1 Graph Partitioning

Expansion of random cuts

\[ \phi = \min_{S \subseteq V} \frac{1}{n} E(S, S^c) \] (1)

can relax to real numbers

\[ d - \lambda_2 = \min_{x \in \mathbb{R}^V} \frac{\sum_{i \neq j} A_{ij} (x_i - x_j)^2}{\sum_{i \neq j} (x_i - x_j)^2} \rightarrow \text{spectral} \] (2)

or can relax to a vector

Claim:

\[ d - \lambda_2 = \min_{x \in \mathbb{R}^n} \frac{\sum_{i \neq j} A_{ij} ||x_i - x_j||_2^2}{\sum_{i \neq j} ||x_i - x_j||_2^2} \] (3)

Proof:

(*3) is relaxation of (*2), the direct solution of (*3) is

Claim

(*3) is equal to SDP

\[ \min \sum_{i \neq j} A_{ij} ||x_i - x_j||_2^2 \]

\[ \text{st} \sum_{i \neq j} ||x_i - x_j||_2^2 = n \]

which is equal to

\[ \min L_{G}[\text{tr}]X \]

\[ \text{st} L_{k_n}[\text{trace}]X = n \]

\[ x \geq 0 \] (4)

Problem harder (SDP versus eigenvalue problem)

useful -

• look at duals

• include “extra” information

Fact:
Dual of *(4)* is

$$\max y$$

$$\text{st } L_G \geq \frac{1}{n} L_n$$

A feasible solution is a number $y$ and a matrix $Y$ such that

$$L_G = \frac{y}{n} L_n + Y$$

Recall:

$$|S| |\overline{S}| = 1^S L_n 1_{\overline{S}}$$

$$E(S, \overline{S}) = 1^T S L_G L_{\overline{S}}$$

but

$$1^T S L_G 1_S = 1_S \left( \frac{y}{n} L_n + Y \right) 1_S \geq 1_S \frac{y}{n} L_n 1_S$$

So cost of cut $\geq y$

What’s going on here?

- embedding a scaled version of the complete graph in $G$
- we know the expansion and cut values for $K_n$ and so relate it to $G$. Note: $K_n$ is an expander.

Recall

Flow - if graph $H$ of known expansion can be embedded in $G$ “as a flow” then $h_H \leq h_G$.

- Then the optimal solution for a fixed $H$ can be computed as the solution to a concurrent multicommodity flow problem.
- $O(\log n)$ approximation, which is tight

Spectral - relax to an “eigenvalue problem” and use Cheeger.

ARV-type methods

- Can I construct iteratively a graph $H$ (and test its expansion) and stop when it’s a good expander and get a bound on $h_G$.
- yes - write as an SDP.
- Can compute faster by using primal-dual ideas

ARV - original $O(\sqrt{\log n})$

AHK - “primal-dual” method in theoretical computer science. Both using multicommodity flows

KRV - single-commodity flows using cut matching game

OSVV - extended KRV

LMO - empirical evaluation. Describes as spectral modified
2 Online Learning

prediction/inference problem - given data predict something.

Ways to formalize this, different assumptions on

• what the data are (real numbers, graphs, strings)
• where they come/generated from (according to an underlying distribution; access to side information)

“Traditional Statistic”

• data generated according to an underlying distribution
• learn parameters describing distribution
• evaluate quality by Risk - expected value of some loss function over the distribution in the data
• ERM→SRM

What if the data are not generated by some underlying process?

with no assumptions, hard to predict

Idea:

get data elements sequentially \( \{y_i, x_i\} \in \mathbb{R} \)
predict the next element.

Evaluated by the loss function e.g. number of incorrect predictions

Access to side information, namely prediction of a set of “experts.”

Experts make predictions according to some rule deterministic, random, adversarial, etc

At each time step, the experts also have a loss

Goal: want loss not too much worse than the “best” expert.

Also: in prediction at time \( t \) you have access to

• your prediction and losses in the past
• predictions and losses of the experts in the past

What are the experts?

• oracle
• statistical model
• certain steps in an algorithm
• basis functions

3 Multiplicative weights update rule

• maintain probability distribution over experts
• at each step, increase or decrease the weight multiplicatively \( \epsilon \) by multiplying by \((1 + \epsilon)\)

• \( \epsilon \) = parameter judges how much confidence to place in expert’s prediction/regularization parameter

Discrete Experts:

• set of experts \( E \) that makes predictions \( f_{E,\epsilon} : \mathbb{R}^n \)

• set of vectors \( \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1\} \) = weights on experts

• \( l_t(i) \) = loss of expert \( i \) at stage \( t \)

\( l_t (\hat{p_t}, y_t) = \text{loss of algorithm} = \sum_{i=1}^n x_i l_t(i) \)

Algorithm

1. \( W_0 = \vec{1} \)

2. when \( y_t \) and the experts prediction algorithm uses this update rule

\[
W_{t+1,i} = W_{t,i} (1 - \epsilon)^{l_t(i)} \\
= (1 - \epsilon)^{\sum_{i=1}^T l_t(i)} \\
= e^{-n \sum_{i=1}^T l_t(i)}, \text{ where } n = -\log (1 - \epsilon)
\]

