The handle http://hdl.handle.net/1887/33204 holds various files of this Leiden University dissertation

Author: Ommen, Thijs van
Title: Better predictions when models are wrong or underspecified
Issue Date: 2015-06-10
Chapter 1

Introduction

Both statistics and machine learning deal with the question of how humans and computers can learn from data. In large amounts of data, we hope to find patterns that express the data’s most important characteristics, and that can be used to make predictions about future data, or to help us understand the process that generated the data better.

To approach such a task, we usually need to draw a line between aspects of reality that we will take into consideration and aspects we will ignore. Once this is done, we can make a model: a simplified description of the part of reality we are interested in. For example, a model used to predict how well a patient will respond to a certain medicine might use that person’s age, gender, and clinical measurements such as blood pressure. Then based on previous observations of other patients and how well they responded to the medicine (the data), the model can be used to predict this for a new, previously unobserved patient. To predict how well the same person will like a movie (such as done by Netflix), we would likely employ a very different model. This model may also use the person’s age and gender. But for such a model, information about other movies that person likes will be very relevant, while white blood cell count will not be so useful.

Ideally, the model includes all aspects of reality we believe might be relevant for the question we want to answer. In practice, this is not always possible. Further simplification may be necessary if the phenomenon under study is too complex to capture entirely, or if with a bigger model, it would take far too long to compute the answer to our question of interest. This dissertation is about understanding what it means for our learning effort if some relevant aspects of reality are not included in the model, or are included in an overly simplified manner.

We already mentioned that one task for which a model might be used is prediction. In this dissertation, we will measure the success at our learning task by our ability to predict new data coming from the same source. Let us look at two more examples illustrating how the ability to predict can be a good measure of learning:
Chapter 1. Introduction

As a first example, one way to understand how a program such as zip can compress files is through prediction. While reading through the file to be compressed, before seeing each character in the file, the program decides for each possibility how it will be encoded (with what sequence of bits in the compressed file). It makes this decision using a prediction of what characters are more likely to occur next: these will be encoded using shorter sequences of bits. If the character that actually appears was likely according to the compressor's prediction, this saves a few bits in the compressed file (Cover and Thomas, 1991).

As a second, much more general example, the predictions of a good scientific theory will correspond to observed data. If new data are observed that differ significantly from the theory's predictions, this is a motivation to find improvements to the theory, or even altogether new theories, that predict the data better. For example, Newton's laws of gravitation can be used to predict the motion of the planets around the sun accurately. But for the planet Mercury, which is closest to the sun, the predictions were very slightly off. In 1915, Einstein's general theory of relativity provided new predictions, which did match the observations of Mercury's orbit; this theory has continued to predict the effects of gravity in the century since then (Misner et al., 1973).

The models we consider in this dissertation are statistical models. Such models give a description of part of reality in the language of mathematics, so that the predictions we are interested in can be found by means of computation. A statistical model consists of several possible explanations, or hypotheses. Each hypothesis is a probability distribution that describes one way in which the data-generating process might work. (One reason for using probabilistic hypotheses is to account for measurement errors in the data.) A model is thus a set of probability distributions.

In addition to the model, we need a method that describes how the data and the model should be used: what computations should be performed to find our prediction?

Whenever a model does not incorporate all aspects of reality that are relevant to our learning task, the model is wrong, underspecified, or both. By wrong,\(^1\) we mean that the model does not contain a hypothesis which exactly describes the true data-generating process (though it may contain hypotheses that are, in some sense, good approximations to the truth). For example, a medical model like the one we mentioned earlier may predict that a specific medicine is more likely to be effective for patients with higher blood pressures, when in fact this is true only up to a point, above which the medicine becomes less effective again. By underspecified, we mean that the hypotheses in the model describe the full process of data generation for only part of the data. For example, the medical model predicts how well a medicine will work given the characteristics of a new patient, but that same model might not predict the characteristics of this patient.

In this dissertation, we discuss three scenarios involving models that are wrong or underspecified. In each case, we find that standard methods for

\[^1\]This is also called misspecified.
1.1 Regression

A regression problem is a kind of learning problem in which the data consist of two parts: the input $X$ and the output $Y$ (Hastie et al., 2001). The task at hand is to predict the unknown value of $Y$ given a known value of $X$. The two examples from the beginning were regression problems: in the medical example, $X$ describes characteristics of a patient and $Y$ is the effectiveness of the medicine for that patient; in the Netflix example, $X$ consists of different characteristics of a user, and $Y$ is the rating that user would assign to some movie.

