Learning Permutations with Exponential Weights

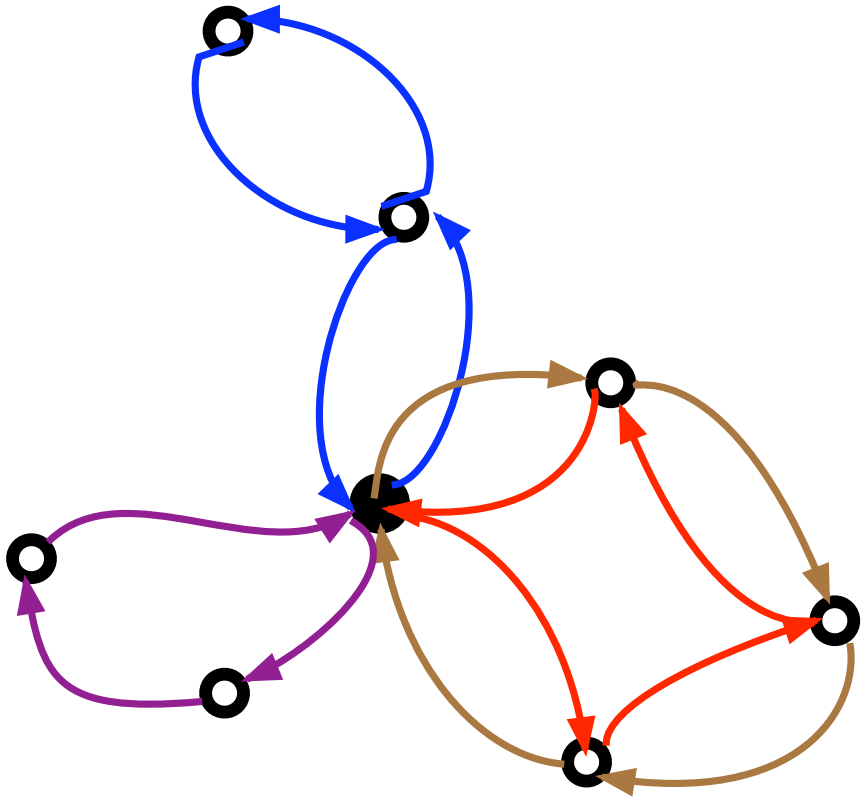
David P. Helmbold Manfred K. Warmuth

University of California - Santa Cruz

Last update - June 18, 2007

Commuter Airline Example

Motivating Problem: Match planes to routes



Commuter Airline Example

- n airplanes fly n daily routes
- Must assign different aircraft to each route
- Aircraft have various sizes, don't (yet) know # of passengers
- Too small/big aircraft can be bad
- At end of day, get the loss (regret) for each aircraft assignment
- Goal: have loss over many days close to best fixed assignment in hindsight

On-line assignment problem – want to pick good assignments (or permutations)

Outline

Outline

Learning Model

On line learning of permututions on n elements
on each trial:

- Algorithm chooses distribution D over permutations ($\boldsymbol{\Pi}$'s)
- "Nature" picks any loss matrix $\mathbf{L}$ in $[0,1]^{n\times n}$;
  $L_{i,j}$ is loss of assigning element i to position j.
- Loss of perm. $\boldsymbol{\Pi}$ is $\sum_{i=1}^{n} \mathbf{L}_{i,\boldsymbol{\Pi}(i)} = \boldsymbol{\Pi} \bullet \mathbf{L}$ (matrix dot product)
- Algorithm incurs expected loss $\mathrm{E}_{\boldsymbol{\Pi}\sim D}[\boldsymbol{\Pi} \bullet \mathbf{L}]$

Goal: expected loss not much more than best permutation in hindsight

Special Case: number of incorrect assignments

Nature picks permutation; loss is number of items assigned to wrong position. If nature picks permutation *abcd*:

Loss matrix $\mathbf{L}$ is

$$\begin{array}{c} a \\ b \\ c \\ d \end{array} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

For $\boldsymbol{\Pi} = bacd = \begin{pmatrix} 0 & \underline{1} & 0 & 0 \\ \underline{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

the loss is 2.

Nature can permute the rows to pick different permutations.

