ECE 8803: Online Decision Making in Machine Learning **Fall 2022**

Lecture 5: September 6

Lecturer: Vidya Muthukumar

Last week, we introduced the simplest interesting setting of online learning, i.e. predicting a binary sequence of 0's and 1's. We have looked at explicit algorithms that predict this sequence as accurately as possible without making any assumptions on how the binary sequence was generated; in fact, the sequence could be adversarial. Last lecture, we introduced the *multiplicative/exponential weights* algorithm and showed that it achieved regret $R_T \leq c\sqrt{T}$ with the choice of parameter $\eta = 1/\sqrt{T}$ against any binary sequence, even one that is adversarially generated.

This leads us to two natural questions:

- Is the $\sqrt{T}$ guarantee on regret the best we can do?
- How much of this generalizes to large-scale online learning settings, i.e. beyond binary prediction?

In this lecture, we will provide answers to both of these questions.

5.1. Recap: Binary prediction, regret, and the multiplicative weights algorithm

We first recap our relevant notion for binary sequence prediction and regret. Our stated goal has been to predict the next value of a binary sequence, $X_t \in \{0,1\}$, from the past realizations, denoted by $X^{t-1} := \{X_1, \ldots, X_{t-1}\}$. Our prediction is denoted by $\widehat{X}_t$ (which, in general, can be randomized), and our loss function is given by $\ell(\widehat{X}_t; X_t)$. The overall goal is to choose a prediction strategy $f_{\mathsf{predict}}(\cdot)$ to minimize the total loss $H_T := \mathbb{E}\left[\sum_{t=1}^T \ell(\widehat{X}_t, X_t)\right]$, where the expectation is with respect to the randomness in the algorithm only. Importantly, the sequence $\{X_1, \ldots, X_T\}$ can be completely arbitrary, even generated *adversarially* to the prediction process, i.e. in order to try and *maximize* the algorithm's expected loss H_T. We introduced the metric of performance of *regret* with respect to the best constant predictor in hindsight (who is able to see the entire stream of data at once) as

$$R_T(X^T) := \underbrace{H_T}_{\text{our algorithm}} - \underbrace{\min_{x\in\{0,1\}} L_{T,x}}_{\text{best fixed prediction}} . \tag{5.1}$$

Above, we defined $L_{t,x} := \sum_{s=1}^t \mathbb{I}[X_s \neq x]$ for each $x \in \{0,1\}$. Under this notation, it is worth noting that we can write $L_T^* := \min\{L_{T,0}, L_{T,1}\}$, and so from here on we denote $\min_{x\in\{0,1\}} L_{T,x} := L_T^*$ as shorthand.

Therefore, the best fixed/constant prediction in hindsight does the following: it looks at the entire sequence all at once, and picks the letter that minimizes the total loss, which simply corresponds to the letter that appeared more often (since we are measuring loss using the 0-1 loss function).

The *multiplicative/exponential weights algorithm* that we introduced last week is a randomized algorithm and predicts $\widehat{X}_t = x$ with probability $p_{t,x}$, where we defined

$$p_{t,x} = \frac{e^{-\eta L_{t-1,x}}}{e^{-\eta L_{t-1,1}} + e^{-\eta L_{t-1,0}}}. \tag{5.2}$$

Equation (5.2) is written in the form of an *exponential-weight update*. We also noted that we could write the un-normalized weights as $w_{t,x} = e^{-\eta L_{t-1,x}}$ (which leads to $p_{t,x} = \frac{w_{t,x}}{w_{t,1}+w_{t,0}}$, and using this expression note that the weights are updated in a *multiplicative* fashion, i.e.

$$w_{t+1,x} = w_{t,x} \cdot e^{-\eta \cdot \mathbb{I}[X_t \neq x]} \text{ for each } x \in \{0,1\}. \tag{5.3}$$

This is the reason that this algorithm is also commonly called the *multiplicative weights algorithm* (MWA).

Intuitively, the parameter η controls the amount of randomization incorporated in the algorithm; a higher value of η means less randomization while a lower value of η means more randomization. Last lecture, we saw that this algorithm for the choice of parameter $\eta = 1/\sqrt{T}$ optimizes the tradeoff between randomizing (to avoid being misled by the adversary) and exploiting past information to achieve a universal regret guarantee of $R_T \leq c\sqrt{T}$ against *any* binary sequence.

