The many faces of online learning

Hoeven, D. van der

CHAPTER 1

Introduction

Online Learning is a fundamental task in Machine Learning. It is the task of sequentially making predictions given previous feedback and possibly additional information. Online Learning tasks have many interesting theoretical properties and practical applications. Examples of Online Learning tasks include gambling on sport matches (Vovk and Zhdanov, 2009), spam filtering, weather prediction, portfolio selection (Cover, 1991), training machine learning models on very large data, and many more.

Online Learning proceeds in a sequence of rounds $t = 1, \ldots, T$, where in each round t the learner has to issue a prediction. In specialized cases of Online Learning such as Online Classification these predictions may come from a very limited set of a finite number of possible labels. In other settings in Online Learning such as Online Convex Optimization predictions may come from any convex set. After the learner has issued his prediction he receives feedback from the environment in the form of a loss function ℓ_t, which tells the learner the error in his prediction. In the full-information setting the learner sees the entire loss function, which means that the learner can also assess the quality of alternative predictions. In the more difficult bandit setting the learner only sees the value of the loss function evaluated at his prediction, which means the learner *cannot* assess the quality of alternative predictions.

As a simple example let us consider predicting whether or not it is going to rain tomorrow. Suppose that we encode "it will not rain" as -1 and "it will rain" as 1. We will denote the prediction of the learner with $\hat{y}_t \in \{-1, 1\}$ and the actual outcome as $y_t \in \{-1, 1\}$. If the learner makes a prediction today he will only get feedback tomorrow, which he may use to make his prediction for the next day. The loss function in this case could be the zero-one loss, i.e. $\ell_t(\hat{y}_t) = \mathbb{1}[\hat{y}_t \neq y_t]$, where $\mathbb{1}$ is the indicator function. This loss function is easy for the learner to come by as he only has to pay attention to the weather throughout the day. The

goal here is straightforward, the learner wants to make as few mistakes as possible. As a more involved example let us consider online regression. Suppose that the learner is interested in predicting how much it is going to rain. Before making his prediction the learner might have some additional information, such as barometric pressure, whether or not there are clouds, and all the tweets from the preceding twenty-four hours that contain the word weather. In each round t, the learner could use a hypothesis h_t from hypothesis set $\mathcal{H}$ to map the additional information, which we will denote by $\boldsymbol{x}_t$, to amount of rain. In this example, the learner could be interested in minimizing the quadratic difference between his prediction and the actual amount of rain $y_t \in [0, m]$, i.e. $\ell_t(h_t) = (h_t(\boldsymbol{x}_t) - y_t)^2$, where m is the size of the cup the learner uses to measure the amount of rain with.

In both examples the goal of the learner is to minimize the cumulative loss he suffers, summed over rounds $t = 1, \ldots, T$. The learner could try to use the feedback he received the previous days to continuously improve his predictions. However, the learner could have a neighbour who is accidentally disturbing his measurements by spraying his garden with water. Clearly, this makes the task of the learner harder as the relation between his observations is weakened. Even more problematic, the learner could face an adversarial neighbour who actively tries to disrupt the learner's measurements. This adversarial neighbour makes learning very difficult, as any correlation between previous days and the next is removed.

Unlike in classical statistical learning theory, which makes probabilistic assumptions about the environment, Online Learning is able to provide some guarantees about the cumulative loss of the learner without probabilistic assumptions, even in adversarial environments. The guarantees of Online Learning are about the regret, which measures how "sorry" the learner is for making the predictions he has made. To ensure that the regret can be suitably bounded we often require that the predictions and best fixed prediction strategy come from the same set, which is often bounded and convex. Suppose that the predictions h_t of the learner come from an abstract hypothesis class $\mathcal{H}$. This hypothesis class can be anything from the set of vectors in a unit ball to a set of functions that map additional information to outcomes, to a simple class that only consists of -1 and 1 as in the example above. The regret is defined as the difference between the cumulative loss of the learner and the cumulative loss of the best fixed prediction strategy in hindsight:

$$\mathcal{R}_T = \sum_{t=1}^{T} \ell_t(h_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell_t(h).$$

The goal of the learner is to have small regret. This does not mean that the best strategy is a good strategy; it is merely a benchmark with which the learner compares

himself. The learner is satisfied as long as the regret is sublinear, which is to say that the average loss of the learner minus the average loss of the best fixed strategy goes to zero as T grows.

