A Collaborative Mechanism for Crowdsourcing Prediction Problems

Jacob Abernethy
Division of Computer Science
University of California at Berkeley
jake@cs.berkeley.edu

Rafael M. Frongillo
Division of Computer Science
University of California at Berkeley
raf@cs.berkeley.edu

Abstract

Machine Learning competitions such as the Netflix Prize have proven reasonably successful as a method of "crowdsourcing" prediction tasks. But these competitions have a number of weaknesses, particularly in the incentive structure they create for the participants. We propose a new approach, called a Crowdsourced Learning Mechanism, in which participants collaboratively "learn" a hypothesis for a given prediction task. The approach draws heavily from the concept of a prediction market, where traders bet on the likelihood of a future event. In our framework, the mechanism continues to publish the current hypothesis, and participants can modify this hypothesis by wagering on an update. The critical incentive property is that a participant will profit an amount that scales according to how much her update improves performance on a released test set.

1 Introduction

The last several years has revealed a new trend in Machine Learning: prediction and learning problems rolled into prize-driven competitions. One of the first, and certainly the most well-known, was the *Netflix prize* released in the Fall of 2006. Netflix, aiming to improve the algorithm used to predict users' preferences on its database of films, released a dataset of 100M ratings to the public and asked competing teams to submit a list of predictions on a test set withheld from the public. Netflix offered $1,000,000 to the first team achieving prediction accuracy exceeding a given threshold, a goal that was eventually met. This competitive model for solving a prediction task has been used for a range of similar competitions since, and there is even a new company (kaggle.com) that creates and hosts such competitions. Such prediction competitions have proven quite valuable for a couple of important reasons: (a) they leverage the abilities and knowledge of the public at large, commonly known as "crowdsourcing", and (b) they provide an incentivized mechanism for an individual or team to apply their own knowledge and techniques which could be particularly beneficial to the problem at hand. This type of prediction competition provides a nice tool for companies and institutions that need help with a given prediction task yet can not afford to hire an expert. The potential leverage can be quite high: the Netflix prize winners apparently spent more than $1,000,000 in effort on their algorithm alone.

Despite the extent of its popularity, is the Netflix competition model the ideal way to "crowdsource" a learning problem? We note several weaknesses:

It is anti-collaborative. Competitors are strongly incentivized to keep their techniques private. This is in stark contrast to many other projects that rely on crowdsourcing – Wikipedia being a prime example, where participants must build off the work of others. Indeed, in the case of the Netflix prize, not only do leading participants lack incentives to share, but the work of non-winning competitors is effectively wasted.

The incentives are skewed and misaligned. The winner-take-all prize structure means that second place is as good as having not competed at all. This ultimately leads to an equilibrium where only a few teams are actually competing, and where potential new teams never form since catching up seems so unlikely. In addition, the fixed achievement benchmark, set by Netflix as a 10% improvement in prediction RMSE over a baseline, leads to misaligned incentives. Effectively, the prize structure implies that an improvement of %9.9 percent is worth nothing to Netflix, whereas a 20% improvement is still only worth $1,000,000 to Netflix. This is clearly not optimal.

The nature of the competition precludes the use of proprietary methods. By requiring that the winner reveal the winning algorithm, potential competitors utilizing non-open software or proprietary techniques will be unwilling to compete. By participating in the competition, a user must effectively give away his intellectual property.

In this paper we describe a new and very general mechanism to crowdsource prediction/learning problems. Our mechanism requires participants to place bets, yet the space they are betting over is the set of *hypotheses* for the learning task at hand. At any given time the mechanism publishes the current hypothesis $\mathbf{w}$ and participants can wager on a modification of $\mathbf{w}$ to $\mathbf{w}'$, upon which the modified $\mathbf{w}'$ is posted. Eventually the wagering period finishes, a set of test data is revealed, and each participant receives a payout according to their bets. The critical property is that every trader's profit scales according to how well their modification *improved the solution* on the test data.

The framework we propose has many qualities similar to that of an *information* or *prediction market*, and many of the ideas derive from recent research on the design of *automated market makers* [7, 8, 3, 4, 1]. Many information markets already exist; at sites like `Intrade.com` and `Betfair.com`, individuals can bet on everything ranging from election outcomes to geopolitical events. There has been a burst of interest in such markets in recent years, not least of which is due to their potential for combining large amounts of information from a range of sources. In the words of Hanson et al [9]: "Rational expectations theory predicts that, in equilibrium, asset prices will reflect all of the information held by market participants. This theorized information aggregation property of prices has lead economists to become increasingly interested in using securities markets to predict future events." In practice, prediction markets have proven impressively accurate as a forecasting tool [11, 2, 12].

The central contribution of the present paper is to take the framework of a prediction market as a tool for information aggregation and to apply this tool for the purpose of "aggregating" a hypothesis (classifier, predictor, etc.) for a given learning problem. The crowd of ML researchers, practitioners, and domain experts represents a highly diverse range of expertise and algorithmic tools. In contrast to the Netflix prize, which pitted teams of participants against each other, the mechanism we propose allows for everyone to contribute whatever knowledge they may have available towards the final solution. In a sense, this approach decentralizes the process of solving the task, as individual experts can potentially apply their expertise to a subset of the problem on which they have an advantage. Whereas a market price can be thought of as representing a consensus estimate of the value of an asset, our goal is to construct a consensus hypothesis reflecting all the knowledge and capabilities about a particular learning problem[1].

