**Lecture 2: Multiplicative weights**
CSE 599S: Entropy optimality, Winter 2016
**Instructor:** James R. Lee **Last updated:** January 23, 2016

# 1 An iterative algorithm

In the last lecture, we considered the *matrix scaling problem:* Given non-negative matrices $X, T \in \mathbb{R}_+^{n \times n}$, our goal was to find non-negative diagonal matrices $D_1, D_2$ so that $D_1 X D_2$ had the same row and column sums as the target matrix $T$. In other words, we sought to weight the rows and columns of $X$ by positive numbers in order to achieve this. We used entropy optimization to prove the following theorem.

**Theorem 1.1.** *Give $X = (x_{ij})$ and $T = (t_{ij})$, if it holds that $x_{ij} = 0 \iff t_{ij} = 0$ for all $i, j \in [n]$, then such a weighting exists.*

Our previous method showed the existence of such a weighting. Now we will study the structure of the optimization in order to find a simple algorithm to (approximately) find it. For simplicity, let us consider the case when $t_{ij} = 1/n^2$ for all $i, j \in [n]$. We may assume (by a global scaling) that $\sum_{i,j} x_{ij} = 1$.

We will start with the initial solution $Y^{(0)} = X$. And recall the constraints we would like to hold for our final matrix $Y = (y_{ij})$:

1. For all $i \in [n]$, $\sum_{j=1}^n y_{ij} = \frac{1}{n}$.
2. For all $j \in [n]$, $\sum_{i=1}^n y_{ij} = \frac{1}{n}$.

Of course, if $Y^{(0)}$ already satisfies these constraints, we can pack up and go home, but we are likely not so fortunate.

A natural approach is to consider a single violated constraint: A row or column with the wrong sum. We can fix it (by rescaling it to have sum $1/n$), and then iterate. The only subtle issue is whether this actually converges; when fixing one row, for instance, we could mess up all the column sums we already had correct! We will prove the following.

Let $X = (x_{ij})$ be a matrix with all positive entries. Define

$$\mathcal{E}(X) = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(1 - \sum_{j=1}^n X_{ij}\right)^2}.$$

This is the mean squared error of the row sums. We will think of $\bar{X} = X / \sum_{i,j} x_{ij}$ and $T$ as distributions on $[n] \times [n]$. (The following bound on the rate of convergence is decent, but certainly not the best possible.)

**Theorem 1.2** (Knopp-Sinkhorn scaling algorithm)**.** *Given a matrix $X = (x_{ij})$ and an accuracy parameter $0 < \delta < 1$, the algorithm which iteratively normalizes all rows and columns repeatedly yields a matrix $X'$ all of whose column sums are one and such that $\mathcal{E}(X') \leqslant \delta$. The number of iterations required is at most*

$$\frac{6}{\delta^2} D(T \,\|\, \bar{X}) \leqslant \frac{6}{\delta^2} \log\left(\frac{\max(X)}{\min(X)}\right)$$

*where* $\min(X)$ *and* $\max(X)$ *are the smallest and largest entries in* $X$.

To analyze a measure of progress for such algorithms, let's think slightly more generally.

## 1.1 An algorithm in continuous time

We will design an algorithm that runs in continuous time. Let us think about a matrix $Y \in \mathbb{R}^{n \times n}$ as a function $Y : [n] \times [n] \to \mathbb{R}$. For $i, j \in [n]$, define the functions $\{R_i, C_j : [n] \times [n] \to \mathbb{R}\}$ by

$$R_i(a,b) = \begin{cases} 1 & a = i \\ 0 & \text{otherwise.} \end{cases}$$

$$C_j(a,b) = \begin{cases} 1 & b = j \\ 0 & \text{otherwise.} \end{cases}$$

As matrices, $R_i$ is the $\{0,1\}$ matrix with 1's on the $i$th row, and $C_j$ is the $\{0,1\}$ matrix with 1's on the $j$th column.

For each time $s \geqslant 0$, let $\psi_s$ be one of $\{R_i, C_j, -R_i, -C_j : i, j \in [n]\}$. These correspond to the $4n$ fundamental directions in which we can move: Either scaling up or down a row or column. We define the function

$$Y^{(s)}(i,j) = \exp\left(\int_0^s \psi_\tau(i,j)\, d\tau\right) X(i,j)\,.$$

The important point here is that $Y^{(s)}$ is obtained by multiplying the rows and columns of $X$ by non-negative weights. For instance, when $\psi_s = R_i$, we are (infinitesimally) multiplying the $i$th row of our current matrix by $e^{d\tau}$ (scaling it up). When $\psi_s = -C_j$, we are multiplying the $j$th column by $e^{-d\tau}$ (scaling it down).

For the sake of analysis, we want our matrices to remain on the probability simplex, so let us also define the scaled matrices:

$$\bar{Y}^{(s)}(i,j) = \frac{Y^{(s)}}{\sum_{i,j} Y^{(s)}(i,j)}\,.$$

**The potential function.** We will analyze convergence by showing that as long as we are fixing violated constraints, there is a potential function measuring our progress. Consider:

$$\Phi(s) = D(T \,\|\, \bar{Y}^{(s)})\,.$$

Observe that $\Phi(0) = D(T \,\|\, X) < \infty$ and $\Phi(s) \geqslant 0$ for all $s \geqslant 0$ because, as we saw in the last lecture, relative entropy is always non-negative.