Loss matrix for pick *bcad* is:

$$\begin{array}{c} a \\ b \\ c \\ d \end{array} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Some other special cases

- Ranking: put best two items in top three positions (best two change each trial)

$$\mathbf{L} = \begin{array}{c}\text{some row}\\ \text{permutation}\\ \text{of}\end{array} \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Note:
any permutation
has loss 1, 3 or 5

- Permutation reflects list order, minimize number of links to find the desired element:

$$\mathbf{L} = \begin{array}{c}\text{some row}\\ \text{permutation}\\ \text{of}\end{array} \frac{1}{5} \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

better algorithm
for this case in
Blum-Chawla-Kalai03

Note that:

- Loss of permutation decomposed into sum of per-assignment losses
- Entries of $\mathbf{L}$ in $[0, 1]$, per trial losses in $[0, n]$.
- Loss matrix provides important structure allowing reasonably efficient algorithm with $\mathcal{L}_{\text{bestP}} + \sqrt{2\mathcal{L}_{\text{est}} n \ln n} + n \ln n$ loss bounds
- Standard Hedge (or WMR, Experts) too expensive - $n!$ permutations (and gets worse bound)
- Could use FPL* (Kalai-Vempala '05), faster but worse bounds

($n \ln n$ terms become $n^3 \ln n$)

We will:

- Use exponential weights with difficult normalization
- Weight of permutation proportional to product of assignment weights – keep n^2 rather than $n!$ weights.
- Bound with relative entropy analysis and Bregman projection techniques
- Get "bounds" for un-normalized version of WMR/Hedge

Outline

Distributions on Permutations and Weight Matrix

- For any D over permutations, define $n \times n$ weight matrix **W** where $W_{i,j} = \Pr(\text{item } i \text{ gets mapped to position } j)$
- **W** is doubly stochastic (each row and col sums to 1)
- Expected loss on trial is $\mathbf{L} \bullet \mathbf{W}$
- Many D's lead to same **W**; and can expand any $n \times n$ doub.stoch. **W** into a D using few ($\sim n^2$) permutations. Example:

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$= \frac{1}{6} \sum_{3\times 3 \text{ perms } \boldsymbol{\Pi}} \boldsymbol{\Pi}$$

A rose by any other name ...

- Permutation Learning: PermLearn

A rose by any other name ...

- Permutation Learning: PermLearn
- Permutation Learning with Exponential Weights: PermLearnEW

A rose by any other name ...

- Permutation Learning: PermLearn
- Permutation Learning with Exponential Weights: PermLearnEW
- Permutation Learning with Exponential-Weights: PermLearnE

A rose by any other name ...

- Permutation Learning: PermLearn
- Permutation Learning with Exponential Weights: PermLearnEW
- Permutation Learning with Exponential-Weights: PermLearnE
- Permutation Learning with Exponential-Weights: PermELearn

Algorithm PermELearn

Keeps $n \times n$ doubly stochastic weight matrix $\mathbf{W}$: initially each entry $1/n$

Need to predict and update each trial.

Predict step:

- Decompose $\mathbf{W}$ into D on few permutations
- Randomly pick $\widehat{\Pi}$ according to D

Update step:

- Loss update: $W'_{i,j} := W_{i,j}\, e^{-\eta \mathbf{L}_{i,j}}$ (η is learning rate)
- "Normalize" $\mathbf{W}'$ back into doubly stochastic $\mathbf{W}''$.

Making a matrix doubly stochastic

- Use Sinkhorn Balancing (Sinkhorn '64), also called matrix scaling
- finds row and column factors (r_i's and c_j's) such that $W''_{i,j} := \frac{W'_{i,j}}{r_i c_j}$ is doubly stochastic.
- Many iterative algorithms, pretty combinatorics, connection to permanent (Lineal-Samorodnitsky-Wigderson '00)
- but: no closed form for r_i, c_j!

