d_{\text{Max}}}$ different values. This is useful, since we have the CART algorithm for determining an approximate solution to the minimization problem of step (b). Furthermore, we can modify the step (c) slightly to take advantage of that we can decide the value of the tree estimator at each leaf rather than scaling them all together.

Gradient Boosting Machines v1 (With Trees) Friedman (2001)

Hyperparameters:

d_{Max} - Integer. Maximal depth allowed for weak learner trees.

η - Learning rate. Controls the rate of learning. Higher learning rates may lead to instability. We require $0 < \eta \leq 1$.

B - Number of iterations.

1) Set

$$\hat{m}_n^{(0)}(x) = \text{argmin}_{m \in \mathcal{G}} \hat{R}_n(m)$$

2) For $b = 1, \dots, B$, do:

(a) For each observation $i = 1, \dots, n$, do:

i. Calculate the (negative) gradients

$$g_{ib} = -\frac{\partial L(Y_i, y)}{\partial y}\bigg|_{y=\hat{m}_n^{(b-1)}(X_i)}$$

(b) Fit the tree T by using the CART(d_{Max}) algorithm on $(X_i, g_{ib})_{i=1,\dots,n}$ where the squared loss is used for splitting nodes.

(c) The estimator based on this tree is

$$\hat{m}_{n,b}(x) = \sum_{A \in \mathcal{A}_T} \mathbb{1}\{x \in A\} \nu_A$$

Where

$$\nu_A = \text{argmin}_{\nu \in \mathbb{R}} \sum_{i: X_i \in A} L(Y_i, \hat{m}_n^{(b-1)}(X_i) + \nu)$$

is chosen such that the risk arising from the leaf A is minimal.

(d) Define $\hat{m}_n^{(b)} = \hat{m}_n^{(b-1)} + \eta \hat{m}_{n,b}$.

In the following, we will consider two extensions to the gradient boosting algorithm, known as xgboost algorithmChen and Guestrin (2016). The first of these is the introduction of a penalty term to empirical risk used for determining the optimal split in the CART algorithm. That is, we replace $\hat{R}_n$ (by some abuse of notation))with

$$\hat{R}_{n,\gamma,\lambda}(\hat{m} + m_T) = \hat{R}_n(\hat{m} + m_T) + J_{\gamma,\lambda}(m_T) \quad , \quad J_{\gamma,\lambda}(m_T) = \gamma|\mathcal{A}_T| + \frac{\lambda}{2n}\sum_{A\in\mathcal{A}_T}\nu_A^2.$$

Essentially, the penalization simply ensures that the tree model is allowed less flexibility compared to an non-penalized tree model.

The second extension to gradient boosting we consider is a replacement of the step (b) with a method based on expanding a second order Taylor expansion of loss function. We have that

$$\begin{aligned}
&\hat{R}_{n,\gamma,\lambda}\Big(\hat{m}_n^{(b-1)} + \hat{m}_{n,b}\Big) \\
&\quad= \frac{1}{n}\sum_{i=1}^n L(Y_i, \hat{m}_n^{(b-1)}(X_i) + \hat{m}_{n,b}(X_i)) + \gamma|\mathcal{A}_T| + \frac{\lambda}{2n}\sum_{A\in\mathcal{A}_T}\nu_A^2 \\
&\quad\approx \frac{1}{n}\sum_{i=1}^n \left[L\Big(Y_i, \hat{m}_n^{(b-1)}(X_i)\Big) + \left(\frac{\partial L(Y_i,m)}{\partial m}\Big|_{m=\hat{m}_n^{(b-1)}(X_i)}\right)\cdot \hat{m}_{n,b}(X_i)\right. \\
&\qquad\qquad \left. + \frac{1}{2}\left(\frac{\partial^2 L(Y_i,m)}{\partial m^2}\Big|_{m=\hat{m}_n^{(b-1)}(X_i)}\right)\cdot \hat{m}_{n,b}(X_i)^2\right] + \gamma|\mathcal{A}_T| + \frac{\lambda}{2n}\sum_{A\in\mathcal{A}_T}\nu_A^2 \\
&\quad:= \frac{1}{n}\sum_{i=1}^n \left[L\Big(Y_i, \hat{m}_n^{(b-1)}(X_i)\Big) + g_i\cdot\hat{m}_{n,b}(X_i) + \frac{1}{2}h_i\cdot\hat{m}_{n,b}(X_i)^2\right] + \gamma|\mathcal{A}_T| + \frac{\lambda}{2n}\sum_{A\in\mathcal{A}_T}\nu_A^2
\end{aligned}$$

Note that we can write the sum $\sum_{i=1}^n = \sum_{A\in\mathcal{A}_T}\sum_{i:X_i\in A}$ such that we sum over the i'th where X_i falls within a given leaf. Using this, we find that

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^n \left[L\Big(Y_i, \hat{m}_n^{(b-1)}(X_i)\Big) + g_i\cdot\hat{m}_{n,b}(X_i) + \frac{1}{2}h_i\cdot\hat{m}_{n,b}(X_i)^2\right] + \gamma|\mathcal{A}_T| + \frac{\lambda}{2n}\sum_{A\in\mathcal{A}_T}\nu_A^2 \\
&= \frac{1}{n}\sum_{A\in\mathcal{A}_T}\sum_{i:X_i\in A}\left[L\Big(Y_i, \hat{m}_n^{(b-1)}(X_i)\Big) + g_i\cdot\hat{m}_{n,b}(X_i) + \frac{1}{2}h_i\cdot\hat{m}_{n,b}(X_i)^2\right] + \gamma|\mathcal{A}_T| + \frac{\lambda}{2n}\sum_{A\in\mathcal{A}_T}\nu_A^2 \\
&= \sum_{A\in\mathcal{A}_T}\left(\sum_{i:X_i\in A}\left[\frac{1}{n}L\Big(Y_i, \hat{m}_n^{(b-1)}(X_i)\Big) + \frac{g_i}{n}\cdot\nu_A + \frac{1}{2}\frac{h_i}{n}\cdot\nu_A^2\right] + \gamma + \frac{\lambda}{2n}\nu_A^2\right)
\end{aligned}$$

Since each term of the outer sum only depends on one ν_A and we are free to chose the value of the leaf as we prefer, this can be minimized without considering the other terms of the outer sum. Hence we find by differentiating with respect to ν_A and setting the derivative equal to zero that

$$\left(\sum_{i:X_i\in A}\frac{g_i}{n} + \frac{h_i}{n}\cdot\nu_A\right) + \frac{\lambda}{n}\nu_A = 0 \Leftrightarrow \nu_A = -\frac{\sum_{i:X_i\in A} g_i}{\lambda + \sum_{i:X_i\in A} h_i}$$

Inserting it back into the expression for the empirical risk yields

$$\begin{aligned}
&\sum_{A\in\mathcal{A}_T}\left(\gamma+\frac{\lambda}{2n}\nu_A^2+\sum_{i:X_i\in A}\left[\frac{1}{n}L\Big(Y_i,\hat{m}_n^{(b-1)}(X_i)\Big)+\frac{g_i}{n}\cdot\nu_A+\frac{1}{2}\frac{h_i}{n}\cdot\nu_A^2\right]\right)\\
&=\hat{R}_n(\hat{m}_n^{(b-1)})+\sum_{A\in\mathcal{A}_T}\left(\gamma+\left(\sum_{i:X_i\in A}\frac{g_i}{n}\right)\nu_A+\frac{1}{2}\left(\lambda+\sum_{i:X_i\in A}\frac{h_i}{n}\right)\nu_A^2\right)\\
&=\hat{R}_n(\hat{m}_n^{(b-1)})+\sum_{A\in\mathcal{A}_T}\left(\gamma-\frac{1}{n}\frac{\left(\sum_{i:X_i\in A}g_i\right)^2}{\lambda+\sum_{i:X_i\in A}h_i}+\frac{1}{2n}\frac{\left(\sum_{i:X_i\in A}g_i\right)^2}{\lambda+\sum_{i:X_i\in A}h_i}\right)\\
&=\hat{R}_n(\hat{m}_n^{(b-1)})+\gamma|\mathcal{A}_T|-\frac{1}{2n}\sum_{A\in\mathcal{A}_T}\left(\frac{\left(\sum_{i:X_i\in A}g_i\right)^2}{\lambda+\sum_{i:X_i\in A}h_i}\right)
\end{aligned}$$

The particularly nice part about this expression is that it yields a workable expression for the reduction in the empirical risk we obtain by adding the optimal weak learner $\hat{m}_{n,b}$ based on the tree T to the estimator $\hat{m}_n^{(b)}$. E.g. if T' is obtained by splitting the node A from T into A_1 and A_2, then

$$\begin{aligned}
&\hat{R}_{n,\gamma,\lambda}(\hat{m}_n^{(b-1)}+\hat{m}_{n,b,T})-\hat{R}_{n,\gamma,\lambda}(\hat{m}_n^{(b-1)}+\hat{m}_{n,b,T'})\\
&\approx\gamma|\mathcal{A}_T|-\frac{1}{2n}\sum_{A\in\mathcal{A}_T}\left(\gamma-\frac{\left(\sum_{i:X_i\in A}g_i\right)^2}{\lambda+\sum_{i:X_i\in A}h_i}\right)-\gamma|\mathcal{A}_{T'}|+\frac{1}{2n}\sum_{A\in\mathcal{A}_{T'}}\left(\gamma-\frac{\left(\sum_{i:X_i\in A}g_i\right)^2}{\lambda+\sum_{i:X_i\in A}h_i}\right)\\
&=\frac{1}{2n}\left\{\frac{\left(\sum_{i:X_i\in A_1}g_i\right)^2}{\lambda+\sum_{i:X_i\in A_1}h_i}+\frac{\left(\sum_{i:X_i\in A_2}g_i\right)^2}{\lambda+\sum_{i:X_i\in A_2}h_i}-\frac{\left(\sum_{i:X_i\in A}g_i\right)^2}{\lambda+\sum_{i:X_i\in A}h_i}-\right\}-\gamma
\end{aligned}$$

We can compare this improvement to other split options and if splitting into A_1, A_2 turns out to be the best option and the improvement is positive, then the split option is taken.

Remark. Gradient boosting machines are known to often provide the strongest predictive performance. `xgboost`, `lightgbm` `catboost` are three popular implementations. They are quite fast, but not as fast as random forests and are also more reliant on optimal parameter tuning. The most relevant parameters are tree depth number of trees and learning rate. Why are gradient boosting machines so powerful? A contributing factor may be that a small max depth restricts the number of interactions fitted. E.g. a maximal depth of 1 corresponds to an additive model. Gradient boosting machines, in contrast to random forests have also less of a problem to fit additive functions.

Chapter 6

Neural Networks

In this chapter, we provide an introduction to neural networks. These are arguably responsible for some 'recent' large advances in the fields of image recognition and natural language processing. The class of models that classify as neural networks is somewhat broad, but since we are interested in tabular data, we will focus our attention to the 'classical' feed-forward networks.

6.1 Motivation

To build up to the idea of a neural network, we will start with a linear model. For this model, the true relation between features X and response Y may be assumed to be

$$Y = \beta^T X + \varepsilon \quad , \quad X \in \mathbb{R}^p, \quad \beta \in \mathbb{R}^p, \quad Y, \in \mathbb{R}, \quad \varepsilon \in \mathbb{R}.$$

We can generalize this situation in the sense that we say

$$Y = BX + \varepsilon \quad , \quad X \in \mathbb{R}^p, \quad B \in \mathbb{R}^{q \times p}, \quad Y \in \mathbb{R}^q, \quad \varepsilon \in \mathbb{R}^p.$$

This really just amounts to that instead of having to predict a one-dimensional Y from X, we instead need to predict a q-dimensional Y from X. And to do this, we simply stack q separate linear models on top of each other. Considering the above expression row-by-row, it simply becomes the ordinary linear model.

The issue with the linear model is that data may be described accurately with a linear decision function. The simplest way to introduce non-linearity would arguably be by introducing some non-linear transformation $f : \mathbb{R}^q \to \mathbb{R}^q$ such that

$$Y = f(BX) + \varepsilon$$

This clearly extends our class of models, since setting f to be the identity will let us have the linear model back. However, the extended model may still not be general enough to capture the true relation between X and Y. However, we may have that $f(BX) \approx Y$ is a better approximation compared to $BX \approx Y$. This leads us to believe that repeating the process

$$X \mapsto f^1(B_1 X) \mapsto f^2(B_2 f^1(B_1 X)) \mapsto \cdots$$

may lead to better and better approximations of Y if sequence matrices $B_1, B_2, \ldots$ and sequence of functions $f^1, f^2, \ldots$ are chosen suitably.

This idea of applying a sequence of alternating linear and non-linear transformations to obtain an estimate is exactly the key behind a feed-forward neural network.

Definition 6.1.1. Let $\ell \in \mathbb{N}$, let $(d_j)_{j=0,\ldots,\ell} \in \mathbb{N}^{\ell+1}$ with $d_0 = p$ and $d_\ell = \dim Y$. Let $B_j \in \mathbb{R}^{d_j \times d_{j-1}}$

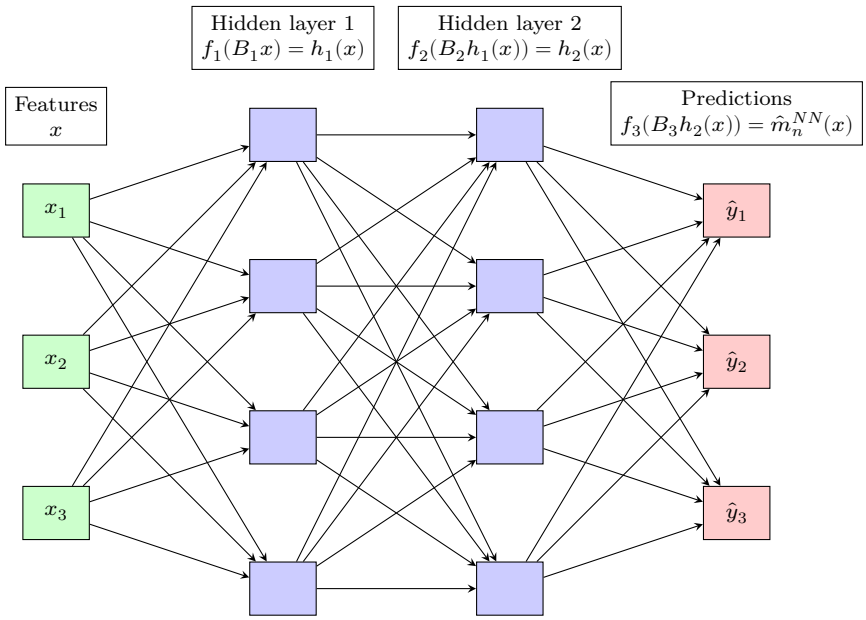

Figure 6.1: Visual representation of a neural network. The arrows represent the linear contributions that arise from matrix multiplication. Each blue box is a coordinate in an intermediate prediction vector, obtained by applying an activation function to the results from the matrix multiplication represented by the incoming arrows.

and $f_h : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ for all $j \in \{1, \dots, \ell\}$. Writing

$$\tilde{B}_j : \mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_j} \quad , \quad \tilde{B}_h(x) = f_j(B_h x),$$

we say that the corresponding neural network is

$$\hat{m}_n^{NN}(x) = \tilde{B}_\ell \circ \dots \circ \tilde{B}_1(x).$$

We say that ℓ is the depth of the network and we refer to the functions $f_1, \dots, f_\ell$ as activation functions. The functions

$$h_j(x) = \tilde{B}_j \circ \dots \circ \tilde{B}_1(x), \quad j = 1, \dots, \ell$$

are called hidden layers and d_j is the width of layer j.

Remark. The choice of activation functions is usually to have each activation function be the same and of the form

$$f_j(x) = (g(x_1), \dots, g(x_{d_j}))^T$$

Typical choices of the function $g : \mathbb{R} \to \mathbb{R}$ are

- Sigmoid - $g(x) = \frac{e^x}{1+e^x}$
- Hyperbolic tangent - $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Rectified linear unit (ReLU) - $g(x) = x^+ = \max\{0, x\}$
- Leaky ReLU - $g(x) = x^+ + \alpha(-x)^+$ for $\alpha \in (0, 1)$.