Figure 1.1 shows a much simpler but illustrative example where $X$ and $Y$ are single numbers. We have observed 100 data points: the training data. For learning from these data, two different models have been tried. Model 1, the simpler of the two, considers all linear functions as possible explanations (hypotheses) of how $Y$ is generated given $X$. The straight blue line is a hypothesis from the model that might be used to predict $Y$ for new data points (the
Model 2 is more complex than model 1: it includes all hypotheses from model 1, but also curved lines (quadratic functions). If we use this model to predict new values of $Y$, we might use the red line in the figure.

In the regression problems we consider, the output variable is a single number, and the input variable may be either a single number or a vector of numbers.

We will consider *linear models*. Each hypothesis in a linear model can be described by a vector of parameters $\beta$. The hypothesis corresponding to $\beta$ then states that for a data point with input variable $X$ consisting of the $p$ numbers $X_1, X_2, \ldots, X_p$, the output variable $Y$ is given by

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon.$$

Here $\epsilon$ is a random noise term, and the other terms on the right hand side define the regression curve.

This type of model is much less restrictive than it may at first appear, because we still have a lot of freedom in choosing the values we put in the input vector $X$. For example, if we believe some real-world quantity $S$ should be taken into account by the model but do not believe that $S$ and $Y$ are linearly related, we may let $X_i = S^i$ for $i \in 1, \ldots, p$: then the model contains a hypothesis for each polynomial up to degree $p$. Model 2 from the figure is an example of this, so it is also a linear model.

After observing a number of data points, we will see that some hypotheses fit the data better than others. Thus we can learn the parameter vector $\beta$ from the data. One method that can be used for this task is to find the maximum likelihood parameter: the value of $\beta$ for which the probability of observing the data would be largest. This is the main approach in the first of our two scenarios (and the lines shown in Figure 1.1 were found this way). In the second scenario another approach is used, which will be explained in Section 1.1.2.

To fully specify a probability distribution of the output variable given the input variable, the model must also describe the distribution from which the noise term $\epsilon$ is drawn. We may for example assume these follow a Gaussian (also called normal) distribution. For this distribution, we still need to specify a variance. Maybe we have a good idea of what this variance is; then we can just let all hypotheses in the model use this fixed value. Alternatively, our model may include a parameter for the variance in addition to the parameter vector $\beta$ that describes the regression curve; then we can learn the variance from the data using our model.

Besides learning the parameters within a model, we may also face the problem of model selection. It is often impossible in practice to formulate and use a model that incorporates all aspects of reality we think might be relevant, so that any model we end up using must be a compromise. It may not be clear in advance how to make this compromise: a model that is too ‘simple’ may not contain any hypotheses that even resemble the data-generating distribution, while a model that is too ‘complex’ may contain so many hypotheses that it may be nearly impossible to learn which ones are good. So we may want
to consider a set of candidate models, and learn from the training data which one to use to make our predictions. (Alternatively, we may want to combine the predictions from all these models into a single prediction, giving more weight to some models and less weight to others.) Even with the help of the training data, choosing a model can be a difficult task, as it is in Figure 1.1. The red line was chosen from a larger set of options, and so could be chosen to fit the data we have seen slightly better than the blue line. But how can we tell if model 2 will also be better for data points we have not seen yet?

Again, many different statistical methods exist that answer these questions in as many different ways. In the two regression scenarios, two different methods take the spotlight: AIC in Chapter 2, and Bayesian model averaging in Chapters 3 to 5. We introduce these scenarios in the next two sections.

1.1.1 Extra-sample prediction

You are given 100 data points \((X, Y)\) as shown in Figure 1.1, and are asked to predict \(y'\) for a new point with \(x' = 2.5\) (marked by the dashed vertical line). Do you base your prediction on the linear or the quadratic model? This is the main question of Chapter 2.

The regression models we consider may be regarded as underspecified, because they only describe how the output is generated given the input, but not how the input is generated. This is enough to have each model give predictions of \(Y\) given \(X\) (the behaviour of the inputs is not relevant for this task), so this is not a failing of the models. But to assess which model might give the best predictions on future ‘test’ data, we would need some information (for example in the form of predictions from an auxiliary model, independent of the other models) on the distribution of inputs (for this task, that information is relevant). Standard methods for choosing a model or averaging over models do not consider such information.

