Solving Online Learning Through Matrix Decomposition

Xingyou Song

Abstract

Recently, work has been made on various online problems, many of which can be translated to online learning using convex matrix sets. However, one difficulty arising in solving these problems is the sheer size of the matrix convex set, if one naively applies ordinary regularization techniques. In this paper, we will show and explain the motivations and techniques to move around this difficulty, as well as the underlying background and interpretations.

Contents

1 Online Learning through Games

In this section we will introduce extensively studied matrix online learning problems. In online learning problems, given some set of observations, we are interested in being able to predict the next outcomes given future data. Here, our regret is from comparing our performance to the best fixed characteristics that are consistent with the current data. In many situations, finding the exact fixed characteristic is NP-HARD and intractable. However, the algorithms presented still do surprisingly well, only given local data.

Why Matrix Learning?

There have been a recent surge in works related to inputting combinatorial online games into a matrix format, and applying well known convex optimization methods on certain measures of entropy of matrices. These have been inspired by methods including SDP, quantum mechanics, and general matrix methods. Because many games are symmetric in terms of the structure, this means that many tools analogous to regular online learning can be used. For instance, the matrix exponential and log behave well with symmetric matrix inputs, due to certain spectral properties, and these relate to the singular values and eigenvalues of the matrix.

Before this, such games were difficult to format because of the high dimension and complexity of the choices available (e.g. $n!$ permutations for the action space of the learner in the online gambling problem, is discrete and is too combinatorial to classify properly and apply regularization in). Instead, relaxations were used by interpreting these permutations in terms of choices in a convex matrix set. However, previous works allowed the convex set to be too "generalized" in the sense that the convex sets were too relaxed, and regularization techniques did not give satisfying bounds. The online matrix prediction algorithm balances between these problems by giving a relatively simple description of the convex sets, without them being too general, by using clever matrix techniques.

1.1 Collaborative Filtering

This game is representative of the explicit matrix prediction problem. In this game, each turn, the adversary outputs a location pair in the matrix (i_t, j_t), and then the learner outputs a predictor value $\hat{y}_t \in [-1, 1]$ and the adversary outputs the exact value y_t. The loss is $\ell(\hat{y}_t, y_t) = |\hat{y}_t - y_t|$ and the regret is defined as

$$\text{Regret } = \sum_{t=1}^{T} \ell(\hat{y}_t, y_t) - \inf_{W \in \mathcal{W}} \sum_{t=1}^{T} \ell(W_{i_t, j_t}, y_t)$$

where the convex set $\mathcal{W}$ has bounded trace norm and $m \leq n$; i.e.

$$\mathcal{W} = \{W \in [-1, 1]^{m \times n} | \sum \sigma_i \leq \tau\}$$

where σ_i were the singular values in the SVD of W.

Unfortunately, using naive mirror descent will give the bound $O(\tau\sqrt{T})$, which is useless when in the practical setting $\tau = n$, since this implies $T = O(n^2)$ (i.e. we saw almost the entire matrix anyways).

Using the matrix multiplicative weights described in [HKS12], this will give the bound $O(\sqrt{\tau\sqrt{n}\log(n)T})$, which is close to the naive MW bound. The naive MW information theoretic bound is given by looking at experts which represent matrices with $\{-1, 1\}$ values with trace norm bounded above by τ and applying MW.

The technique by using randomized rounding proposed in [BS13] has a worse running time and slightly better regret bound, and is inspired by ERM and applying Rademacher random variables by choosing the values at the boundary, which gives a bound of $O(\tau\sqrt{n})$ However, this is not too generalizable to other games because the matrices representing the choices can have high trace norms.

In order to fit the format of the matrix algorithm, the comparison class is clearly $(\sqrt{m+n}, 2\tau)$-decomposable, which is defined later in [HKS12].