The essential part is that activation functions are non-linear, since linear activation functions would cause neural network to be entirely comprised of linear transformations, which would thus result in the neural network itself being linear. And thus we would be back to the linear model.

For some choice of depth and activation functions $f_1, \dots, f_\ell$, a neural network is parameterized by the choice of weight matrices $B_1, \dots, B_\ell$. For a random assortment of weights, the decision function is really

just some arbitrary function $\hat{m}^{NN} : \mathbb{R}^p \to \mathbb{R}^{d_\ell}$, but the idea is that we can tune the weight in such a way that $Y \approx \hat{m}_x^{NN}(X)$.
Before we discuss how to tune the weights of a neural network, we should discuss what progress we have made. We motivated the introduction of neural networks by pointing out that a linear model may be too restrictive, e.g. it would lead to a large approximation error.

The question to ask for neural networks in then what class of functions can be described by them. It turns out that the answer is very uplifting - There are a number of "Universal Approximation Theorem's" available for neural networks, saying that neural networks can approximate just about anything if the hidden layers are allowed to be arbitrarily high-dimensional.

This does not mean that neural networks with any fixed architecture will approximate well or that the solution to all problems with neural networks is to increase the dimensionality of the hidden layers. The universal approximation theorem(s) merely state(s) that neural networks can be arbitrarily good function approximates, not that they will be or that they can do it in a computationally feasible way.

6.2 Training a Neural Network

The issue we face when training a neural network is that we wish to minimize the loss on training data. That is, we wish to solve the minimization problem

$$\text{argmin}_{B_1,\dots,B_\ell} \frac{1}{n} \left(\sum_{i=1}^{n} L(Y_i, \hat{m}^{NN}(X_i)) \right).$$

We will assume $f(x) = (g(x_1), \dots, g(x_k))^T$ and introduce the following notation for convenience

$$B_j = \begin{pmatrix} \beta_{j,1}^T \\ \vdots \\ \beta_{j,d_j}^T \end{pmatrix}, \quad f_k'(x) = \text{diag}(g'(x_1), \dots, g'(x_{d_k})).$$

If the loss function and activation functions are chosen suitably, the expression we wish to minimize is differentiable with respect to each $\beta_{j,k}$, and hence we can apply gradient descent to find the minimum. Furthermore, since the decision function produced by the neural network is really just a composition of activation functions and linear transformations, the derivative with respect to some $\beta = \beta_{j,k}$ is calculated using the chain rule:

$$D_{\beta_{j,k}} L(Y, h_\ell) = D_{m_\ell} L(Y, h_\ell) \cdot D_{\beta_{j,k}} h_\ell$$

as well as

$$\begin{aligned}
D_{\beta_{j,k}} h_\ell &= D_{\beta_{j,k}} f_\ell(B_\ell h_{\ell-1}) \\
&= (f_\ell'(B_\ell h_{\ell-1}) \cdot B_\ell) \cdot D_{\beta_{j,k}} h_{\ell-1} \\
&= (f_\ell'(B_\ell h_{\ell-1}) \cdot B_\ell) \cdot (f_{\ell-1}'(B_{\ell-1} h_{\ell-2}) \cdot B_{\ell-1}) \cdot D_{\beta_{j,k}} h_{\ell-2} \\
&= \cdots \\
&= \left(\prod_{k=j+1}^{\ell} f_k'(B_k h_{k-1}) \cdot B_k \right) \cdot f_j'(B_j h_{j-1}) \cdot D_{\beta_{j,k}} B_j h_{j-1} \\
&= \left(\prod_{k=j+1}^{\ell} f_k'(B_k h_{k-1}) \cdot B_k \right) \cdot f_j'(B_j h_{j-1}) \cdot \left\{ \begin{pmatrix} \mathbf{0}^T \\ \vdots \\ \mathbf{0}^T \\ h_{j-1}^T \\ \mathbf{0}^T \\ \vdots \\ \mathbf{0}^T \end{pmatrix} \right\} h_{j-1}^T \text{ at row } k \text{ (out of } d_j).
\end{aligned}$$

Note that the gradient for $\beta_{j,k}$ (apart of the weights B_k) depends on $D_{h_\ell}L(Y, h_\ell)$ and on the objects $(f_k'(B_k h_{k-1}))_{k=j,\dots,\ell}$. If one has already calculated the gradients for the later weights $B_{j'}$, for all $j' > j$, then $D_{h_\ell}L(Y, h_\ell)$ and $(f_k'(B_k h_{k-1}))_{k=j+1,\dots,\ell}$ are already calculated and don't need to be calculated again. This strategy is known as backpropagation. Having all gradients at hand (calculated for each data point), we can update the weights. In step r and given a learning rate γ the weights are updated as

$$\beta_{j,k}^{(r+1)} = \beta_{j,k}^{(r)} - \gamma \frac{1}{n} \sum_{i=1}^{n} D_{\beta_{j,k}} L(Y_i, h_\ell^{(r)}(X_i)).$$

6.3 Regularisation and Speed-Up Methods

While the backpropagation algorithm is quite reasonable, it does not prevent overfitting and may be very computationally intensive. There are many ways to modify the algorithm, but we present a few options here.

To reduce the computational costs of backpropagation, one can switch the type of optimization technique from gradient descend to stochastic gradient descend. The motivation for this is that for large data sets, it is expensive to calculate the gradients as this computation has to be performed for each data point. To avoid this, stochastic gradient descend randomly partitions the data into batches and applies gradiient descent consecutively on each batch.

Stochastic Gradient Descent (SGD)

Hyperparameters:

1. `batch-size` (integer valued)
2. early stopping criteria (most often whether improvement is better than some treshold on the validation set)

---

1) For $j = 1, \dots$ (=epochs) do:
   (a) For $k = 1, \dots, n/$`batch-size` do:
      i. Sample without replacement `batch-size` from $\{1, \dots, n\} \setminus \cup_{l=1}^{k-1} S_l^j$, resulting in the kth mini-batch, S_k^j.
      ii. Perform gradient descent using only observations $i \in S_k^j$.
   (b) Check improvement of the jth epoch on some validation set. If improvement is not big enough: STOP.

For preventing overfitting, one usually stops the algorithm when performance on a separate validation set does not improve significantly between iterations. Additioanlly, one can introduce a penalty on the coefficients. This corresponds to changing the minimization of interest problem to

$$\text{argmin}_{B_1,\dots,B_\ell} \frac{1}{n} \left(\sum_{i=1}^{n} L(Y_i, \hat{m}^{NN}(X_i)) \right) + J_\lambda(B_1, B_2, \dots, B_\ell)$$

The type of penalty that is commonly used is either ridge or lasso, corresponding to

$$J_\lambda^{LASSO} = \lambda \sum_{j=1}^{\ell} ||B_j||_1 \quad , \quad J_\lambda^{Ridge} = \lambda \sum_{j=1}^{\ell} ||B_j||_2^2$$

Here, the norms are usually applied entry-wise, but matrix norms would clearly also work. In the machine learning community lasso and ridge penalization is also known as weight decay.
Another popular option for regularization is dropout, introduced in Srivastava et al. (2014) It is an analogue to random forest, but the obvious idea of averaging the outputs of many separately trained nets is prohibitively expensive.

Stochastic Gradient Descent (SGD) with Dropout Learning

Hyperparameters:

1. `batch-size` (integer valued)
2. early stopping criteria (most often whether improvement is better than some treshold on the validation set)
3. Drop-out probability p

---

1) For $j = 1, ...$ (=epochs) do:
   (a) For $k = 1, ..., n/$`batch-size` do:
      i. Sample without replacement `batch-size` from $\{1, ..., n\} \setminus \cup_{l=1}^{k-1} S_l^j$, resulting in the kth mini-batch, S_k^j.
      ii. Randomly remove a fraction of the units in a layer, i.e for every layer r, unit $s = 1, ..., d_s$, and $i \in S_k^j$

$$\tilde{h}_{r,s}(j,k,i) = \begin{cases} 0 & \text{with probability } p \\ h_{r,s}(j,k,i)/(1-p) & \text{else} \end{cases}$$

      iii. Perform gradient descent using observations $i \in S_k^j$ where for each observation the gradient is calculated based on the network $(\tilde{h}_r(j,k,i))_{r=1,...,\ell}$.
   (b) Check improvement of the jth epoch on some validation set. If improvement is not big enough: STOP.

Note that the scaling factor in drop-out is chosen such that $\mathbb{E}_p[\tilde{h}(j,k,i)] = h(j,k,i)$. The heuristic of drop-out is that many thinned networks are trained in parallel. Lastly, it should be noted that there are many alternatives to vanilla gradient descent, e.g. momentum

$$\beta_{j,k}^{(r+1)} = \beta_{j,k}^{(r)} + \alpha(\beta_{j,k}^{(r)} - \beta_{j,k}^{(r-1)}) - \gamma \frac{1}{n} \sum_{i=1}^{n} D_{\beta_{j,k}} L(Y_i, h_\ell^{(r)}(X_i)),$$

which we however do not cover in this lecture.

Chapter 7

Post hoc explanations

For this chapter, we will consider methods that aim to expose the workings of trained machine learning models. These methods are relevant, because machine learning models can get very hard to interpret. The primary issue this brings around revolves around model trust. If we cannot understand, why the model makes the predictions that it makes, why should we trust them? This is critical, if the model is deployed in some high-states setting - Examples are medical diagnosis, legal judgement, loan approval, fraud detection or insurance pricing.

A useful distinction to know about in the context of explainability is that of global explanations versus local explanations. Global explanations focus on the behaviour of the model at scale, thus answering questions such as: "What is the effect of this feature?". Local explanations focus on individual predictions, thus answering the question of: "Why did the model predict this value?".

7.1 Shapley Values

We consider some trained model having produced a decision function $\hat{m}$ that we are interested in understanding. Note initially that we can represent

$$\hat{m}_n(x) = \phi_0 + \sum_{j=1}^{p} \phi_j(x)$$

with many different functions $\phi_j : \mathbb{R}^p \to \mathbb{R}$. For a fixed $x = x_0$, such a decomposition provides a local explanation, since we may interpret $\phi_j(x_0)$ as the contribution of the j'th feature to the prediction $\hat{m}_n(x_0)$.

Example 15. A possible decomposition is $\phi_1(x) = \hat{m}(x)$ and $\phi_j(x) = 0$ for $j \neq 1$. While this makes sense mathematically, it is probably nonsense from a heuristic point of view, since our interpretation of the decomposition is that the feature X_1 is responsible for the value of $\hat{m}(X)$ entirely.

There are many more such nonsense decompositions, for example $\phi_0 = 1, \phi_1(x) = \frac{1}{2}\hat{m}(x) - 1, \phi_2(x) = \frac{1}{2}\hat{m}_n(x)$ and $\phi_j(x) = 0$ for $j \geq 3$. The reason that these explanations are nonsense is that they do not reflect the $\hat{m}_n$ actually depends on its arguments. To get sensible explanations, we should find some way to extract this information.

Fix $x_0 \in \mathbb{R}^p$. To obtain meaningful explanations, it is useful to consider some set function ν_{x_0} defined on the power set of $I_p = \{1, \dots, p\}$. Our interpretation of $\nu_{x_0}(S)$ with $S \subseteq I_p$ is that ν measures the contribution of $\{X_j | j \in S\}$ to the prediction $\hat{m}_n(x_0)$.

Definition 7.1.1 (Shapley values)**.** Assume ware given a point $x_0 \in \mathbb{R}^p$ and a value function $\nu_{x_0} : \mathcal{P}(I_p) \to \mathbb{R}$ with $v(I_p) = \hat{m}_n(x)$. We say that a decomposition $\phi_0, \dots, \phi_p$ satisfies the Shapley axioms with respect to ν if

S1) **(Efficiency)**
$$\phi_0(x_0) = \nu_{x_0}(\emptyset), \qquad \sum_{j=0}^{p} \phi_j = \nu_{x_0}(I_p)$$

S2) **(Symmetry)**: Fix any $k, l \in I_P, k \neq l$.
If $v_{x_0}(S \cup k) = v_{x_0}(S \cup l)$, for all $S \subseteq I_p \setminus \{k, l\}$, then $\phi_k(x_0) = \phi_l(x_0)$.

S3) **(Dummy)**: For all $k \in I_p$:
If $v_{x_0}(S \cup k) = v_{x_0}(S)$, for all $S \subseteq I_p \setminus \{k\}$, then $\phi_k(x_0) = 0$.

S4) **(Linearity)** Assume that $\nu^1_{x_0}, \nu^2_{x_0} : I_p \to \mathbb{R}$ with $\nu_{x_0} = \nu^1_{x_0}, +\nu^2_{x_0}$, are set functions and $(\phi_0^{\nu_1}(x_0), \dots, \phi_p^{\nu_1}(x_0)$ resp. $(\phi_0^{\nu_2}(x_0) \dots, \phi_p^{\nu_2}(x_0))$ are decompositions satisfying the Shapley axioms with respect to ν^1 resp. ν^2. Then $\phi_k(x_0) = \phi_k^{\nu_1}(x_0) + \phi_k^{\nu_2}(x_0)$ for all $k \in I_p$.

We should think of ν_{x_0} as complex object that correctly encodes the information about how relevant each set of features is for the prediction $\hat{m}_n(x_0)$. However, since ν_{x_0} is complex, we need to translate to a decomposition $\phi_0(x_0), \dots, \phi_p(x_0)$, which is easier to understand. However, we would like this decomposition to remain faithful to ν_{x_0}, and by that we mean that it should satisfy the Shapley axioms.

The natural question to ask is whether there exists a decomposition, which satisfies these axioms, and in that case whether or not it is unique. It turns out that the answer to both questions is 'yes'. In the following we suppress the dependence on a point x_0.

Theorem 7.1.2. Let $\nu : \mathcal{P}(I_p) \to \mathbb{R}$ be an arbitrary set function. Then there exists a unique decomposition $\phi_0, \dots, \phi_p$ such that this decomposition satisfies the Shapley axioms. The decomposition is given by

$$\begin{aligned}\phi_k &= \frac{1}{p!} \sum_{\pi \in \Pi_p} \Delta_\nu(k, \{\pi(1), \dots, \pi(\pi^{-1}(k) - 1)\}) \\ &= \frac{1}{p!} \sum_{S \subseteq I_p \setminus \{k\}} |S|!(p - |S| - 1)! \Delta_\nu(k, S),\end{aligned}$$

where $\Delta_\nu(k, S) = \nu(S \cup \{k\}) - \nu(S)$ and Π_p is the set of permutations over the elements of I_p.

Proof. The proof consists of three major steps. These are:

1) Start by establishing a representation of an arbitrary $\nu : I_p \to \mathbb{R}$ in terms of the basis $\{\nu_T \mid T \subseteq I_p\}$ with $\nu_T(S) = \mathbb{1}\{T \subseteq S\}$.
2) Find the only possible Shapeye decomposition for ν_T.
3) Use linearity to translate this into the (only possible) shapely decomposition for ν.
4) Verify that ϕ_k satisfies S1)– S4).

We will omit details for step 4, as S1)– S4) are quickly verified starting from the definition of ϕ_k.