In Chapter 2, we take the classical method AIC (Akaike’s Information Criterion;\(^2\) Akaike (1973)), and adapt it to take information on the inputs into consideration. We call our modification \(\text{XAIC}\).

When comparing AIC and \(\text{XAIC}\), we see that AIC is based on an estimate of the prediction error that would be obtained if the test inputs were randomly chosen from the inputs already seen in the training sample (the in-sample error). \(\text{XAIC}\) does not assume that the test inputs will be so alike the training inputs, and computes the extra-sample error. (\(\text{XAIC}\) stands for eXtra-sample AIC.) If the training and test inputs are drawn from the same probability distribution, the difference between the two error measures becomes smaller and smaller as the number of training data points grows. So in this case, AIC eventually performs well. However, the difference between AIC and \(\text{XAIC}\) never vanishes entirely, even in this case; in a different case where training and test inputs do not come from the same distribution, it becomes very important to estimate the extram-sample and not the in-sample error, and \(\text{XAIC}\) is strongly recommended over AIC.

---

\(^2\)According to Akaike himself, AIC stands for ‘An Information Criterion’.
XAIC requires some information about test inputs. Predicting these with an auxiliary model may be hard (and is not the problem we are really interested in, namely predicting outputs given inputs). So one special case of XAIC is of special interest: FAIC (Focused AIC), which does not require predictions of $X$, but tailors its model selection choice to each possible value of the test input: for the same training data but different test inputs $X = x$, a different model may be selected. This makes FAIC radically different from most model selection methods,\(^3\) which usually choose a model (or a weighting over models) without any knowledge of the task for which this model will then be used. If instead we take the prediction task as fundamental and view model selection as part of this task, then focused model selection is a natural approach.

For the data shown in Figure 1.1, AIC tells us to use the more complex model 2 for all our predictions, even though the data were actually generated according to the simpler model 1. FAIC will also use model 2 for test inputs near 0 (where the red and blue regression curves are very similar), but will use model 1 for test inputs farther away from 0, like $x' = 2.5$.

The method AIC has existed for over forty years, and in that time, many variations and improvements have been proposed: AIC\(_C\) (Hurvich and Tsai, 1989), BPIC (Ando, 2007), DIC (Spiegelhalter et al., 2002), GIC (Konishi and Kitagawa, 1996), NIC (Murata et al., 1994), TIC (see Burnham and Anderson, 2002), WAIC (Watanabe, 2010), …. Thus a reader might think that XAIC is just another slight variation. However, none of these criteria address the issue that XAIC and FAIC address. Thus XAIC is certainly not ‘yet another information criterion’.\(^4\)

Some more details on our approach are given in Section 1.1.3. In Chapter 2, we compare XAIC and FAIC to other methods, both theoretically and experimentally.

### 1.1.2 Bayesian inconsistency

The Bayesian approach to dealing with uncertainty is to represent it in terms of a probability distribution. For example, if we think one of the models in a regression problem is correct but we do not know which one, we may make our uncertainty precise by specifying a prior distribution over the models. This prior may put the same amount of probability mass on each of the models to represent that we think they are all equally likely to be the correct model, or it may put more mass on some than on others.

After seeing data, we can use Bayes’ theorem to compute a posterior distribution over the models. This distribution tells us exactly what would be rational to believe after seeing the data, if our initial beliefs corresponded to the prior distribution (Bernardo and Smith, 1994).

---

\(^3\)One exception is the Focused Information Criterion (FIC) of Claeskens and Hjort (2003), which shares the idea of ‘focus’ on a prediction task of interest, and hence gave FAIC its name; yet it works out this idea in a completely different manner.

\(^4\)That would be YAIC.
1.1. Regression

Figure 1.2: Bayesian inconsistency in regression: When the training data contain many 'easy' points (at (0, 0)), the Bayesian posterior assigns most of its mass to the most complex models available, even though a much simpler model is actually much closer (under a variety of distance measures) to being correct.

Instead of using AIC or XAIC as in Chapter 2, we may use the posterior distribution over the models to select a model for giving predictions, or a weighting over the models. The latter approach is a natural way of taking into account all the uncertainty over the choice of model represented by the posterior. This approach is called Bayesian model averaging (BMA). This is the method for dealing with multiple models we investigate in Chapters 3, 4 and 5.