Layout: We begin in Section 2.1 by introducing the simple notion of a *generalized scoring rule* $L(\cdot, \cdot)$ representing the "loss function" of the learning task at hand. In Section 2.2 we describe our proposed Crowdsourced Learning Mechanism (CLM) in detail, and discuss how to structure a CLM for a particular scoring function L, in order that the traders are given incentives to minimize L. In Section 3 we give an example based on the design of Huffman codes. In Section 4 we discuss previous work on the design of prediction markets using an *automated prediction market maker* (APMM). In Section 5 we finish by considering two learning settings (e.g. linear regression) and we construct a CLM for each. The proofs have been omitted throughout, but these are available in the full version of the present paper.

Notation: Given a smooth strictly convex function $R : \mathbb{R}^d \to \mathbb{R}$, and points $\mathbf{x}, \mathbf{y} \in \text{dom}(R)$, we define the *Bregman divergence* $D_R(\mathbf{x}, \mathbf{y})$ as the quantity $R(\mathbf{x}) - R(\mathbf{y}) - \nabla R(\mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})$. For any convex function R, we let R^* denote the *convex conjugate* of R, that is $R^*(\mathbf{y}) := \sup_{\mathbf{x} \in \text{dom}(R)} \mathbf{y} \cdot \mathbf{x} - R(\mathbf{x})$. We shall use $\Delta(S)$ to refer to the set of integrable probability distributions over the set

[1] It is worth noting that Barbu and Lay utilized concepts from prediction markets to design algorithms for classifier aggregation [10], although their approach was unrelated to crowdsourcing.

S, and Δ_n to refer to the set of probability vectors $\mathbf{p} \in \mathbb{R}^n$. The function $H : \Delta_n \to \mathbb{R}$ shall denote the *entropy function*, that is $H(\mathbf{p}) := -\sum_{i=1}^n \mathbf{p}(i) \log \mathbf{p}(i)$. We use the notation $\mathrm{KL}(\mathbf{p}; \mathbf{q})$ to describe the *relative entropy* or *Kullback-Leibler divergence* between distributions $\mathbf{p}, \mathbf{q} \in \Delta_n$, that is $\mathrm{KL}(\mathbf{p}; \mathbf{q}) := \sum_{i=1}^n \mathbf{p}(i) \log \frac{\mathbf{p}(i)}{\mathbf{q}(i)}$. We will also use $\mathbf{e}_i \in \mathbb{R}^n$ to denote the ith standard basis vector, having a 1 in the ith coordinate and 0's elsewhere.

2 Scoring Rules and Crowdsourced Learning Mechanisms

2.1 Generalized Scoring Rules

For the remainder of this section, we shall let $\mathcal{H}$ denote some set of *hypotheses*, which we will assume is a convex subset of $\mathbb{R}^n$. We let $\mathcal{O}$ be some arbitrary set of *outcomes*. We use the symbol X to refer to either an element of $\mathcal{O}$, or a random variable taking values in $\mathcal{O}$.

We recall the notion of a *scoring rule*, a concept that arises frequently in economics and statistics [6].

Definition 1. *Let $\mathcal{P} \subseteq \Delta(\mathcal{O})$ be some convex set of distributions on an outcome space $\mathcal{O}$. A* scoring rule *is a function $S : \mathcal{P} \times \mathcal{O} \to \mathbb{R}$ where, for all $P \in \mathcal{P}$, $P \in \operatorname{argmax}_{Q \in \mathcal{P}} \mathbb{E}_{X \sim P} S(Q, X)$.*

In other words, if you are paid $S(P, X)$ upon stating belief $P \in \mathcal{P}$ and outcome X occurring, then you maximize your expected utility by stating your true belief. We offer a much weaker notion:

Definition 2. *Given a convex hypothesis space $\mathcal{H} \subset \mathbb{R}^n$ and an outcome space $\mathcal{O}$, let $L : \mathcal{H} \times \mathcal{O} \to \mathbb{R}$ be a continuous function. Given any $P \in \Delta(\mathcal{O})$, let $W_L(P) := \operatorname{argmin}_{\mathbf{w} \in \mathcal{H}} \mathbb{E}_{X \sim P}[L(\mathbf{w}; X)]$. Then we say that L is a* Generalized Scoring Rule *(GSR) if $W_L(P)$ is a nonempty convex set for every $P \in \Delta(\mathcal{O})$.*

The generalized scoring rule shall represent the "loss function" for the learning problem at hand, and in Section 2.2 we will see how L is utilized in the mechanism. The hypothesis $\mathbf{w}$ shall represent the advice we receive from the crowd, X shall represent the test data to be revealed at the close of the mechanism, and $L(\mathbf{w}; X)$ shall represent the loss of the advised $\mathbf{w}$ on the data X. Notice that we do not define L to be convex in its first argument as this does not hold for many important cases. Instead, we require the weaker condition that $\mathbb{E}_X[L(\mathbf{w}; X)]$ is minimized on a convex set for any distribution on X.

Our scoring rule differs from traditional scoring rules in an important way. Instead of starting with the desire know about the true value of X, and then designing a scoring rule which incentivizes participants to elicit their belief $P \in \mathcal{P}$, our objective is precisely to minimize our scoring rule. In other words, traditional scoring rules were a means to an end (eliciting P) but our generalized scoring rule is the end itself. One can recover the traditional scoring rule definition by setting $\mathcal{H} = \mathcal{P}$ and imposing the constraint that $P \in W_L(P)$.

