



## UvA-DARE (Digital Academic Repository)

### Combining strategies efficiently: high-quality decisions from conflicting advice

Koolen, W.M.

**Publication date**  
2011

[Link to publication](#)

#### **Citation for published version (APA):**

Koolen, W. M. (2011). *Combining strategies efficiently: high-quality decisions from conflicting advice*. [Thesis, fully internal, Universiteit van Amsterdam].

#### **General rights**

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

#### **Disclaimer/Complaints regulations**

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

# $L - L^*$

One of the main tasks in Artificial Intelligence is to construct computer systems that *learn* to be successful in any task assigned to them. Indeed, a goal that is as fascinating as it is difficult! Such an ambitious project can only be realised in stages. This dissertation is such a stage.

### 1.1 The Stage

This dissertation realises successful learning systems for a practically important and diverse class of tasks, known as the *recurring decision problems with full feedback*. That is, problems in which a decision maker repeatedly chooses actions with uncertainty about the quality of each available action. If you perchance want to solve a problem of this kind, then this book is for you. But even if you do not, its overarching approach may provide guidance in addressing other decision problems, the specific methods developed may be used as modular building blocks in the construction of advanced learning systems and the theoretical results derived may provide insight into the philosophy of learning.

This dissertation describes a series of systems that, without any prior knowledge of the task at hand, incrementally learn to choose high-quality actions, in a rigorously defined sense. Namely, each individual system comes with a specific performance guarantee, that relates the quality of its actions to those that would have been chosen by a “pre-scient” system, or equivalently, by an external observer with hindsight about the quality of all actions. Moreover, we show that each system is computationally efficient in the sense that the required computation time remains modest even if a large amount of data is processed.

**Goal & Outline** This chapter gently introduces recurring decision problems with full feedback and existing online learning solutions, motivating concepts, goals and methods by example, and incrementally building up to a high-level overview of the research presented in this dissertation. Although the research chapters are self-contained, this informal introduction explains their common conceptual background.

This introduction is itself structured as follows. First, in Section 1.2 we give a definition of and numerous examples from the class of recurring decision problems with full feedback. The first modular step in solving such problems is identifying or creating expert advice, as explained in Section 1.3. The second modular step is to use online learning methods to combine the available expert advice into high-quality actions, as discussed in Section 1.4. We review two fundamental online learning algorithms in Section 1.5, and discuss their guarantees and efficiency. We summarise the online learning approach in Section 1.7. We then place online learning into context by describing related research in Section 1.8. Subsequently, in Section 1.9, we motivate the extended learning tasks studied in this dissertation and discuss their challenges. Then in Section 1.10, we give a short overview of this dissertation’s five main chapters. We conclude in Section 1.11.

## 1.2 Recurring Decision Problems

This dissertation is about *recurring decision problems with full feedback*. That is, problems in which a decision maker repeatedly chooses actions with uncertainty about the quality of each available action. The following concrete examples illustrate the diversity of such problems, their frequency of occurrence and their practical importance.

- A diabetic regularly needs to inject insulin, the desired amount depending on future sugar intake and exercise level.
- A farmer decides whether to irrigate his fields, risking to waste water when it rains later.
- A banker invests her capital in a portfolio of stocks with payoffs determined by future prices.
- A commuter chooses a route daily, whose duration depends on the severity of traffic jams.
- A hybrid car (equipped with both combustion and electric engines) has to select its power source, with overall efficiency depending on price and availability of refuelling and recharging stations.
- An SMS typing assistant predicts the continuation of the current word, with usefulness proportional to the number of key presses saved.

Note that the first few problems are currently addressed by humans, the final few problems are already completely automated, and the problems in the middle are faced by humans with plenty of machine assistance. These boundaries are constantly shifting as more decision problems become automated or assisted.

**Side Information** Decisions are usually based on *side information*. For example, diabetics typically measure their current blood glucose level, and integrated hybrid car controllers use the route chosen by the on-board navigator as side information. Trading with insider side information is highly profitable and usually illegal.

**Loss Function** In each problem, the so-called *loss function* governs the set of available actions and the way their quality is measured: better actions incur less loss. The SMS assistant's mistakes are counted by the *0/1 loss* [25]. The banker's capital growth rate is scored by *Cover's loss* [33], and the related *log loss* is appropriate for data compression [34], probability forecasting [39] and hypothesis testing [15]. *Square loss* is also widely used, e.g. in regression and clustering. Total commute time is measured by the *dot loss*, which scores so-called *structured concepts* [92] like home-to-work routes.