Step 1:
We wish to find a representation $\nu = \sum_{T \subseteq I_p} \alpha_T \nu_T$ where $\forall T \subseteq I_p : \alpha_T \in \mathbb{R}$. Fix an arbitrary set $U \subseteq I_p$. Then trivially

$$\nu(U) = \sum_{S \subseteq U} \mathbb{1}\{S = U\} \nu(S)$$

Now, using that $S \subseteq U$ for the first equality below and (Lemma A.0.1) for the second, we have

$$\mathbb{1}\{S = U\} = \mathbb{1}\{U \setminus S = \emptyset\} = \sum_{D \subseteq U \setminus S} (-1)^{|D|}$$

Inserting this back into the above expression, we obtain

$$\begin{aligned}\nu(U) &= \sum_{S\subseteq U} \mathbb{1}\{S = U\}\nu(S) \\ &= \sum_{S\subseteq U}\left(\sum_{D\subseteq U\setminus S}(-1)^{|D|}\nu(S)\right) \\ &= \sum_{S\subseteq U}\left(\sum_{T\subseteq U: S\subseteq T}(-1)^{|T\setminus S|}\nu(S)\right)\end{aligned}$$

The last equality is obtained from a change-of-index such that we sum over the T such that $S \subseteq T \subseteq U$ with $D = T \setminus S$. Now, switching the order of the finite summation yields

$$\begin{aligned}\sum_{S\subseteq U}\left(\sum_{T\subseteq U: S\subseteq T}(-1)^{|T\setminus S|}\nu(S)\right) &= \sum_{T\subseteq U}\left(\sum_{S\subseteq T}(-1)^{|T\setminus S|}\nu(S)\right) \\ &= \sum_{T\subseteq I_p}\left(\sum_{S\subseteq T}(-1)^{|T\setminus S|}\nu(S)\right)\nu_T(U) \\ &:= \sum_{T\subseteq I_p}\alpha_T\nu_T(U)\end{aligned}$$

As desired.

Step 2:
Note initially for $k \in I_p \setminus T$, we have that

$$\nu_T(S\cup\{k\}) = \mathbb{1}\{T\subseteq S\cup\{k\}\} = \mathbb{1}\{T\subseteq S\} = \nu_T(S)$$

Since adding k to the set S cannot change if $T \subseteq S$ when $k \notin T$. Hence the dummy-axiom (S3) implies that $\phi_k(\nu_T) = 0$.

Now, consider $k, l \in T$ with $k \neq l$. Take some $S \subseteq I_p \setminus \{k, l\}$. Noting that $k \notin S \cup \{l\}$ and $l \notin S \cup \{k\}$, it follows that $T \not\subseteq S \cup \{k\}$ and $T \not\subseteq S \cup \{l\}$. Hence we have

$$\nu_T(S\cup\{k\}) = \mathbb{1}\{T\subseteq S\cup\{k\}\} = 0 = \mathbb{1}\{T\subseteq S\cup\{l\}\} = \nu_T(S\cup\{l\})$$

By the symmetry axiom (S2), this implies that $\phi_k(\nu_T) = \phi_l(\nu_T)$. This shows that $\phi_k(\nu_T)$ is invariant of k for $k \in T$ and $\phi_k(\nu_T) = 0$ for $k \notin T$. Using the efficiency axiom (S1), we have that for $k \in T$

$$|T|\phi_k(\nu_T) = \sum_{j\in T}\phi_j(\nu_T) = \sum_{j=1}^{p}\phi_j(\nu_T) = \nu_T(I_p) - \phi_0(\nu_T) = \mathbb{1}\{T\neq\emptyset\}$$

Thus implying for $k \in I_p$ that $\phi_k(\nu_T) = \frac{\mathbb{1}\{k\in T\}}{|T|}$ is the only possible Shapely decomposition with respect to ν_T.

⚠ The case $T = \emptyset$ is a little different from the rest, but I believe that nothing significant changes, the notation simply gets more involved if both cases should be considered together. Hence we will politely ignore $\emptyset$ in the following calculations, knowing that this is formally an issue.

Step 3:

Using the linearity axiom for $\nu = \sum_{T \subseteq I_p} \alpha_T \nu_T$, we find that

$$\begin{aligned}
\phi_k(\nu) &= \sum_{T \subseteq I_p} \alpha_T \phi_k(\nu_T) \\
&= \sum_{T \subseteq I_p} \alpha_T \frac{\mathbb{1}\{k \in T\}}{|T|} \\
&= \sum_{T \subseteq I_p} \sum_{S \subseteq T} \left((-1)^{|T \setminus S|} \nu(S) \frac{\mathbb{1}\{k \in T\}}{|T|} \right) \\
&= \sum_{S \subseteq I_p} \sum_{T \subseteq I_p : S \subseteq T} \left((-1)^{|T \setminus S|} \nu(S) \frac{\mathbb{1}\{k \in T\}}{|T|} \right) \\
&= \sum_{S \subseteq I_p} \nu(S) \left(\sum_{T \subseteq I_p : S \subseteq T} (-1)^{|T \setminus S|} \frac{\mathbb{1}\{k \in T\}}{|T|} \right) \\
&:= \sum_{S \subseteq I_p} \nu(S) \gamma_k(S)
\end{aligned}$$

At this point, we have a unique expression for the Shapley decomposition of ν. All that is left from here is simplifying the expression.

Start by observing that

$$\begin{aligned}
\gamma_k(S) &= \sum_{T \subseteq I_p : S \subseteq T} (-1)^{|T \setminus S|} \frac{\mathbb{1}\{k \in T\}}{|T|} \\
&= \sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \frac{(-1)^{|T \setminus S|}}{|T|}
\end{aligned}$$

Take S with $k \notin S$. Then

$$\begin{aligned}
\gamma_k(S) &= \sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \frac{(-1)^{|T \setminus S|}}{|T|} \\
&= \sum_{T \subseteq I_p : (S \cup \{k\}) \cup \{k\} \subseteq T} (-1) \cdot \frac{(-1)^{|T \setminus (S \cup \{k\})|}}{|T|} \\
&= -\gamma_k(S \cup \{k\})
\end{aligned}$$

Rewriting our expression for $\phi_k(\nu)$ to take advantage of this, we get that

$$\begin{aligned}
\phi_k(\nu) &= \sum_{S \subseteq I_p} \gamma_k(S) \nu(S) \\
&= \sum_{S \subseteq I_p \setminus \{k\}} \gamma_k(S \cup \{k\}) \nu(S \cup \{k\}) + \gamma_k(S) \nu(S) \\
&= \sum_{S \subseteq I_p \setminus \{k\}} -\gamma_k(S) \Big(\nu(S \cup \{k\}) - \nu(S) \Big) \\
&= \sum_{S \subseteq I_p \setminus \{k\}} -\gamma_k(S) \Delta_\nu(k, S)
\end{aligned}$$

To get further from this point, we will need an explicit expression for $\gamma_k(S)$. Note that for $k \notin S$

$$\begin{aligned}
\gamma_k(S) &= \sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \frac{(-1)^{|T \setminus S|}}{|T|} \\
&= \sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \left(\sum_{m=|S|+1}^{p} \mathbb{1}\{|T| = m\} \right) \frac{(-1)^{|T \setminus S|}}{|T|} \\
&= \sum_{m=|S|+1}^{p} \sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \mathbb{1}\{|T| = m\} \frac{(-1)^{|T \setminus S|}}{|T|} \\
&= \sum_{m=|S|+1}^{p} \sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \mathbb{1}\{|T| = m\} \frac{(-1)^{m-|S|}}{m} \\
&= \sum_{m=|S|+1}^{p} \frac{(-1)^{m-|S|}}{m} \left(\sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \mathbb{1}\{|T| = m\} \right)
\end{aligned}$$

Noting that the inner sum counts the subsets of I_p that are of size m and contain set set $S \cup \{k\}$ of size $|S| + 1$. Selecting such a subset corresponds to selecting $m - |S| - 1$ elements from $I_p \setminus S$, which can be done in

$$\left(\sum_{T \subseteq I_p : S \cup \{k\} \subseteq T} \mathbb{1}\{|T| = m\} \right) = \binom{p - |S| - 1}{m - |S| - 1}$$

different ways. Inserting this, we get

$$\begin{aligned}
\gamma_k(S) &= \sum_{m=|S|+1}^{p} \frac{(-1)^{m-|S|}}{m} \binom{p - |S|}{m - |S| - 1} \\
&= \sum_{m=0}^{p-|S|-1} \frac{(-1)^{m+1}}{m + |S| + 1} \binom{p - |S| - 1}{m}
\end{aligned}$$

To deal with the fraction denominator, note that $\int_0^1 x^{m+|S|} dx = \frac{1}{m+|S|+1}$. Inserting this, we get

$$\begin{aligned}
\gamma_k(S) &= \sum_{m=0}^{p-|S|-1} (-1)^{m+1} \int_0^1 x^{m+|S|} dx \binom{p - |S| - 1}{m} \\
&= -\int_0^1 x^{|S|} \sum_{m=0}^{p-|S|-1} \binom{p - |S| - 1}{m} (-x)^m dx
\end{aligned}$$

By the binomial theorem, the sum can be rewritten such that we get

$$\begin{aligned}
\gamma_k(S) &= -\int_0^1 x^{|S|} (1 - x)^{p-|S|-1} dx \\
&= -\beta(|S|, p - |S| - 1) \\
&= -\frac{|S|!(p - |S| - 1)!}{p!}
\end{aligned}$$

Where we used the well-known expression for the β-function:

$$\beta(a, b) = \int_0^1 x^a (1 - x)^b dx = \frac{a!b!}{(a + b + 1)!}$$

Inserting this into the expression for $\phi_k(\nu)$ yields

$$\phi_k(\nu) = \frac{1}{p!} \sum_{S \subseteq I_p \setminus \{k\}} |S|!(p - |S| - 1)! \Delta_\nu(k, S)$$

Which is the desired expression for the Shapley decomposition. □

Having established that Shapley values exist and are unique, it is relevant to discuss what value function ν should be used to generate them.

In the following, the notation X_S or x_S are shorthand for the entries of X resp. x at the indices contained in S, e.g. $X_{\{1,4,5\}} = (X_1, X_4, X_5)$. We will also slightly abuse notation by ignoring ordering in the input of the functions below. Lastly, we write $-S$ for $S^{\complement}$.

Definition 7.1.3. Fix a point $x \in \mathbb{R}^p$ and let $X \in \mathbb{R}^p$ be some random variable. We say that the observational SHAP value function corresponding to X and point x is

$$\nu(S) = \mathbb{E}_X[\hat{m}_n(X_S, X_{-S}) \mid X_S = x_S] = \int \hat{m}_n(x_1, \dots, x_p) p_{X_{-S}|X_S}(x_{-S}|x_S) \mathrm{d}x_{-S}$$

and the interventional SHAP value function corresponding to X is

$$\nu(S) = \mathbb{E}_X[\hat{m}_n(x_s, X_{-S})] = \int \hat{m}_n(x_1, \dots, x_p) p_{X_{-S}}(x_{-S}) \mathrm{d}x_{-S}$$

Note that these value functions explain the attribution of value to each feature X_j for $j = 1, \dots, p$ at the point $x = (x_1, \dots, x_p)$. The difference between observational and interventional SHAP is what distribution is used for X_{-S}. For observational SHAP, the distribution used incorporates the information that $X_S = x_S$, while for interventional SHAP, the distribution used for X_{-S} is the marginal distribution.

The naming is an artifact that is supposed to highlight the connection to the analog observational and interventional distributions from the field of causality.

While we present both observational and interventional SHAP, there are two good arguments for always using interventional SHAP.

The first is that observational SHAP is a measure of the expected model prediction given the value of some of the features. This means that if features are correlated, the information about the values of the features in S will give information about the features in $S^{\complement}$. Hence assigning the value that should rightfully be assigned to those in $S^{\complement}$ may leak to the value of S. Interventional SHAP avoids this issue by using the marginal distributions for the unobserved features, hence the information in $S^{\complement}$ does not 'leak' to S.

Example 16 (Janzing et al. (2020))**.** Assume

$$\hat{m}_n(x_1, x_2) = x_1$$

Let X_1, X_2 be binary with

$$p(x_1, x_2) = \begin{cases} \frac{1}{2} & \text{if } x_1 = x_2 \\ 0 & \text{else} \end{cases}$$

For observational SHAP, we have

- $v_x(\emptyset) = \mathbb{E}[\hat{m}_n(X_1, X_2)] = 0.5$
- $v_x(\{1\}) = \mathbb{E}[\hat{m}_n(X_1, X_2)|X_1 = x_1] = x_1$
- $v_x(\{2\}) = \mathbb{E}[\hat{m}_n(X_1, X_2)|X_2 = x_2] = x_2$
- $v_x(\{1, 2\}) = \hat{m}_n(x_1, x_2) = x_1$

Hence,

- $\Delta(2, \emptyset) = v_x(\{2\}) - v_x(\emptyset) = x_1 - 0.5$
- $\Delta(2, \{1\}) = v_x(\{1, 2\}) - v_x(\{1\}) = x_1 - x_1 = 0$

Such that

$$\phi_2 = \frac{1}{2}(x_1 - 0.5) \neq 0$$

For interventional SHAP, we have

- $v_x(\emptyset) = \mathbb{E}[\hat{m}_n(X_1, X_2)] = 0.5$
- $v_x(\{1\}) = \mathbb{E}[\hat{m}_n(x_1, X_2)] = x_1$
- $v_x(\{2\}) = \mathbb{E}[\hat{m}_n(X_1, x_2)] = 0.5$
- $v_x(\{1,2\}) = [\hat{m}_n(x_1, x_2) = x_1$

Hence,

- $\Delta(2, \emptyset) = v_x(\{2\}) - v_x(\emptyset) = 0$
- $\Delta(2, \{1\}) = v_x(\{1,2\}) - v_x(\{1\}) = 0$

Such that

$$\phi_2 = 0$$

The second problem with observational SHAP is a more practical one - It is simply harder to calculate compared to interventional SHAP. We will comment on that in the next section.

7.1.1 Estimation

We present a simple way to calculate Shapley values here and give references to modern (faster) techniques. It is technically possible to calculate Shapley values exactly by virtue of (Theorem 7.1.2). However, for this calculation it is needed to evaluate the value function ν on the order of 2^p times, which quickly becomes computationally intensive. The obvious solution to this issue is to simply only evaluate some terms at random, hoping that this will approximate the true exact Shapley value well. In more formal terms, we have the following algorithm:

Permutation Sampling for Shapley Values (for interventional SHAP)

Goal: To estimate $\phi_k(x)$.

1. For $r = 1, \dots, N$, do:
   (a) Randomly sample a permutation $\pi \in \Pi_p$ and set $S = \{\pi(1), \dots, \pi(\pi^{-1}(k) - 1)\})$.
   (b) Calculate $\Delta^{(r)} = \Delta_\nu(k, S) = \nu(S \cup \{k\}) - \nu(S)$ by:
      i. Randomly sample m observations $X_1, \dots, X_m$.
      ii. Estimate $\nu(S) = \mathbb{E}[\hat{m}(x_S, X_{-S})]$ by $\frac{1}{m}\sum_{i=1}^{m} \hat{m}(x_S, X_{i,-S})$ and similarly for $\nu(S \cup \{k\})$. (Note that we are assuming interventional SHAP)
2. The final estimate for $\phi_k(x)$ is then $\frac{1}{N}\sum_{r=1}^{N} \Delta^{(r)}$.

Remark. The algorithm above makes two approximations: Firstly only N randomly drawn permutations are used instead of all $p!$ permutations. Secondly, in step (b)ii, the expectation $v(S) = \mathbb{E}[\hat{m}_n(x_S, X_{-S})]$ (interventional SHAP) is approximated by its empirical counterpart using m observations. Note that If $v(S) = \mathbb{E}[\hat{m}_n(X_S, X_{-S})|X_S = x_S]$ (observational SHAP), it is not clear how an empirical approximation would look like. The reason is that we may not have many observations i with $X_{i,S} = x_S$. One would need to estimate the distribution of X first using some kind of smoothing; which is a high dimensional estimation problem.