5.2. Why $\mathcal{O}(\sqrt{T})$-regret is the best we can do: An informal discussion

It turns out that this guarantee of $\mathcal{O}(\sqrt{T})$-regret (i.e. sublinear regret) is the best we can do, at least in a worst-case sense. This makes MWA one out of many *optimal* algorithms for adversarial sequence prediction.

To see why $\mathcal{O}(\sqrt{T})$ is unimprovable, it is instructive to consider the following sequence:

$$X_t \text{ i.i.d. } \sim \text{Bernoulli}(1/2),$$

that is, $X_t = 1$ with probability $1/2$ on every round and the sequence values are generated independently. Note that this is a very different case from the stochastic sequence with $p > 1/2$ or $p < 1/2$, where there was a "drift" in one direction (we either see many more 1's than 0's or vice-versa). This special case $p = 1/2$ renders the sequence *fundamentally unpredictable* because we are equally likely to see a 0 or a 1 on every round; thus, it is apriori unclear whether the better decision in hindsight would have been to go with all 1's or all 0's. Moreover, (unlike our example of alternating 0's and 1's that we saw last Tuesday) there is no discernible pattern in the sequence: the past does not yield any side information about the future and whether we are more or less likely to see one of these.

To see this more clearly, we consider an arbitrary prediction algorithm that yields $\widehat{X}_t = 1$ with probability $p_{t,x}$. Then, since $X_t = 1$ with probability $1/2$ (and independent of the past as well as $\widehat{X}_t$), we are equally likely to get a loss of 1 or 0 on every round regardless of what

we do. Formally, we can take an expectation both over the algorithm *and* the randomness in the sequence to get

$$\mathbb{E}[H_T] = \mathbb{E}\left[\sum_{t=1}^{T} \ell(\widehat{X}_t, X_t)\right] = \sum_{t=1}^{T} \frac{1}{2} = \frac{T}{2}.$$

This tells us that, on average, any algorithm would experience a loss of $T/2$ on this unpredictable stochastically generated sequence. Now, consider a fixed realization of a stochastic sequence — it turns out that the realization itself will encode a tiny fluctuation of $\Theta(\sqrt{T})$ in either the number of 1's or 0's: in other words, for half the realizations, we would have seen $\Theta(\sqrt{T})$ more 1's than 0's; and for the other half of the realizations, we would have seen $\Theta(\sqrt{T})$ more 0's than 1's. More formally, this yields

$$\mathbb{E}[L_T^*] = \mathbb{E}[\min\{L_{T,0}, L_{T,1}\}] \leq \frac{T}{2} - C\sqrt{T} \tag{5.4}$$

(where the expectation was taken over the randomness in the sequence realizations) for some constant $C > 0$. Putting these facts together yields a regret of $\mathbb{E}[R_T] \geq \Theta(\sqrt{T})$ for *any* online algorithm, on average on this sequence model.

Intuitively, the reason that this type of stochastic sequence imposes this fundamental limitation on performance is that the infinitesmal, $\Theta(\sqrt{T})$-fluctuation: a) is *impossible to anticipate* by any online algorithm, even if the probability model for the sequence was known beforehand, b) can *go in either direction, i.e. is equally likely to encode more* 0*'s than* 1*'s or* 1*'s than* 0*'s*, and c) is not a *determinstic, predictable pattern in the sequence.* A formal proof of Equation (5.4) can be conducted using the *central limit theorem* and is outside the scope of this class.