Similarly to putting a prior on the models, we may put a prior on the hypotheses within each model. Then the posterior over the hypotheses can be used to form the Bayesian predictive distribution, which can be used in place of the prediction suggested by the maximum likelihood parameter. The predictive distributions of all our models can be combined into a single overall predictive distribution by BMA.

Instead of considering multiple models, we can also use a single complex model, and use the prior on that model’s hypotheses to express that we find ‘simpler’ a hypothesis (that is, with parameter value $\beta$ closer to 0) more likely. This Bayesian ridge regression setting is also investigated in Chapters 3 to 5.

We conduct two kinds of experiments. In one kind, one of the models is correct: the data are generated by a distribution in one of the models (in Figure 1.2 it is in fact generated by $Y = 0 +$ Gaussian noise, which corresponds to the simplest model under consideration). As predicted by the theory, in these experiments, Bayesian inference methods perform very well. This can be seen in Figure 1.2, where the green line is always close to $Y = 0$.

In the other kind of experiment, half the data are generated by the same model as before, but the other half are all in the same place: (0, 0). With the
inclusion of these data points, all models are wrong, because the models predict that all data points will have the same noise level regardless of the input. However, the point \((0, 0)\) is on the same regression curve as the other data (the black line at \(Y = 0\) in Figure 1.2) and has no noise, so it would seem that these data points should only make the problem easier. It turns out this is not the case for Bayesian model averaging: it severely overfits, which essentially means that it gives far too much attention to other, complex models, as the red line in Figure 1.2 shows.

It is known from other work that Bayesian inference can be made robust against this kind of misbehaviour by selecting a small enough learning rate (Walker and Hjort, 2002; Zhang, 2006a; Grünwald, 2012), and using the generalized Bayesian posterior with this learning rate (instead of the standard Bayesian posterior, which corresponds to a learning rate of 1). With a properly chosen learning rate, Bayes learns to predict as well as the best hypothesis in the model (if appropriate conditions hold). By comparison, under weak conditions, standard Bayes learns to predict as well as the true distribution if it is in the model, but cannot be guaranteed to find a good hypothesis in general if the model is wrong.

To put this theory into practice, we need methods that can determine a good value for the learning rate, tuned for the data and the model. In Chapters 3 to 5, we consider several related methods for this task and evaluate them experimentally in regression problems. We use the name SafeBayes for these methods. The blue line in Figure 1.2 shows that SafeBayes predicts very well (close to the true \(Y = 0\)) in the wrong-model experiment.

### 1.1.3 Details on XAIC and SafeBayes

In a regression problem, there are infinitely many possible outcomes for \(Y\). No matter how good a prediction method is, it cannot be expected to predict the true outcome exactly. So we cannot measure the quality of predictions by simply counting the number of mistakes. Instead, we need other loss functions to compare different predictions.

We distinguish two types of predictions. A point prediction simply states a value for the outcome. The closer the predicted value \(\hat{y}\) is to the actual value \(y\), the better the prediction is, and the smaller the loss. A standard loss function is squared error loss, which assigns a loss of \((y - \hat{y})^2\) to this prediction.

The alternative to a point prediction is a probabilistic prediction, which takes the form of a probability distribution over all possible outcomes. Such predictions allow the predictor to express his degree of uncertainty over different possible outcomes. For example, a Gaussian prediction with a small variance represents that the actual outcome is expected to be very close to the mean of the distribution, while a large variance means that the predictor would not be surprised if the outcome turns out to be far away.

A loss function that is commonly used to evaluate probabilistic predictions is logarithmic loss (Bernardo and Smith, 1994). If the prediction is given by a

\[5\text{In the sense of KL divergence (see Section 1.1.3).}\]
density function \( \hat{f} \) and \( y \) is the actual outcome, this loss function assigns a loss of \(- \log \hat{f}(y)\) to the prediction. Thus the larger the probability density that the predictor gave to the actual outcome, the smaller the loss. (Loss functions for probabilistic predictions are often called scoring rules, but we will usually just call them loss functions.)

For Gaussian predictions with mean \( \hat{y} \) and fixed variance, logarithmic loss behaves very much like squared error loss: logarithmic loss is then also a quadratic function of \( y \) and \( \hat{y} \), minimized when \( y = \hat{y} \), so that a prediction that minimizes (an expectation over) one of these two loss functions also minimizes (the same expectation over) the other. This holds for arbitrary expectations: the true density does not need to be a Gaussian. If we consider Gaussian probabilistic predictions with different variances, or even altogether different distributions, the two loss functions become less closely related.