A useful class of GSRs L are those based on a Bregman divergence.

Definition 3. *We say that a GSR $L : \mathcal{H} \times \mathcal{O} \to \mathbb{R}$ is* divergence-based *if there exists an alternative hypothesis space $\mathcal{H}' \subset \mathbb{R}^m$, for some m, where we can write*

$$L(\mathbf{w}; X) \equiv D_R(\rho(X), \psi(\mathbf{w})) + f(X) \tag{1}$$

for arbitrary maps $\rho : \mathcal{O} \to \mathcal{H}'$, $f : \mathcal{O} \to \mathbb{R}$, and $\psi : \mathcal{H} \to \mathcal{H}'$, and any closed strictly convex $R : \mathcal{H}' \to \mathbb{R}$ whose convex conjugate R^* is finite on all of $\mathbb{R}^m$.

This property allows us to think of $L(\mathbf{w}; X)$ as a kind of distance between $\rho(X)$ and $\psi(\mathbf{w})$. Clearly then, the minimum value of L for a given X will be attained when $\psi(\mathbf{w}) = \rho(X)$, given that $D_R(\mathbf{x}, \mathbf{x}) = 0$ for any Bregman divergence. In fact, as the following proposition shows, we can even think of the expected value $\mathbb{E}[L(\mathbf{w}; X)]$, as a distance between $\mathbb{E}[\rho(X)]$ and $\psi(\mathbf{w})$.

Proposition 1. *Given a divergence-based GSR $L(\mathbf{w}; X) = D_R(\rho(X), \psi(\mathbf{w})) + f(X)$ and a belief distribution P on $\mathcal{O}$, we have $W_L(P) = \psi^{-1}\big(\mathbb{E}_{X \sim P}[\rho(X)]\big)$.*

We now can see that the divergence-based property greatly simplifies the task of minimizing L; instead of worrying about $\mathbb{E}[L(\cdot; X)]$ one can simply base the hypothesis directly on the expectation $\mathbb{E}[\rho(X)]$. As we will see in section 4, this also leads to efficient prediction markets and crowdsourcing mechanisms.

2.2 The Crowdsourced Learning Mechanism

We will now define our actual mechanism rigorously.

Definition 4. *A* Crowdsourced Learning Mechanism (CLM) *is the procedure in Algorithm 1 as defined by the tuple* $(\mathcal{H}, \mathcal{O}, \texttt{Cost}, \texttt{Payout})$. *The function* $\texttt{Cost} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ *sets the cost charged to a participant that makes a modification to the posted hypothesis. The function* $\texttt{Payout} : \mathcal{H} \times \mathcal{H} \times \mathcal{O} \to \mathbb{R}$ *determines the amount paid to each participant when the outcome is revealed to be* X.

Algorithm 1 Crowdsourced Learning Mechanism for $(\mathcal{H}, \mathcal{O}, \texttt{Cost}, \texttt{Payout})$

Mechanism sets initial hypothesis to some $\mathbf{w}_0 \in \mathcal{H}$
for rounds $t = 0, 1, 2, \ldots$ **do**
    Mechanism posts current hypothesis $\mathbf{w}_t \in \mathcal{H}$
    Some participant places a bid on the update $\mathbf{w}_t \mapsto \mathbf{w}'$
    Mechanism charges participant $\texttt{Cost}(\mathbf{w}_t, \mathbf{w}')$
    Mechanisms updates hypothesis $\mathbf{w}_{t+1} \leftarrow \mathbf{w}'$
end for
Market closes after T rounds and the outcome (test data) $X \in \mathcal{O}$ is revealed
for each t **do**
    Participant responsible for the update $\mathbf{w}_t \mapsto \mathbf{w}_{t+1}$ receives $\texttt{Payout}(\mathbf{w}_t, \mathbf{w}_{t+1}; X)$
end for

The above procedure describes the process by which participants can provide advice to the mechanism to select a good $\mathbf{w}$, and the profit they earn by doing so. Of course, this profit will precisely determine the incentives of our mechanism, and hence a key question is: how can we design Cost and Payout so that participants are incentivized to provide good hypotheses? The answer is that we shall structure the incentives around a GSR $L(\mathbf{w}; X)$ chosen by the mechanism designer.

Definition 5. *For a* CLM $A = (\mathcal{H}, \mathcal{O}, \texttt{Cost}, \texttt{Payout})$, *denote the ex-post profit for the bid* $(\mathbf{w} \mapsto \mathbf{w}')$ *when the outcome is* $X \in \mathcal{O}$ *by* $\texttt{Profit}(\mathbf{w}, \mathbf{w}'; X) := \texttt{Payout}(\mathbf{w}, \mathbf{w}'; X) - \texttt{Cost}(\mathbf{w}, \mathbf{w}')$. *We say that* A implements *a GSR* $L : \mathcal{H}' \times \mathcal{O} \to \mathbb{R}$ *if there exists a surjective map* $\varphi : \mathcal{H} \to \mathcal{H}'$ *such that for all* $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{H}$ *and* $X \in \mathcal{O}$,

$$\texttt{Profit}(\mathbf{w}_1, \mathbf{w}_2; X) = L(\varphi(\mathbf{w}_1); X) - L(\varphi(\mathbf{w}_2); X). \tag{2}$$

If additionally $\mathcal{H}' = \mathcal{H}$ *and* $\varphi = id_{\mathcal{H}}$, *we call* A *an* L*-CLM and say that* A *is* L-incentivized.