5.3. From binary sequence prediction to decision-making using experts

We now turn to the question of how to generalize the ideas we have explored thus far from binary sequence prediction to more general and real-world online decision-making settings. We have been using the notation $\mathbb{I}[X_t \neq x]$ to indicate the "instantaneous" losses that are incurred by predicting 0 and 1 respectively. It turns out that all of the ideas that we have explored in class can be significantly extended to a online decision making setting where we employ the advice of K "experts". Our decision now constitutes the choice of expert $\widehat{X}_t \in [K] := \{1, \ldots, K\}$ on every round, and every expert $x \in [K]$ incurs a loss of $\ell_{t,x}$. Each of the losses incurred by each expert is revealed, but only after our choice of expert $\widehat{X}_t$ is made. The goal remains the same as before: to aggregate the performance of all experts in an optimal way regardless of how these loss functions are chosen. *We will assume that the loss functions remain bounded between* 0 *and* 1*, i.e.* $\ell_{t,x} \in [0, 1]$ *for all experts* $x \in [K]$ *and all rounds* t*.*

Accordingly, we as the online learner now choose a distribution over all K experts, given by $\boldsymbol{p}_t := \begin{bmatrix} p_{t,1} & \ldots & p_{t,K} \end{bmatrix}$. Our expected loss is now defined as $H_T := \mathbb{E}[\sum_{t=1}^{T} \ell_{t,\widehat{X}_t}] = \sum_{t=1}^{T} \sum_{x \in [K]} p_{t,x} \ell_{t,x}$. The regret is defined in a similar manner as before, but now with respect to the *best expert in hindsight*:

$$R_T := H_T - \min_{x \in \{1,\ldots,K\}} L_{T,x}. \tag{5.5}$$

5.3.1 Applications of the experts paradigm

This paradigm of *decision-making using expert advice* (DEA) was first considered in the 1950's (Hannan, 1957) with a relatively abstract motivation of game playing[1]: the learner has K actions available to her, and her loss functions are induced by the actions of an opponent who is adversarial to her. However, this paradigm has several applications that are significantly less abstract. DEA saw a significant resurgence in the 1980's and 1990's through the motivations of binary sequence compression (the Lempel-Ziv compression scheme (Ziv and Lempel, 1977) is actually an instance of online learning!), financial forecasting (Cover, 2011), and ensemble methods in machine learning (Freund and Schapire, 1997). DEA-based applications are heavily used in all of these domains. Here, we provide a high-level and informal description of each of these applications in the DEA framework.

Example 1 (Weather forecasting with experts and 0-1 loss) *Our classic example of binary sequence prediction can be written in the DEA paradigm with $K = 2$ experts: expert* 0 *would always predict* 0 *("rain") and incur a loss of $\ell_{t,0} = \mathbb{I}[X_t \neq 0]$, while expert* 1 *would always predict* 1 *and incur a loss of $\ell_{t,1} = \mathbb{I}[X_t \neq 1]$. We can make this weather-prediction problem more realistic and interesting by considering a larger set of K experts who make more sophisticated* weather forecasts*. At each round, expert $x \in [K]$ would make a forecast of weather $f_{t,x}$ on day t, and if we choose expert $\widehat{X}_t = x$, we go with the prediction $f_{t,x}$ of the binary sequence. Accordingly, the loss function of expert x on day t is given by $\ell_{t,x} = \mathbb{I}[X_t \neq f_{t,x}]$. Our goal is to* ***aggregate*** *these experts' forecasts to come up with our own that does as well as possible on possibly unpredictable weather. For example, we could be trying to aggregate the forecasts of a) AccuWeather, b) Weather.com, and c) NOAA to predict as accurately as possible. This would constitute a weather prediction problem with* 3 *experts. Our notion of regret is with respect to accuracy that was achieved by the best single forecaster in hindsight, a natural benchmark for this kind of application.*

Example 2 (A simplified[2] variant of portfolio optimization) *Suppose we have K assets that we wish to invest in, and a fixed capital that we can spread across these assets — we denote our investment at "time step" t by the K-dimensional probability vector $\boldsymbol{p}_t$, where $p_{t,x}$ denotes the fraction of capital that we invest in asset x. The value of these assets changes across time, and at every round we can denote the profit that we would have made had we invested in asset x by $r_{t,x}$. Naturally, our goal here is to maximize our long-term profit. We can write this in the DEA paradigm by denoting the K assets as experts, our decision at every time step as the fraction of capital $\{p_{t,x}\}_{x\in[K]}$, and our loss function as $\ell_{t,x} = -r_{t,x}$ (i.e. the negative profit). The regret is then measured with respect to the best fixed asset we should have invested in in hindsight.*

The stock market is a quintessential example of unpredictability: assets heavily fluctuate in their value, and accurately forecasting them is extremely difficult (although statistical models for this do exist, their relative success continues to be heavily debated). Thus, the adversarial model actually makes sense to assume: in particular, a single asset may not consistently be the best across time.