Logarithmic loss arises as a natural choice of loss function for comparing probabilistic predictions in several settings. For example, it is a fundamental quantity in data compression (Cover and Thomas, 1991). It is also related to the information-theoretic concept of Kullback-Leibler (KL) divergence. The KL divergence between true density \( f^* \) and predicted density \( \hat{f} \) is

\[
D(f^* \| \hat{f}) := \mathbb{E}_{Y \sim f^*} \left[ \log f^*(Y) - \log \hat{f}(Y) \right] = \mathbb{E}_{Y \sim f^*} \left[ - \log \hat{f}(Y) \right] - \mathbb{E}_{Y \sim f^*} \left[ - \log f^*(Y) \right].
\tag{1.1}
\]

This can be thought of as a distance measure between distributions. We see that the first term in (1.1) is the expected logarithmic loss of \( \hat{f} \), and the second term does not depend on \( \hat{f} \). Thus the smaller the expected logarithmic loss of \( \hat{f} \), the closer it is to the ‘truth’ \( f^* \) in terms of KL divergence.

Logarithmic loss and KL divergence play a central role in both of our regression scenarios. We discuss this role in more detail now.

**AIC and XAIC** Given a set of models, each a parameterized set of densities \( M_i = \{ f_\theta \mid \theta \in \Theta_i \} \), AIC (Akaike, 1973) tells us to compute

\[
-2 \log f_{\hat{\theta}(X,Y)}(Y \mid X) + 2k
\tag{1.2}
\]

for each model, and select the model that minimizes this value. Here \( \hat{\theta} \) denotes the maximum likelihood estimator and \( k \) is the number of free parameters in the model. Under some regularity conditions, the quantity (1.2) is an asymptotically unbiased estimator of the KL divergence from the maximum likelihood estimate to the truth, up to a constant (the second term in (1.1)) that is the same for all models.

XAIC replaces the estimator (1.2) by an estimator of the KL divergence at a given set of (or distribution over) test inputs:

\[
-2 \log f_{\hat{\theta}(X,Y)}(Y \mid X) + k + \kappa_{X'},
\tag{1.3}
\]

where \( \kappa_{X'} \) is a quantity that depends on the model, the training data, and the test input(s) \( X' \) or some prediction thereof, arrived at by other means. We refer
to Theorem 2.1 for the definition of $\kappa_{X'}$. In the special case where the set of test inputs is exactly identical to the set of training inputs, $\kappa_{X'} = k$ and XAIC reduces to AIC; if $X'$ contains only one point, we find the special case FAIC.

**Bayes and SafeBayes**  For a prior on $\Theta$ with density $\pi$ relative to measure $\rho$, the standard Bayesian posterior has density

$$
\pi(\theta \mid x^n, y^n) = \frac{f_\theta(y^n \mid x^n) \pi(\theta)}{\int f_\theta(y^n \mid x^n) \pi(\theta) \rho(d\theta)}.
$$

(1.4)

(Here $\Theta$ may be the parameter space of a single model; or, regarding the model index as another parameter, $\Theta$ may represent all models combined.) If the posterior converges to a hypothesis in the model, then, under weak conditions, it is to a hypothesis that minimizes the KL divergence to the true distribution. In this case, the Bayesian predictive distribution in our regression models will become more and more similar to a Gaussian distribution with the same variance as the noise, and as a result, logarithmic loss and squared error loss behave similarly.

In the wrong-model experiments of Chapter 3 (of which we saw one example in Figure 1.2), however, the posterior does not concentrate, and Bayes’ predictions do not start to resemble Gaussian distributions with a fixed variance. While these predictions still perform well when evaluated by logarithmic loss, they prove inadequate for other tasks, such as point prediction under squared error loss. (See Section 3.2 and the rest of Chapters 3 to 5 for details.)

The generalized Bayesian posterior with learning rate $\eta$ has density

$$
\pi(\theta \mid x^n, y^n, \eta) = \frac{(f_\theta(y^n \mid x^n))^{\eta} \pi(\theta)}{\int (f_\theta(y^n \mid x^n))^{\eta} \pi(\theta) \rho(d\theta)}.
$$

(1.5)

SafeBayes considers different values of $\eta$ and chooses one for which Bayes still performs well when constrained to a probabilistic prediction corresponding to a hypothesis in the model (evaluated using logarithmic loss). Two of the versions of SafeBayes discussed in Chapter 3 may also be interpreted as choosing $\eta$ by evaluating Bayes’ point predictions (using squared error loss).