Overview of the dissertation The chapters in this dissertation can be read independently, but for a full appreciation of each chapter it is recommended that the reader starts with the current chapter. The remainder of this chapter serves to provide a gentle introduction for the chapters that follow.

A common theme in several chapters of this dissertation is how to design algorithms that can handle adversarial environments but also exploit benign environments. In Chapter 2 we show how we can design adaptive algorithms for the Prediction with Expert Advice setting, which we introduce in Section 1.1, by using a reduction based on the Exponential Weights algorithm (Vovk, 1990; Littlestone and Warmuth, 1994). In Chapter 3 we derive algorithms that are adaptive to unknown noise in the Online Convex Optimization setting, a setting which we will introduce together with the Bandit Convex Optimization setting in Section 1.2. In Chapter 4 we derive the first algorithms that are adaptive to the norm of the offline optimizer for the Bandit Convex Optimization setting. Finally, in Chapter 5 we describe MetaGrad, which operates in the Online Convex Optimization setting. MetaGrad is adaptive to a large class of loss functions, including exp-concave and various other types of functions.

Since the learner updates his predictions each round it is important that the updates can be performed reasonably fast. For many Online learning settings algorithms with per round running time larger than quadratic in the dimension of the problem are considered impractical. In several chapters we will show how to improve the running time of Online Learning algorithms while maintaining similar or even improved guarantees. In Chapter 6 we consider the full-information and bandit Online Multiclass Classification settings, which we introduce in Section 1.3. We introduce a new approach to Online Multiclass Classification which allows us to use an algorithm that has a per round running time that is linear in the dimension of the problem that guarantees small regret bounds. Interestingly, our algorithm often improves upon the regret bounds of slower algorithms. In Chapter 5 we show how to improve the running time of MetaGrad by using matrix sketching methods at the cost of a slightly larger regret bound. Finally, in Chapter 7 we pose an open problem that asks for a fast and optimal algorithm for online portfolio selection. We propose a fast algorithm and an analysis of this algorithm that shows that in some special cases of online portfolio selection this algorithm indeed obtains the optimal regret bound.

1.1 Prediction with Expert Advice

Our first setting in Online Learning is perhaps the most well-studied setting: the prediction with expert advice setting. In the prediction with expert advice setting the learner has access to d experts. In a given round t, each expert i sends his prediction $\hat{y}_t^i$ to the learner, who may use these expert predictions to form his own prediction. These experts can be anything, for example the learner's neighbours who predict how much rain is going to fall, static experts $i = 1, \ldots, d$ who always say it is going to rain i mm, or arbitrary points in a convex set. To issue his predictions, the learner forms a distribution p_t over the experts. The learner's loss becomes $\hat{\ell}_t = \mathop{\mathbb{E}}_{i \sim p_t}[\ell_t^i]$, where $\ell_t^i = \ell_t(\hat{y}_t^i)$ is the loss of expert i at time t. Loss $\hat{\ell}_t$ can be motivated in several ways:

(a) If the learner randomly chooses an expert $i \sim p_t$ then this is the expected loss.

(b) If ℓ_t is convex and the learner predicts $\hat{y}_t = \mathbb{E}_{p_t}[\hat{y}_t^i]$ then by Jensen's inequality $\hat{\ell}_t$ is an upper bound on the learner's loss.

The goal of the learner in the prediction with expert advice setting is to predict almost as well as the best expert in hindsight, which is to say that the regret with respect to the best expert in hindsight is sublinear.

A fundamental algorithm in the prediction with Expert Advice setting and Online Learning in general is the Exponential Weights algorithm. With a discrete set of experts, the distribution of Exponential Weights has the following form:

$$p_t(i) \propto \pi(i) e^{-\eta \sum_{s=1}^{t-1} \ell_s^i},$$

where $\eta > 0$ is called the learning rate and $\pi(i)$ is the prior mass on expert i. Unsurprisingly, Exponential Weights gets its name from the exponentially weighted losses of each expert. Somewhat surprisingly, Exponential Weights can be applied in many different settings and several other algorithms are special cases of it. For example, in Chapter 2 we will see that with a continuous set of experts and a Gaussian prior Online Gradient Descent (Zinkevich, 2003) is a special case of Exponential Weights.