It is clear that this a natural manifestation of the DEA paradigm; in particular, "hedging" across assets in the sense of MWA makes intuitive sense. Using these algorithms for portfolio

1. We will return to this game-theoretic motivation at the end of the course.

optimization implies that we can achieve a profit that is almost as good *(in the sense of regret) as the best fixed allocation across time in hindsight. See Cover (2011) for a description of "universal portfolio" algorithms that achieve this kind of guarantee (in a more complex setting than the one presented here).*

Example 3 (Aggregating "weak" machine learning models on stochastic data) *The classification task, which involves predicting a discrete-valued label y from input x, is a classical goal of machine learning (ML). Suppose that we have a training dataset given by $\{(x_t, y_t)\}_{t=1}^T$, and we run an iterative algorithm to **train** a ML model on this dataset where at every iteration t, we see x_t and make a prediction $\widehat{f}(x_t)$, after which we see the output y_t. Our goal is to eventually obtain a ML model (at the end of T iterations) that predicts well on fresh **test samples**.*

*In the 1990's, the development of state-of-the-art ML models was still in flux, and ML researchers frequently encountered a scenario in which they had access to K "weak" classifiers that did well on some of the training examples, but not others. Concretely speaking, classifier $x \in [K]$ made predictions of the form $y = f_k(x)$ and would do slightly better than random guessing on average, i.e. $\mathbb{P}[f_k(x) = y] \geq \frac{1}{2} + \gamma$ for some $\gamma > 0$, but not by much (i.e. γ would be very very small). It turns out that we can get **much** better performance by aggregating these K classifiers using an adaptive online majority-voting scheme that is directly inspired by MWA (Freund et al., 1997; Freund and Schapire, 1997). As we are boosting the performance of these individually weak classifiers to produce a strong classifier, this scheme is commonly called **boosting**. Boosting still finds widespread use as a technique today, especially in applications where the design of state-of-the-art models and algorithms remains in flux.*

Example 4 (Online supervised prediction against poisoned data) *In the more modern ML milieu, we are typically able to find a single classifier that achieves state-of-the-art performance on data that is generated stochastically and iid, i.e. (x_i, y_i) i.i.d. $\sim \mathcal{D}$ for some fixed distribution $\mathcal{D}$. However, in some applications of ML, notably those where the data is generated by people (common examples involve detection of spam, fraud or explicit content), the data could be generated adversarially with the aim of **worsening** prediction performance as much as possible. Concretely, the data can be poisoned by generating $y_i \in \{-1, 1\}$ arbitrarily rather i.i.d. One could apply the methods from this class and generate an **ensemble** of ML models that achieves relatively low prediction loss even against such poisoned data, doing almost as well as the fixed model that would have performed best, in hindsight, over all of the poisoned data.*

*In general, these models will come from a model family — such as the set of linear models, decision trees or neural networks. Ensembling ML models involves treating different models in the model family as "experts" and randomizing our choice of model to make predictions. For model families like linear models and neural networks, the models are parameterized by weights that can take on **uncountably infinite** numbers of values — so the number of experts $K = \infty$ and we cannot directly apply the DEA paradigm (although we could if we discretized, or quantized our weight values appropriately). A more directly applicable case of the DEA paradigm arises in the model family of **decision trees** for classification tasks: an example is provided in Figure 5.1. A decision tree constitutes rule-based ML and can be modified by changing the decisions at each of the leaf nodes of the tree — leading to a*

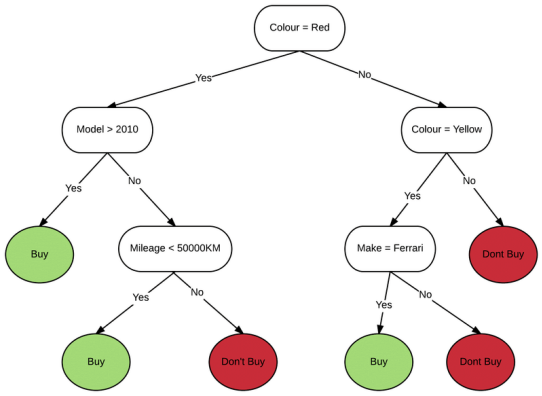

Figure 5.1: A depiction of a decision tree for binary classification. In this example, 1 would represent the choice to buy and -1 would represent the choice to not buy. Note that we could change the predictions made by this decision tree by changing any of the leaf nodes. This generates a *model family* of size equal to $2^{\text{number of leaf nodes}} = 2^6$ for this particular example.