Matrix Scaling Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{matrix} 2 \\ 3 \end{matrix}$$

Matrix Scaling Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \quad \begin{matrix} 2 \\ 3 \end{matrix} \qquad \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \quad \begin{matrix} \frac{5}{6} & \frac{7}{6} \end{matrix}$$

Matrix Scaling Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{matrix} 2 \\ 3 \end{matrix} \qquad \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \begin{matrix} \\ \\ \frac{5}{6} \end{matrix} \quad \begin{matrix} \\ \\ \frac{7}{6} \end{matrix} \qquad \begin{pmatrix} \frac{3}{5} & \frac{3}{7} \\ \frac{2}{5} & \frac{4}{7} \end{pmatrix} \begin{matrix} \frac{36}{35} \\ \frac{34}{35} \end{matrix} \quad \ldots$$

Matrix Scaling Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \quad \begin{matrix} 2 \\ 3 \end{matrix} \qquad \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \quad \begin{matrix} \\ \frac{5}{6} \quad \frac{7}{6} \end{matrix} \qquad \begin{pmatrix} \frac{3}{5} & \frac{3}{7} \\ \frac{2}{5} & \frac{4}{7} \end{pmatrix} \quad \begin{matrix} \frac{36}{35} \\ \frac{34}{35} \end{matrix} \quad \cdots \quad \begin{pmatrix} \frac{\sqrt{2}}{1+\sqrt{2}} & \frac{1}{1+\sqrt{2}} \\ \frac{1}{1+\sqrt{2}} & \frac{\sqrt{2}}{1+\sqrt{2}} \end{pmatrix}$$

Matrix Scaling Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \quad \begin{matrix} 2 \\ 3 \end{matrix} \qquad \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \quad \begin{matrix} 1 \\ 1 \end{matrix} \qquad \begin{pmatrix} \frac{3}{5} & \frac{3}{7} \\ \frac{2}{5} & \frac{4}{7} \end{pmatrix} \quad \begin{matrix} \frac{36}{35} \\ \frac{34}{35} \end{matrix} \quad \cdots \quad \begin{pmatrix} \frac{\sqrt{2}}{1+\sqrt{2}} & \frac{1}{1+\sqrt{2}} \\ \frac{1}{1+\sqrt{2}} & \frac{\sqrt{2}}{1+\sqrt{2}} \end{pmatrix}$$

Why $\sqrt{2}$?

- "heft" of permutation is product of its assignment weights
- re-scaling rows and/or cols preserves heft ratios
- $\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$ scales to $\begin{pmatrix} a & b \\ b & a \end{pmatrix}$ where $a + b = 1$ and $a^2 = 2b^2$.

Outline

Method (Kivinen-Warmuth '97): for every trial and every doubly stochastic comparator $\mathbf{U}$ show the key invariant

$$\Delta(\mathbf{U},\mathbf{W}) - \Delta(\mathbf{U},\mathbf{W}'') \geq (1-e^{\eta})(\overbrace{\mathbf{W}\bullet\mathbf{L}}^{\text{Alg's loss}}) - \eta(\overbrace{\mathbf{U}\bullet\mathbf{L}}^{\mathbf{U}\text{'s loss}})$$

where relative entropy $\Delta(\mathbf{U},\mathbf{W}) = \sum_{i,j} U_{i,j} \ln \frac{U_{i,j}}{W_{i,j}} + W_{i,j} - U_{i,j}$,

Method (Kivinen-Warmuth '97): for every trial and every doubly stochastic comparator $\mathbf{U}$ show the key invariant

$$\Delta(\mathbf{U},\mathbf{W}) - \Delta(\mathbf{U},\mathbf{W}'') \geq (1-e^{\eta})(\overbrace{\mathbf{W}\bullet\mathbf{L}}^{\text{Alg's loss}}) - \eta(\overbrace{\mathbf{U}\bullet\mathbf{L}}^{\mathbf{U}\text{'s loss}})$$

where relative entropy $\Delta(\mathbf{U},\mathbf{W}) = \sum_{i,j} U_{i,j} \ln \frac{U_{i,j}}{W_{i,j}} + W_{i,j} - U_{i,j}$,

Summing over trials gives

$$\Delta(\mathbf{U},\mathbf{W}_{\text{init}}) - \Delta(\mathbf{U},\mathbf{W}_{\text{fin}}) \geq (1-e^{\eta})(\text{Alg's tot loss}) - \eta \sum_t (\mathbf{U}\bullet\mathbf{L}_t)$$

$$\text{Alg's tot loss} \leq \frac{\Delta(\mathbf{U},\mathbf{W}_{\text{init}}) + \eta\sum_t(\mathbf{U}\bullet\mathbf{L}_t)}{1-e^{-\eta}} = \frac{n\ln n + \eta\mathcal{L}_{\text{bestP}}}{1-e^{-\eta}}$$