For losses such that $\ell_t(\hat{y}_t^i) \in [0, 1]$, Exponential Weights with learning rate $\eta = \sqrt{\frac{8 \ln(d)}{T}}$ provides the following guarantee (see for example Theorem 2.2 by Cesa-

Bianchi and Lugosi (2006)):

$$\sum_{t=1}^{T} \hat{\ell}_t - \min_i \sum_{t=1}^{T} \ell_t^i \leq \sqrt{\frac{\ln(d)T}{2}}.$$

As we can see, as T grows the difference between the average loss of the learner and the average loss of the best expert decreases.

In an adversarial environment the learner can not do better than the above regret bound (see section 3.7 by Cesa-Bianchi and Lugosi (2006)). However, in more benign environments the learner could have had a better guarantee. For example, it could have been clear from the start that the predictions from the neighbour who works at the KNMI[1] are the best predictions. In fact, if the learner only listens to the KNMI neighbour after a few rounds he will no longer suffer any additional regret after these initial rounds. This means that the learner will have to quickly learn to only listen to the KNMI neighbour to get a small regret bound.

With the Exponential Weights algorithm the speed at which the algorithm learns is governed by the learning rate η. In adversarial settings the learning rate is set such that the distribution over the experts p_t does not change drastically between rounds. However, to quickly learn that one expert is clearly the best expert, η would have to be tuned so that p_t quickly converges to a point mass on the best expert. Unfortunately, the learner usually does not know beforehand whether or not his environment is adversarial, benign, or something in between adversarial and benign. This means that the safest thing for the learner to do is to tune his algorithms to deal with an adversarial setting. In Chapter 2 we will show that with a reduction based on Exponential Weights we recover the Squint (Koolen and Van Erven, 2015) and coin betting for experts (Orabona and Pál, 2016) algorithms that adjust their learning rate automatically, which allows these algorithms to adapt to different environments.

1.2 Online Convex Optimization

The Prediction with Expert Advice setting is a special case of the Online Convex Optimization setting. In the Online Convex Optimization setting the predictions of the learner can come from any convex set, for example an L_2 ball or the probability simplex. This setting is called the Online *Convex* Optimization setting because the loss functions are assumed to be convex. Applications of Online Convex

[1] Dutch National Weather Institute

Optimization include training machine learning models on very large data and online classification.

In Online Convex Optimization, in each of $t = 1, \ldots, T$ rounds, the learner has to make a prediction $\boldsymbol{w}_t$ in a convex domain $\mathcal{W}$ before observing a convex loss function $\ell_t : \mathcal{W} \to \mathbb{R}$. The goal is to obtain a guaranteed bound on the regret

$$\mathcal{R}_T = \sum_{t=1}^{T} \ell_t(\boldsymbol{w}_t) - \min_{\boldsymbol{w} \in \mathcal{W}} \sum_{t=1}^{T} \ell_t(\boldsymbol{w})$$

that holds for any possible sequence of loss functions ℓ_t. To be able to bound the regret a standard assumption is that the domain is bounded, but in Chapter 3 we consider algorithms that are able to achieve suitable regret bounds with an unbounded domain.

To see how the Prediction with Expert Advice setting is a special case of the Online Convex Optimization setting we set the domain $\mathcal{W} = \{\boldsymbol{w} \in \mathbb{R}_+^d \mid \sum_{i=1}^d w_i = 1\}$ to be the probability simplex and let the losses be linear: $\ell_t(\boldsymbol{w}_t) = \boldsymbol{w}_t^\intercal \boldsymbol{g}_t = \hat{\ell}_t$, where $\boldsymbol{g}_t = (\ell_t^1, \ldots, \ell_t^d)$. With this loss the definition of the regret in the Online Convex Optimization setting coincides with the regret of the Prediction with Expert Advice setting.