When a CLM implements a given L, the incentives are structured in order that the participants will work to minimize $L(\mathbf{w}; X)$. Of course, the input X is unknown to the participants, yet we can assume that the mechanism has provided a public "training set" to use in a learning algorithm. The participants are thus asked not only to propose a "good" hypothesis $\mathbf{w}_t$ but to wager on whether the update $\mathbf{w}_{t-1} \mapsto \mathbf{w}_t$ improves generalization error. It is worth making clear that knowledge of the true distribution on X provides a straightforward optimal strategy.

Proposition 2. *Given a GSR* $L : \mathcal{H} \times \mathcal{O} \to \mathbb{R}$ *and an* L*-CLM* $(\texttt{Cost}, \texttt{Payout})$, *any participant who knows the true distribution* $P \in \mathcal{P}$ *over* X *will maximize expected profit by modifying the hypothesis to any* $\mathbf{w} \in W_L(P)$.

Cost of operating a CLM. It is clear that the agent operating the mechanism must pay the participants at the close of the competition, and is thus at risk of losing money (in fact, it is possible he may gain). How much money is lost depends on the bets $(\mathbf{w}_t \mapsto \mathbf{w}_{t+1})$ made by the participants, and of course the final outcome X. The agent has a clear interest in knowing precisely the potential cost – fortunately this cost is easy to compute. The loss to the agent is clearly the total ex-post profit earned by the participants, and by construction this sum telescopes: $\sum_{t=0}^{T} \texttt{Profit}(\mathbf{w}_t, \mathbf{w}_{t+1}; X) = L(\mathbf{w}_0; X) - L(\mathbf{w}_T; X)$. This is a simple yet appealing property of the CLM: the agent pays only as much in reward to the participants as it benefits from the improvement of $\mathbf{w}_T$ over the initial $\mathbf{w}_0$. It is worth noting that this value could be *negative* when $\mathbf{w}_T$ is actually "worse" than $\mathbf{w}_0$; in this case, as we shall see in section 3, the CLM can act as an insurance policy with respect to the mistakes of the participants. A more typical scenario, of course, is where the participants provide an improved hypothesis, in which case the CLM will run at a cost. We can compute the WorstCaseLoss(L-CLM) $:= \max_{\mathbf{w} \in \mathcal{H}, X \in \mathcal{O}} (L(\mathbf{w}_0; X) - L(\mathbf{w}; X))$. Given a budget

of size $\$B$, the mechanism can always rescale L in order that WorstCaseLoss(L-CLM) $= B$. This requires, of course, that the WorstCaseLoss is finite.

Computational efficiency of operating a CLM. We shall say that a CLM has the *efficient computation (EC)* property if both `Cost` and `Payout` are efficiently computable functions. We shall say a CLM has the *tractable trading (TT)* property if, given a current hypothesis $\mathbf{w}$, a belief $P \in \Delta(\mathcal{O})$ and a budget B, one can efficiently compute an element of the set

$$\operatorname*{argmin}_{\mathbf{w}' \in \mathcal{H}} \Big\{ \mathbb{E}_{X \sim P} \big[\texttt{Profit}(\mathbf{w}, \mathbf{w}', X) \big] \,:\, \texttt{Cost}(\mathbf{w}, \mathbf{w}') \leq B \Big\}.$$

The EC property ensures that the mechanism operator can run the CLM efficiently. The TT property says that *participants* can compute the optimal hypothesis to bet on given a belief on the outcome and a budget. This is absolutely essential for the CLM to successfully aggregate the knowledge and expertise of the crowd – without it, despite their motivation to lower $L(;)$, the participants would not be able to compute the optimal bet.

Suitable collateral requirements. We say that a CLM has the *escrow (ES)* property if the `Cost` and `Payout` functions are structured in order that, given any wager $(\mathbf{w} \mapsto \mathbf{w}')$, we have that $\texttt{Payout}(\mathbf{w}, \mathbf{w}'; X) \geq 0$ for all $X \in \mathcal{O}$. It is clear that, when designing an L-CLM for a particular L, the `Payout` function is fully specified once `Cost` is fixed, since we have the relation $\texttt{Payout}(\mathbf{w}, \mathbf{w}'; X) = L(\mathbf{w}; X) - L(\mathbf{w}'; X) + \texttt{Cost}(\mathbf{w}, \mathbf{w}')$ for every $\mathbf{w}, \mathbf{w}' \in \mathcal{H}$ and $X \in \mathcal{O}$. A curious reader might ask, why not simply set $\texttt{Cost}(\mathbf{w}, \mathbf{w}') \equiv 0$ and $\texttt{Payout} \equiv \texttt{Profit}$? The problem with this approach is that potentially $\texttt{Payout}(\mathbf{w}, \mathbf{w}'; X) < 0$ which implies that the participant who wagered on $(\mathbf{w} \mapsto \mathbf{w}')$ can be indebted to the mechanism and could default on this obligation. Thus the `Cost` function should be set in order to require every participant to deposit at least enough collateral in escrow to cover any possible losses.