finite-sized family of models. More concretely, a tree of depth d leads to $L = 2^d$ leaves, and thus $K = 2^{2^d}$ possible decision trees (which are the "experts" in this application).

5.3.2 MWA for decision-making using expert advice

Generalizations of FTL and MWA can be obtained to this setting of decision-making using K experts and are defined below.

Definition 1 (FTL for DEA) *The FTL algorithm uses the expert that accumulated the minimal loss up to round $(t-1)$, i.e. $\widehat{X}_t = \arg\min_{x\in[K]} L_{t-1,x}$.*

Definition 2 (MWA for DEA) *The MWA algorithm randomizes the choice of expert at round t as $p_{t,x} = \mathbb{P}[\widehat{X}_t = x]$ for every $x \in [K]$, where*

$$p_{t,x} = \frac{e^{-\eta L_{t-1,x}}}{\sum_{x'\in[K]} e^{-\eta L_{t-1,x'}}}.$$

As with the special case of binary prediction, we can express this as a multiplicative-weights update as well: denote $w_{t,x} = e^{-\eta L_{t-1,x}}$ as the un-normalized weight on every expert (and start with $w_{1,x} = 1$ for all $x \in [K]$). Then, we have $w_{t+1,x} = w_{t,x} \cdot e^{-\eta \ell_{t,x}}$ and $p_{t,x} = \frac{w_{t,x}}{\sum_{x'\in[K]} w_{t,x'}}$.

Since FTL failed to obtain sublinear in T regret even in the binary prediction setting, it clearly will not succeed in this more general K-experts setting. However, MWA will continue to enjoy a $\mathcal{O}(\sqrt{T})$ regret guarantee. Of particular interest is the dependence on the number of experts K, which the following foundational result shows is very small (only logarithmic!)

Theorem 3 (Cesa-Bianchi et al. (1997); Freund and Schapire (1997)) *Consider the K-experts setting with losses $l_{t,x} \in [0,1]$ for all $x \in \{1,\ldots,K\}$. MWA with the learning rate $\eta = \sqrt{\frac{\log K}{T}}$ achieves the regret guarantee*

$$R_T = \mathcal{O}(\sqrt{T \log K}).$$

Thus, the regret continues to grow at the rate of $\sqrt{T}$ in the total number of prediction rounds, and also grows logarithmically in the number of experts K. This logarithmic dependence is very nice and particularly useful in large-scale applications like Example 4 (prediction on poisoned data), where the number of experts (ML models) can be very large. Remarkably, the proof of the theorem above turns out to directly use the proof idea that you saw in Lecture 4 and generalize it to this much more widely applicable setting of a larger number of experts and general loss functions! We provide a brief sketch of how and why this works below.

Proof sketch: We recall the decomposition that we introduced in Lecture 4 of the form

$$R_T = H_T - L_T^* = \underbrace{H_T - M_T}_{\text{mix regret}} + \underbrace{M_T - L_T^*}_{\text{mix loss approximation error}}.$$

For the general DEA paradigm, we can naturally define the cumulative mix loss M_T as

$$M_T := \sum_{t=1}^{T} m_t \text{ where}$$
$$m_t := -\frac{1}{\eta} \log \left(\sum_{x' \in [K]} p_{t,x'} \cdot e^{-\eta \ell_{t,x'}} \right).$$

It is a worthwhile exercise to to show that this definition of mix loss generalizes the one that we saw for binary prediction. Note that, just as in the binary case, we can write $m_t = -\frac{1}{\eta} \log \left(\mathbb{E}[e^{-\eta \ell_{t,\widehat{X}_t}}] \right)$ (where the expectation is only over the randomness in the algorithm, $\widehat{X}_t \sim \mathbf{p}_t$). Moreover, we can write $H_T = \sum_{t=1}^{T} h_t$ where $h_t = \mathbb{E}[\ell_{t,\widehat{X}_t}]$. Then, we can use similar ideas from Lecture 4 to characterize the difference terms $\delta_t = h_t - m_t$; i.e. the random variable $\ell_{t,\widehat{X}_t}$ is still bounded between $[0,1]$ (*why?*) and we can still apply Hoeffding's inequality.