**Informal Definition** A *recurring decision problem* is a task that proceeds in trials. Each trial the decision maker, or *agent*, receives the side information relevant to the current trial. Then the agent chooses an action from a set of available actions, and subsequently incurs the loss associated to the chosen action by the loss function. In recurring decision problem with *full feedback*, the loss of all possible actions is revealed to the agent at the end of the trial. The agent's *cumulative loss* after  $T$  trials is the sum of the losses incurred in trials 1 through  $T$ .

Our goal is to build an automated agent that learns to exploit the available side information and feedback to incur small cumulative loss. This is accomplished in two modular steps. The first step is mustering or creating experts to interpret the available side information, as explained in the next section. The second step is using the feedback to sequentially combine the expert advice using online learning methods, as explained in Section 1.4.

### 1.3 Experts

The technical term *expert* denotes any strategy, agent, method or algorithm, for choosing actions based on the available side information. An expert, either human or machine, thus represents a particular approach to a recurring decision problem. An expert may choose actions based on common sense, rules of thumb, human experience and expertise, scientific theories and statistical models. For example,

- In the diabetic example one expert may prescribe to follow the catalogued average insulin production curve of healthy comparable individuals. The patient's physician may recommend a dose based on the patient's medical record. And as a third source of expert advice we may consult a statistical biological model after extracting dietary and exercise clues from the patient's electronic agenda.
- An SMS typing assistant is usually equipped with a dictionary and a set of morphological rules, both annotated with frequencies of occurrence. We may instantiate one expert based on the Dutch language, one on the English language, and one on so-called Textese, used 4 gr8er input r8.

- In the banking example a human expert may read all international news and daily trade reports, and recommend investing all capital in shares of IBM today, then in Coca Cola tomorrow etc. A fully automated expert may recommend the portfolio that is optimal under the assumption that stock prices evolve according to some fixed probability distribution.

We assume throughout that suitable experts are present. It is often possible to find human experts, and it is relatively straightforward to create simple automated experts for any decision problem. Even cutting-edge automated experts, based on problem-tailored modelling and incorporating domain-specific knowledge, can in some cases be acquired.

Evidently, different experts capture different aspects of the decision problem, and we desire to intelligently integrate their conflicting advice into high quality decisions. In other words, we want to construct a single superior master expert that combines the advice of many simpler experts.

Recurrence with full feedback allows us to achieve this goal by *learning*, i.e. using the quality of past advice to adjust the relative importance of each expert's advice in our next decision. Let's see how this is done.

## 1.4 Online Learning

The study of recurring decision problems in the presence of expert advice is the domain of *online learning* [25], an emerging discipline on the interface between computer science, information theory and statistics.

Online-learning methods have been successfully applied to a diverse range of practical problems. Results include e.g. state-of-the-art data compression software [189, 31], improved statistical methods for hypothesis testing and model selection [178, 162, 40], fast natural language entry assistants [184], competitive stock market predictors [33], improved portable device power managers [78, 64] and highly adaptive caching policies [70].

The goal of online learning theory [25] is to develop for each combination of loss function and set of experts an efficient algorithm that guarantees small *regret*, where regret is defined as the difference between the loss of the algorithm and the loss of the best expert. An algorithm with small regret learns to act like the best expert. This im-

---

**Protocol 1.1** Sequential point forecasting
 

---

**for** trial  $t = 1, 2, \dots$  **do**  
 Receive a prediction  $p_{i,t} \in \{y, n\}$  for each expert  $i \in \{1, \dots, n\}$ .  
 Choose a prediction  $q_t \in \{y, n\}$ .  
 Observe the actual outcome  $x_t \in \{y, n\}$ .  
 Incur a mistake if  $q_t \neq x_t$ . Expert  $i$  incurs a mistake if  $p_{i,t} \neq x_t$ .  
**end for**

Regret w.r.t. best expert after  $T$  trials:  $L - L^*$  mistakes, where

$$L := \sum_{t=1}^T \mathbf{1}_{q_t \neq x_t} \quad \text{and} \quad L^* := \min_i \sum_{t=1}^T \mathbf{1}_{p_{i,t} \neq x_t}.$$


---

plies that if at least one of the experts that the algorithm has access to incurs small loss, then the algorithm itself suffers small loss as well, and hence, as desired, learns to perform its task well. We now introduce the two fundamental regret guarantees that can be obtained.