Subsidizing with a voucher pool. One practical weakness of a wagering-based mechanism is that individuals may be hesitant to participate when it requires depositing actual money into the system. This can be allayed to a reasonable degree by including a *voucher pool* where each of the first m participants may receive a voucher in the amount of $\$C$. These candidates need not pay to participate, yet have the opportunity to win. Of course, these vouchers must be paid for by the agent running the mechanism, and hence a value of mC is added to the total operational cost.

3 A Warm-up: Compressing an Unfamiliar Data Stream

Let us now introduce a particular setting motivated by a well-known problem in information theory. Imagine a firm is looking to do *compression* on an unfamiliar channel, and from this channel the firm will receive a stream of m characters from an n-sized alphabet which we shall index by $[n]$. The goal is to select a binary encoding of this alpha in such a way that minimizes the total bits required to store the data, as a cost of \$1 is required for each bit.

A first-order approach to encode such a stream is to assign a probability distribution $\mathbf{q} \in \Delta_n$ to the alphabet, and to select an encoding of character i with a binary word of length $\log(1/\mathbf{q}(i))$ (we ignore round-off for simplicity). This can be achieved using Huffman Codes for example, and we refer the reader to Cover and Thomas ([5], Chapter 5) for more details. Thus, given a distribution $\mathbf{q}$, the firm pays $L(\mathbf{q}; i) = -\log \mathbf{q}(i)$ for each character i. It is easy to see that if the characters are sampled from some "true" distribution $\mathbf{p}$, then the expected cost $L(\mathbf{q}; \mathbf{p}) := \mathbb{E}_{i \sim p} [L(\mathbf{q}; i)] = \mathrm{KL}(\mathbf{p}; \mathbf{q}) + H(\mathbf{p})$, which is minimized at $\mathbf{q} = \mathbf{p}$. Not knowing the true distribution $\mathbf{p}$, the firm is thus interested in finding a $\mathbf{q}$ with a low expected cost $L(\mathbf{q}; \mathbf{p})$.

An attractive option available to the firm is to *crowdsource* the task of lowering this cost $L(\cdot; \cdot)$ by setting up an L-CLM. It is reasonably likely that outside individuals have private information about the behavior of the channel and, in particular, may be able to provide a better estimate $\mathbf{q}$ of the true distribution of the characters in the channel. As just discussed, the better the estimate the cheaper the compression.

We set $\mathcal{H} = \Delta_n$ and $\mathcal{O} = [n]$, where a hypothesis $\mathbf{q}$ represents the proposed distribution over the n characters, and X is some character *sampled uniformly* from the stream after it has been observed.

We define $\texttt{Cost}$ and $\texttt{Payout}$ as

$$\texttt{Cost}(\mathbf{q}, \mathbf{q}') := \max_{i \in [n]} \log(\mathbf{q}(i)/\mathbf{q}'(i)), \qquad \texttt{Payout}(\mathbf{q}, \mathbf{q}'; i) := \log(\mathbf{q}(i)/\mathbf{q}'(i)) + \texttt{Cost}(\mathbf{q}, \mathbf{q}'),$$

which is clearly an L-CLM for the loss defined above. It is worth noting that L is a divergence-based GSR if we take $R(\mathbf{q}) = -H(\mathbf{q})$, $\rho(i) = \mathbf{e}_i$, $f \equiv 0$, $\psi \equiv \mathrm{id}_{\Delta_n}$, using the convention $0 \log 0 = 0$ (in fact, L is the LMSR). Finally, the firm will initially set $\mathbf{q}_0$ to be its best guess of $\mathbf{p}$, which we will assume to be uniform (but need not be).

We have devised this payout scheme according to the selection of a single character i, and it is worth noting that because this character is sampled uniformly at random from the stream (with private randomness), the participants *cannot* know which character will be released. This forces the participants to wager on the empirical distribution $\hat{\mathbf{p}}$ of the characters from the stream. A reasonable alternative, and one which lowers the payment variance, is to payout according to the $L(\mathbf{q}; \hat{\mathbf{p}})$, which is also equal to the average of $L(\mathbf{q}; i)$ when i is chosen uniformly from the stream.

The obvious question to ask is: how does this CLM benefit the firm that wants to design the encoding? More precisely, if the firm uses the final estimate $\mathbf{q}_T$ from the mechanism, instead of the initial guess $\mathbf{q}_0$, what is the trade-off between the money paid to participants and the money gained by using the crowdsourced hypothesis? At first glance, it appears that this trade-off can be arbitrarily bad: the worst case cost of encoding the stream using the final estimate $\mathbf{q}_T$ is $\sup_{i, \mathbf{q}_T} -\log(\mathbf{q}_T(i)) = \infty$. Amazingly, however, by virtue of the aligned incentives, the firm has a very strong control of its total cost (the CLM cost plus the encoding cost). Suppose the firm scales L by a parameter α, to separate the scale of the CLM from the scale of the encoding cost (which we assumed to be \$1 per bit). Then given any initial estimate $\mathbf{q}_0$ and final estimate $\mathbf{q}_T$, the expected total cost over $\mathbf{p}$ is

$$\begin{aligned} \text{Total expected cost} &= \overbrace{H(\mathbf{p}) + \mathrm{KL}(\mathbf{p}; \mathbf{q}_T)}^{\text{Encoding cost of using } \mathbf{q}_T \text{ given } \mathbf{p}} + \overbrace{\alpha(\mathrm{KL}(\mathbf{p}; \mathbf{q}_0) - \mathrm{KL}(\mathbf{p}; \mathbf{q}_T))}^{\text{Mechanism's cost of getting advice } \mathbf{q}_T} \\ &= H(\mathbf{p}) + (1-\alpha)\mathrm{KL}(\mathbf{p}; \mathbf{q}_T) + \alpha \mathrm{KL}(\mathbf{p}; \mathbf{q}_0) \end{aligned}$$