1.2 Gambling Problem

Consider the case when there exist n teams $1, 2, ..., n$. The learner wishes to predict accurately, given teams (i_t, j_t), the relaxed outcome $\hat{y}_t \in [-1, 1]$ of the game, where

$\hat{y}_t = 1$ implies that the learner absolutely believes that i_t will beat j_t, and $\hat{y}_t = -1$ implies the reverse. The non-absolute case will just mean, using randomized rounding, that the learner outputs 1 with probability $\frac{1+\hat{y}_t}{2}$ and -1 with the complement.
After the prediction, the true outcome $y_t \in \{-1, 1\}$ is shown, and suppose the loss function $\ell(\hat{y}_t, y_t) = (1 - \hat{y}_t y_t)/2$ (i.e. in absolute terms, 1 if wrong, 0 if correct).
Now suppose that after t rounds, we consider the set of all permutations $\pi \in \mathcal{F}, \pi : [n] \times [n] \to \{-1, 1\}$ such that they agree with the data shown; i.e. $\pi(i, j) = 1, \pi(j, k) = 1 \implies \pi(i, k) = 1$ and obviously $\pi(i, j) = -\pi(j, i)$ (transitive property, as well as the obvious zero-sum symmetric conclusion).
The goal is to minimize regret, which is defined to be

$$\text{Regret} = \sum_{t=1}^{T} \ell(\hat{y}_t, y_t) - \min_{\pi \in \mathcal{F}} \sum_{t=1}^{T} \ell(\pi(i_t, j_t), y_t)$$

However, note that this may be turned into the formulation in terms of matrices; i.e. for permutation π, we have that W the corresponding permutation $n \times n$ matrix, such that:

$$W_\pi(i, j) = \begin{cases} 1 \text{ if } \pi(i, j) = 1 \\ 0 \text{ if otherwise} \end{cases}$$

Then, instead, the formulation becomes (in order to be consistent with the matrix formulation)

$$\ell_{\mathcal{W}}(\hat{y}_t, y_t) = |y_t - \hat{y}_t|$$

and we wish to analyze the regret:

$$\text{Regret} = \sum_{t=1}^{T} \ell(\hat{y}_t, y_t) - \sum_{W_\pi \in \mathcal{W}} \sum_{t=1}^{T} \ell(W_\pi(i_t, j_t), y_t)$$

There is a basic but highly *inefficient* algorithm such that Regret $= O(\sqrt{Tn \log n})$; we just treat every $\pi \in \mathcal{F}$ as an expert, and run the standard multiplicative weights algorithm to get the bound of $O(\sqrt{T \log(n!)}) = O(\sqrt{Tn \log n})$. Since this treats every single permutation, $n!$ as an expert, and calculates the loss from each "expert", this is clearly an inefficient algorithm. This is due to insufficient use of the structure of the problem by interpreting each lone expert as a typical, independent expert. More importantly as well however, this is information theoretically the best bound as well. The adversary himself may use randomness in order to enforce a limit on how well the learner can perform, in expectation. The most simple case is uniform randomness, as seen here:

Theorem 1 (Regret Bounds for Sleeping Experts)**.** *There exists a set of moves* $(i_1, j_1), ..., (i_T, j_T) \in [n] \times [n]$ *for any learner-algorithm that achieves the regret bound of* $\Omega(Tn \log n)$

Proof: The proof of this is essentially finding a good ordering, using "mergesort", but only using two experts at a time. That is, split the time T into $Q(n)$ intervals. For each interval $t = 1, ..., Q(n)$, pick $\{x_t, y_t\}$, which will be two players, and only play the

experts game with these two players, over $T/Q(n)$ rounds. For each $T/Q(n)$ round, let the adversary uniformly at random choose either $(x_t, y_t) \in (1,0),(0,1)$. Denote $g^{(i)}(x_i)$ to be the accumulated losses outputted by x_i over the entire interval. Note that over $T/Q(n)$ steps, the expected gap between the accumulated losses of x_t and y_t will be:

$$\mathbb{E}|g^{(t)}(x_t) - g^{(t)}(y_t)| = \mathbb{E}|\sum_{i=1}^{T/Q(n)} \epsilon_i|$$

where ϵ_i are iid chosen from the uniform $1/2$ distribution with $\{-1, 1\}$; i.e. a Rademacher distribution. Noting that this is essentially the L_1 norm on the distribution, it is a general result that $L_i(x) = O(L_j(x))$ for all i, j (i.e. L_p norms are "between" each other, which forms the Khintchine inequality), which implies that

$$\mathbb{E}\left|\sum_{i=1}^{T/Q(n)} \epsilon_i\right| = O\left(\left(\sum_{i=1}^{T/Q(n)} 1\right)^2\right)^{1/2} = O\left(\sqrt{T/Q(n)}\right)$$