## 1.5 Fundamental Online Learning Problems

Recall our farming example, where a farmer needed to decide whether to irrigate or not, and hence needed to predict the rain. We abbreviate the event that it rains to  $y$ , and denote its complement by  $n$ . We analyse this online learning problem for two different loss functions: 0/1 loss and log loss. In each case our goal is to minimise our *regret*, i.e. the difference between our cumulative loss  $L$  and the cumulative loss  $L^*$  of the best expert. The regret formula  $L - L^*$  serves as this chapter's logo.

### 1.5.1 Prediction with 0/1 Loss

Say that we have  $n$  weather forecasting experts, e.g. employed by  $n$  television channels. Every evening, each expert predicts whether it will rain tomorrow or not by quoting either  $y$  or  $n$ . We then form our own binary prediction based on this advice. The weather is observed the following day, revealing correct and erroneous predictions, that is, providing full feedback about the loss of all possible predictions. Namely,

on outcome  $x$ , prediction  $p$  suffers 0/1 loss

$$\mathbf{1}_{p \neq x} = \begin{cases} 0 & \text{if } p = x, \\ 1 & \text{if } p \neq x. \end{cases}$$

This prediction problem is repeated daily, and our goal is to minimise our regret  $L - L^*$ , i.e. the difference between our cumulative mistake count  $L$  and the cumulative mistake count  $L^*$  of the best expert. See Protocol 1.1 for a systematic rendering of the timing of the actions and for the formal definition of  $L$  and  $L^*$ . We illustrate the setup by example. Say after  $T = 5$  days the actual rain sequence was

$$x = y, y, y, n, y$$

while the  $n = 3$  experts that we consulted sequentially predicted

$$\begin{array}{ll} p_1 = y, y, y, y, y & \text{making 1 mistake,} \\ p_2 = y, n, y, n, y & \text{making 1 mistake,} \\ p_3 = n, n, n, n, n & \text{making 4 mistakes,} \end{array}$$

and we sequentially combined these predictions into our predictions

$$q = y, n, n, y, y \quad \text{making 3 mistakes.}$$

Then our cumulative loss is  $L = 3$ , the cumulative loss of the best expert is  $L^* = 1$ , so that our regret is  $L - L^* = 2$  mistakes.

**The Hedge Algorithm** The Hedge algorithm [59] is a systematic way to combine point predictions. At a high level, the idea is to give more weight to predictions by experts that have suffered small 0/1 loss in the past. Full details are given in Chapter 6. The Hedge algorithm issues randomised predictions, so that  $L$  is a random variable. Applied to  $n$  experts, the Hedge algorithm guarantees expected regret

$$\mathbb{E}[L] - L^* \leq \sqrt{2L^* \ln(n)} + \ln(n) \quad \text{mistakes.} \quad (1.1)$$

This guarantee holds irrespective of the number of days  $T$  and without making any stochastic assumptions about the way outcomes arise; it

---

**Protocol 1.2** Sequential probability forecasting
 

---

**for** trial  $t = 1, 2, \dots$  **do**

    Receive a probability  $p_{i,t} \in [0, 1]$  of  $y$  for each expert  $i \in \{1, \dots, n\}$ .

    Choose a probability  $q_t \in [0, 1]$  of  $y$ .

    Observe the actual outcome  $x_t \in \{y, n\}$ .

    Suffer log loss  $-\log q_t(x_t)$ . Expert  $i$  suffers log loss  $-\log p_{i,t}(x_t)$ .

**end for**

Regret w.r.t. best expert after  $T$  trials:  $L - L^*$  bits, where

$$L := \sum_{t=1}^T -\log q_t(x_t) \quad \text{and} \quad L^* := \min_i \sum_{t=1}^T -\log p_{i,t}(x_t).$$


---

holds for each individual sequence of outcomes  $x$ . It implies that the quality of our predictions is close to that of the best expert. In fact, dividing both sides by the number of days  $T$ , we see that the overhead per day tends to zero since  $L^* \leq T$ . As one would expect, it is harder to approximate the overall best expert if there are many experts ( $n$  is large). Apparently, it is also harder to predict like the best expert if all experts are bad ( $L^*$  is large). This phenomenon is explained in Chapter 2.