While AIC and Bayes can both be understood as optimizing KL divergence, they do so in very different ways. AIC usually converges to the KL-optimal predictions more quickly than Bayes. On the other hand, if one of the models contains the true data-generating distribution, AIC may continue to try more complex models instead, regardless of the amount of training data (Yang, 2007a).

In their respective chapters, XAIC and SafeBayes are considered as solutions to two different problems appearing in regression. XAIC (or, more specifically, FAIC) addresses the problem that the accuracy of predictions of a linear regression model may vary with the test input, and introduces the quantity $\kappa_x$ to measure this effect. SafeBayes addresses the problem of potential Bayesian inconsistency on wrong models; one of several explanations of how this inconsistency may occur in regression involves the way in which the variance of the Bayesian predictive distribution depends on $x$ (see Figure 4.4).
In Section 2.6.1, we see that the variance of the Bayesian predictive distribution in a linear regression model is linearly related to $\kappa_x$. Looking further, in Section 3.4.2 we find that $\kappa_x$ also appears in the versions of SafeBayes that randomize over the posterior ($\kappa_x$ occurs as the last term in (3.23), and the second to last in (3.24)), but not in the versions that pick a single parameter value in the model. This is remarkable, but it is not clear what conclusions can be drawn from this fact: there are significant differences between the ways in which FAIC and SafeBayes use $\kappa_x$. Also, our experiments in Chapters 3 to 5 do not conclusively answer the question which version of SafeBayes is better: one that includes $\kappa_x$, or one that excludes it.

So, although the methods we develop bear similarities, we cannot draw hard conclusions from that, and we will consider these methods separately in the chapters to come.

1.2 Probability updating with underspecified distributions

And now for something completely different:

1.2.1 The Monty Hall problem

Monty Hall was the host of the TV show *Let’s Make a Deal* (Selvin (1975); see also vos Savant (1990)). He used to play games like the following with the contestants.

The contestant faces three closed doors. Behind one of the doors, a great prize is hidden (say, a car), while the other doors hide less appealing objects (say, goats). After the contestant has picked one of the doors, Monty Hall does not simply tell the contestant whether he won the prize or not. Instead, he opens one of the other two doors, revealing a goat behind it. (He knows what is behind each door, so he never accidentally reveals where the car is.) Now he asks the contestant if he would like to change his mind and switch doors. What should the contestant do?

Most people’s first impression is that switching does not change the chances of winning the car. Both doors were equally likely to hide the car before Monty Hall opened a door, so why would one be more likely than the other now?

However, this is not the right answer. The location of the car does not change when Monty Hall opens a door, and so the probability\(^6\) that it is behind the initially chosen door remains $1/3$, so the probability for the other closed door must be $2/3$, and it is wise to switch.

The situation becomes easier to understand intuitively when we consider a game with 100 doors. After the contestant’s initial pick, Monty Hall opens 98 doors, revealing 98 goats. Two doors remain shut: the door the contestant

---

\(^6\)To be precise: this is the *unconditional* probability, which does not take into account which door the quizmaster opened.
picked, and another door Monty Hall chose to leave shut. However, the con-
testant picked his door without knowing where the prize is, while Monty Hall
does know where the prize is. Clearly, the two doors are not symmetric, and
it would be wise to switch. (This is the explanation given by vos Savant in the
article (1990) on the Monty Hall game that made this problem famous.)

1.2.2 Generalizing the problem

In the Monty Hall game, the contestant has an initial probability distribution
of where the car is hidden. After seeing a goat behind one of the doors, he
may (if he is a probabilist or statistician) want to update his probabilities. The
standard way to update probabilities after receiving such information is by
conditioning on the set of remaining options. This would tell us that if the car
was equally likely to be behind the two remaining doors initially, then it is still
equally likely to be behind those doors now.$^\text{7}$

As the Monty Hall problem shows, this is not always the right answer. The
fact that this does not always work correctly motivates us to study a more gen-
eral question: how should we update our probabilities in similar situations?

The problems we study correspond to the second half of the Monty Hall
game: the car has already been hidden (at random) behind one of the three
doors, and the contestant has already made an initial guess. How does the
quizmaster now decide which door to open, and how can the contestant inter-
pret this new information? We generalize this problem in the following ways:
we allow the set of possible outcomes to contain any number of values (instead
of just three doors as possible locations of the car in the Monty Hall game); we
allow any number of messages that may be received by the contestant (corre-
spending to the two doors the quizmaster may choose to open); and the initial
distribution on the outcomes may be any probability distribution.