Another example of an Online Convex Optimization task is online portfolio selection (Cover, 1991). Online portfolio selection corresponds to the special case that the domain is the probability simplex and the loss functions are restricted to be of the form $\ell_t(\boldsymbol{w}) = -\ln(\boldsymbol{w}^\intercal \boldsymbol{x}_t)$ for vectors $\boldsymbol{x}_t \in \mathbb{R}_+^d$. With online portfolio selection the goal of the learner is to distribute his funds over several assets. Online portfolio selection was introduced by Cover (1991) with the interpretation that $x_{t,i}$ represents the factor by which the value of an asset $i \in \{1, \ldots, d\}$ grows in round t and $w_{t,i}$ represents the fraction of our capital we re-invest in asset i in round t. The factor by which our initial capital grows over T rounds then becomes $\prod_{t=1}^T \boldsymbol{w}_t^\intercal \boldsymbol{x}_t = e^{-\sum_{t=1}^T \ell_t(\boldsymbol{w}_t)}$.

Cover (1991); Cover and Ordentlich (1996) show that the best possible guarantee on the regret is of order $\mathcal{R}_T = O(d \ln T)$ and that this is achieved by choosing $\boldsymbol{w}_{t+1}$ as the mean of a continuous Exponential Weights distribution $\mathrm{d}P_{t+1}(\boldsymbol{w}) \propto e^{-\sum_{s=1}^t \ell_s(\boldsymbol{w})} \mathrm{d}\pi(\boldsymbol{w})$ with Dirichlet-prior π (and learning rate $\eta = 1$). Unfortunately, this approach has a runtime of order $O(T^d)$, which scales exponentially in the number of assets d, and is therefore computationally infeasible when d exceeds, say, 3. A sampling-based implementation by Kalai and Vempala (2002) greatly improves the runtime to $\tilde{O}(T^4(T+d)d^2)$, but even this is still infeasible already for modest d and T.

A common approach to runtime problems in Online Convex Optimization is instead of optimizing the loss ℓ_t directly, optimizing a linear or quadratic approximation of ℓ_t. With a linear approximation, we make use of the convexity to upper bound the regret

$$\sum_{t=1}^{T} (\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})) \leq \sum_{t=1}^{T} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \nabla \ell_t(\boldsymbol{w}_t) = \sum_{t=1}^{T} \tilde{\ell}_t(\boldsymbol{w}_t) - \tilde{\ell}_t(\boldsymbol{u}),$$

where $\boldsymbol{u} = \arg\min_{\boldsymbol{w} \in \mathcal{W}} \sum_{t=1}^{T} \ell_t(\boldsymbol{w})$, $\tilde{\ell}_t(\boldsymbol{w}) = \boldsymbol{w}^\intercal \nabla \ell_t(\boldsymbol{w}_t)$, and $\nabla \ell_t(\boldsymbol{w}_t)$ is the gradient of ℓ_t evaluated at $\boldsymbol{w}_t$. Instead of having to optimize the complicated ℓ_t we can now run our algorithms on the linear $\tilde{\ell}_t$. For example, we could now run Online Gradient Descent (Zinkevich, 2003), which has a running time of order $O(dT)$, and obtain a regret bound of $O(G\sqrt{T})$, where G is an upper bound on the L_2 norm of $\nabla \ell_t(\boldsymbol{w}_t)$.

For many loss functions running Online Gradient Descent or a related algorithm often gives satisfying guarantees. However, for online portfolio selection assuming a bound on $\nabla \ell_t(\boldsymbol{w}_t) = -\frac{\boldsymbol{x}_t}{\boldsymbol{w}_t^\intercal \boldsymbol{x}_t}$ involves making assumptions on either $\boldsymbol{x}_t$ or $\boldsymbol{w}_t$. This means that bounding the gradients is very restrictive: we either need to (i) assume that $\boldsymbol{x}_t$ is lower bounded i.e. the asset prices do not fluctuate too rapidly, which defeats the purpose of using adversarial online learning; or (ii) we need to allocate a minimum amount of capital $w_{t,i} \geq \alpha$ to each asset, which means we cannot drop any poorly performing assets from our portfolio. To see how these assumptions affect the gradient suppose that we manage two assets and that up to round t the first asset has been performing poorly compared to the second asset. This means that we would want to put (almost) all of our money on the second asset to maximize revenue. Suppose that in round t the asset prices fluctuate rapidly: in round t the first asset remains constant, i.e. $x_{t,1} = 1$, but the second asset loses all of its value, i.e. $x_{t,2} = 0$. The gradient would be $\frac{1}{w_{t,2}}$, which would ruin the regret bound of for example Online Gradient Descent if $w_{t,2}$ is very small, for example of order $O(\frac{1}{T})$. Even when we are willing to assume a bound on the gradient, running Online Gradient Descent gives unsatisfactory results as the regret is $O(\sqrt{T})$, which is far from optimal. For online portfolio selection and other problems with exp-concave losses Online Newton Step (Hazan et al., 2007) often has better regret at the cost of increased running time, which is of order $O(d^3 T)$. However, with Online Newton Step we still need a bound on the gradient as its regret bound is of order $O(Gd \ln(T))$, making the algorithm too restrictive. Because of these issues with standard algorithms, online portfolio selection is a challenging research area, as illustrated by the open problem in Chapter 7.