An alternative to permutation sampling is the co-called 'Kernel-SHAP', which relies on re-writing the Shapley decomposition as the solution to a minimization problem. We given the following result without proof:

Theorem 7.1.4 (Shapley kernel Charnes et al. (1988))**.** Shapley values ϕ_k, $k = 0, \dots, p$, can be written

as the solution of the following constrained minimization problem:

$$\phi(x) = \text{argmin}_{\phi \in \mathbb{R}^{p+1}} \sum_{S \subseteq I_p; S \neq \emptyset, I_p} \mu(S) \left(\nu(S) - \left(\phi_0 + \sum_{k \in S} \phi_k \right) \right)^2,$$

under the constraint

$$\phi_0 = \nu(\emptyset), \qquad \phi_0 + \sum_{k=1}^{p} \phi_k = \nu(I_p),$$

where

$$\mu(S) = \frac{p-1}{\binom{p}{|S|}|S|(p-|S|)}.$$

While we do not go through the details, the theorem can be proven by writing down the solution explicitly and thereafter show that equals the Shapley values as described in Theorem 7.1.2. The weights for the empty set all features are $\mu(\emptyset) = \mu(I_p) = \infty$, and hence the minimisation would not be well defined if they where included in the sum. While Shapley kernel is a quadratic programming problem and thus lends itself to well-known optimization techniques, it requires estimation of the value function for all 2^p subsets of I_p, which is the same problem that motivated the introduction of permutation sampling. To solve the problem for this method, we would instead sample subsets $S_1, \dots, S_m$ from $\mathcal{A} = \mathcal{P}(I_p) \setminus \{\emptyset, I_p\}$ according to $P(S_i = s) = \frac{\mu(s)}{\sum_{s^* \in \mathcal{A}} \mu(s^*)}$. The final constrained minimization problem to solve is

$$\text{argmin}_{\phi \in \mathbb{R}^p} \sum_{i=1}^{m} \left(\nu(S_i) - \left(\nu(\emptyset) + \sum_{k \in S_i} \phi_k \right) \right)^2$$

Subject to $\sum_{k=1}^{p} \phi_k = \nu(I_p) - \nu(\emptyset)$. In practice, this is often then solved by reformulating the minimization as a least squares linear regression problem:

$$\text{argmin}_{\phi \in \mathbb{R}^p} \sum_{i=1}^{m} \left((\nu(S_i) - \nu(\emptyset)) - M_i \phi \right)^2,$$

subject to $\sum_{k=1}^{p} \phi_k = \nu(I_p) - \nu(\emptyset)$ and where M_i is the ith row of the matrix M with m rows p columns that binary encodes S_i: $M_{ij} = \mathbb{1}(j \in S_i)$, $j = 1, \dots, p$. This optimization procedure is also known as KernelSHAP.

Tree-SHAP

For tree-based algorithms, it is possible to exploit the tree structure to obtain a more efficient method for calculating Shapley values. This was originally proposed in the form of the TreeSHAP-algorithm Lundberg et al. (2020). Note that we only need cover Tree-SHAP for decision trees, not for ensemble estimators such as random forests or gradient boosting machines based on trees. The reason for this is that these other estimators are based on sums of decision trees, and by virtue of the linearity axiom for Shapley values (S4), the Shapley values for these more advanced estimators can be computed by summing over the Shapley-values for each decision tree that enters the model.

Example 17. Let $\hat{m} = \hat{m}^1 + \hat{m}^2$ be an ensemble estimator, where $\hat{m}^1$ and $\hat{m}^2$ are decision trees, then $\phi_k(\hat{m}) = \phi_k(\hat{m}^1) + \phi_k(\hat{m}^2)$. Hence we only need to be able to calculate the Shapley-values for decision trees $\hat{m}^1$ and $\hat{m}^2$ to get the Shapley values for the more advanced estimator $\hat{m}$.

For a tree $T = (\mathcal{B}, f)$, denote the fraction of observations falling into node B_j by

$$r_{B_j} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{X_i \in B_j\}.$$

A formal description of the TreeSHAP algorithm is the following.

TreeSHAP - Lundberg et al. (2020)

Input: A tree T, a subset $S \subseteq I_p$, a point $x \in \mathcal{X} \subseteq \mathbb{R}^p$.
Goal: Approximate $\nu(S)$ as `EXPVALUE`(x, S, T) .

Function `G`(B, x, S, T)**:**
  if $B \in \mathcal{A}_T$ *(i.e. is B a leaf?)* **then**
    return $\widehat{m}_n(B)$
  else
    Denote by B_1 and B_2 the daughter nodes of B (i.e. $f^{-1}(B) = \{B_1, B_2\}$).
    if $P((x_s, X_{-S}) \in B_1 \mid (x_s, X_{-S}) \in B) = 1$ **then**
      return `G`(B_1)
    else
      if $P((x_s, X_{-S}) \in B_2 \mid (x_s, X_{-S}) \in B) = 1$ **then**
        return `G`(B_2)
      else
        return `G`$(B_1) \cdot \frac{r_{B_1}}{r_B}$ + `G`$(B_2) \cdot \frac{r_{B_2}}{r_B}$

Function `EXPVALUE`(x, S, T)**:**
  return `G`$(\mathcal{X})$

The algorithm relies on the recursive function G, which transverse the tree from top to bottom, calculating the contribution of each leaf to the expectation $\nu(S) = E[\widehat{m}(x_S, X_{-S})]$. There are four possible situations that this function may encounter:

1. If B is a leaf node, then the function G simply returns the value predicted at that leaf.
2. If B is not a leaf, then it is possible to split B into $B_1 \cup B_2$.
   (a) If the information contained within x_S is sufficient to decide that $(x_S, X_{-S}) \in B_1$, then the algorithm continues to node B_1.
   (b) If the information contained within x_S is sufficient to decide that $(x_S, X_{-S}) \in B_2$, then the algorithm continues to node B_2.
   (c) If the information contained within x_S is insufficient to decide what node the algorithm should continue to, it continues to both nodes, but weights the results obtained by the empirical probabilities r_{B_j}/r_B of reaching each node.

While this algorithm calculates an estimator for $\nu(S)$ for any particular S, we need to know this value for each $S \subseteq I_p$ to be able to calculate Shapley values. Hence, naively going through each subset S would result in a $O(2^p)$ run-time, since there are 2^p such subsets. The actual Tree-SHAP algorithm improves this to $O(p^2)$ by performing the computations for each $S \subseteq I_p$ in parallel.
Which value function is TreeSHAP actualy approximating? The answer to that question is unfortunately none. Even if infinite data is available TreeSHAP will neither be equal to observational SHAP nor interventional SHAP. In fact, the values that TreeHSAP provides are not fully determined by the tree estimator $\widehat{m}_n$ alone but they do also depend on the paths of the tree. Two trees with the same leaves and thereby leading to the same estimator $\widehat{m}_n$, may still have different TreeSHAP values if the splits are done in a different order.

Example 18. Let $\mathcal{X} =\in \{-1, 1\}^2$ and

$$\widehat{m}_n(x_1, x_2) = \texttt{xor}(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 = x_2, \\ 1 & \text{else.} \end{cases}$$

From 7.1.3, we have that for $x = (x_1, x_2)^T = (-1, -1)^T$

$$\begin{aligned} v_x^{\texttt{observational SHAP}}(\{1\}) &= \mathbb{E}[\hat{m}_n(X_1, X_2)|X_1 = -1] \\ &= P(X_2 = -1|X_1 = -1) \cdot 0 + P(X_2 = 1|X_1 = -1) \cdot 1 \\ &= P(X_2 = 1|X_1 = -1), \end{aligned}$$

as well as

$$v_x^{\texttt{interventional SHAP}}(\{1\}) = \mathbb{E}[\hat{m}_n(-1, X_2)] = P(X_2 = -1) \cdot 0 + P(X_2 = 1) \cdot 1 = P(X_2 = 1).$$

Assume that we have two tree models T^1, T^2 that both lead to $\hat{m}_n(x_1, x_2)$. However, T^1 and T^2 have different paths.. In the following we will use the notation as in Figure 5.1. In T^1, the first split is with respect to X_1:

$$\begin{aligned} B_2^1 = \{x : x_1 \leq 0\}, B_3^1 = \{x : x_1 > 0\}, \\ B_4^1 = \{x \in B_2^1 : x_2 \leq 0\}, B_5^1 = \{x \in B_2^1 : x_2 > 0\}, \\ B_6^1 = \{x \in B_3^1 : x_2 \leq 0\}, B_7^1 = \{x \in B_3^1 : x_2 > 0\}. \end{aligned}$$

While in T^2, the first split is with respect to X_2:

$$\begin{aligned} B_2^2 = \{x : x_2 \leq 0\}, B_3^2 = \{x : x_2 > 0\}, \\ B_4^2 = \{x \in B_2^2 : x_1 \leq 0\}, B_5^2 = \{x \in B_2^2 : x_1 > 0\}, \\ B_6^2 = \{x \in B_3^2 : x_1 \leq 0\}, B_7^2 = \{x \in B_3^2 : x_1 > 0\}. \end{aligned}$$

We now estimate $v_x(\{X_1\})$ at the point $x = (-1, -1)$ using the TreeSHAp algorithm. Assume that

- For T^1 we get:

$$G(\mathcal{X}) = G(B_2^1) = r_{B_4^1} \cdot G(B_4^1) + r_{B_5^1} \cdot G(B_5^1) = r_{B_4^1} \cdot 0 + r_{B_5^1} \cdot 1 = r_{B_5^1}.$$

- For T^2 we get:

$$G(\mathcal{X}) = r_{B_2^2} G(B_2^2) + r_{B_3^2} G(B_3^2) = r_{B_2^2} G(B_4^2) + r_{B_3^2} G(B_6^2) = r_{B_2^2} \cdot 0 + r_{B_3^2} \cdot 1 = r_{B_3^2}.$$

  Noting that $r_{B_5^1} \approx P(X_2 = 1|X_1 = -1)$ and $r_{B_3^2} \approx P(X_2 = 1)$, we conclude that TreeSHAP estimates for $S = \{1\}$ observational SHAP in the case of T^1 and interventional SHAP in the case of T^2. By symmetry, for $S = \{2\}$ the roles will be interchanged.

The example illustrates that the SHAP values estimated by TreeSHAP depend on the paths of the tree. A computational more expensive alternative is to use $J \leq n$ background instances $(\tilde{X}_j)_{j=1,\ldots,J}$ i which is obtained by randomly sampling J times without replacement from the original data $(X_i)_{i=1,\ldots,n}$. Then, for each background sample j an alternative TreeShap algorithm is run: For a subset S, $v_x(S)$ is approximated by following the path according to $(x_S, \tilde{X}_{j,-S})$, leading to a unique leaf value $\hat{m}(x_S, X_{i,-S})$. The final estimate is given as simple average over all background instances. If $J = n$, then the average of the J TreeSHAP with backgroudnd data outputs will be equal to the empirical interventional SHAP value:

$$\frac{1}{J} \sum_{j=1}^{J} \hat{m}(x_S, \tilde{X}_{j,-S}) = \frac{1}{n} \sum_{i=1}^{n} \hat{m}(x_S, X_{i,-S}).$$

TreeSHAP with background data - Lundberg et al. (2020)

Input: A tree T, a subset $S \subseteq I_p$, a point $x \in \mathcal{X} \subseteq \mathbb{R}^p$, a backgound instancece $\tilde{x} \in \mathbb{R}^p$.
Goal: Calculate $\hat{m}(x_S, \tilde{x}_{-S}) = \texttt{EXPVALUE}(x, S, T)$.

Function `G`(B, x, S, T)**:**
  if $B \in \mathcal{A}_T$ *(i.e. is B a leaf?)* **then**
    return $\hat{m}_n(B)$
  else
    Denote by B_1 and B_2 the daughter nodes of B (i.e. $f^{-1}(B) = \{B_1, B_2\}$).
    if $P((x_s, \tilde{x}_{-S}) \in B_1 \mid (x_s, \tilde{x}_{-S}) \in B) = 1$ **then**
      return `G`(B_1)
    else
      return `G`(B_2)

Function `EXPVALUE`(x, S, T)**:**
  return `G`$(\mathcal{X})$

Similar to the original TreeSHAP algorithm without background data, the computational run-time can be reduced by running the algorithm for all $S \subset I_p$ at the same time, and thereby running down the tree only once. Still since this has to be done for every background instance $j = 1, ... J$, the run-time is still around J time longer than in the case where no backgroubd data is used. The reward for the additional computational time is an exact calculation of the empirical interventional SHAP value, where the unkown expectation is approximated by the empirical probabilities stemming from the J instances.

⚠ A word of caution: Even assuming that one has the computational resources to run TreeSHAP with background data, not everything is solved. The algorithm will provide an empirical version of the nterventional SHAP value:

$$v(S) = \sum_{j=1}^{J} [\hat{m}_n(x_S, X_{j,-S})].$$

This will usually be a good approximation of

$$v(S) = \mathbb{E}_X[\hat{m}_n(x_S, X_{-S})].$$

But there are two important points we haven't discussed yet: (a) We are actually approximating an explanation of $\hat{m}_n$ and not of m^*. (b) While Interventional SHAP has some advantages compared to observational SHAP it has its own problems: it might suffer from heavy extrapolations if $\mathcal{X}$ is not rectangular, because $\hat{m}_n(x_S, X_{j,-S})] \notin \mathcal{X}$ for some $\tilde{X}_j$. We will come back to those points in later sections

7.2 LIME

In the previous section we introduced Shapley values as an example of a local explanation. Another class of local explanations is known as LIME. We start by defining an interpretable model.

Definition 7.2.1 (interpretable model)**.** Consider a supervised learning problem with $(X, Y) \in \mathbb{R}^p \times \mathcal{Y}$. Given an "*interpretable representation*" of data points

$$h : \mathbb{R}^p \to \{0,1\}^{p'}; \quad x \mapsto x',$$

the set G is a class of interpretable models if all $g \in \mathcal{G}$ are "*understandable*" mappings

$$g : \{0,1\}^{p'} \to \mathcal{Y}.$$

Given a class of interpretable models we can now define LIME.

Definition 7.2.2 (LIME). Given a data instance $x \in \mathbb{R}^p$, a local interpretable model-agnostic explanation (LIME) is given by

$$\xi(x) = \text{argmin}_{g \in \mathcal{G}} \mathcal{L}(\hat{m}_n, g \circ h, \pi_x) + J(g),$$

where $\mathcal{G}$ is a class of interpretable model, $\mathcal{L}$ is some loss function measuring the distance between the functions $\hat{m}_n$ and $g \circ h$, π_x is a weight function (called kernel) and J is a penalty for the complexity of g.

The definition of LIME is very general and its exact implementation depends on the problem at hand. LIME explanations are mostly popular in the case of unstructured data, but they can also be used for tabular data. Here, the following implementation is common:

Example 19. Common settings for LIME are

$$\begin{aligned} \pi_x(z) &= e^{-\sum_{k=1}^{p} \left(\frac{x_k - z_k}{\sigma_k}\right)^2}, \ \sigma_k > 0, \\ \mathcal{G} &= \{\text{linear functions}\} = \{g | g(z') = w^T z', \text{for some} w \in \mathbb{R}^{p'}\}, \\ L(\hat{m}_n, g \circ h, \pi_x) &= \sum_{z \in \mathcal{Z}} \pi_x(z)(\hat{m}_n(z) - g \circ h(z))^2. \end{aligned}$$

For tabular data, the following remaining choices are common.

- The interpretable representation h: Categorical Features are one-hot encoded, while continuous features remain unchanged.
- Let μ_k and $\tilde{\sigma}_k$ be the emiprical mean and standard deviation of $h(X)_k$.
- The kernel π: $\sigma_k = 0.75\sqrt{p}\tilde{\sigma}_k$. Furthermore π is applied on the transformed features $x' = h(x)$, $z' = h(z)$:
$$\pi_x(z) = e^{-\sum_{k=1}^{p'} \left(\frac{x'_k - z'_k}{\sigma_k}\right)^2}.$$
- The set $\mathcal{Z}$: A pertubed instance $z = (z_k)_{k=1,\dots,p} \in \mathcal{Z}$ is derived by sampling each component z_k, $k = 1, \dots, p$, independently. If k is numerical z_k is sampled from $N(\mu_k, \sigma_k^2)$. In the case of a categorical variable k, z_k according to the category frequency.
- The penalty: $J(g) = \infty \mathbb{1}\{||g||_0 > K\}$. Note that optimizing g with this penalty is only done approximately, e.g. via forward search or selecting K features via Lasso penalization.