**Lemma 1.3.** *It holds that*

$$\frac{d}{ds}\Phi(s) = \sum_{i,j} \psi_s(i,j)\left(\bar{Y}^{(s)}(i,j) - \frac{1}{n^2}\right)\,.$$

Before proving this lemma, let's try to understand its content. Recall that $\psi_s \in \{R_i, -R_i, C_j, -C_j\}$, so the right-hand side is precisely the violation of $\bar{Y}^{(s)}$ on some row or column constraint (whichever one we have chosen to focus on at time $s$).

For instance, if for some $i \in [n]$, we have

$$\sum_j \bar{Y}^{(s)}(i,j) = \frac{1}{n} - \varepsilon, \qquad \varepsilon > 0$$

so that the $i$th row sum of $\bar{Y}$ is too small, then by choosing $\psi_s = R_i$, we will increase the $i$th row sum, and correspondingly the derivative of the potential function will be $-\varepsilon$.

*Proof.* Let us calculate

$$\begin{aligned}\frac{d}{ds}\Phi(s) &= \frac{d}{ds}\sum_{i,j}\frac{1}{n^2}\log\frac{1/n^2}{\bar{Y}^{(s)}(i,j)} \\ &= -\frac{1}{n^2}\sum_{i,j}\frac{d}{ds}\log\bar{Y}^{(s)}(i,j) \\ &= -\frac{1}{n^2}\sum_{i,j}\frac{d}{ds}\log\frac{Y^{(s)}(i,j)}{\sum_{i,j}Y^{(s)}(i,j)}\,.\end{aligned}$$

First, we compute

$$\frac{d}{ds}\log Y^{(s)}(i,j) = \psi_s(i,j)\,.$$

And then:

$$\frac{d}{ds}\log\left(\sum_{i,j}Y^{(s)}(i,j)\right) = \frac{\sum_{i,j}Y^{(s)}(i,j)\psi_s(i,j)}{\sum_{i,j}Y^{(s)}(i,j)} = \sum_{i,j}\bar{Y}^{(s)}(i,j)\psi_s(i,j)\,.$$

We conclude that

$$\begin{aligned}\frac{d}{ds}\Phi(s) &= \frac{1}{n^2}\sum_{i,j}\left(\sum_{i,j}\bar{Y}^{(s)}(i,j)\psi_s(i,j) - \psi_s(i,j)\right) \\ &= \sum_{i,j}\bar{Y}^{(s)}(i,j)\psi_s(i,j) - \frac{1}{n^2}\sum_{i,j}\psi_s(i,j) \\ &= \sum_{i,j}\psi_s(i,j)\left(\bar{Y}^{(s)}(i,j) - \frac{1}{n^2}\right)\,.\end{aligned}$$

□

**Fixing a violated constraint.** In the preceding algorithm, we can focus on any constraint at any time $s$. A natural thing to do is find a violated constraint, and then improve it until it becomes satisfied. Let us suppose we have

$$\sum_j \bar{Y}^{(s)}(i,j) = \frac{1}{n} - \varepsilon, \qquad \varepsilon \in \left(0, \frac{1}{n}\right)$$

for some $i \in [n]$.

In order to see how long we spend fixing this constraint, we need to compute the change in the violation when $\psi_s = R_i$, i.e.

$$\frac{d}{ds}\sum_k \bar{Y}^{(s)}(i,k) = \sum_k \frac{d}{ds}\frac{Y^{(s)}(i,k)}{\sum_{i,j}Y^{(s)}(i,j)}\,.$$

We have

$$\frac{d}{ds}\frac{Y^{(s)}(i,k)}{\sum_{i,j} Y^{(s)}(i,j)} = \bar{Y}^{(s)}(i,k) - \frac{\sum_j \bar{Y}^{(s)}(i,j)}{\sum_{i,j} Y^{(s)}(i,j)} \leqslant \bar{Y}^{(s)}(i,k)$$

so that

$$\frac{d}{ds}\sum_k \bar{Y}^{(s)}(i,k) \leqslant \sum_k \bar{Y}^{(s)}(i,k)\,.$$

By a simple application of Grönwall's inequality, we conclude that at time $s' \geqslant s$,

$$\sum_k \bar{Y}^{(s')}(i,k) \leqslant e^{s'-s}\sum_k \bar{Y}^{(s)}(i,k)\,.$$

In particular, we spend time at least

$$\log\left(\frac{1}{1-\varepsilon n}\right) \geqslant \varepsilon n\,.$$

Similarly, if

$$\sum_j \bar{Y}^{(s)}(i,j) = \frac{1}{n} + \varepsilon, \qquad \varepsilon \in \left(0, \frac{1}{n}\right),$$

then the time spent is

$$\log(1+\varepsilon n) \geqslant \frac{\varepsilon n}{2}\,.$$

Recall that the potential function is initially decreasing at the rate $-\varepsilon$. And this happens for time $\approx \varepsilon n$. Thus the total decrease in fixing such a constraint is at least $c\varepsilon^2 n$ for some constant $c > 0$. (One can do the integral carefully and conclude that $c \geqslant 1/6$.) We conclude that we can fix at most

$$6\frac{D(T \,\|\, X)}{\varepsilon^2 n}$$

$\varepsilon$-violated constraints before they are all satisfied within $\varepsilon$.