Let us spend a moment to analyze the above expression. Imagine that the firm set $\alpha = 1$. Then the total cost of the firm would be $H(\mathbf{p}) + \mathrm{KL}(\mathbf{p}; \mathbf{q}_0)$, which is bounded by $\log n$ for $\mathbf{q}_0$ uniform. Notice that this expression *does not depend on* $\mathbf{q}_T$ – in fact, this cost precisely corresponds to the scenario where the firm had not set up a CLM and instead used the initial estimate $\mathbf{q}_0$ to encode. In other words, for $\alpha = 1$, the firm is entirely neutral to the quality of the estimate $\mathbf{q}_T$; even if the CLM provided an estimate $\mathbf{q}_T$ which performed worse than $\mathbf{q}_0$, the cost increase due to the bad choice of $\mathbf{q}$ is recouped from payments of the ill-informed participants.

The firm may not want to be neutral to the estimate of the crowd, however, and under the reasonable assumption that the final estimate $\mathbf{q}_T$ will improve upon $\mathbf{q}_0$, the firm should set $0 < \alpha < 1$ (of course, positivity is needed for nonzero payouts). In this case, the firm will strictly gain by using the CLM when $\mathrm{KL}(\mathbf{p}; \mathbf{q}_T) < \mathrm{KL}(\mathbf{p}; \mathbf{q}_0)$, but still has some insurance policy if the estimate $\mathbf{q}_T$ is poor.

4 Prediction Markets as a Special Case

Let us briefly review the literature for the type of prediction markets relevant to the present work. In such a prediction market, we imagine a future event to reveal one of n uncertain outcomes. Hanson [7, 8] proposed a framework in which traders make "reports" to the market about their internal belief in the form of a distribution $\mathbf{p} \in \Delta_n$. Each trader would receive a reward (or loss) based on a function of their proposed belief and the belief of the previous trader, and the function suggested by Hanson was the *Logarithmic Market Scoring Rule* (LMSR). It was shown later that the LMSR-based market is equivalent to what is known as a *cost function based automated market makers*, proposed by Chen and Pennock [3]. More recently a much broader equivalence was established by Chen and Wortman Vaughan [4] between markets based on cost functions and those based on scoring rules.

The market framework proposed by Chen and Pennock allows traders to buy and sell *Arrow-Debreu* securities (equivalently: shares, contracts), where an Arrow-Debreu security corresponding to outcome i pays out \$1 if and only if i is realized. All shares are bought and sold through an *automated market maker*, which is the entity managing the market and setting prices. At any time period, traders can purchase *bundles* of contracts $\mathbf{r} \in \mathbb{R}^n$, where $\mathbf{r}(i)$ represents the number of shares purchased on

outcome i. The price of a bundle $\mathbf{r}$ is set as $C(\mathbf{s}+\mathbf{r}) - C(\mathbf{s})$, where C is some differentiable convex cost function and $\mathbf{s} \in \mathbb{R}^n$ is the "quantity vector" representing the total number of outstanding shares. The LMSR cost function is $C(\mathbf{s}) := \frac{1}{\eta} \log\left(\sum_{i=1}^{n} \exp(\eta \mathbf{s}(i))\right)$.

This cost function framework was extended by Abernethy et al. [1] to deal with prohibitively large outcome spaces. When the set of potential outcomes $\mathcal{O}$ is of exponential size or even infinite, the market designer can offer a restricted number of contracts, say n ($\ll |\mathcal{O}|$), rather than offer an Arrow-Debreu contract for each member of $\mathcal{O}$. To determine the payout structure, the market designer chooses a function $\boldsymbol{\rho} : \mathcal{O} \to \mathbb{R}^n$, where contract i returns a payout of $\rho_i(X)$ and, thus, a contract bundle $\mathbf{r}$ pays $\boldsymbol{\rho}(X) \cdot \mathbf{r}$. As with the framework of Chen and Pennock, the contract prices are set according to a cost function C, so that a bundle $\mathbf{r}$ has a price of $C(\mathbf{s}+\mathbf{r}) - C(\mathbf{s})$. The design of the function C is addressed at length in Abernethy et al., to which we refer the reader.

For the remainder of this section we shall discuss the prediction market template of Abernethy et al. as it provides the most general model; we shall refer to such a market as an Automated Prediction Market Maker. We now precisely state the ingredients of this framework.

Definition 6. *An* Automated Prediction Market Maker (APMM) *is defined by a tuple* $(\mathcal{S}, \mathcal{O}, \boldsymbol{\rho}, C)$ *where* $\mathcal{S}$ *is the* share space *of the market, which we will assume to be the linear space* $\mathbb{R}^n$*;* $\mathcal{O}$ *is the set of outcomes;* $C : \mathcal{S} \to \mathbb{R}$ *is a smooth and convex cost function with* $\nabla C(\mathcal{S}) = \text{relint}(\nabla C(\mathcal{S}))$ *(here, we use* $\nabla C(\mathcal{S}) := \{\nabla C(\mathbf{s}) \mid \mathbf{s} \in \mathcal{S}\}$ *to denote the* derivative space *of* C*); and* $\boldsymbol{\rho} : \mathcal{O} \to \nabla C(\mathcal{S})$ *is a payoff function*[2].