The default LIME settings, as specified in the example, are quite arbitrary. Unfortunately LIME values are quite sensitive to a specific kernel choice, hence making LIME explanations questionable. If one is willing to change the perspective and aims to explain a value function, as defined in the previous section, at a fixed point x instead of the estimator $\hat{m}_n$, then Shapley values can be seen as a special case of LIME via the Shapley kernel representation.

7.3 ICE plots, PDP and ALE plots

While Shapley values explain the model at a local scale, e.g. they answer the question of why the model predicted what it did for a single observation, they don't provide an easily interpretable description for the general trends caught in the model.

It would clearly be optimal if humans could simply look at high-dimensional plots of $x \mapsto \hat{m}_n(x)$ to understand the behaviour of decision function $\hat{m}_n$. However, anything more complex than an ordinary 2dimensional -plot is generally too complex to be well understood. Thus, to understand the estimator $\hat{m}_n$, it seems reasonable to plot the different possible marginalisation of it.

7.3.1 Individual Conditional Expectation (ICE) plots and partial dependence plots

Individual Conditional Expectation (ICE) plots display one line for every observation $i = 1, \dots, n$.

Definition 7.3.1 (ICE plots)**.** The ICE plots for feature k are defined as

$$\eta_i(x_k) = \hat{m}_n(x_k, X_{i,-k}), \quad i = 1, \dots, n.$$

Each curve $\eta_i(x_k)$ displays how a feature x_k affects an individual given that the other features remain unchanged. Hence, if all curves $\eta_i(x_k)$, $i = 1, \dots, n$ are plotted together this may give insight into the effect of x_k in different conditions. A drawback can be that similar to the case of interventional SHAP values, some parts of the plots are the result of extrapolation, making the values questionable in those areas. A partial dependence plot (PDP) is the expected value of an ICE plot.

Definition 7.3.2 (PDP)**.** The PDP plot for feature k is defined as

$$\xi_k(x_k) = \mathbb{E}_X[\hat{m}_n(x_k, X_{-k})].$$

Its empirical version is given as

$$\hat{\xi}_k(x_k) = \frac{1}{n}\sum_{i=1}^{n} \eta_i(x_k) = \frac{1}{n}\sum_{i=1}^{n} \hat{m}_n(x_k, X_{i,-k}).$$

More generally, the PDP plot for a set of features $S \subseteq I_p$ is defined as

$$\xi_S(x_S) = \mathbb{E}_X[\hat{m}_n(x_S, X_{-S})] = \int \hat{m}(x_S, x_{-S}) p_{X_{-S}}(x_{-S}) dx_{-S}.$$

The empirically version is given as

$$\hat{\xi}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{m}(x_S, X_{i,-S}).$$

While one can potentially have $|S| > 1$, this means that plotting ξ_S takes up more than two dimensions and hence such plots are much more difficult to get anything out of. The notable exception to this is the case of one continuous feature and one categorical. If the categorical feature has sufficiently few categories, a meaningful plot could be produced by producing a graph for each category separately.

Example 20. Let $X \sim N_2(0, I_2)$ and let $Y = X_1 + X_2^2 + \varepsilon$ with $\varepsilon \sim N(0, 1)$. We generate 1000 samples from this distribution and fit a random forest to the resulting data set. Based on the estimator we obtain, we compute partial dependence plots for X_1 and X_2, these are show in Figure 7.1 and Figure 7.2 respectively.
What we can conclude is that the random forest seems to have caught the marginal effects of both X_1 and X_2 pretty well.

7.3.2 The Extrapolation Issue

One important drawback of partial dependence plots is the extrapolation issue. If features are correlated, their joint support may not be equal to the product of their marginal support. If this is the case, then the empirical partial dependence plot evaluates the estimator $\hat{m}_n$ outside the data support. But for nonparametric algorithms, the value of $\hat{m}_n$ is quite arbitrary in regions without data. The result is that the empirical partial dependence plot is highly influenced by the behaviour of $\hat{m}_n$ in regions where predictions are arbitrary. We illustrate the problem in the following example.

Example 21. Assume that $p = 2$ and that X_1 and X_2 are highly correlated:

$$X_1 = U + N(0, 0.05^2), \quad X_2 = U + N(0, 0.05^2), \quad U = \mathit{Unif}[0, 1].$$

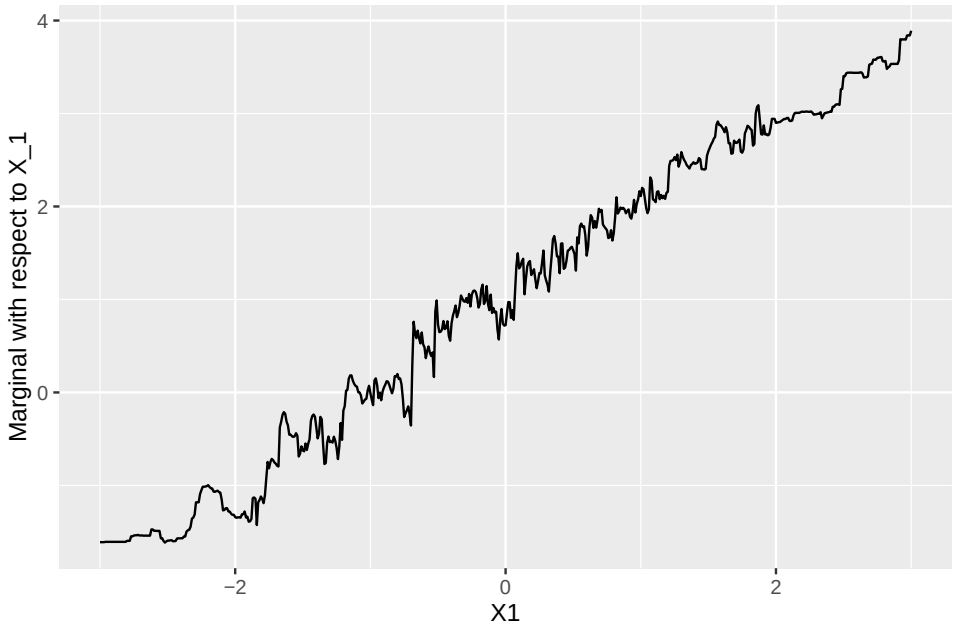

Figure 7.1: Partial dependence plot for X_1

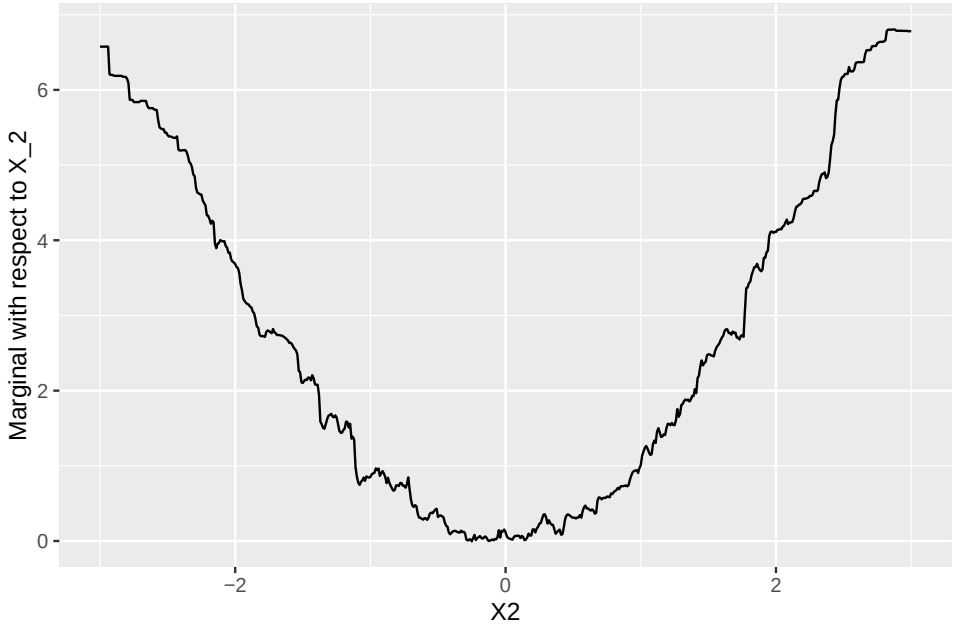

Figure 7.2: Partial dependence plot for X_2

A scatterplot from 100 observation is shown in Figure 7.3. To The empirical partial dependence plot of X_1 at $x_1 = 0.3$ is calculated by taking a simple average of predictions on the blue dots in Figure 7.3. The problem is that most of the blue points are outside the data support. Hence, especially for complex algorithms these estimates can be quite random. This problem is known as extrapolation issue.

The problem comes from the fact that the integral in the definition of $\xi_S(x_S)$ runs over the support of $p_{X_{-S}}$. One solution to that problem is to modify the definition and let the bounds of the integral depend

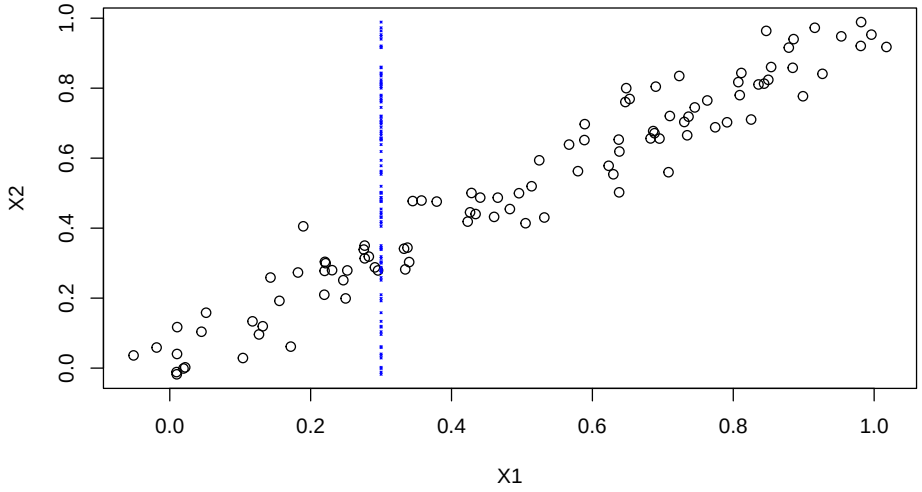

Figure 7.3: Scatterplot of 100 observations with highly correlated X_1 and X_2. $X_1 = U + N(0, 0.05^2)$, $X_2 = U + N(0, 0.05^2)$, $U = Unif[0, 1]$. When calculating the empirical partial dependent plot for X_1 at $x_1 = 0.3$, one would need to evaluate the regression estimator at every blue point.

on the support of $X \in \mathbb{R}^p$:

$$\tilde{\xi}_S(x_S) = \frac{\int_{x_{-S}:(x_S,x_{-S})\in\mathcal{X}} \hat{m}_n(x_S, x_{-S}) p_{X_{-S}}(x_{-S}) \mathrm{d}x_{-S}}{\int_{x_{-S}:(x_S,x_{-S})\in\mathcal{X}} p_{X_{-S}}(x_{-S}) \mathrm{d}x_{-S}}.$$

This solution does not seem to be very popular yet in applications and one would need to develop an empirical version of $\tilde{\xi}_S(x_S)$ that is stable and can be calculated fast. One such proposal can be found in Taufiq et al. (2023). A more popular solution to the extrapolation issue are ALE plots which will be introduced in the next section. Before we come to this, it is worth mentioning that a marginal plot for X_S at x_S,

$$\ell_S(x_S) = \mathbb{E}_X[\hat{m}_n(X_S, X_{-S})|X_S = x_S] = \int \hat{m}_n(x_S, x_{-S}) p_{X_{-S}|X_S}(x_{-S}|x_S) dx_{-S},$$

may no be so useful to interpret the effect the direct effect of X_S on $\hat{m}_n$. The issue here is the same as the issue with observational SHAP (cf. Definition 7.1.3 and Example 16): If X_S is correlated with X_{-S}, then $\ell_S(x_S)$ will reflect both the direct effect of X_S on $\hat{m}_n$ and the indirect effect of X_S on $\hat{m}_n$ via X_{-S}. This is also analogue to the omitted variable bias in regression.

Example 22. Consider the same setting as in Example 16, i.e.,

$$\hat{m}_n(x_1, x_2) = x_1$$

Let X_1, X_2 be binary with

$$p(x_1, x_2) = \begin{cases} \frac{1}{2} & \text{if } x_1 = x_2 \\ 0 & \text{else} \end{cases}$$

We get $\ell_2(x_2) = x_2$. In other words: Although X_2 has no direct impact on $\hat{m}_n$, the marginal plot attributes a non-zero contribution stemming from the correlation with X_1.

7.3.3 Accumulated Local Effects (ALE) Plots

Accumulated Local Effects (ALE) Plots have been introduced in Apley and Zhu (2020) as solution to the extrapolation issue of partial dependence plots.

Definition 7.3.3 (uncentered ALE plot)**.** Assume that $\hat{m}_n$ is differentiable and denote by $\hat{m}_n^k$ the partial derivative of $\hat{m}_n$ with respect to x_k. Let $x_{k,\min}$ be the lower bound of the support of X_k The uncentered ALE plot for feature k is defined as

$$g_k(x_j) = \int_{x_{k,\min}}^{x_k} \mathbb{E}\left[\hat{m}_n^k(X_k, X_{-k}) \mid X_k = z_k\right] \mathrm{d}z_k = \int_{x_{k,\min}}^{x_k} \int \hat{m}_n^k(x_k, x_{-k})\, p_{X_{-k}|X_k}(x_{-k}|x_k) \mathrm{d}x_{-k} \mathrm{d}z_k.$$

Considering a subset $S \subseteq I_p$, we denote the cross partial derivative by

$$\hat{m}_n^S = \frac{\partial^{|S|} \hat{m}_n}{\prod_{j \in S} \partial x_j}.$$

The uncentered ALE plot for the set of features S is defined as

$$g_S(x_j) = \int_{x_{S,\min}}^{x_S} \mathbb{E}\left[\hat{m}_n^S(X_S, X_{-S}) \mid X_S = z_S\right] \mathrm{d}z_S.$$

Here $\int_{x_{S,\min}}^{x_S}$ denotes the $|S|$ dimensional integral running over the hyper-rectangle with edges $[x_{j,\min}, x_j],\ j \in S$.

The heuristic for the ALE plot g_k is the following. Looking at the derivative $\hat{m}_n^k$ removes all indirect effects of X_k. Afterwards the conditional mean can be taken that avoids the extrapolation issue. In contrast to marginal plots or observational SHAP, indirect effects will not enter because they have already been removed. As a last step, one takes the integral as an antiderivative. If one is only interested in the shape of g_k, then the exact value of $x_{k,\min}$ does not matter much as different values only lead to vertical shifts of g_k.

Example 23. Let

$$\hat{m}_n(x_1, x_2) = x_1 + x_2.$$

Then,

$$g_1(x_1) = x_1 - x_{1,\min} = x_1 \pm \text{constant}, \quad g_2(x_2) = x_2 - x_{2,\min} = x_2 \pm \text{constant},$$

showing that ALE plots recover additive components. Now, assume

$$\hat{m}_n(x_1, x_2) = x_1 + x_2 + 2x_1x_2,$$

and let (X_1, X_2) be multivariate standard normal with correlation equal 0.3. Noting that $\mathbb{E}[X_2|X_1 = x_1] = 0.3x_1$, we get

$$g_1(x_1) = \int_{x_{1,\min}}^{x_1} \mathbb{E}\left[1 + X_2 | X_1 = z_1\right] \mathrm{d}z_1 = \int_{x_{1,\min}}^{x_1} 1 + 0.6z_1 \mathrm{d}z_1 = x_1 - x_{1,\min} + 0.3(x_1^2 - x_{1,\min}^2).$$

We observe two things: Firstly $x_{1,\min}$ should be chosen as data driven value close to the minimal observed value as the theoretical value of $-\infty$ would make the ALE plot undefined. Secondly, interaction complicate the question of what we actually want to see. As a comparison, the partial dependent plot for x_1 (only using that the variables have mean zero and no further assumption on the distribution or covariance) is

$$\xi_1(x_1) = \mathbb{E}\left[x_1 + X_2 + 2x_1X_2\right] = x_1.$$

While the partial dependence plot has a simpler form here it is hard to formally argue why ξ_1 represents the effect of X_1 on $\hat{m}_n$ better than g_1.