Furthermore, for any strategy that the learner picks, because we are uniformly outputting $\{0, 1\}$ values, the accumulated loss in expectation will be at least $T/(2Q(n))$. Therefore the regret bound for every interval will be at least

$$R = T/(2Q(n)) - \sqrt{T/Q(n)} = \Omega\left(\sqrt{T/Q(n)}\right)$$

The construction of the pair of experts for each interval will be essentially Mergesort; we let form a permutation of the $\{1, 2, ..., n\}$ experts that will rank them. The idea here is that the pairwise ranking, i.e. $x_t > y_t$ for an operator $>$, will be equivalent to seeing if x_t gave a higher loss than y_t, if we used the sub-game on $\{x_t, y_t\}$. Therefore, we apply Mergesort in order to get a consistent ranking, which will then imply that the time taken is $Q(n) = n \log n$. Then the total regret will be

$$Q(n) \cdot \Omega(\sqrt{T/Q(n)}) = \Omega(\sqrt{TQ(n)}) = \Omega(\sqrt{Tn \log n})$$

which is the bound we wanted.

The exploitation of the structure will lead to all such matrices in this game to be $(O(\log n), O(n \log n))$-decomposable, which will imply that the algorithm achieves a regret of $O(\sqrt{n \log^3(n) T})$.

1.3 Online MAX-CUT

In this problem, the input $(i_t, j_t) \in [n] \times [n]$ represents the vertices i_t, j_t. Given these two vertices, the learner predicts if the two vertices are supposed to be joined by an edge or not, by again, outputting $\hat{y}_t \in [-1, 1]$ The adversary outputs the true outcome, $y_t \in \{-1, 1\}$, with $y_t = 1$ meaning that (i_t, j_t) are joined by an edge, and $y_t = -1$ if not. Then here the loss will be

$$\ell(\hat{y}_t, y_t) = \frac{1}{2}|\hat{y}_t - y_t|$$

Now again, regret is formulated the same way.
We may reformulate this in terms of a matrix problem as well, where this cutting problem implies that we are taking a set of vertices $A \subseteq [n]$, and separating it from $[n] - A$, which implies our matrices are of the form:

$$W_A(i,j) = \begin{cases} 1 \text{ if } \text{ if } (i \in A, j \notin A), \text{ OR } (i \notin A, j \in A) \\ -1 \text{ if otherwise} \end{cases}$$

For this problem, the information theoretic approach shows that, ignoring structure, there is a minimal, but again, *highly inefficient and exponential time* algorithm that obtains $O(\sqrt{nT})$ bound. Like the above, it takes all possible 2^n cuts and treats them as experts; this will give us using MW, $O(\sqrt{T\log(2^n)}) = O(\sqrt{Tn})$. Similarly, this the minimum possible regret. But of course, this is extremely intractable given the exponential memory and time use.

A more efficient algorithm, discovered before the matrix-prediction algorithm, will obtain a $O(\sqrt{Tn^2})$ bound by just exploiting structure, and replacing all cuts with $\{-1,1\}^{n\times n}$, which will give n^2 edges to be predicted.
The matrix algorithm however, applied to this setting will achieve $O(\sqrt{Tn\log n})$, which is very close to the information theoretic-bound.
Using the decomposition, this set of matrices will be $(1,n)$-decomposable.

2 Matrix-Type Multiplicative Weights

In all of the problems above, the structure of the matrices are exploited by looking at specific attributes. Specifically, this is denoted (β,τ)-decomposability. Firstly, define the symmetric operator:

Definition 1. If W is $m \times n$ matrix, then we generate a new, symmetric matrix:

$$\text{sym}(W) := \begin{pmatrix} 0 & W \\ W^T & 0 \end{pmatrix}$$

Otherwise, if W is symmetric $n \times n$ already, then $\text{sym}(W) := W$.