In Chapter 2 we will provide a unifying view of several algorithms in the Online

Convex Optimization setting by viewing them as special cases of (Continuous) Exponential Weights. This unified view leads to a straightforward analysis of various algorithms in Online Convex Optimization, including Exponentiated Gradient Plus-Minus (Kivinen and Warmuth, 1997), Online Mirror Descent (Beck and Teboulle, 2003), and Online Newton Step (Hazan et al., 2007).

1.2.1 Bandit Convex Optimization

The Bandit Convex Optimization setting (Flaxman et al., 2005; Kleinberg, 2005) is a more difficult version of the Online Convex Optimization setting. In the Bandit Convex Optimization setting, rather than seeing the loss function ℓ_t, the learner only observes the loss function evaluated at his prediction, $\ell_t(\boldsymbol{w}_t)$. This significantly hinders the learner, as there is less information to improve his predictions for the next round. An application of the Bandit Convex Optimization setting is online auctions (Kleinberg and Leighton, 2003). In online auctions the learner has to set a price for products he wants to sell. Unfortunately the price at which people are willing to buy the learner's products is unknown, so he will have to guess a price and infer information based on whether or not people are buying the product at the guessed price. The learner's goal is to maximize his revenue, but the buyers are trying to get the lowest price, which means that learner could be facing an adversarial environment: a perfect place to apply ideas from Online Learning. Other applications of the Bandit Convex Optimization setting include recommendation systems and the online shortest path problem (see for example Hazan et al. (2016)).

The most straightforward instance of the Bandit Convex Optimization setting is when the losses are linear, i.e. $\ell_t(\boldsymbol{w}) = \boldsymbol{w}^\intercal \boldsymbol{g}_t$, where $\boldsymbol{g}_t \in \mathbb{R}^d$ is a loss vector. To still be able to update the predictions the learner will estimate the loss vector, which the learner will do by randomizing his predictions. Since the learner randomizes his predictions the guarantees in the Bandit Convex Optimization setting are about the expected regret, where the expectation is with respect to the randomness of the learner. To see how the learner estimates the loss vectors with the use of randomisation suppose that the learner plays $\tilde{\boldsymbol{w}}_t = \boldsymbol{w}_t + \varepsilon_t$, where ε_t is sampled from a distribution with mean $\mathbb{E}[\varepsilon_t] = \mathbf{0}$ and covariance matrix $\mathbb{E}[\varepsilon_t \varepsilon_t^\intercal] = \Sigma$. This means that the learner observes $\tilde{\boldsymbol{w}}_t^\intercal \boldsymbol{g}_t$, which in expectation is equivalent to $\boldsymbol{w}_t^\intercal \boldsymbol{g}_t$ because $\mathbb{E}[\varepsilon_t] = \mathbf{0}$. To estimate $\boldsymbol{g}_t$ the learner can now use $\hat{\boldsymbol{g}}_t = \Sigma^{-1} \varepsilon_t \tilde{\boldsymbol{w}}_t^\intercal \boldsymbol{g}_t$. If we take the expectation of $\hat{\boldsymbol{g}}_t$ we can see that $\hat{\boldsymbol{g}}_t$ is an unbiased estimate of $\boldsymbol{g}_t$:

$$\mathbb{E}[\hat{\boldsymbol{g}}_t] = \mathbb{E}\left[\Sigma^{-1} \varepsilon_t (\boldsymbol{w}_t + \varepsilon_t)^\intercal \boldsymbol{g}_t\right] = \Sigma^{-1}\, \mathbb{E}\left[\varepsilon_t \varepsilon_t^\intercal\right] \boldsymbol{g}_t = \boldsymbol{g}_t.$$