We now illustrate how uncentered ALE plots can be estimated. The main ingredient for deriving the ALE plot for a single feature k is to partition the support of X_k after which the empirical equivalent of the population version can be calculated. For second order effects, we will need a two dimensional grid

and so on. This hints that estimates of higher order effect are expected to quickly get more and more unstable.

Calculate ALE plots

First order effects for feature k:

1. Partition the support for feature k into a grid with B bins: $X_{k,min} = z_{k0} < \cdots < z_{kB} = X_{k,max}$. The partitioning doesn't need to be equidistant and can e.g. coincide with quantiles of X_k.
2. Denote by $\iota(x_k)$ the interval index where x_k falls: $x_k \in (z_{k,j-1}, z_{kj}]$ if $\iota(x_k) = j \neq 1$ and $x_k \in [z_{k,j0}, z_{k1}]$ if $\iota(x_k) = 1$. Denote by $N_k(j)$ all observation i with $\iota(X_{ik}) = j$. We will also write $|N_k(j)| = n_k(j)$.
3. Calculate

$$\hat{g}_k(x_k) = \sum_{j=1}^{\iota_k(x_k)} \frac{1}{n_k(j)} \sum_{\{i:X_{i,k} \in N_k(j)\}} \left\{ \hat{m}_n\left(z_{k,j}, X_{i,-j}\right) - \hat{m}_n\left(z_{k-1,j}, X_{i,-j}\right) \right\}.$$

Second order effects for features (j, k):

1. Partition the support for features j, k into B bins each : $X_{k,min} = z_{k0} < \cdots < z_{kB} = X_{k,max}$, $X_{j,min} = z_{j0} < \cdots < z_{jB} = X_{j,max}$. The partitioning doesn't need to be equidistant and can e.g. coincide with quantiles of X_j and X_k.
2. We use the same notation as before for the first order effect. Additionally: Denote by $N_{jk}(l, m)$ all observation i with $\iota(X_{ik}) = l$ and $\iota(X_{ik}) = m$. We also write $|N_{jk}(l, m)| = n_{j,k}(l, m)$.
3. $$\hat{g}_{jk}(x_j, x_k) = \sum_{l=1}^{\iota_j(x_j)} \sum_{m=1}^{\iota_k(x_k)} \frac{1}{n_{\{j,k\}}(l, m)} \sum_{\left\{i:X_{i,jk} \in N_{\{j,k\}}(l,m)\right\}} \Delta_{\hat{m}_n}^{\{j,k\}}\left(l, m; X_{i,-\{j,k\}}\right),$$

   where

$$\begin{aligned}\Delta_{\hat{m}_n}^{\{j,k\}}\left(l, m; X_{i,-\{j,k\}}\right) =& \hat{m}_n\left(z_{j,l}, z_{k,m}, X_{i,-\{j,k\}}\right) - \hat{m}_n\left(z_{j,l-1,}, z_{k,m}, X_{i,-\{j,k\}}\right) \\ &- \left\{\hat{m}_n\left(z_{j,l}, z_{k,m-1}, X_{i,-\{j,k\}}\right) - \hat{m}_n\left(z_{j,l-1}, z_{k,m-1}, X_{i,-\{j,k\}}\right)\right\}.\end{aligned}$$

7.4 Functional Decomposition

We consider $\hat{m}$ some decision function that we wish to explain. The fundamental idea is to realize that this decision function can be represented through

$$\begin{aligned}\hat{m}_n(x) &= m + \sum_{k=1}^{p} \hat{m}_k(x_k) + \sum_{j,k:k<j} \hat{m}_{k,j}(x_k, x_j) + \cdots + \sum_{k=1}^{p} \hat{m}_{-k}(x_{-k}) + \hat{m}_{1,\ldots,p}(x) \\ &= \sum_{S \subseteq I_p} \hat{m}_S(x_S)\end{aligned}$$

This decomposition may not be uniquely identified, and in the spirit of example 15 an arbitrary decomposition may not carry relevant information. Hence we would be looking to find a decomposition of this type, where the decomposition carries useful information.

7.4.1 Marginal Identification

Definition 7.4.1 (marginal identification). We say that a decomposition of $\hat{m}_n$ satisfies the marginal

identification if for any $S \subseteq I_p$

$$\sum_{T \cap S \neq \emptyset} \int \hat{m}_T(x_T) \hat{p}_S(x_S) \mathrm{d}x_S = 0, \tag{7.1}$$

where $\hat{p}_S$ is some estimator of the density p_S of X_S.

It turns out that for any estimator $\hat{m}_n$, there exists exactly one decomposition that satisfies the marginal identification.

Theorem 7.4.2. Let $\hat{m}_n$ be some estimator. Then the only decomposition that satisfies the marginal identification is given by

$$\hat{m}^*_S(x_S) = \sum_{V \subseteq S} (-1)^{|S \setminus V|} \int \hat{m}_n(x) \hat{p}_{-V}(x_{-V}) dx_{-V}$$

Proof. We start showing that $\{\hat{m}^*_S(x_S)\}$ as defined in the theorem satisfies the marginal identification. By definition, for any $U \subseteq I_p$, we have

$$\begin{aligned}
\sum_{T \subseteq U} \hat{m}^*_T(x_T) &= \sum_{T \subseteq U} \sum_{V \subseteq T} (-1)^{|T \setminus V|} \int \hat{m}_n(x) \hat{p}_{-V}(x_{-V}) \mathrm{d}x_{-V} \\
&= \sum_{V \subset U} \int \hat{m}_n(x) \hat{p}_{-V}(x_{-V}) \mathrm{d}x_{-V} \sum_{S \subseteq \{U \setminus V\}} (-1)^{|S|} \\
&\quad + \int \hat{m}_n(x) \hat{p}_{-U}(x_{-U}) \mathrm{d}x_{-U} \\
&= \int \hat{m}_n(x) \hat{p}_{-U}(x_{-U}) \mathrm{d}x_{-U},
\end{aligned} \tag{7.2}$$

where the last equation follows from $\sum_{S \subseteq \{U \setminus V\}} (-1)^{|S|} = 0$, noting that a non-empty set has an equal number of subsets with an odd number of elements as subsets with an even number of elements (cf. Lemma A.0.1). From this derivation we can conclude that $\hat{m}^*$ is a solution of the marginal identification. To see this, for $S \subseteq I_p$, consider the following decomposition of $\int \hat{m}_n(x) \hat{p}_S(x_S) \mathrm{d}x_S$

$$\int \hat{m}_n(x) \hat{p}_S(x_S) \mathrm{d}x_S = \sum_{T \cap S \neq \emptyset} \int \hat{m}^*_T(x_T) \hat{p}_S(x_S) \mathrm{d}x_S + \sum_{T \cap S = \emptyset} \hat{m}^*_T(x_T). \tag{7.3}$$

While from the derivation above (7.2), we have

$$\int \hat{m}_n(x) \hat{p}_S(x_S) \mathrm{d}x_S = \sum_{T \subseteq S^c} \hat{m}^*_T(x_T) = \sum_{T \cap S = \emptyset} \hat{m}^*_T(x_T),$$

which with the previous statement implies that $\hat{m}^*$ is a solution to the marginal identification:

$$\sum_{T \cap S \neq \emptyset} \int \hat{m}^*_T(x_T) \hat{p}_S(x_S) \mathrm{d}x_S = 0.$$

It is left to show that the solution is unique. Assume that there are two set of functions $\{\hat{m}^\circ_S\}$ and $\{\hat{m}^*_S\}$ that satisfy the marginal identification with $\sum_S \hat{m}^\circ_S = \sum_S \hat{m}^*_S$. The marginal identification says that the first summand in (7.3) is zero, hence for all $S \subseteq I_p$

$$\sum_{T \cap S = \emptyset} \hat{m}^\circ_T(x_T) = \sum_{T \cap S = \emptyset} \hat{m}^*_T(x_T),$$

implying $\hat{m}^\circ_T(x_T) = \hat{m}^*_T(x_T)$ for all $T \subseteq I_p$.

□

If some decomposition of the estimator is known, the decomposition satisfying the marginal identification can be computed via the following corollary.

Corollary 7.4.3. Let

$$\hat{m_n}(x) = \sum_{S \subseteq I_p} \hat{m}_S^{(0)}(x_S),$$

be any functional decomposition of $\hat{m}_n$. Then the decomposition of $\hat{m}_n$ that satisfies the marginal identification is

$$\hat{m}_S^*(x_S) = \sum_{T:T \supseteq S} \sum_{U:T \setminus S \subseteq U \subseteq T} (-1)^{|S|-|T \setminus U|} \int \hat{m}_T^{(0)}(x_T) p_U(x_U) dx_U.$$

Proof. We consider a fixed $S \subseteq I_p$. We will make use of the fact that for a set $T \not\supseteq S$

$$\sum_{V \subseteq S} (-1)^{|S \setminus V|} \int \hat{m}_T^{(0)}(x_T) p_{-V}(x_{-V}) \mathrm{d}x_{-V} = 0, \tag{7.4}$$

and for a set $T \subseteq I_p, T \supseteq S$

$$\{U : T \setminus S \subseteq U \subseteq T\} = \{T \setminus V : V \subseteq S\} \tag{7.5}$$

Combining (7.4)–(7.5), we get

$$\begin{aligned}
&\sum_{V \subseteq S} (-1)^{|S \setminus V|} \int \hat{m}_n(x) p_{-V}(x_{-V}) \mathrm{d}x_{-V} \\
&= \sum_{T \subseteq I_p} \sum_{V \subseteq S} (-1)^{|S \setminus V|} \int \hat{m}_T^{(0)}(x_T) p_{-V}(x_{-V}) \mathrm{d}x_{-V} \\
&= \sum_{T \supseteq S} \sum_{V \subseteq S} (-1)^{|S|-|V|} \int \hat{m}_T^{(0)}(x_T) p_{T \setminus V}(x_{-T \setminus V}) \mathrm{d}x_{T \setminus V} \\
&= \sum_{T \supseteq S} \sum_{T \setminus S \subseteq U \subseteq T} (-1)^{|S|-|T \setminus U|} \int \hat{m}_T^{(0)}(x_T) p_U(x_U) \mathrm{d}x_U.
\end{aligned}$$

It is left to show (7.4)–(7.5). Equation (7.5) follows from straight forward calculations. To see 7.4, note

$$\begin{aligned}
&\sum_{V \subseteq S} (-1)^{|S \setminus V|} \int \hat{m}_T^{(0)}(x_T) p_{-V}(x_{-V}) \mathrm{d}x_{-V} \\
&= \sum_{U \subseteq S \cap T} \sum_{W \subseteq S \setminus T} (-1)^{|S \setminus \{W \cup U\}|} \int \hat{m}_T^{(0)}(x_T) p_{-\{U \cup W\}}(x_{-\{U \cup W\}}) \mathrm{d}x_{-\{U \cup W\}} \\
&= \sum_{U \subseteq S \cap T} \int \hat{m}_T^{(0)}(x_T) p_{-U}(x_{-U}) \mathrm{d}x_{-U} \sum_{W \subseteq S \setminus T} (-1)^{|S \setminus \{W \cup U\}|} \\
&= \sum_{U \subseteq S \cap T} \int \hat{m}_T^{(0)}(x_T) p_{-U}(x_{-U}) \mathrm{d}x_{-U} \left(\sum_{W \subseteq S \setminus T, |W| = \text{odd}} (-1)^{|S \setminus U|-1} + \sum_{W \subseteq S \setminus T, |W| = \text{even}} (-1)^{|S \setminus U|} \right) \\
&= 0,
\end{aligned}$$

where the last equality follows from the fact that every non-empty set has an equal number of odd and even subsets. □

While the idea of a well-behaved decomposition of the decision function $\hat{m}_n$ should in itself be interesting, it turns out that this particular decomposition has strong ties to both Shapley values and partial dependence plots.

Lemma 7.4.4 (PDP from marginal identification)**.** Let $\{\hat{m}_S^*\}$ be a functional decomposition of $\hat{m}_n$ that satisfies the marginal identification. It holds that

$$\hat{\xi}_S(x_s) = \sum_{T \subseteq S} \hat{m}_T^*(x_T),$$

where $\hat{\xi}_S(x_s) = \int \hat{m}_n(x_S, x_{-S}) \hat{p}_{X_{-S}}(x_{-S}) dx_{-S}$ is the emiprical PDP.

Proof. This is just equation (7.2). □

Corollary 7.4.5 (Interventional SHAP from marginal identification)**.** Let $\{\hat{m}_S^*\}$ bet the functional decomposition of $\hat{m})_n$ that satisfies the marginal identification. It holds that

$$\hat{\phi}_k(x) = \hat{m}_k^*(x_k) + \frac{1}{2} \sum_{j=1}^{p} \hat{m}_{k,j}^*(x_k, x_j) + \cdots + \frac{1}{p} \hat{m}_{I_p}^*(x),$$

where $\hat{\phi}_k = \int \hat{m}_n(x_1, \ldots, x_p) \hat{p}_{-S}(x_{-S}) \mathrm{d}x_{-S}$ is the empirical interventional SHAP value.

Proof. From Lemma 7.4.4 we have $\int \hat{m}_n(x) \hat{p}_{-U}(x_{-U}) \mathrm{d}x_{-U} = \sum_{T \subseteq U} \hat{m}_T^*(x_T)$. Hence the game $\hat{m}_n$ with value function

$$v_{\hat{m}_n}(U) = \int \hat{m}_n(x) \hat{p}_{-U}(x_{-U}) \mathrm{d}x_{-U}$$

equals the game $\hat{m}^*$ with value function

$$v_{\hat{m}^*}(U) = \sum_{S \subseteq I_p} \hat{m}_S^*(x_S) \delta_S(U), \quad \delta_S(U) = \mathbb{1}\{S \subseteq U\}.$$

We now concentrate on the function $\hat{m}_S^*$ with value function $\hat{m}_S^*(x_S)\delta_S(U)$. Combining the Linearity Axiom and step 2 of the proof of Theorem 7.1.2 we have that for every non-empty $S \subseteq I_p$,

$$\phi_k(x, \hat{m}_S^*(x_S)\delta_S(U)) = \mathbb{1}\{k \in S\} |S|^{-1} \hat{m}_S^*(x_S).$$

Here, $\phi_k(x, v)$ denotes the Shapley value for feature k at point x in a game with value function v. The proof is then completed by the linearity axiom, together with

$$\phi_k(x, \hat{m}_\emptyset^* \delta_\emptyset(U)) = \begin{cases} \hat{m}_\emptyset^*, & k = 0, \\ 0 & \text{else.} \end{cases}$$

The last statement follows from the dummy axiom. □

Thus, if one knows the marginal decomposition of $\hat{m}_n$, the empirical interventional SHAP value at x can be obtained by properly weighting the components evaluated in x.

7.4.2 Centered ALE plots

The accumulated local effect plots introduced in Section 7.3.3 can – as partial dependence plots – be connected to a functional decomposition. Indeed, we will now define centered ALE plots, g_S^{centered} such that

$$\hat{m}_n(x) = \sum_{S \subseteq I_p} g_S^{\text{centered}}(x_S).$$

To this end, define the linear operator $\mathcal{L}_S$ that maps a p-dimensional function to its uncentered ALE plot for the set S.

$$\mathcal{L}_S(f) = \int_{x_{S,\min}}^{x_S} \mathbb{E}\left[f^S(X_S, X_{-S}) \mid X_S = z_S\right] \mathrm{d}z_S.$$

As before, here f^S denotes the cross partial derivative of f and $\int_{x_{S,\min}}^{x_S}$ is an integral over the hyper-rectangle defined by the marginals (cf. Definition 7.3.3).