Moreover, the Hedge algorithm runs in time  $O(n)$  per day. This is clearly optimal, simply consulting the  $n$  experts already takes this many steps.

### 1.5.2 Prediction with Log Loss

Now suppose that our  $n$  weather forecasting experts report a *probability* of rain, as is commonly done e.g. on national television, and that we want to combine these forecasts into our own probability. The loss function commonly used for such probability forecasts is the so-called log loss, which is the information-theoretic measure of surprise measured in bits<sup>1</sup>. More precisely, the *log loss* of predicting outcome  $y$  with probability  $p \in [0, 1]$  (and hence outcome  $n$  with probability  $1 - p$ ) on

---

<sup>1</sup>Log loss originated in coding theory, where code lengths are measured in bits. Probability forecasting with log loss and sequential coding are essentially the same, as shown by the Kraft Inequality and Shannon-Fano coding. [34]

actual outcome  $x \in \{y, n\}$  is

$$-\log p(x) := \begin{cases} -\log(p) & \text{if } x = y, \\ -\log(1-p) & \text{if } x = n, \end{cases}$$

where  $\log$  denotes the logarithm to base 2. As expected, assigning high probability to the actual outcome means small loss and vice versa. Note that again by observing the outcome  $x$  we get full feedback about the loss of all possible predictions  $p$ .

In contrast to the 0/1 loss used for counting mistakes, the log loss is not bounded. In particular, any  $p$  that assigns zero probability to the actual outcome  $x$  suffers infinite loss. The log loss setup is displayed as Protocol 1.2.

Recall that in the example of the previous section the actual rain sequence after  $T = 5$  days was

$$x = y, y, y, n, y.$$

Suppose that our  $n = 3$  experts sequentially quoted rain probability

$$\begin{array}{ll} p_1 = 0.6, 0.6, 0.6, 0.6, 0.6 & \text{incurring 4.270 bits,} \\ p_2 = 0.8, 0.7, 0.6, 0.5, 0.4 & \text{incurring 3.895 bits,} \\ p_3 = 0, 1, 0, 1, 0 & \text{incurring } \infty \text{ bits,} \end{array}$$

and we sequentially combined these predictions into our predictions

$$q = 0.7, 0.6, 0.6, 0.5, 0.5 \quad \text{incurring 3.989 bits.}$$

Then our cumulative loss is  $L = 3.989$  bits, the cumulative loss of the best expert is  $L^* = 3.895$  bits, so that our regret is  $L - L^* = 0.093$  bits.

**The Bayesian Strategy** The Bayesian strategy is a systematic way to combine probabilistic predictions. At a high level, the idea is again to give more weight to predictions by experts that have suffered small log loss in the past. Full details are given in Chapter 3. Applied to  $n$  experts, the Bayesian strategy guarantees regret

$$L - L^* \leq \log(n) \text{ bits.} \quad (1.2)$$

Like Hedge, the Bayesian strategy runs in optimal time  $O(n)$  per day.

### 1.5.3 Comparison

Thus, in both cases there is a computationally efficient strategy that guarantees that the quality of our predictions is close to that of the best expert. This means that we can learn to approximately predict like the best expert. Here, as always, “best” means “best with hindsight” at the time  $T$  of evaluation.

These results also allow us to quantitatively compare the two prediction problems. The fact that the point forecasting regret bound (1.1) grows as the square root of the loss of the best expert whereas the probability forecasting regret bound (1.2) is completely independent of the expert losses indicates that the first task is harder than the second.

Finally, note that both cases share the same interesting underlying tradeoff. To see this, consider the loss as a function of the number  $n$  of experts that we feed into either algorithm. The loss of the best expert is non-increasing in the number of experts used. But the regrets (or at least both bounds) are increasing in the number of experts used. The best cumulative loss is thus obtained by running these algorithms with a reasonably sized set of reasonable experts, so that both the regret and the loss of the best expert are small.

In this section we did not use any background knowledge about the predictions of the experts, and hence obtained regret guarantees that hold irrespective of the expert advice. On the other hand, sometimes explicit knowledge about the experts functioning is available, and may be used to our advantage, to obtain better regret guarantees. In the next section we classify the different types of experts, according to what we know about them.