We can use this to say something about the expected regret because $\mathbb{E}\left[\tilde{\boldsymbol{w}}_t^\intercal \boldsymbol{g}_t\right] = \boldsymbol{w}_t^\intercal \boldsymbol{g}_t = \mathbb{E}\left[\boldsymbol{w}_t^\intercal \hat{\boldsymbol{g}}_t\right]$, where the expected regret is defined as $\mathbb{E}\left[\sum_{t=1}^T \tilde{\boldsymbol{w}}_t^\intercal \boldsymbol{g}_t\right] -$

$\min_{\boldsymbol{u}\in\mathcal{W}} \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{u}^\intercal \boldsymbol{g}_t\right]$. Algorithms in bandit information setting often suffer an additional factor d regret compared to their counterparts in the full-information setting. One of the most well-known algorithm in the Bandit Convex Optimization setting is SCRiBLe (Abernethy et al., 2012), which achieves an $O(d^{3/2}\sqrt{T})$ expected regret bound with linear losses.

In several chapters we will provide new algorithms or analysis of algorithms in the Bandit Convex Optimization setting. In Chapter 2 we use Continuous Exponential Weights to sample $\tilde{\boldsymbol{w}}_t$. As already observed by Bubeck and Eldan (2015), this improves the regret of SCRiBLe by a factor of $d^{1/2}$. As we mentioned before, in Chapter 4 we provide several Bandit Convex Optimization algorithms that are adaptive to the norm of the offline optimizer.

1.3 Online Multiclass Classification

Another setting in Online Learning is the Online Multiclass Classification setting. In each round t in the Online Multiclass Classification setting the learner has to predict the true label y_t out of N possible labels given some extra information $\boldsymbol{x}_t \in \mathbb{R}^d$. An example of this setting is the weather forecasting example. The learner might want to predict whether it is going to rain or not and he probably has some extra information such as humidity or barometric pressure. As in the Online Convex Optimization setting there is a distinction between the full-information setting, in which the learner gets to see the true label, and the bandit setting, in which the learner only sees whether his prediction was correct or not. The loss function in both the full-information and bandit settings is the zero-one loss: $\ell_t(\hat{y}_t) = \mathbb{1}[\hat{y}_t \neq y_t]$, where $\hat{y}_t$ is the prediction of the learner and $\mathbb{1}$ is the indicator function. The goal of both settings is to control the number of mistakes the learner makes i.e. $\sum_{t=1}^{T} \ell_t(\hat{y}_t)$.

1.3.1 Full-Information

We start by introducing the full-information multiclass classification setting. Since the zero-one loss is a non-convex loss the standard approach is to make use of a surrogate loss, which is a convex upper bound on the zero-one loss. In the case where there are only two possible labels the surrogate loss is a function of $z_t = y_t \boldsymbol{w}_t^\intercal \boldsymbol{x}_t$, where $y_t \in \{-1, 1\}$. The learner predicts with $\hat{y}_t = \operatorname{sign}(\boldsymbol{w}_t^\intercal \boldsymbol{x}_t)$ (if $\boldsymbol{w}_t^\intercal \boldsymbol{x}_t = 0$ the learner can arbitrarily pick -1 or 1). One of the most well-known surrogate losses is the hinge loss: $\tilde{\ell}(z) = \max\{1 - z\}$. Another well known

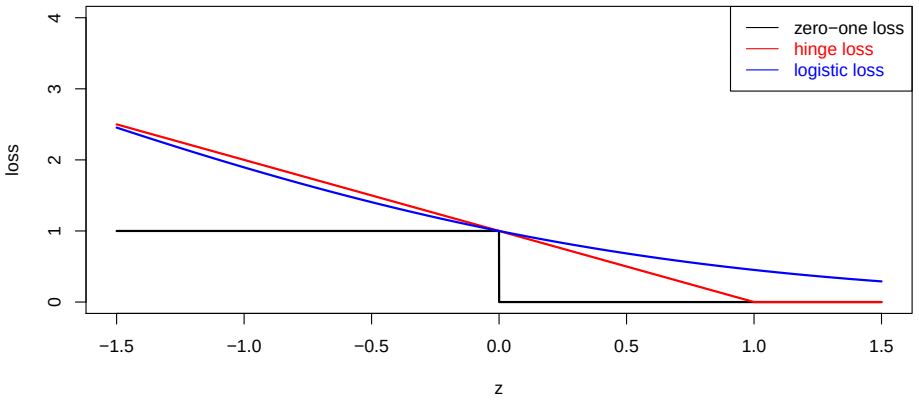

Figure 1.1: A depiction of the zero-one loss and various surrogate losses.

surrogate loss function is the logistic loss, $\tilde{\ell}(z) = \log_2(1+\exp(z))$,[2] which is used for logistic regression. In Figure 1.1 we can see how the hinge loss and the logistic loss are convex upper bounds for the zero-one loss.