Definition 7.4.6. The centered ALE plot for a non-empty set of features S is defined as

$$g_S^{\text{centered}}(x_S) = \left[(I - \mathcal{L}_\emptyset) \circ \left(I - \sum_{T \subseteq S : |T|=1} \mathcal{L}_T\right) \circ \cdots \circ \left(I - \sum_{T \subseteq S : |T| = |S|-1} \mathcal{L}_T\right) \circ \mathcal{L}_S\right](\hat{m}_n)(x_S),$$

where I is the identity map. Furthermore

$$g_\emptyset^{\text{centered}} = \mathcal{L}_\emptyset(\hat{m}_n) = \mathbb{E}[\hat{m}_n].$$

Note in particular that $\mathcal{L}_\emptyset(f) = \mathbb{E}[f]$ such that for $k = 1, \dots, p$,

$$g_k^{\text{centered}}(x_k) = [(I - \mathcal{L}_\emptyset) \circ \mathcal{L}_k](\hat{m}_n)(x_k) = \mathcal{L}_k(\hat{m}_n)(x_k) - \mathcal{L}_\emptyset \circ \mathcal{L}_k(\hat{m}_n)(x_k) = g_k(x_k) - \mathbb{E}[g_k(x_k)].$$

We present the following theorem without proof (formally it is rather a conjecture as the original proof in Apley and Zhu (2020) is missing an essential part which has not been fixed yet up to my knowledge):

Theorem 7.4.7. It holds that

$$\hat{m}_n(x) = \sum_{S \subseteq I_p} g_S^{\text{centered}}(x_S).$$

Example 24. Let

$$\hat{m}_n(x_1, x_2) = x_1 + x_2 + 2x_1x_2,$$

and let (X_1, X_2) be multivariate standard normal with correlation equal 0.3. Noting that $\mathbb{E}[X_2|X_1 = x_1] = 0.3x_1$, we calculated in Example 23 that

$$g_1(x_1) = x_1 - x_{1,\min} + 0.3(x_1^2 - x_{1,\min}^2).$$

Hence,

$$\begin{aligned}
g_\emptyset^{\text{centered}} &= 0.6 \\
g_1^{\text{centered}}(x_1) &= x_1 - x_{1,\min} + 0.3(x_1^2 - x_{1,\min}^2) - \mathbb{E}[X_1 - x_{1,\min} + 0.3(X_1^2 - x_{1,\min}^2)] = x_1 + 0.3(x_1^2 - 1), \\
g_2^{\text{centered}}(x_1) &= x_2 + 0.3(x_2^2 - 1), \\
g_{12}^{\text{centered}}(x_1, x_2) &= \hat{m}_n(x) - g_\emptyset^{\text{centered}} - g_1^{\text{centered}}(x - 1) - g_2^{\text{centered}}(x_2) = x_1x_2 - 0.3(x_1^2 + x_2^2)
\end{aligned}$$

This can be compared to the marginal identification:

$$\begin{aligned}
m_0^* &= 0.6, \\
m_1^*(x_1) &= x_1 - 0.6, \\
m_2^*(x_2) &= x_2 - 0.6, \\
m_{12}^*(x_1, x_2) &= 2x_1x_2 + 0.6.
\end{aligned}$$

Chapter 8

Causality

In this chapter, we consider the topic of causality. We do not provide comprehend introduction to the field here, but just as much we need to derive some interesting results. The reader interested to learn more about the field of causality is referred to the excellent treatment by Peters et al. (2017).

Definition 8.0.1 (Graph Terminology)**.** We consider random variables $X = (X_1, ..., X_p)$ with index set $V := I_p = \{1, ..., p\}$. A graph $G = (V, \mathcal{E})$ consists of

- nodes or vertices V, and
- edges $\mathcal{E} \subseteq V^2$ with $(v, v) \notin \mathcal{E}$ for any $v \in V$.
- A node k is called a
  - parent of j if $(k, j) \in \mathcal{E}$ and $(j, k) \notin \mathcal{E}$.
    - The set of parents of k is denoted by $pa_G(k)$.
  - a child if $(j, k) \in \mathcal{E}$ and $(k, j) \notin \mathcal{E}$.
    - The set of childrens of k is denoted by $ch_G(k)$.
- Two nodes k and j are adjacent if either $(k, j) \in \mathcal{E}$ or $(j, k) \in \mathcal{E}$.
- We say that there is an undirected edge between two adjacent nodes k and j if $(k, j) \in \mathcal{E}$ and $(j, k) \in \mathcal{E}$.
- An edge between two adjacent nodes (k, j) is directed if $(k, j) \in \mathcal{E}$ and $(j, k) \notin \mathcal{E}$ or vice versa.
  - We write $k \to j$ for $(k, j) \in \mathcal{E}$, $(j, k) \notin \mathcal{E}$ and $j \to k$ for $(j, k) \in \mathcal{E}$, $(k, j) \notin \mathcal{E}$
- G is called directed if all its edges are directed.
- A path in G is a sequence of (at least two) distinct vertices $k_1, \dots, k_m$, such that there is an edge between k_l and k_{l+1} for all $l = 1, \dots, m-1$.
  - If $k_{l-1} \to k_l$ and $k_{l+1} \to k_l$ ($k_{l-1} \to k_l \leftarrow k_{l+1}$), k_l is called a collider relative to this path.
  - If $k_l \to k_{l+1}$ for all l, we speak of a directed path from k_1 to k_m
    - In this case, we call k_1 an ancestor of k_m and
    - k_m a descendant of k_1.
- G is called a directed acyclic graph (DAG) if all edges are directed and there is no pair (j, k) with directed paths from j to k and from k to j.

Definition 8.0.2 (Pearl's d-separation)**.** In a DAG G, a path between nodes k_1 and k_m is blocked by a set S (with neither k_1 nor k_m in S) if there is a node k_l fulfilling one of the two points:

(a) $k_l \in S$ and

$$\begin{aligned} & k_{l-1} \to k_l \to k_{l+1} \\ \text{or } & k_{l-1} \leftarrow k_l \leftarrow k_{l+1} \\ \text{or } & k_{l-1} \leftarrow k_l \to k_{l+1} \end{aligned}$$

b) neither k_l nor any of its descendants is in S, and

$$k_{l-1} \to k_l \leftarrow k_{l+1}$$

In a DAG G, we say that two disjoint subsets of vertices A and B are d-separated by a third (also disjoint) subset S if every path between nodes in A and B is blocked by S. We then write

$$A \perp\!\!\!\perp_G B \mid S$$

Definition 8.0.3 (Structural causal models)**.** A structural causal model (SCM) $\mathfrak{C} := (S, P_N)$ consists of a collection S of p (structural) assignments

$$X_j := f_j\left(pa(j), N_j\right), \quad j = 1, \dots, p,$$

and a distribution $P_N = \bigtimes_{j=1}^{p} P_{N_j}$ of jontly independent noise variables $N_1 \dots, N_p$.

Note that an SCM $\mathfrak{C}$ defines a unique graph G and a unique distribution, $P_X^{\mathfrak{C}}$, over the variables $X_1, ..., X_p$. The reverse is not true.

Example 25 (Structural causal models with acyclic graph structure)**.**

$$\begin{aligned} X_1 &:= f_1\left(X_3, N_1\right) \\ X_2 &:= f_2\left(X_1, N_2\right) \\ X_3 &:= f_3\left(N_3\right) \\ X_4 &:= f_4\left(X_2, X_3, N_4\right), \end{aligned}$$

$N_1, \dots, N_4$ jointly independent.

Definition 8.0.4 (Interventional distribution/ do-operator)**.** Consider an SCM $\mathfrak{C} := (S, P_N)$ and its entailed distribution $P_X^{\mathfrak{C}}$. We define a new SCM, $\tilde{\mathfrak{C}}$, by replacing the assignment for X_k by

$$X_k = \tilde{f}(\tilde{pa}(k), \tilde{N}_k).$$

The resulting entailed distribution of the new SCM is called interventional distribution. We write short:

$$do(X_k = \tilde{f}(\tilde{pa}(k), \tilde{N}_k)), \quad P_X^{\tilde{\mathfrak{C}}} = P_X^{\mathfrak{C}, do(X_k = \tilde{f}(\tilde{pa}(k), \tilde{N}_k))}$$

An intervention can also simply assign a fixed value a (i.e., $\tilde{N}_k = a$ (deterministic) and the set of parents is empty). This is called an atomic or deterministic intervention and can be denoted by $do(X_k = a)$.

In the following we will, with some abuse of notation, work with a generic probability density function $p(\cdot)$. For example $p(x_1) = p_{X_1}(x_1)$, while $p(x_1, x_2) = p_{X_1, X_2}(x_1, x_2)$.

Definition 8.0.5 (Markov property)**.** Given a DAG G and an entailed joint absolutely continuous distribution P_X, this distribution is said to satisfy

(i) the global Markov property with respect to the DAG G if

$$A \perp\!\!\!\perp_{\mathcal{G}} B | C \Rightarrow A \perp\!\!\!\perp B | C$$

for all disjoint sets of nodes A, B, C.

(ii) the local Markov property with respect to the DAG G if each variable is independent of its non-descendants given its parents, and

(iii) the Markov factorization property with respect to the DAG G if

$$p(x) = p(x_1, \dots, x_d) = \prod_{j=1}^{d} p\left(x_j \mid pa(j)\right).$$

We state the following theorem without proof.

Theorem 8.0.6 (equivalence of Markov properties, see e.g. Lauritzen (1996)). If P_X is absolutely continuous, then all three Markov properties above are equivalent.

Note that every SCM induced Graph satisfies the Markovian properties.

8.1 Adjustment Criteria

We start by defining what a valid adjustment set is.

Definition 8.1.1. Consider a SCM $\mathfrak{C}$ with nodes $X, Y \in V$. We say that $Z \subseteq V$ is a valid adjustment set for the causal effect of X on Y, if

$$p^{\mathfrak{C};do(X=x)}(y) = \int p^{\mathfrak{C}}(y \mid x, z) p^{\mathfrak{C}}(z) dz.$$

In other words - Conditioning on Z allows us to use the observational distribution $p^{\mathfrak{C}}$ to calculate probabilities in the interventional distribution $p^{\mathfrak{C};do(X:=x)}$.

To be able to characterize valid adjustment sets, we will make use of the following lemma.

Lemma 8.1.2. Consider a SCM $\mathfrak{C}$ and a subset of nodes $X_1, \dots, X_p \subseteq V$.

- If X_1 has no parents, then

$$p^{\mathfrak{C};do(X_1=a)}(x_2, \dots, x_p) = p^{\mathfrak{C}}(x_2, \dots, x_p \mid x_1 = a).$$

- If X_1 has no children, then

$$p^{\mathfrak{C};do(X_1=a)}(x_2, \dots, x_p) = p^{\mathfrak{C}}(x_2, \dots, x_p).$$

Proof. Both statements follow directly from the definitions of an SCM and an do-operator. □

With the above lemma, we are able to prove the parent adjustment and backdoor criteria.

Theorem 8.1.3 (Backdoor Criterion). Set $\mathfrak{C}$ be a SCM and let $X, Y \in V$ be nodes in the induced graph. A set $Z \subseteq V \setminus \{X, Y\}$ is a valid adjustment set for (X, Y) if

- Z contains no descendants of X, and
- Any path of the form $X \leftarrow \cdots Y$ is blocked by Z.

Proof. We wish to show that

$$p^{\mathfrak{C};do(X=x)}(y) = \int p^{\mathfrak{C}}(y \mid x, z) p^{\mathfrak{C}}(z) dz$$

Define the extended SCM $\tilde{\mathfrak{C}}$ analogously to $\mathfrak{C}$ with the following changes:

- We include the node I, which is assumed to have no parents.
- We change the structural assignment for X to be

$$X \leftarrow X\mathbb{1}(I = 0) + x\mathbb{1}(I = 1)$$

Hence setting $I = 0$ corresponds to not intervening on X, while $I = 1$ corresponds to intervening on X. The advantage of introducing this additional node is that we will be able to use it to explicitly connect the two SCM's $\mathfrak{C}$ and $\mathfrak{C}; do(X = x)$. Before we do so, we prove the following two d-separation statements:

$$Y \perp\!\!\!\perp_{\tilde{\mathfrak{C}}} I \mid X, Z \quad , \quad Z \perp\!\!\!\perp_{\tilde{\mathfrak{C}}} I \mid \emptyset$$

To prove the first statement, consider some path connecting Y and I. Since I is only connected to the remaining nodes through $I \to X$, the path must be of either of the forms

$$I \to X \to \cdots Y,$$
$$I \to X \leftarrow \cdots Y.$$

In case of the first form $\to X \to$ blocks the path, since we condition on X. For the second form, the path enters X through the backdoor $X \leftarrow$, hence by assumption that path is blocked by Z. Hence the path is blocked in all cases, and thus we get the first of the desired d-separation statements. For the other statement, note that any path connecting I to $Z_0 \in Z$ must be of the form $I \to X \cdots Z_0$. Note that if the path is directed, e.g. it is of the form

$$I \to X \to \cdots \to Z_0$$

we have that Z_0 is a descendant of X, which we know by assumption is not the case. Hence one of the edges must be a '$\leftarrow$', thus producing a collider in the path. Since we condition on $\emptyset$, any collider will block the path, thus we get the second d-separation statement. Having obtained the two d-separation statements, the Markov-property of $\tilde{\mathfrak{C}}$ implies that under the distribution implied by $\tilde{\mathfrak{C}}$

$$Y \perp\!\!\!\perp I \mid X, Z \quad , \quad Z \perp\!\!\!\perp I.$$

By lemma 8.1.2,

$$p^{\tilde{\mathfrak{C}}}(y \mid I = 0) = p^{\tilde{\mathfrak{C}}; do(I=0)}(y) = p^{\mathfrak{C}}(y)$$

as well as

$$p^{\tilde{\mathfrak{C}}}(y \mid I = 1) = p^{\tilde{\mathfrak{C}}; do(I=1}(y) = p^{\mathfrak{C}; do(X=x)}(y),$$

such that

$$\begin{aligned} p^{\mathfrak{C}; do(X=x)}(y) &= p^{\tilde{\mathfrak{C}}}(y \mid I = 1) \\ &= \int p^{\tilde{\mathfrak{C}}}(y \mid I = 1, z) p^{\tilde{\mathfrak{C}}}(z \mid I = 1) dz \\ &\overset{(\dagger)}{=} \int p^{\tilde{\mathfrak{C}}}(y \mid I = 1, z, x) p^{\tilde{\mathfrak{C}}}(z \mid I = 1) dz \\ &\overset{(\dagger\dagger)}{=} \int p^{\tilde{\mathfrak{C}}}(y \mid I = 0, z, x) p^{\tilde{\mathfrak{C}}}(z \mid I = 0) dz \\ &= \int p^{\mathfrak{C}}(y \mid, z, x) p^{\mathfrak{C}}(z) dz \end{aligned}$$

In (†) we used that $P^{\tilde{\mathfrak{C}}}(X = x \mid I = 1, Z = z) = P^{\mathfrak{C}; do(X=x)}(X = x \mid Z = z) = 1$ and the fact that conditioning on an almost-sure event does not change any probabilities. In (††) we used the two conditional independence statements derived above. Hence we arrive at the desired conclusion. □

Corollary 8.1.4. The set of parents of X, $Z = pa(X)$, satisfies the backdoor criterion.

Proof. Note that $Z = pa(X)$ fulfills the backdoor criterion. Hence this statement follows from Theorem 8.1.3. □

8.2 Average Natural Direct Effect

In the previous section, we considered random variables $(X_1, \dots, X_{\tilde{p}})$ with a structural causal model with acyclic graph structure. At first glance, this could seem quite restrictive because (a) no cycles are permitted (in real life we often observe feedback loops) (b) no unmeasured confounders are allowed because the noise variables are independent. However both (a) and (b) may not be too strong, if we allow $(X_1, \dots, X_{\tilde{p}})$ to be random vectors.