Method (Kivinen-Warmuth '97): for every trial and every doubly stochastic comparator $\mathbf{U}$ show the key invariant

$$\Delta(\mathbf{U},\mathbf{W}) - \Delta(\mathbf{U},\mathbf{W}'') \geq (1-e^{\eta})(\overbrace{\mathbf{W}\bullet\mathbf{L}}^{\text{Alg's loss}}) - \eta(\overbrace{\mathbf{U}\bullet\mathbf{L}}^{\mathbf{U}\text{'s loss}})$$

where relative entropy $\Delta(\mathbf{U},\mathbf{W}) = \sum_{i,j} U_{i,j} \ln \frac{U_{i,j}}{W_{i,j}} + W_{i,j} - U_{i,j}$,

Summing over trials gives

$$\Delta(\mathbf{U},\mathbf{W}_{\text{init}}) - \Delta(\mathbf{U},\mathbf{W}_{\text{fin}}) \geq (1-e^{\eta})(\text{Alg's tot loss}) - \eta \sum_t (\mathbf{U}\bullet\mathbf{L}_t)$$

$$\text{Alg's tot loss} \leq \frac{\Delta(\mathbf{U},\mathbf{W}_{\text{init}}) + \eta\sum_t(\mathbf{U}\bullet\mathbf{L}_t)}{1-e^{-\eta}} = \frac{n\ln n + \eta\mathcal{L}_{\text{bestP}}}{1-e^{-\eta}}$$

Tune η (Freund-Schapire '97):

$$\text{loss} \leq \mathcal{L}_{\text{bestP}} + \sqrt{2\mathcal{L}_{\text{est}} n \ln n} + n\ln n$$

Problem: don't know r_i's, c_j's – can't work with $\Delta(\mathbf{U}, \mathbf{W}'')$ directly

Solution:

- Prove key invariant for $\mathbf{U}$, un-normalized $\mathbf{W}'$
- Normalization projects onto convex sets containing $\mathbf{U}$, so $\Delta(\mathbf{U}, \mathbf{W}'') \leq \Delta(\mathbf{U}, \mathbf{W}')$ (Herbster-Warmuth '01)

Implications:

- $\mathbf{W}'$ already close enough to $\mathbf{U}$, normalization to $\mathbf{W}''$ only helps.
- Can interleave element loss updates and row/column normalization
- Can bound "Un-normalized" versions of Weighted Majority etc. (what does this mean?)

Outline

Lower Bound

Consider just one row of L (where item 1 gets mapped to)

- Nature can assign arbitrary [0..1] loss to positions
- positions are experts in allocation (hedge) setting (FS '97)
- allocation algorithms "generalize" experts setting
- for large enough n, $\mathcal{L}_{\text{bestE}}$ any algorithm can have loss

  $\mathcal{L}_{\text{bestE}} + (1-\epsilon)\sqrt{\mathcal{L}_{\text{bestE}} \ln n}$ (CFHHSW '97)

Lower Bound

Repeat on different rows – each row uses up 1 position.
Use $\frac{n}{2}$ positions on $\frac{n}{2}$ rows. Bound becomes:

$$\frac{n}{2}\left(\mathcal{L}_{\text{bestE}} + (1-\epsilon)\sqrt{\mathcal{L}_{\text{bestE}} \ln \frac{n}{2}}\right)$$

$$= \mathcal{L}_{\text{bestP}} + (1-\epsilon)\sqrt{\mathcal{L}_{\text{bestP}} \frac{n}{2} \ln \frac{n}{2}}$$

where $\mathcal{L}_{\text{bestP}} = \frac{n}{2}\mathcal{L}_{\text{bestE}}$

upper bound: $\mathcal{L}_{\text{bestP}} + \sqrt{2\mathcal{L}_{\text{est}} n \ln n} + n \ln n$

Outline

Open problems

- Can interleaving update and normalization operations improve efficiency?
- Can bounds for FPL* be improved for permutations? (Trials can't be decomposed as in expert setting)
- What kind of other matrices can be learned? (Orthonormal?)

Open problems

- Can interleaving update and normalization operations improve efficiency?
- Can bounds for FPL* be improved for permutations? (Trials can't be decomposed as in expert setting)
- What kind of other matrices can be learned? (Orthonormal?)

Thank You!

Notes:

-
-
-
-