Fortunately, we need not provide a full description of the procedure of the APMM mechanism: The APMM is precisely a special case of a CLM! Indeed, the APMM framework can be described as a CLM $(\mathcal{H}, \mathcal{O}, \texttt{Cost}, \texttt{Payout})$ where

$$\mathcal{H} = \mathcal{S}(= \mathbb{R}^n) \qquad \texttt{Cost}(\mathbf{s}, \mathbf{s}') = C(\mathbf{s}') - C(\mathbf{s}) \qquad \texttt{Payout}(\mathbf{s}, \mathbf{s}'; X) = \boldsymbol{\rho}(X) \cdot (\mathbf{s}' - \mathbf{s}). \quad (3)$$

Hence we can think of APMM prediction markets in terms of our learning mechanism. Markets of this form are an important special class of CLMs – in particular, we can guarantee that they are efficient to work with, as we show in the following proposition.

Proposition 3. *An APMM* $(\mathcal{S}, \mathcal{O}, \boldsymbol{\rho}, C)$ *with efficiently computable* C *satisfies EC and TT.*

We now ask, what is the learning problem that the participants of an APMM are trying to solve? More precisely, when we think of an APMM as a CLM, does it implement a particular L?

Theorem 1. *Given APMM* $A := (\mathcal{S}, \mathcal{O}, \boldsymbol{\rho}, C)$*, then* A *implements* $L : \nabla C(\mathcal{S}) \times \mathcal{O} \to \mathbb{R}$ *defined by*

$$L(\mathbf{w}; X) = D_{C^*}(\boldsymbol{\rho}(X), \mathbf{w}), \quad (4)$$

where C^* *is the conjugate dual of the function* C*.*

There is another more subtle benefit to APMMs – and, in fact, to most prediction market mechanisms in practice – which is that participants make bets via purchasing of shares or share bundles. When a trader makes a bet, she purchases a contract bundle $\mathbf{r}$, is charged $C(\mathbf{s}+\mathbf{r}) - C(\mathbf{s})$ (when the current quantity vector is $\mathbf{s}$), and shall receive payout $\boldsymbol{\rho}(X) \cdot \mathbf{r}$ if and when X is realized. But at any point before X is observed and trading is open, the trader can sell off this bundle, to the APMM or another trader, and hence neutralize her risk. In this sense bets made in an APMM are stateless, whereas for an arbitrary CLM this may not be the case: the wager defined by $(\mathbf{w}_t \mapsto \mathbf{w}_{t+1})$ can not necessarily be sold back to the mechanism, as the posted hypothesis may no longer remain at $\mathbf{w}_{t+1}$.

Given a learning problem defined by the GSR $L : \mathcal{H} \times \mathcal{O} \to \mathbb{R}$, it is natural to ask whether we can design a CLM which implements this L and has this "share-based property" of APMMs. More precisely, under what conditions is it possible to implement L with an APMM?

Theorem 2. *For any divergence-based GSR* $L(\mathbf{w}; X) = D_R(\boldsymbol{\rho}(X), \psi(\mathbf{w})) + f(X)$*, with* $\psi : \mathcal{H} \to \mathcal{H}'$ *one-to-one,* $\mathcal{H}' = \text{relint}(\mathcal{H}')$*, and* $\boldsymbol{\rho}(\mathcal{O}) \subseteq \psi(\mathcal{H})$*, there exists an APMM which implements* L*.*

We point out, as a corollary, that if an APMM implements some arbitrary L, then we must be able to write L as a divergence function. This fully specifies the class of problems solvable using APMMs.

[2]The conditions that $\boldsymbol{\rho}(\mathcal{O}) \subseteq \nabla C(\mathcal{S})$ and $\nabla C(\mathcal{S}) = \text{relint}(\nabla C(\mathcal{S}))$ are technical but important, and we do not address these details in the present extended abstract although they will be considered in the full version. More relevant discussion can also be found in Abernethy et al. [1].

Corollary 1. *If APMM $(\mathcal{S}, \mathcal{O}, \boldsymbol{\rho}, C)$ implements a GSR $L : \mathcal{H} \times \mathcal{O} \to \mathbb{R}$, then L is divergence-based.*

Theorem 1 establishes a strong connection between prediction markets and a natural class of GSRs. One interpretation of this result is that any GSR based on a Bregman divergence has a "dual" characterization as a share-based market, where participants buy and sell shares rather than directly altering the share prices (the hypothesis). This has many advantages for prediction markets, not least of which is that shares are often easier to think about than the underlying hypothesis space.

Our notion of a CLM offers another interpretation. In light of Proposition 3, any machine learning problem whose hypotheses can be evaluated in terms of a divergence leads to a tractable crowdsourcing mechanism, as was the case in Section 3. Moreover, this theorem does not preclude efficient yet non-divergence-based loss functions as we see in the next section.