We are thus looking at a generalization of conditioning where a set $y$ is re-
vealed containing true outcome $x$. In standard situations, the different sets $y$
that may be revealed are disjoint. (For example, in regression we condition
on the precise value of the input variable; these different values do not ‘over-
lap’). But here, the sets $y$ can overlap: calling the doors $\{a, b, c\}$, assuming the
contestant’s initial pick is $a$, Monty Hall reveals either $\{a, b\}$ or $\{a, c\}$.

We can also cast the contestant’s problem into the form of the regression
problem / statistical models. Some process generates the outcome $X$, and an-
other (the quizmaster) generates the message $Y$ given $X$; this is very similar to
the regression setting. Now the contestant observes $Y$ (instead of $X$), and has
to predict $X$ (instead of $Y$). Another important difference with the regression
setting is that in the probability updating problem, there is no previous data
for the contestant to learn from.

$^\text{7}$This is the answer we get by ‘naive conditioning’ (Grünwald and Halpern, 2003). As we will
see below (and in detail in Chapter 6), if we had access to additional information and would formu-
larize the problem in a larger space in which this information can be represented, then conditioning
would give the right answer. The problem is thus not with conditioning per se, but with the choice
of space: in the naive space, conditioning is not right, while in the right space, we do not have
sufficient information to condition.
We assume the contestant knows the process generating $X$. However, the process generating $Y$ given $X$ is unknown to him. The only thing he knows about this process is that some pairs of outcomes and messages cannot occur together: the quizmaster could not reveal a goat behind the door that has the car behind it. The set of all distributions satisfying these constraints form a model. However, with no previous data, we have no way to learn which of these distributions to prefer, as we could in the regression setting. The model only tells us what is possible and what is not. Thus we may say the model is underspecified.

1.2.3 Our approach

We do not know how the quizmaster in the Monty Hall game chooses which door to open; or, in the general setting, how the message $Y$ is generated. We need to know this if we want to know the conditional probability that the contestant should assign to each outcome given some message. To do this, we take a worst-case approach: we assume that the contestant and the quizmaster are two players, playing the game against each other. (In the Monty Hall problem, this may very well be what is actually going on.) The contestant wants to predict as well as possible, while the quizmaster's goal is to make the contestant's task as hard as he can.

This worst-case approach is standard in game theory, and has been previously applied to the Monty Hall problem by Gill (2011); Gnedin (2011). But the applicability of this theory is not restricted to games between two opponents. If we do not believe the data-generating process is fully adversarial, but merely want to be careful about the conclusions we draw, a worst-case approach can ensure that our predictions will not be terrible, no matter what happens.

In order to make this approach precise, we need to assign numeric scores to the contestant's predictions. For this purpose, we again use loss functions, which were introduced in the beginning of Section 1.1.3. In the probability updating problem, a loss function takes a prediction and the actual outcome, based on which it assigns a loss to the contestant. A smaller loss means the contestant predicted well; a higher loss makes the quizmaster happy. In the terminology of game theory, our game is zero-sum: the amount that one player wins always equals the amount that the other player loses.

There are many loss functions that we could choose to use, and in general the players may want to play this game differently for different loss functions. For example, if we use 0-1 loss as our loss function, then the contestant must pick a single outcome; if this was the actual outcome, his loss is 0, otherwise his loss is 1. (This is essentially the original Monty Hall problem. We did not consider this loss function in the regression setting, because there it would almost surely give loss 1 to every prediction.)

We can also allow the contestant to give a probabilistic prediction, allowing him to express his uncertainty over the true outcome. Then we may again use logarithmic loss to judge these predictions. In terms of Kelly gambling (Cover and Thomas, 1991), this can be thought of as the contestant betting money
on each of the possible outcomes, and winning an amount of money depend-
ing on the amount he bet on the true outcome. Many other loss functions are
possible. We explore their implications for the probability updating game in
Chapter 6.