Exercise 1 *Do this calculation to show that the mix regret is bounded in an identical manner to binary prediction, i.e. $H_T - M_T \leq \frac{\eta T}{8}$.*

Next, we look at the approximation error $M_T - L_T^*$.

Exercise 2 *Show that we can write $m_t = \frac{1}{\eta} \log \left(\sum_{x' \in [K]} w_{t,x} \right) - \frac{1}{\eta} \log \left(\sum_{x' \in [K]} w_{t+1,x} \right)$.*

We now use Exercise 2 to get a nice expression for M_T. As with the calculation from Lecture 4, most of the terms in the sum cancel out and we get

$$\begin{aligned} M_T = \sum_{t=1}^{T} m_t &= \frac{1}{\eta}\log(\sum_{x' \in [K]} w_{1,x'}) - \frac{1}{\eta}\log(\sum_{x' \in [K]} w_{T+1,x'}) \\ &= -\frac{1}{\eta}\log\left(\frac{\sum_{x' \in [K]} e^{-\eta L_{T,x'}}}{K}\right) \end{aligned}$$

where in the last step we substituted the expressions for the weights on each expert at the beginning and the end of the prediction process. Now, just as in Lecture 4 we get

$$\begin{aligned} M_T &= -\frac{1}{\eta}\log\left(\frac{\sum_{x' \in [K]} e^{-\eta L_{T,x'}}}{K}\right) \leq -\frac{1}{\eta}\log\left(\frac{e^{-\eta L_T^*}}{K}\right) \\ &= L_T^* + \frac{\log K}{\eta}. \end{aligned}$$

Note that this is identical to the calculation we did for binary prediction, except that we now have a $\log K$ dependence (as there are K experts rather than just 2)! Putting everything together gives us our regret bound of

$$R_T \leq \frac{\eta T}{8} + \frac{\log K}{\eta},$$

and substituting $\eta = \sqrt{\frac{\log K}{T}}$ yields the desired result: $R_T = \mathcal{O}(\sqrt{T \log K})$.

5.4. Additional references

- For a historical perspective on the discoveries of various online learning algorithms as well as historical motivation of the online learning paradigm, see the excellent survey by Foster and Vohra (1999).
- For an excellent machine-learning-oriented survey and history of online learning, see Blum (1998). This includes the inspiration for MWA (or the weighted-majority algorithm) from the perceptron, "winnowing" algorithms and mistake bounds.
- For recent perspective on how MWA is used for algorithm design, see the excellent survey by Arora et al. (2012).
- Ideas from DEA continue to be used in diverse applications today. In particular, DEA is used to aggregate "experts" in a number of ML and algorithmic applications. For example, the *meta-learning* application constitutes aggregating the output of ML algorithms trained for different sub-tasks, to generalize well on a new task. Here, the various ML algorithms are the "experts". See Finn et al. (2019) for a recent example of DEA applied to meta-learning. Another example lies in the design of a *meta-algorithm* for a challenging NP-hard optimization problem (such as the k-satisfiability problem (k-SAT)) that optimally combines heuristics that are each good for certain subclasses

of instances. Here, the heuristics are the "experts". See Xu et al. (2008) and code there-in for a description of the meta-algorithm *SAT-Zilla*, which has won several SAT solver competitions.

References

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

Avrim Blum. On-line algorithms in machine learning. In *Online algorithms*, pages 306–325. Springer, 1998.

Nicolo Cesa-Bianchi, Yoav Freund, David Haussler, David P Helmbold, Robert E Schapire, and Manfred K Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44 (3):427–485, 1997.

Thomas M Cover. Universal portfolios. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*, pages 181–209. World Scientific, 2011.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.

Dean P Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 334–343, 1997.

James Hannan. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.