5 Example CLMs for Typical Machine Learning Tasks

Regression. We now construct a CLM for a typical regression problem. We let $\mathcal{H}$ be the ℓ_2-norm ball of radius 1 in $\mathbb{R}^d$, and we shall let an outcome be a batch of a data, that is $X := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ where for each i we have $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in [-1, 1]$, and we assume $\|\mathbf{x}_i\|_2 \leq 1$. We construct a GSR according to the mean squared error, $L(\mathbf{w}; \{(\mathbf{x}_i, y_i)\}_{i=1}^n) = \frac{\alpha}{2n} \sum_{i=1}^n (\mathbf{w} \cdot \mathbf{x}_i - y_i)^2$ for some parameter $\alpha > 0$. It is worth noting that L is not divergence-based.

In order to satisfy the escrow property (ES), we can set $\texttt{Cost}(\mathbf{w}, \mathbf{w}') := 2\alpha\|\mathbf{w} - \mathbf{w}'\|_2$ because the function $L(\mathbf{w}; X)$ is 2α-lipschitz with respect to $\mathbf{w}$ for any X. To ensure that the CLM is L-incentivized, we must set $\texttt{Payout}(\mathbf{w}, \mathbf{w}'; X) := \texttt{Cost}(\mathbf{w}, \mathbf{w}') + L(\mathbf{w}; X) - L(\mathbf{w}'; X)$.

If we set the initial hypothesis $\mathbf{w}_0 = \mathbf{0}$, it is easy to check that WorstCaseLoss $= \alpha/2$. It remains to check whether this CLM is tractable. It's clear that we can efficiently compute `Cost` and `Payout`, hence the EC property holds. Given how `Cost` is defined, it is clear that the set $\{\mathbf{w}' : \texttt{Cost}(\mathbf{w}, \mathbf{w}') \leq B\}$ is just an ℓ_2-norm ball. Also, since L is convex in $\mathbf{w}$ for each X, so is the function $\mathbb{E}_{X \sim P}\big[\texttt{Profit}(\mathbf{w}, \mathbf{w}', X)\big]$ for every P. A budget-constrained profit-maximizing participant must simply solve a convex optimization problem, and hence the TT property holds.

Betting Directly on the Labels. Let us return our attention to the Netflix Prize model as discussed in the Introduction. For this style of competition a host releases a dataset for a given prediction task. The host then requests participants to provide predictions on a specified set of instances on which it has correct labels. For every submission the agent computes an error measure, say the MSE, and reports this to the participants. Of course, the correct labels are withheld throughout.

Our CLM framework is general enough to apply to this problem framework as well. Define $\mathcal{H} = \mathcal{O} = K^m$ where $K \subseteq \mathbb{R}$ bounded is the set of valid labels, and m is the number of requested test set predictions. For some $\mathbf{w} \in \mathcal{H}$ and $\mathbf{y} \in \mathcal{O}$, $\mathbf{w}(k)$ specifies the kth predicted label, and $\mathbf{y}(k)$ specifies the true label. A natural scoring function is the total squared loss, $L(\mathbf{w}; \mathbf{y}) := \sum_{k=1}^m (\mathbf{w}(k) - \mathbf{y}(k))^2$. Of course, this approach is quite different from the Netflix Prize model, in two key respects: (a) the participants have to wager on their predictions and (b) by participating in the mechanism they are required to *reveal their modification* to all of the other players. Hence while we have structured a competitive process the participants are de facto forced to collaborate on the solution.

A reasonable critique of this collaborative mechanism approach to a Netflix-style competition is that it does not provide the instant feedback of the "leaderboard" where individuals observe performance improvements in real time. However, we can adjust our mechanism to be online with a very simple modification of the CLM protocol, which we sketch here. Rather than make payouts in a large batch at the end, the competition designer could perform a mini-payout at the end of each of a sequence of time intervals. At each interval, the designer could select a (potentially random) subset S of user/movie pairs in the remaining test set, freeze updates on the predictions $\mathbf{w}(k)$ for all $k \in S$, and perform payouts to the participants on only these labels. What makes this possible, of course, is that the generalized scoring rule we chose decomposes as a sum over the individual labels.

Acknowledgments. We gratefully acknowledge the support of the NSF under award DMS-0830410, a Google University Research Award, and the National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

References

[1] J. Abernethy, Y. Chen, and J. Wortman Vaughan. An optimization-based framework for automated market-making. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, 2011.

[2] J. E. Berg, R. Forsythe, F. D. Nelson, and T. A. Rietz. Results from a dozen years of election futures markets research. In C. A. Plott and V. Smith, editors, *Handbook of Experimental Economic Results*. 2001.

[3] Y. Chen and D. M. Pennock. A utility framework for bounded-loss market makers. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.

[4] Y. Chen and J. Wortman Vaughan. A new understanding of prediction markets via no-regret learning. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, 2010.

[5] T.M. Cover, J.A. Thomas, J. Wiley, et al. *Elements of information theory*, volume 6. Wiley Online Library, 1991.

[6] T. Gneiting and A.E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[7] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):105–119, 2003.

[8] R. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15, 2007.

[9] R. Hanson, R. Oprea, and D. Porter. Information aggregation and manipulation in an experimental market. *Journal of Economic Behavior & Organization*, 60(4):449–459, 2006.

[10] Nathan Lay and Adrian Barbu. Supervised aggregation of classifiers using artificial prediction markets. In *ICML*, pages 591–598, 2010.

[11] J. Ledyard, R. Hanson, and T. Ishikida. An experimental test of combinatorial information markets. *Journal of Economic Behavior and Organization*, 69:182–189, 2009.

[12] J. Wolfers and E. Zitzewitz. Prediction markets. *Journal of Economic Perspective*, 18(2):107–126, 2004.