We wish to find a ‘worst-case optimal’ strategy in such a game. But what
do such strategies look like? To answer this, consider a different game: rock-
paper-scissors. If you know your opponent is going to play rock next, you
play paper and you win. We say that always playing rock is not a worst-case
optimal strategy, because if player A is using this strategy and player B figures
it out, player B will have a huge advantage. Now consider a different strategy:
your opponent plays rock, paper or scissors each with probability $1/3$. Then
even if he tells you what his strategy is, this would not give you any advantage.
A strategy like this will be called worst-case optimal: it gives a player the best
possible result against an opponent with the exact opposite goal, even if the
opponent figures out what strategy the player is using and then picks his own
strategy to exploit this information as much as possible (the worst case).

Similarly, in the Monty Hall game, if the car happens to be behind the door
the contestant guessed, then the quizmaster will have a choice of which door
to open. It is a worst-case optimal strategy for the quizmaster to make this
decision at random, choosing either door with probability $1/2$. If he uses another
strategy, such as always opening the leftmost door out of his two options, then
this might give an advantage to a contestant who figures this out: if the quiz-
master ever opens the rightmost door, then the contestant can bet all his money
that the car is behind the leftmost door; depending on the loss function, this
may improve (decrease) his loss on average. (In the Monty Hall game with 0-1
loss, it does not make a difference if the contestant tells the quizmaster before-
hand that he will always switch doors. However, it does make a difference in
other instances of the generalized game we consider. So in general, we want
both players’ strategies to be worst-case optimal.)

For many loss functions, worst-case optimal strategies for both players have
the property that neither the quizmaster nor the contestant can benefit from
knowing the other’s strategy. Such a pair of strategies is called a Nash equilib-
rium (Nash, 1951).

Worst-case optimal strategies may be difficult to compute. In particular, it is
not always a worst-case optimal strategy for the quizmaster to pick a message
uniformly at random, as is sometimes assumed in solutions to the Monty Hall
game (when the marginal is also uniform). For example, in the Monty Hall
game with a nonuniform marginal (that is, some doors have a larger probabili-
ty of containing the prize), the worst-case optimal strategy for the quizmas-
ter may require him to open one of the doors with a higher probability than
the other. How hard it is to find a worst-case optimal strategy in a probabil-
ity updating game depends in large part on the arrangement of the possible
messages, as we will see in Chapter 7. For some message structures, efficient
methods exist that allow a worst-case optimal strategy to be computed quickly.
This is the topic of Chapter 8.
1.3 Overview of this dissertation

In Chapter 2, we investigate the problem of model selection for extra-sample prediction. Based on a novel, unbiased expression for KL divergence, we propose XAIC and its special case FAIC as versions of AIC intended for this task, and show that both may significantly improve predictive performance compared to standard methods, including AIC and Bayesian model averaging.

Chapters 3 to 5 concern inconsistency of Bayesian inference for wrong models. Experiments with linear models exhibiting such inconsistency are shown in Chapter 3, both in a model averaging/selection and in a Bayesian ridge regression setting. To remedy the problem, we equip the likelihood in Bayes’ theorem with an exponent called the learning rate, and we propose the SafeBayesian method to learn the learning rate from the data. SafeBayes tends to select small learning rates as soon the standard posterior is not ‘cumulatively concentrated’, and its results on our data are quite encouraging.

In Chapter 4, we give several explanations of how this inconsistency may occur under ‘bad’ misspecification, and why SafeBayes provides a solution to this problem. We also discuss how our inconsistency example and the SafeBayes method relate to other work.

Chapter 5 provides additional regression experiments to test whether the results of Chapter 3 also hold with different priors, models, methods, and data-generating distributions. We find that three versions of SafeBayes consistently perform well, while other methods, including Bayes and AIC, perform badly.

The final three chapters of this dissertation deal with worst-case optimal probability updating, an alternative to conditioning that may be used when the distribution is not fully specified. In Chapter 6, we introduce the problem, and find how optimal solutions may be recognized for different loss functions; our main tool is convex analysis. We find that for logarithmic loss, optimality is characterized by the elegant RCAR (reverse coarsening at random) condition.

In Chapter 7, we analyse the combinatorial aspect of the probability updating problem, and present some theoretical tools that may help us compute worst-case optimal solutions to a probability updating problem, as opposed to merely recognizing such solutions. Further, we see that the applicability of the RCAR condition is not restricted to the cases discovered in Chapter 6, and explore the consequences.

In Chapter 8, we give algorithms that automate the task of finding worst-case optimal solutions, for restricted classes of probability updating problems. Section 8.3 in Chapter 8 is an intermezzo that investigates a notion of fairness in the theory of maximum flows. While needed to understand the subsequent developments of Chapter 8, it may also be of independent interest.