## 1.6 Nomenclature and Taxonomy of Experts

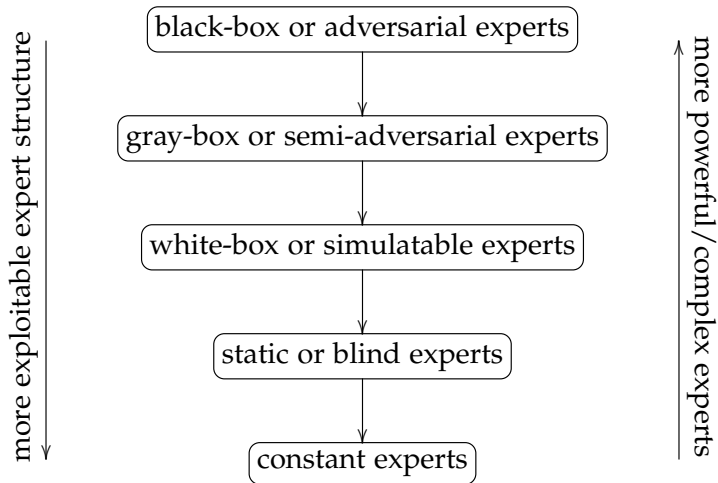
We already pointed out in Section 1.3 that the notion of expert is extremely general: an expert is any strategy, agent, method or algorithm, for choosing actions in a given recurring decision problem. In this section we identify different types of experts, and classify them according to how much information we have about their functioning. The resulting hierarchy is displayed in Figure 1.1.

At the bottom we find the simplest *constant experts*. A constant expert always recommends the same action. Next we encounter the

---

**Figure 1.1** Hierarchy of expert types
 

---



slightly more general *static* or *blind experts*. A static expert follows a fixed, known sequence of actions, so that its actions only depend on time, but not on the loss of its past actions.

Experts in the subsequent category of *white-box* or *simulatable experts* do adapt their recommendation based on the loss of their past actions, and they do this in a way that is known to us. That is, given past outcomes we can always compute the action they will recommend next. The most general class of experts are the *black-box* or *adversarial experts*. This class includes human experts, which we can consult but which we can not simulate. To obtain performance guarantees for algorithms for decision problems with black-box experts, we have to assume the worst case, that is, we assume that the advice of black-box experts is determined by a malevolent adversary. Finally, the *gray-box* or *semi-adversarial experts* arise when we combine the advice of black-box experts in a simulatable fashion.

Before going into the details of this dissertation, we first summarise what has been presented so far, and discuss related research.

## 1.7 Summary

In recurring decision problems a decision maker repeatedly chooses actions with uncertainty about the quality of each available action. With full feedback, the quality of each action is revealed after a decision is committed. Online learning advocates the following approach to solving recurring decision problems with full feedback:

1. Identify the set of actions.
2. Identify the loss function.
3. Identify the available side information.
4. Gather or create experts. That is, get access to agents that, each trial, use the side information to recommend an action.
5. Run an algorithm that uses the feedback to combine the expert advice, and that guarantees small regret w.r.t. the best expert.

For step 5 we need algorithms that guarantee small regret w.r.t. the best expert, i.e. that learn to act approximately like the best expert. This dissertation investigates the design and analysis of such algorithms.

Note that the resulting algorithm itself qualifies as an expert, since it sequentially chooses actions. Its output can therefore serve as the expert input in step 4 to an even higher-level algorithm. Such hierarchical combinations of experts can be used to obtain advanced learning algorithms.

## 1.8 Related Research

Different models of learning are actively researched [126, 154, 8, 95, 106, 127, 179, 35, 74, 160, 17, 150, 25, 71]. There are two main approaches: learning as displaying the correct behaviour vs learning as identifying the underlying rule or process.

### 1.8.1 Decision-Making

A large class of models for learning agrees with online learning that the objective is to build systems that make good decisions. These models differ in the exact type of problems considered. The closest relative is called *bandit learning*. [176, 155, 158, 50, 24, 55, 10, 118, 143, 5, 26] This

name refers to a gambler who wants to learn which slot machine, or *one-armed bandit*, yields the best overall payoff. In contrast to online learning, the learner only gets *partial feedback*. He observes the loss of his chosen action, but not the loss of the other possible actions. A typical bandit problem is *ad(vertisement) selection*. An ad has to be selected from a large set for presentation to the user, say in the margin of some web page. The system's owner receives some financial reward if the user clicks on the ad. In this setting, the system clearly only observes the user's reaction to the presented ad, and not to all possible ads. Despite the difference, algorithms for the bandit setting often strongly resemble online learning algorithms.