Definition 2. A $m \times n$ matrix W is considered (β,τ)-decomposable if there exist symmetric, PSD matrices $P, N \in \mathbb{R}^{p\times p}$ (where $\text{sym}(W)$ is $p \times p$),

$$\text{sym}(W) = P - N$$

$$\text{Tr}(P) + \text{Tr}(N) \leq \tau$$

$$P(i,i), N(i,i) \leq \beta \quad \forall i \in [p]$$

Denote an entire set $\mathcal{W}$ of matrices to be (β,τ) decomposable if all of its elements are. A key insight to these series of transformations, is that firstly, symmetrization does not change the complexity of the convex set $\mathcal{W}$. Furthermore, the separation of the convex set into, essentially at most the difference between two convex sets in the sense of additive combinatorics. The trace is the most important element here, because shown in 3.3, it bounds the spectrum of the eigenvalues.

Now, for each problem, written in the method of Online Matrix Prediction, the problem can be rewritten into the general form using the Matrix Dot Product, which is essentially the trace of the products:

$$\mathrm{Tr}(AB) = A \bullet B = \sum_{i,j=1}^{n} A(i,j) \cdot B(i,j)$$

Then note that we can rewrite many games to be of the form:

$$\text{Regret} = \sum_{t=1}^{T} X_t \bullet L_t - \min_{X \in \mathcal{W}} \sum_{t=1}^{T} X \bullet L_t$$

where at each time step t, the learner chooses a $X_t \in \mathcal{W}$ and the adversary outputs the loss L_t.

One reason this is used is that the trace bounds the sum of the eigenvalues of a matrix. Furthermore, due to the spectral theorem, symmetric matrices can be decomposed into the eigenvalue-eigenvector spectrum, which makes it easy to analyze.

3 Follow Regularized Leader

In this section, we show the motivations for algorithms such as the Multiplicative weights algorithm, and certain variations using convex optimization. In the case where our regret is based on maximization of payoffs, the general framework lies in maximizing the expression containing both the payoff summed with a strongly concave function (called the regularizer). In cases where our regret is based on minimization of losses, we can just apply the negative of the payoff, and have a *convex* regularizer.
Here in our definition, for sake of clarity, we use payoff, but allow the parameter η to be negative for the loss case.

3.1 Regularization

Definition 3. If $\mathcal{W}$ is a convex set, then let $\mathcal{R} : \mathcal{W} \to \mathbb{R}$ be a regularizer function, that is strongly concave.
Strongly Concave here means, for any $\epsilon \in [0, 1]$,

$$\mathcal{R}(\epsilon W_1 + (1-\epsilon)w_2) \geq \epsilon \mathcal{R}(W_1) + (1-\epsilon)\mathcal{R}(W_2) + \epsilon(1-\epsilon)\gamma\delta^2$$

for some $\gamma = \gamma(\mathcal{W}, \mathcal{R}) > 0$, and where δ is the absolute difference between the payoffs of W_1 and W_2. (Note that we must have at least a quadratic term with respect to ϵ; it's easy to show that a linear term is impossible).

Now the expression to be maximized is

$$\Phi_t(W) = \eta \left(\sum_{i=1}^{t-1} \text{payoff}(W) \right) + \mathcal{R}(W)$$

In choosing the minimizer W_t^*, i.e.

$$W_t^* = \arg\min_{W \in \mathcal{W}} \Phi_t(W_t^*)$$

It suffices to use the Bregman Divergence, with respect to Φ, i.e.

$$D_\Phi(a,b) = \Phi(a) - (\Phi(b) + \nabla\Phi(b)\cdot(a-b))$$

Then if

$$\tilde{W}_t^* = \arg\min_{W\in\mathbb{R}^{n\times n}} \Phi_t(W)$$

then the actual constrained W_t^* will be the Bregman Projection of $\tilde{W}_t^*$, i.e.

$$W_t^* = \arg\min_{W\in\mathcal{W}} D_{\Phi_t}(W, \tilde{W}_t^*)$$

The motivation for efficient algorithms for regret minimization in these games then lies in trying to fit the problem into a convex set $\mathcal{W}$, and finding a good regularizer $\mathcal{R}$ so that $\gamma(\mathcal{W})$ is high, but the Bregman divergence value is usually low.

3.2 Original Multiplicative Weights

Definition 4. For inspiration, the *Multiplicative Weights* algorithm, given a n experts, time T game, and parameter η, consists of the player assigning weights uniformly at random, then updating the new weights according to the losses from the experts at the previous time. If η is negative, then we consider this as loss minimization, which is the standard way of interpretation.

$$w_{0,i} = \frac{1}{n}$$
$$w_{t+1,i} = w_{t,i}\exp(\eta\cdot\ell(t,i))$$

The learner then outputs the the probabilistic distribution of choices, according to the proportion $\propto w_{t,i}\ \ \forall i$ at time t for prediction.