Below we will provide a standard approach to controlling the number of mistakes the learner makes with the use of surrogate losses. The analysis of the well-known Perceptron (Rosenblatt, 1958), which uses the hinge loss as the surrogate loss, as well as many other algorithms follows a similar approach. We start the analysis by bounding the zero-one loss in terms of the surrogate loss:

$$\sum_{t=1}^{T} \ell_t(\hat{y}_t) \leq \sum_{t=1}^{T} \tilde{\ell}(z_t). \tag{1.3.1}$$

Now, we will slightly change notation and write $\tilde{\ell}_t(\boldsymbol{w}) = \tilde{\ell}(y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t)$. In the next step we will add and subtract the surrogate loss again, but now evaluated at the offline optimizer $\boldsymbol{u} \in \arg\min_{\{\boldsymbol{w}\in\mathcal{W}\}} \sum_{t=1}^{T} \tilde{\ell}_t(\boldsymbol{w})$:

$$\sum_{t=1}^{T} \ell_t(\hat{y}_t) \leq \sum_{t=1}^{T} \tilde{\ell}_t(\boldsymbol{w}_t) = \sum_{t=1}^{T} \left(\tilde{\ell}_t(\boldsymbol{w}_t) - \tilde{\ell}_t(\boldsymbol{u}) + \tilde{\ell}_t(\boldsymbol{u}) \right).$$

We now almost have a guarantee on the number of mistakes we make, we only need to control $\sum_{t=1}^{T} \left(\tilde{\ell}_t(\boldsymbol{w}_t) - \tilde{\ell}_t(\boldsymbol{u}) \right)$. Fortunately we can use various tools from Online Convex Optimization, for example Continuous Exponential Weights

[2] The logarithm has base 2 because with base 2 at $z = 0$ the surrogate loss is equivalent to the zero-one loss.

or Online Gradient Descent, to guarantee that $\sum_{t=1}^{T}\left(\tilde{\ell}_t(\boldsymbol{w}_t) - \tilde{\ell}_t(\boldsymbol{u})\right)$ is suitably bounded. As we have seen before, each of these Online Convex Optimization algorithms have their own advantages and disadvantages: Continuous Exponential Weights can be slow but has good guarantees and while Online Gradient Descent is very fast it may not have optimal guarantees for some (surrogate) loss functions.

In the end, the guarantee of this type of classifiers is of the form

$$\tilde{\mathcal{R}}_T = \sum_{t=1}^{T} \ell_t(\hat{y}_t) - \sum_{t=1}^{T} \tilde{\ell}_t(\boldsymbol{u}).$$

We will refer to $\tilde{\mathcal{R}}_T$ as the surrogate regret. In the worst-case, the Perceptron, which uses Online Gradient Descent to optimize the surrogate loss, has $O(\sqrt{T})$ surrogate regret with respect to the hinge loss. An alternative to the Perceptron is Online Logistic Regression. Foster et al. (2018a) show that if we use continuous Exponential Weights to optimize the logistic loss the surrogate regret is $O(dN\log(T+1))$, with the drawback that continuous Exponential Weights on the logistic loss has running time $O(\max\{dN, T\}^{12}T)$.

Even though more sophisticated versions of the analysis above exist, many algorithms in the Online Multiclass Classification setting roughly follow the same approach. In Chapter 6 we will introduce a new approach that provides a randomized linear time algorithm with $O(K)$ expected surrogate regret, where the expectation is with respect to the learner's randomness. In particular, we will exploit the gap between the zero-one loss and the surrogate loss from equation 1.3.1. As can be seen in Figure 1.1 this upper bound is wasteful for many values of z as the gap between the zero-one loss and the surrogate loss can be quite substantial. By exploiting the aforementioned gap we are able to significantly reduce the impact of $\sum_{t=1}^{T} \tilde{\ell}_t(\boldsymbol{w}_t) - \tilde{\ell}_t(\boldsymbol{u})$ on the surrogate regret bound, which leads to our new result.