The even more difficult setting called *Reinforcement Learning* arises when the quality of chosen actions is only revealed later, indirectly, or not at all. [1, 9, 12, 13, 14, 20, 11, 22, 36, 30, 46, 48, 45, 42, 44, 49, 52, 37, 47, 54, 56, 57, 61, 60, 67, 62, 69, 63, 68, 76, 82, 83, 88, 89, 90, 96, 98, 94, 93, 97, 103, 109, 105, 110, 86, 107, 131, 114, 122, 116, 134, 133, 113, 121, 123, 120, 132, 124, 130, 119, 111, 115, 117, 135, 125, 137, 136, 138, 145, 141, 142, 139, 140, 144, 148, 152, 149, 151, 157, 156, 153, 147, 167, 159, 165, 169, 168, 164, 166, 175, 170, 171, 174, 183, 191, 187, 188, 51]

In these settings it is often necessary to make assumptions about the environment before anything can be provably learnt.

Methods for making good decisions compared to a class of experts can be found in the literature on different fields. Examples include *universal modelling* in statistics [71], *universal coding* in data compression [34] and *universal portfolios* in finance [33].

### 1.8.2 Inference

A second set of models for learning pursues a different goal altogether. The crucial difference is that there is no extrapolation into the future. Instead, the system has to create a pattern, rule, description, model, hypothesis or explanation of the situation at hand from examples. Applications include data analysis, data mining, data compression, density estimation, archaeology, clustering and de-noising. Although the goal is exploratory, decision making is sometimes used as a sanity check.

The background so far provided by the introduction allows us to explain the content of this dissertation in slightly more detail.

## 1.9 Meta-Experts — This Dissertation

This introduction discussed a basic case: learning to behave like the best expert. In the remainder of this dissertation we consider four complex online learning problems, with more ambitious goals. In each case, we start out with a set of “basic” experts, and then create a huge variety of derived “meta”-experts, in such a way that we expect the best meta-expert to be much better than the best basic expert. We consider the following three sets of meta-experts.

- Switching between basic experts. Here a meta-expert switches between basic experts as time progresses. Such meta-experts outperform the best basic expert when the relative quality of the basic experts varies over time. Competing with the best meta-expert means learning when to switch between basic experts.
- Switching between learning basic experts. When basic experts are themselves learning in the sense that their predictions depend on the data that they have observed, we create a more ambitious goal. Namely, each meta-expert switches between basic experts as time progresses, but only lets the currently selected basic expert observe the current outcome. Competing with the best meta-expert means learning how to segment the data, in such a way that for each segment, the best basic expert for that segment can learn to exploit the local patterns well.
- Structured concepts. Consider home-to-work routes. Say that we have for each directed road segment a basic expert that specialises in driving that segment. A meta-expert now is a route from home to work, that uses *all* the basic experts along the path. The loss (commute time) of a meta-expert is the sum of the losses of the basic experts used. Competing with the best meta-expert means learning the shortest path from home to work. Analogous constructions exist for other combinatorial combinations of experts, e.g. sets, spanning trees, permutations etc.

In each case, the meta-experts that we construct are gray-box experts according to the classification of Section 1.6. We may not know how the underlying basic experts function, but we do know exactly how the basic experts are combined.

As before, we then try to build algorithms that have small regret w.r.t. the best meta-expert, i.e. that learn to approximate the best meta-expert. This cannot be achieved by simply applying the fundamental algorithms from Section 1.5 on the set of meta-experts, for two reasons:

- **Efficiency.** The set of meta-experts is huge, often exponential in the number of basic experts and sometimes even infinite, so that running the algorithm is intractable.
- **Performance.** The regret bounds (1.1) and (1.2) of the standard algorithms are tight when the experts' actions are independent. But the actions of different meta-experts are highly dependent, since they are derived from the same set of basic experts, so we expect that much better bounds are achievable.

The challenge is hence to exploit the internal structure of the set of meta-experts. That is, we want to design new algorithms that are efficient in terms of the number of basic experts, and that use the dependence between meta-experts to obtain tight regret bounds. This dissertation contains efficient algorithms for each of these classes of meta-experts.