Note that if we consider the potential function to decide the current probability distribution $\mathbf{p^t} = (p_1^t, ..., p_n^t)$,

$$\Psi(\mathbf{p^t}) = \eta\sum_{j=1}^{t-1}\sum_{i=1}^{n} p_i^t\ell(j,i) + \sum_{i=1}^{n} p_i^t \ln\left(\frac{1}{p_i^t}\right)$$

which is the sum of current gains, and the entropy. Intuitively, the learner wants to optimize both the cumulative gains, and the unpredictability of the adversary. The parameter η attempts to balance the two, for the optimum gain.

Using Lagrange multipliers, under the condition $\left(\sum_{i=1}^n p_i^t\right) - 1 = 0$, and looking at every component of the gradient,

$$\frac{\partial\Phi}{\partial p_i^t} = \lambda_t\frac{\partial g}{\partial p_i^t}$$

which implies that

$$\eta\left(\sum_{j=1}^{t-1}\ell(j,i)\right) - \ln(p_i^t) - 1 = \lambda_t,\ \ \forall i$$

solving for each p_i^t,

$$p_i^t = \exp\left(\eta\left(\sum_{j=1}^{t-1}\ell(j,i)\right) - 1 - \lambda_t\right)$$

Applying the same tactic using induction to get a similar equation for p_i^{t-1}, we have that recursively,

$$p_i^t = p_i^{t-1}\exp(\eta\ell(t-1,i))\exp(\lambda_{t-1} - \lambda_t)$$

where ultimately, $\exp(\lambda_{t-1} - \lambda_t)$ is the normalization factor to enforce that the p_i^t's sum to 1. Thus, the MW algorithm will attempt to maximize the potential Ψ.

3.3 Quantum Entropy on Matrices as a Regularizer

The difference here from a regular MW algorithm is that our convex set is not a simplex, but based on a matrix set $\mathcal{W} \subseteq \mathbb{R}^{n\times n}$. However, the regularizer here will be inspired by statistical mechanics, defined as the Von Neumann Entropy, similar to normal Shannon entropy.

Definition 5. Define the Von Neumann Entropy:

$$S(W) = -\mathrm{Tr}(W\log W)$$

where here, we also define the matrix exponential and log operators:

$$\exp(A) = \sum_{k=0}^{\infty}\frac{1}{k!}A^k = 1 + \frac{A}{1} + \frac{A^2}{2!} + ...$$

$$\log(A) = B, \text{ such that } \exp(B) = A$$

Sidenote: (Note that the log operator is unique only for A within a certain open neighborhood around the identity due to Lie Theory); the series

$$\log(1+A) = A - \frac{A^2}{2} + \frac{A^3}{3}...$$

only converges for an open neighborhood.

A connection here is that for symmetric matrices, due to the spectral theorem, a symmetric matrix W can be decomposed into its spectrum:

$$W = \sum_{i=1}^{n}\lambda_i u_i u^T$$

where the u_i's are mutually orthogonal and λ_i's are the eigenvalues. Then it is clear that

$$\exp(W) = \sum_{i=1}^{n}\exp(\lambda_i)u_i u_i^T$$

and therefore

$$\log(W) = \sum_{i=1}^{n}\log(\lambda_i)u_i u_i^T$$

(The $\log \lambda_i$ is well defined because all the eigenvalues of a symmetric matrix are non-negative). Since the trace of a matrix is the sum of the eigenvalues, i.e.

$$\mathrm{Tr}(W) = \sum_{i=1}^{n} \lambda_i$$

then it follows that the trace of exponentials and logs are easily related:

$$\mathrm{Tr}(\exp(W)) = \sum_{i=1}^{n} \exp(\lambda_i)$$

$$\mathrm{Tr}(\log(W)) = \sum_{i=1}^{n} \log(\lambda_i)$$

In a strong sense, applying the MW algorithm with matrix exponentials is applying MW algorithm on the eigenvalues "in parallel", which is why symmetrization and MW for matrices are so motivated. One can bound the eigenvalues due to the bounding on the trace, and can therefore bound the trace of the exponential or log as well.