1.3.2 Bandit Information

In the Bandit Multiclass Classification setting (Kakade et al., 2008) the learner only receives $\mathbb{1}[\hat{y}_t \neq y_t]$ as feedback. This means that we can no longer directly use the surrogate loss approach to bound the number of mistakes the learner makes. However, since the learner does know $\mathbb{1}[\hat{y}_t \neq y_t]$ in all rounds, in rounds where $\hat{y}_t = y_t$ the learner also knows y_t. So how does the learner leverage this to guarantee a suitable bound on the number of mistakes?

As in Section 1.2.1 the learner will have to randomize his prediction: $\hat{y}_t \sim q_t$. If we use a technique called importance weighting, which multiplies the surrogate loss by

$\mathbb{1}[\hat{y}_t = y_t] q_t(\hat{y}_t)^{-1}$, the learner can use the weighted surrogate losses to update $\boldsymbol{w}_t$. Note that this means that we only update whenever we have guessed the correct label, which hopefully happens often. In expectation the weighted surrogate losses are equivalent to their full-information counterparts, which means the learner could use the standard techniques from the previous section to provide surrogate regret guarantees.

As in the full-information setting the learner can use algorithms from Online Convex Optimization to optimize the weighted surrogate losses. In the bandit setting the predictions of the learner are randomized so the guarantees are for the expected surrogate regret, where the expectation is with respect to the randomness of the learner. Several authors have proposed polynomial time algorithms that have a $O(N\sqrt{dT\ln(T+1)})$ expected surrogate regret bound (see for example Hazan and Kale (2011); Beygelzimer et al. (2017); Foster et al. (2018a)). In Chapter 6 we will exploit the gap between the surrogate loss and the zero-one loss to provide the first linear time algorithm with $O(N\sqrt{T})$ expected surrogate regret bounds with respect to various surrogate losses. Interestingly, our new algorithm improves upon the expected surrogate regret bound of slower algorithms by a factor of $\sqrt{d\log(T+1)}$, which makes our new algorithm the first answer to the open question of Abernethy and Rakhlin (2009) with an expected surrogate regret bound that does not depend on the dimension of the feature vectors.

1.4 Organisation

The remainder of this dissertation is concerned with various settings and algorithms in Online Learning. In Chapter 2 we show that many algorithms in Online Learning are special cases of Exponential Weights. We also provide a reduction for several adaptive expert algorithms based on Exponential Weights, which recovers Squint (Koolen and Van Erven, 2015), iProd (Koolen and Van Erven, 2015), and Coin Betting for experts (Orabona and Pál, 2016).

Throughout this dissertation we provide several new adaptive algorithms. In Chapter 3 we show how we can adapt to unknown noise in the unconstrained Online Convex Optimization setting, which allows users to choose their privacy requirements without having to disclose them to whoever receives their data. In Chapter 4 we study Bandit Convex Optimization methods that adapt to the norm of the offline optimizer, a topic that has only been studied before for its full-information counterpart. We show that algorithms from the full information setting can be adapted to develop algorithms that adapt to the norm of the offline optimizer for linear bandits. These ideas are then extended to the Bandit Convex Optimization

setting by using a new single-point gradient estimator and carefully designed surrogate losses. In Chapter 5 we introduce MetaGrad, which is an algorithm that is adaptive to a broad class of loss functions. We then improve the running time of MetaGrad by applying sketching methods and evaluate the performance of several versions of MetaGrad in numerous experiments.

As we mentioned above, in Chapter 6 we provide a new approach to Online Multiclass Classification based on exploiting the gap between the zero-one loss and a surrogate loss. In the Bandit Multiclass Classification setting we use our new approach to provide the first linear time algorithm with $O(N\sqrt{T})$ surrogate regret. Furthermore, the surrogate regret of this new bandit algorithm is independent of the dimension of the feature vector, contrary to algorithms with similar surrogate regret bounds in the Bandit Multiclass Classification setting.

Finally, in Chapter 7 we pose an open problem which asks for a fast and optimal algorithm for online portfolio selection. We provide an algorithm and the first steps of the analysis which shows that in some special cases this algorithm indeed yields the optimal regret bound.