Thm:

For any expert \( E_j \in [n] \)

\[
\sum_{t=1}^T l_t (\hat{p_t}) \leq \frac{\log n}{\epsilon} + \frac{y}{\epsilon} \sum_{t=1}^T l_t(i)
\]

Proof:

use potential function argument.

\[
W_t = \sum_{i=1}^n W_{t,i}
\]

First, relate potential function

\[
W_{t+1} \geq W_{t+1,i} = (1 - \epsilon)^{\sum_{i=1}^T l_t(i)} \\
= e^{-n \sum_{i=1}^T l_t(i)}
\]

Next relate potential function to performance of algorithm

\[
W_{t+1} = \sum_{i=1}^n w_{t+1,i} = \sum_{i=1}^n w_{t,i} (1 - \epsilon)^{l_t(i)}
\]

Note \((1 - \epsilon)^x \leq 1 - \epsilon x \text{ for } 0 \leq \epsilon \leq 1\)
So

\[ w_{t+1} \leq \sum_i w_{t,i} (1 - d_t(i)) \]

\[ = w_t \left(1 - \frac{\epsilon}{w_t} \sum_i w_{t,i} l_t(i)\right) \]

\[ = w_t (1 - d_t(\hat{p}_t)) \]

\[ \leq w_t \exp \left(-\epsilon \sum_{t=1}^T l_t(\hat{p}_t)\right) \]

\[ e^{-\eta \sum_t l_t(i)} \leq W_{t+1} \leq ne^{-\epsilon \sum_t l_t(\hat{p}_t)} \]

\[ \frac{\eta}{\epsilon} \sum_t l_t(i) \leq \frac{\log n}{\epsilon} - \epsilon \sum_t l_t(\hat{p}_t) \]

\[ \sum_t l_t(\hat{p}_t) \leq \frac{\log n}{\epsilon} + \frac{n}{\epsilon} \sum_t l_t(n) \]

\[ \leq \frac{\log n}{\epsilon} + (1 + \epsilon) \sum_t l_t(i) \]

Define the regret

\[ R_T = \sum_{t=1}^T l_t(p_t) - \min_{E_t} \sum_t l(f_{E,t}) \]

\[ \leq \frac{\log n}{\epsilon} + \epsilon \sum_t l_t(i) \]

if \( \epsilon = \sqrt{\frac{\log n}{T}} \)

\[ \leq 2T \log n \]

Q: is \( \log n \) large or small?

If “extra” information is given that one expert will be perfect

find the best expert in \( \log n \) mistakes

-multiplicative weights update rule says you’re not much worse than this scenario, in more general cases

applications to algorithms

- AHK \( \rightarrow \) generalize the losses to matrix losses to solve SDPs \( \rightarrow O(n^2) \) time

- KRV - “cut-matching” game to solve sparsest cuts. 2 players: a cut player, and a matching player

1. \( G_0 = 0 \)

2. in each round, cut player chooses a bisection \((S, \overline{S})\) and the matching player chooses a perfect matching \( M \) across \((S, \overline{S})\), then \( G_{t+1} \rightarrow G_t + M \).

3. game stops when \( G \) is an expander \( \epsilon G \geq \frac{1}{10} \)

4. value of game is number of steps it took. goal: cut player - stop soon (find expander fast), matching player - delay stop.
Dual algorithm.

1. Let $G' = \gamma G$

2. approximate the 2nd eigenvector of $L_{G'}$. Degree of approximation governed by regularization parameter.

3. use the bisection $(S_{n/2t}, S_{n/2t})$ from the sweep cut. Call flow-based improvement algorithm to get a cut $(T_t, T^c_t)$ and a matching $M_t$ until stopping rule is satisfied. Let $G' = G' + M_t$. Return “best” cut $(T_t)_{t=1}^T$

Why would you hope/expect that these multiplicative weight update algorithms would perform well in practice?

- faster than naive computation
- often give better answers than the exact algorithm

Boosting - example of an ensemble method

Given $X$, learn $C : X \to \{0, 1\}$ a classification rule from some concept class $C$

Risk = $E[error]$.

Define a $\gamma$-weak learning algorithm is one that has error $\leq \frac{1}{2} - \gamma$.

An $\epsilon$-strong learning algorithm with error $\leq \epsilon$.

Can one combine a set of weak learners into a strong learner.

Idea - weak learners are a little better than chance, so combining them doesn’t make things worse. If they are “different” then we can hope for improvement by averaging predictions.

Boosting - AdaBoost - do boosting by sampling. Take a sample of data and use algorithm to boost on that sample

- do this in an iterative manner by “updating” weights on data points to find new classification rule for data points that are misclassified

Events - hypotheses for the classification rule output at each step

At each step - get a classification rule $h_t$ (weak learner) and final classification algorithm use $\sum_t h_t$ as prediction