Furthermore, the matrix convex sets can be interpreted in terms of the components on its eigenspaces which helps with structure.

Once formed, we can allow S to be the regularizer term; Then our Φ function will be:

$$\Phi_t(W) = -\eta \left(\sum_{t=1}^{t} \mathrm{Tr}(W \cdot L_t) \right) + S(W)$$

this implies that the Bregman Divergence will be (called the Quantum Relative Entropy, analogous to the KL Divergence):

$$\triangle(A, B) = \Phi(A) - (\Phi(B) + \nabla\Phi(B) \cdot (A - B)) = \mathrm{Tr}(A \log(A) - A \log(B) - A + B)$$

Using the normal regularizer technique, then we apply the Bergman Projection onto the convex comparison set $\mathcal{W}$. We want

$$X_{t+1} = \Pi^{\Phi}_{\mathcal{W}}(W) = \arg\min_{X \in \mathcal{W}} \triangle(X, W)$$

where

$$W^* = \arg\min_{W \in \mathbb{R}^{n \times n}} \Phi_t(W)$$

i.e. the closest element in our comparison set to the actual min, due to convex optimization. This works out to the analogous Matrix MW algorithm, because

Theorem 2.

$$\arg\min_{W \in \mathbb{R}^{n \times n}} \Phi_t(W) = \exp(\log(X_t) - \eta L_t)$$

Proof: The derivation here is essentially Lagrange Multipliers; i.e. take a transformation and let $\log(W) = Y \iff W = \exp(Y)$. Then

$$\Phi(W) = \Phi(\exp(Y)) = -\eta \cdot \mathrm{Tr}\left(\exp(Y) \cdot (\sum_{t=1}^{T} L_t)\right) - \mathrm{Tr}(\exp(Y) \cdot Y)$$

Taking the matrix derivative with respect to Y (which will ignore the trace) and factoring out $\exp(Y)$, we have that

$$0 = -\eta \sum_{t=1}^{T} L_t - Y + I$$

$$\iff Y = I - \eta(L_1 + ... + L_t)$$

Then inductively,

$$Y_{t+1} = Y_t - \eta L_t$$

Note that we still need to project using the Bergman Divergence, and therefore we make the approximation

$$W_{t+1} = \exp(\log(X_t) - \eta L_t)$$

To find the minimizer of the Bregman Divergence, i.e. finding

$$\arg\min_{X \in \mathcal{W}} \triangle(X, W)$$

It turns out that this is equivalent to a large SDP. The details are in [HKS12], but we provide a sketch: we may convert the convex set $\mathcal{W}$ into a sphere by normalizing every constraint and keep the transformed constraints, into a convex set $\mathcal{K}$, and allowing $\log X = \log W - Q$. Finding out the difference, Q, is by decomposing Y into the constraints, and applying semi-definite programming. (e.g. ellipsoid) method.

The runtime for each X_t is bounded by the log of the trace (which defines the size of the space to search in), and it is shown that due to the Golden Thompson inequality,

$$\mathrm{Tr}(\exp(\log X_t - \eta L_t)) \leq \mathrm{Tr}(X_t \exp(-\eta L_t)) \leq 3\mathrm{Tr}(X_t)$$

Since the trace grows by at most a factor of 3, the runtime is not affected in the asymptotic case.
Then the MW algorithm turns out to be:

Definition 6. Given a constant η, we recursively define X_t as follows:

$$X_1 = \frac{\tau}{n} I$$

after receiving the loss matrix L_t,

$$X_{t+1} = \Pi^{\Phi}_{\mathcal{W}}(\exp(\log(X_t) - \eta L_t)) = \arg\min_{X \in \mathcal{W}} \triangle(X, \exp(\log(X_t) - \eta L_t))$$

From normal regularization theory, this will incur regret bound using linear losses,

$$\text{Regret} \leq \eta \sum_{t=1}^{T} \text{Tr}(X_t \cdot L_t^2) + \frac{\triangle(X, X_1) - \triangle(X, X_{T+1})}{\eta} \leq \eta \sum_{t=1}^{T} \text{Tr}(X_t \cdot L_t^2) + \frac{\tau \log n}{\eta}$$

where n is the size of the matrix and assumption $\text{Tr}(X) \leq \tau$.