Example 26. Consider a supervised learning problem with iid observations of $(X_1, \dots, X_p, Y)$. We divide the features into two sets (we assume that they are already ordered the right way):

$$M = \{X_1, \dots, X_q\} \quad W = \{X_{q+1}, \dots, X_p\}.$$

We can then consider the following structural causal model

$$\begin{aligned} W &= \varepsilon_W, \\ M &= f_M(W) + \varepsilon_M, \\ Y &= f_Y(W, M) + \varepsilon_Y, \end{aligned}$$

where $\varepsilon_W, \varepsilon_M, \varepsilon_Y$ are independent random variables. In this setting, the variables within W and within M are allowed to build cycles. There can also be unmeasured confounding in the sense that there can for example be an unobserved variable U that affects both X_1, X_2, if $q \leq 2$.

We now come to the definition of an average natural direct effect. Consider variables $(X_1, \dots, X_p, Y)$. We may be interested in a set of variables $S \subseteq I_p$. Assume that the response depends on the covariates via the structural assignment

$$Y = m(X) + \varepsilon \quad \text{and} \quad \varepsilon \perp\!\!\!\perp X. \tag{8.1}$$

This models both the conditional distribution of Y on $X = (X_S, X_{-S})$ and any intervention on all covariates $X = (X_S, X_{-S})$ simultaneously. In contrast to a full causal model of all variables, this model does not capture how a marginal intervention, e.g. on X_S or variable within X_S, affects the response (as this requires modeling how the marginal intervention affects the remaining covariates). Starting from such a model with a fixed m, we can define the notion of average natural direct effect of setting the value of X_S from some x'_S to x_S.

Definition 8.2.1 (average natural direct effect)**.** Assume the causal structure (8.1). The average natural direct effect of a set of variables $S \subseteq I_p$ on Y is

$$\begin{aligned} m^*_{X_S}(x_s) = &\int p^{\mathrm{do}(X_S = x'_S)}_{X_{-S}}(x_{-S}) p_{X_S}(x'_S) m(x_S, x_{-S}) \mathrm{d}x'_S \mathrm{d}x_{-S} \\ &- \int p^{\mathrm{do}(X_S = x'_S)}_{X_{-S}}(x_{-s}) p_{X_S}(x'_S) m(x'_S, x_{-S}) \mathrm{d}x'_S \mathrm{d}x_{-S}. \end{aligned}$$

In the definition of the average natural direct effect the left term indicates the expected value of Y if X_S is x_S but X_{-S} behaves as if X_S was intervened on to be x'_S. The right term subtracts the expected value of Y if X_S was intervened on to be x'_S. Different values of x'_S are drawn from the marginal distribution p_{X_S} and averaged.
Note that the effect of the intervention $do(X_S = x'_S)$ is not described via equation (8.1), and an additional causal structure for X will be needed in order to calculate the average natural direct effect. The next theorem describes that the natural direct effect of X_S can be identified from our observations if the features in S can be divided into two sets that together with X_{-S} form a structural causal model with the following property: The first set is not a descendent of X_{-S} and X_{-S} is not a descendent of the second set.

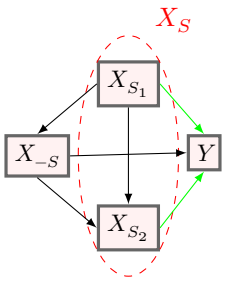

Figure 8.1: Causal structure for calculating the natural direct effect of X_S.

Theorem 8.2.2. Assume the causal structure (8.1). Furthermore, for a set $S \subseteq I_p$, assume that $S = S_1 \cup S_2$, $S_1 \cap S_2 = \emptyset$, with causal structure

$$\begin{aligned} X_{S_1} &= \varepsilon_{S_1} \\ X_{-S} &= m_{-S}(X_{S_1}) + \varepsilon_{-S} \\ X_{S_2} &= m_{S_2}(X_{S_1}, X_{-S}) + \varepsilon_{S_2}, \end{aligned}$$

with $\varepsilon, \varepsilon_{S_1}, \varepsilon_{-S}, \varepsilon_{S_2}$ independent. Then, up to a constant $c \in \mathbb{R}$, the average natural direct effect of S is equal to the partial dependence plot of S with respect to the function m:

$$m^*_{X_S}(x_S) = \int p_{X_{-S}}(x_S) m(x_S, x_{-S}) \mathrm{d}x_{-S} + c = \xi_S(x_S) + c.$$

Proof. Straight forward calculations yield

$$\begin{aligned} m^*_{X_S}(x_S) &= \int p^{\mathrm{do}(X_S = x'_S)}_{X_{-S}}(x_{-S}) p_{X_S}(x'_S) m(x_S, x_{-S}) \mathrm{d}x'_S \mathrm{d}x_{-S} + c \\ &= \int p_{X_{-S}|X_{S_1}}(x_{-S}|x'_{S_1}) p_{X_S}(x'_{S_1}, x'_{S_2}) m(x_{S_1}, x_{S_2}, x_{-S}) \mathrm{d}x'_{S_1} \mathrm{d}x'_{S_2} \mathrm{d}x_{-S} \\ &= \int p_{X_{-S,S_1}}(x_{-S}, x'_{S_1}) m(x_{S_1}, x_{S_2}, x_{-S}) \mathrm{d}x'_{S_1} \mathrm{d}x_{-S} \\ &= \int p_{X_S}(x_S) m(x_S, x_{-S}) \mathrm{d}x_{-S} \\ &= \xi_S(x_S). \end{aligned}$$

□

Remark (explaining a black box)**.** The causal structure (8.1), allows to consider $Y = m = \hat{m}_n(X)$, where $\hat{m}_n(X)$ is a black box we wish to explain. Next, consider a functional decomposition of $\hat{m}_n(X)$

$$\hat{m}_n(x) = \sum_{T \subseteq I_p} \hat{m}_T(x_T),$$

via the marginal identification with $\hat{p} = p$, i.e.,

$$\sum_{T \cap S \neq \emptyset} \int \hat{m}_T(x_T) p_S(x_S) \mathrm{d}x_S = 0,$$

for all $S \subseteq I_p$. For a fixed set $S \subseteq I_p$, we may also write

$$\hat{m}_n(x) = \hat{m}_0 + \widetilde{m}_S(x_S) + \widetilde{m}_{-S}(x_{-S}) + \widetilde{m}_{S,-S}(x_S, x_{-S}),$$

where

$$\widetilde{m}_S(x_S) = \sum_{T \subseteq S, T \neq \emptyset} \hat{m}_T(x_T),$$
$$\widetilde{m}_S(x_{-S}) = \sum_{T \subseteq -S, T \neq \emptyset} \hat{m}_T(x_T),$$
$$\widetilde{m}_{S,-S}(x_S, x_{-S}) = \sum_{T: T \not\subseteq S, T \not\subseteq -S, T \neq \emptyset} \hat{m}_T(x_T).$$

If S fulfils the causal structure as stated in Theorem 8.2.2, then $\widetilde{m}_S(x_S)$ is up to a constant the average natural direct effect of X_S on the black box $\hat{m}_n(X)$.

$$m^*_{X_S}(x_s) = \widetilde{m}_S(x_S) + c.$$

Note that also $-S$ can fulfill the equivalent causal structure at the same time as S. In that case both $\widetilde{m}_S(x_S)$ and $\widetilde{m}_{-S}(x_{-S})$ have the interpretation of being up to constants the average natural direct of their features. The component $\widetilde{m}_{S,-S}(x_S, x_{-S})$ is then the remaining interaction that cannot be attributed to a direct effect of either set of variables.

8.3 Fairness via Average Natural Direct Effect

Consider the case where we have a separation of the feature space into X, A, where X is the set of 'permitted' features and A is the set of 'protected' features. What it means for a feature to be permitted/protected is a matter of debate, but in a rough sense, it means that we don't want our model to discriminate based on the information contained in the features belonging to A.

One may think that the simplest way to avoid discrimination based on A is to not use A for prediction. While this works if X and A are uncorrelated, indirect discrimination may occur if X and A are correlated. This is also known as proxy discrimination.

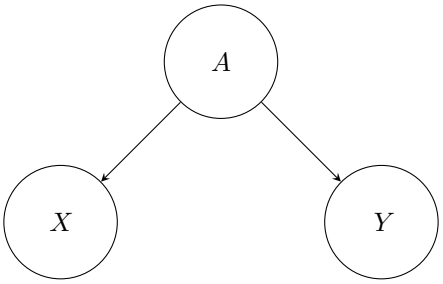

Figure 8.2: The implied DAG for the Gender-Height-Shopping example.

Example 27. Assume that $A =$ `Gender`, $X =$ `Height` and $Y =$ `number of car accidents in a year`. We would like to consider a predictive model for Y based on A, X, but assume that A is a protected feature, since we don't want to discriminate based on gender.
Let us assume the underlying simple model to be

$$Y = \mathbb{1}\{A = \texttt{Male}\}Z_1 + \varepsilon_Y,$$
$$X = \mathbb{1}\{A = \texttt{Male}\}180 + \mathbb{1}\{A = \texttt{Female}\}160 + \varepsilon_X,$$
$$Z_1 \sim Poisson(0.1), \quad \varepsilon_Y \sim Poisson(0.05), \quad \varepsilon_X \sim N(0, 10^2),$$

corresponding to

- males are generally taller than females.
- Females generally have less accidents than males.

Note that X has no causal effect on Y, but it carries information about A, since observing a tall person would imply that the person is probably male, and thus will have more accidents. Hence, only using X for prediction will most probably lead to indirect discrimination.

The example brings to light the idea of indirect discrimination. It is the notion that the statistical dependence between features may allow a model to infer the values of protected features and discriminate based on the inferred values even if these are not provided to the model.

Using the notion of an average natural direct effect, we can define a corresponding fair estimator.

Definition 8.3.1. We call a predictor $(A-X)$ fair if it can be written as a linear transformation of the average natural direct effect of X, i.e., there are constants $\alpha, \beta \in \mathbb{R}$, such that the predictor equals

$$\alpha + \beta m_X^*(x).$$

Note that this definition of fairness is of course not the only possible definition of a fair prediction. It also does not always align with what can be socially expected to be fair. But within this lecture, we will concentrate on this definition. The idea of the defined notion of fairness is that one shall only use the direct effect of X on Y for prediction. In particular, one shall not use A neither directly nor indirectly.

Remark. Let $m(x,a) = \mathbb{E}[Y|X = x, A = a]$. We can look at the functional decomposition with marginal identification where components belonging to X and A are summed together:

- $m_0 = \mathbb{E}[Y]$,
- $m_X(x) \mapsto \int m(x,a)p_A(a)da - m_0$,
- $m_A : a \mapsto \int m(x,a)p_X(x)dx - m_0$,
- $m_{X,A} : (x,a) \mapsto m(x,a) - m_X(x) - m_A(a) - m_0$.

Using these functions, it holds that

$$m(x,a) = m_0 + m_X(x) + m_A(a) + m_{X,A}(x,a).$$

If the assumptions of Theorem 8.2.2 are satisfied with $S = X$, then an (A-X) fair estimator is given by

$$m_0 + m_X(x).$$

The problem here is that neither the function m nor the distribution of (X, A, Y) is known, which are needed to calculate m_0 and $m_X(x)$. In practice, one can replace the integral above with the corresponding empirical average

$$m_0 + m_X(x) \approx \hat{\xi}_X(x) = \frac{1}{n}\sum_{i=1}^{n} \hat{m}(x, A_i),$$

where $\hat{\xi}_X(x)$ is the empirical partial dependence plot of X. Note however that there exist more efficient estimators for the partial dependence plot than its empirical version.

Appendix A

Technical results

Lemma A.0.1. Let A be any finite set. Then

$$\mathbb{1}\{A = \emptyset\} = \sum_{D \subseteq A} (-1)^{|D|}$$

Proof. Note initially that for $A = \emptyset$, the result trivially holds. Assume now that $A \neq \emptyset$ e.g. that $|A| \in \mathbb{N}$. Initially note that

$$\sum_{D \subseteq A} (-1)^{|D|} = |\{D \subseteq A \mid |D| \text{ even}\}| - |\{D \subseteq A \mid |D| \text{ odd}\}|$$

Hence to prove the result, we should prove that is an equal number of odd-sized and even-sized subsets of A. We will do this by induction over $k = |A|$. For $k = 1$, we have $\mathcal{P}(A) = \{\emptyset, A\}$, and hence there is one even-sized subset $\emptyset$ and one odd-sized subset A.

Assume now that there are equally many odd-sized and even-sized subsets for $|A| = k$. Consider some A with $|A| = k+1$. Take some arbitrary $a \in A$ and note that $A = \{a\} \cup (A \setminus \{a\})$ with $|A \setminus \{a\}| = k$. Hence by induction there is equally many even- and odd-sized subsets of $A \setminus \{a\}$. Note that any subset of A is produced from a subset of $A \setminus \{a\}$ by either including a or not including a. Hence the four systems

$$\begin{aligned}
&\{D \subseteq A \mid a \in D, |D| \text{ odd}\},\\
&\{D \subseteq A \mid a \notin D, |D| \text{ odd}\},\\
&\{D \subseteq A \mid a \in D, |D| \text{ even}\},\\
&\{D \subseteq A \mid a \notin D, |D| \text{ even}\}
\end{aligned}$$

are all of equal size, since each of them are derived from either $\{D \subseteq A \setminus \{a\} \mid |D| \text{ odd}\}$ or $\{D \subseteq A \setminus \{a\} \mid |D| \text{ even}\}$. This implies that the systems

$$\begin{aligned}
\{D \subseteq A \mid |D| \text{ even}\} &= \{D \subseteq A \mid a \in D, |D| \text{ even}\} \cup \{D \subseteq A \mid a \notin D, |D| \text{ even}\}\\
\{D \subseteq A \mid |D| \text{ odd}\} &= \{D \subseteq A \mid a \in D, |D| \text{ odd}\} \cup \{D \subseteq A \mid a \notin D, |D| \text{ odd}\}
\end{aligned}$$

are of equal size, hence the induction is complete. □

Bibliography

Apley, D. W. and J. Zhu (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology 82*(4), 1059–1086.

Bates, S., T. Hastie, and R. Tibshirani (2023). Cross-validation: what does it estimate and how well does it do it? *Journal of the American Statistical Association*, 1–12.

Bischl, B., M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, et al. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 13*(2), e1484.

Charnes, A., B. Golany, M. Keane, and J. Rousseau (1988). Extremal principle solutions of games in characteristic function form: core, chebychev and shapley value generalizations. *Econometrics of planning and efficiency*, 123–133.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Freund, Y. and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55*(1), 119–139.

Friedman, J., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics 28*(2), 337–407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Györfi, L., M. Kohler, A. Krzyzak, H. Walk, et al. (2002). *A distribution-free theory of nonparametric regression*, Volume 1. Springer.

Hastie, T., R. Tibshirani, and M. Wainwright (2015). *Statistical learning with sparsity: the lasso and generalizations*. CRC press.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S*, pp. 249–307. Routledge.

Janzing, D., L. Minorics, and P. Blöbaum (2020). Feature relevance quantification in explainable ai: A causal problem. In *International Conference on artificial intelligence and statistics*, pp. 2907–2916. PMLR.

Lauritzen, S. L. (1996). *Graphical models*, Volume 17. Clarendon Press.

Lundberg, S. M., G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee (2020). From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence 2*(1), 56–67.

Peters, J., D. Janzing, and B. Schölkopf (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Silverman, B. W. (1984). Spline smoothing: the equivalent variable kernel method. *The annals of Statistics*, 898–916.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*(1), 1929–1958.

Stone, C. J. (1985). Additive regression and other nonparametric models. *The annals of Statistics 13*(2), 689–705.

Taufiq, M. F., P. Blöbaum, and L. Minorics (2023). Manifold restricted interventional shapley values. In *International Conference on Artificial Intelligence and Statistics*, pp. 5079–5106. PMLR.

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*, Volume 48. Cambridge university press.

Wang, X., P. Du, and J. Shen (2013). Smoothing splines with varying smoothing parameter. *Biometrika 100*(4), 955–970.

Wright, M. N. and I. R. König (2019). Splitting on categorical predictors in random forests. *PeerJ 7*, e6339.