If we are scaling to a doubly-stochastic matrix, it is more natural to require the row and column sums to lie in the interval $[1-\delta, 1+\delta]$, and we can achieve this by setting $\varepsilon = \delta/n$ so that the number of iterations is $\frac{6n}{\delta^2}D(T \,\|\, X)$. This does not immediately yield Theorem 1.2 because we have globally scaled the matrix at every step to maintain the constraint that $\sum_{ij} \bar{Y}_{ij}^{(s)} = 1$.

## 1.2 Analyzing the Knopp-Sinkhorn algorithm

In fact, the algorithm discussed in Theorem 1.2 has slightly nicer properties than ours. It achieves the rescaling $\sum_{ij} \bar{Y}_{ij}^{(s)} = 1$ implicitly because after all the row sums are normalized to $1/n$, the matrix automatically satisfies this global scaling constraint. Since the row (and column) updates are to independent coordinates, the potential function merely adds, and we see that the total potential change is at least

$$\frac{n}{6}\sum_{i=1}^n \varepsilon_i^2\,,$$

where $\varepsilon_i$ is the violation of the $i$th row. Thus if we seek to, say, get all the column sums to be exactly one and have all the row sums $r_1, \ldots, r_n$ satisfy

$$\sqrt{\frac{1}{n}\sum_{i=1}^n (r_i - 1)^2} \leqslant \delta\,,$$

then the total number of iterations required is at most

$$6\frac{D(T \,\|\, X)}{\delta^2}.$$

On the other hand, our analysis has the advantage that we didn't need to use "independence" of the rows (respectively, the columns). This will be a big advantage in the next lecture when we apply it to a problem in discrete Fourier analysis. The quadratic dependence on the error parameter is a hallmark of this type of algorithm; it's important to recall where it comes from: The potential change per step is proportional to $\varepsilon^2$ where $\varepsilon$ is the violation. This is because the potential function is decreasing by $\approx \varepsilon$ for an amount of time which is also $\approx \varepsilon$.

### 1.3 An application to detecting perfect matchings in bipartite graphs

Here is a cute application of the Knopp-Sinkhorn (KS) scaling algorithm (due to [Linial-Samorodnitsky-Wigderson, 2000]). Let $G = (U, V, E)$ be a bipartite graph with vertex set $U \cup V$ and edges $E \subseteq U \times V$. Assume that $|U| = |V|$. A *perfect matching* is a subset $M \subseteq E$ with $|M| = |U| = |V|$ such that every vertex of $G$ touches exactly one edge of $M$.

Let $A$ be the adjacency matrix of $G$. Do $\approx n^4 \log n$ iterations of the KS scaling algorithm (alternating between rescaling of rows and columns) such that all the column sums are equal to one. If all the row sums are in the interval $(1 - \frac{1}{n}, 1 + \frac{1}{n})$, output YES. Otherwise, output NO.

**Exercise 1.4** (Not for credit—done in class)**.** Show that this algorithm is always correct, outputting YES if and only if $G$ has a perfect matching. Note that the $n^4$ bound can be improved at least to $n^2$ by being more careful in the analysis.

## 2 Convergence rates

[This section (which will not be presented in lecture) is still unwritten; there are two exercises in Section 2.2.]

Dependencies like those in Theorem 1.2 on the bit complexity of the input matrix $X$ and the error parameter $\delta > 0$ will become important for some structural results we encounter soon (in the next lecture, in fact). So it is interesting that, in the setting of matrix scaling, one can do significantly better using a more sophisticated algorithm. This material comes from the papers [Linial-Samorodnitsky-Wigderson, 2000] and [Franklin-Lorenz, 1989].

### 2.1 Franklin-Lorenz convergence analysis

### 2.2 Bit complexity of $X$

**Exercise 2.1.** Consider the matrix

$$X = \begin{pmatrix} 1/2 & \varepsilon & \varepsilon \\ 1/2 & \varepsilon & \varepsilon \\ 0 & 1-2\varepsilon & 1-2\varepsilon \end{pmatrix}$$

Prove that it will take $\Omega(\log \frac{1}{\varepsilon})$ rounds of scaling before $A$ is close to doubly-stochastic. Therefore the dependence of our bound on the entries of $X$ is necessary in Theorem 1.2.

**Expository 2.2.** Give an exposition of the algorithm and analysis of LSW. They use the *permanent* of a matrix as a potential function measuring progress of the iterative rescaling algorithm. They are able to remove the dependence on the bit complexity of $X$ by first preprocessing $X \mapsto X'$ so that $\mathrm{per}(X') \geqslant n^{-n}$.

### 2.3 Dependence on the error