Applying this algorithm for when we have the $(\beta, \tau)-$ decomposition, where $\text{sym}(W) = P - N$, we generate the p.s.d. matrix

$$\phi(W) = \begin{bmatrix} P & 0 \\ 0 & N \end{bmatrix}$$

and apply the matrix MW algorithm over all possible $\phi(W)$, with a correct η minimizer. This will end with a regret bound of

$$\text{Regret} \leq O(\sqrt{\tau \beta \log(2p) T})$$

which is much smaller than the result of previous techniques.

3.4 Log-Determinant Regularizer

In [C14], a different regularizer is used, for a specific class of problems, i.e. $\mathcal{R}(W) = -\log\det(kW + I) \quad \forall \ W \in \mathcal{W}$. A similarity between this regularizer and the von-Neumann regularizer is its intention on the eigenvalues of W, i.e. if W has the eigen-decomposition composed of $\lambda_1, ..., \lambda_n$, then

$$\mathcal{R}(W) = -\sum_{i=1}^{n} (k \cdot \lambda_i + 1)$$

Note that the Von-Neumann Entropy will give

$$S(W) = -\sum_{i=1}^{n} \lambda_i \log \lambda_i$$

For a shifted convex space according to the parameter k, this is very close to the Von-Neumann Entropy by a constant.

It introduces the notion of the SDP $\rightarrow$ LP relaxation, in which quadratic constraints, i.e. constraints involving variables $x_i x_j$ are relaxed into LP terms by using $A_{ij} = \mathbb{E}(x_i x_j)$. This is inspired by a long line of work related to SDP lower bounds, termed the "Sherali Adams" hierarchy, that can sometimes be efficient and effective approximations. The result achieves a small boost in the regret for the gambling problem, cutting off a log factor.

4 Conclusion

This new line of work motivates the question of "convexifying" a set of points properly. The common regularization techniques will give better bounds if we have as shown,

(β, τ)-decomposability is extremely powerful because it conveys points in terms of a convex set, but it is optimized for the tradeoff between speed (based on how simple the convex sets can be displayed) and compactness (how well the convex set specifies the points). The very best algorithm in polynomial time to solve the Bregman divergence is due to SDP, and thus our convex sets must not have too many constraints (SDP's run-speed is due to number of constraints and dimension).

This in fact, is a very real issue that many online learning problems face. Hopefully, techniques in computational and combinatorial geometry, as well as approximation algorithms can aid in this. A generalization to this approach is for a symmetric matrix W (or essentially symmetric after the symmetrization operator), is if it can decomposed even more than the original (β, τ) decomposability:

$$W = \sum_{i=1}^{K} \alpha_i P_i$$

where there are constraints on the trace of the positive symmetric semi-definite P_i's. Using the P_i's as block diagonals in a new matrix would allow faster implementation, due to more parallelism and possibly better decomposition constants, and would be an even greater improvement. This area of mathematical research however is not yet developed.

References

[H09] Elad Hazan A Survey: The Convex Optimization Approach to Regret Minimization. 2009

[S11] Shai Shalev-Shwartz Online Learning and Online Convex Optimization 2011

[HKS12] Elad Hazan, Satyen Kale, Shai Shalev-Shwartz Near-Optimal Algorithms for Online Matrix Prediction 2012

[H09] Elad Hazan, The Convex Optimization Approach to Regret Minimization 2009

[A12] Jacob Abernethy. Can We Learn to Gamble Efficiently? 2012

[KS12] Varun Kanade, Thomas Steinke. Learning Hurdles for Sleeping Experts. 2012

[C14] Paul Christiano Online Local Learning via Semidefinite Programming 2014

[KMS10] Robert Kleinberg, Alexandru Mizil, Yogeshwer Sharma. Regret Bounds for Sleeping Experts and Bandits. 2010

[HW98] Mark Herbster, Manfred Warmuth Tracking the Best Expert. 1998

[BS13] Nicolo Cesa-Bianchi, Ohad Shamir Efficient Online Learning via Randomized Rounding 2013

[AK07] Sanjeev Arora, Satyen Kale, A Combinatorial, Primal-Dual Approach to Semidefinite Programs 2007.

[KST 12] Sham Kakade, Shal Shalev-Shwartz, Ambuj Tewari Regularization Techniques for Learning with Matrices 2012.