## 1.10 Organisation of this Dissertation

We now sketch the contents of the main chapters of this dissertation. The next chapter, Chapter 2, is written at an introductory level. It does not assume any prior knowledge about online learning, and contains examples of the concepts used in the remainder. The subsequent four chapters are based on research articles.

### 1.10.1 Chapter 2: Regret Games (Introductory)

We give a game-theoretic analysis of the simplest online learning problem, the prediction of a sequence of binary outcomes under 0/1 loss with the help of 2 experts. For this simple problem, we compute the minimax, i.e. game-theoretically optimal, regret, and show how to implement the optimal strategy efficiently. We then give special attention to the case that one of the experts is good. We conclude with a new result: we give the optimal algorithm for competing with the set of meta-experts that switch between the 2 basic experts.

### 1.10.2 Chapter 3: Expert Hidden Markov Models

We show how models for prediction with expert advice can be defined concisely and clearly using hidden Markov models (HMMs); standard algorithms can then be used to efficiently calculate how the expert predictions should be weighted. We focus on algorithms for “tracking the best expert”, starting from the Fixed Share algorithm [79, 80], and show how most existing models can be cast as HMMs. We recover the running times and loss bounds for each algorithm, and discuss how they are related. We also describe three new models: (i) models with decreasing switching rate, which run in linear time and for which a fixed switching rate does not have to be specified in advance, (ii) a new generalisation of the fixed share algorithm that is especially well equipped to handle switches that occur in clusters, and (iii) a model tailored to the scenario where the experts have a natural order, and where jumps between them are typically small. This last model is relevant for predicting time series data where parameter drift is expected.

### 1.10.3 Chapter 4: Freezing & Sleeping

A problem posed by Yoav Freund at the Conference on Learning Theory (COLT) in 2000 is how to efficiently track a small pool of experts out of a much larger set. This problem was solved when Bousquet and Warmuth introduced their mixing past posteriors (MPP) algorithm in 2001.

In Freund’s problem the experts would normally be considered black boxes. However, in this chapter we re-examine Freund’s problem in case the experts have internal structure that enables them to learn. In this case the problem has two possible interpretations: should the experts learn from all data or only from the subsequence on which they are being followed? The MPP algorithm solves the first case. Our contribution is to generalise MPP to address the second option. The results we obtain apply to any expert structure that can be formalised using (expert) hidden Markov models. Curiously enough, for our interpretation there are *two* natural reference schemes: freezing and sleeping. For each scheme, we provide an efficient prediction strategy and prove the relevant loss bound.

#### 1.10.4 Chapter 5: Switching Investments

We present a simple online two-way trading algorithm that exploits fluctuations in the unit price of an asset. Rather than analysing worst-case performance under some assumptions, we prove a novel, unconditional performance bound that is parameterised either by the actual dynamics of the price of the asset, or by a simplifying model thereof. The algorithm processes  $T$  prices in  $O(T^2)$  time and  $O(T)$  space, but if the employed prior density is exponential, the time requirement reduces to  $O(T)$ . The result translates to the prediction with expert advice framework, and has applications in data compression and hypothesis testing.

#### 1.10.5 Chapter 6: Hedging Structured Concepts

We develop an online algorithm called *Component Hedge* for learning structured concept classes when the loss of a structured concept sums over its components. Example classes include paths through a graph (composed of edges) and partial permutations (composed of assignments). The algorithm maintains a parameter vector with one non-negative weight per component, which always lies in the convex hull of the structured concept class. The algorithm predicts by decomposing the current parameter vector into a convex combination of concepts and choosing one of those concepts at random. The parameters are updated by first performing a multiplicative update and then projecting back into the convex hull. We show that Component Hedge has optimal regret bounds for a large variety of structured concept classes.

### 1.11 Conclusion

We introduced the online learning approach to recurring decision problems with full feedback. We first reviewed prediction with expert advice, where the goal is to suffer small regret w.r.t. the best expert. We then discussed more ambitious goals that arise by combining few basic experts into a huge set of meta-experts, and then trying to approximate the best meta-expert. We briefly summarised the sets of meta-experts that are studied in the four research chapters this thesis, and previewed

the results that we obtain in each case, which take the form of algorithms and corresponding regret bounds.