CS369M: Algorithms for Modern Massive Data Set Analysis **Lecture 16 - 11/11/2009**

Multiplicative Update Algorithms, Boosting and Ensemble Methods

Lecturer: Michael Mahoney *Scribes: Mark Wagner and Yunting Sun*

**Undited Notes*

1 Graph Partitioning

Expansion of random cuts

$$\phi = \min_{S \subset V} \frac{E\left(S, \overline{S}\right)}{\frac{1}{n}|S|\left|\overline{S}\right|} (*1)$$

can relax to real numbers

$$d - \lambda_2 = \min_{x \epsilon \mathbb{R}^V} \frac{\sum A_{ij}\left(x_i - x_j\right)^2}{\sum_{ij}\left(x_i - x_j\right)^2} \rightarrow \text{spectral} (*2)$$

or can relax to a vector

Claim:

$$d - \lambda_2 = \min_{\vec{x_j} \epsilon \mathbb{R}^n} \frac{\sum_{ij} A_{ij}\left|\left|x_i - x_j\right|\right|_2^2}{\sum_{ij}\left|\left|x_i - x_j\right|\right|_2^2} (*3)$$

Proof:

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

Claim

($*3$) is equal to SDP

$$\begin{aligned} \min \sum_{ij} A_{ij}\left|\left|x_i - x_j\right|\right|_2^2 & \\ \text{st} \sum_{ij}\left|\left|x_i - x_j\right|\right|_2^2 &= n \end{aligned}$$

which is equal to

$$\begin{aligned} \min L_G[tr]X & \\ \text{st } L_{k_n}[trace]X &= n \\ x &\geq 0 (*4) \end{aligned}$$

Problem harder (SDP versus eigenvalue problem)

useful -

- look at duals
- include "extra" information

Fact:

Dual of $(*4)$ is

$$\begin{aligned} & \max y \\ \text{st } L_G & \geq y\frac{1}{n}L_n \end{aligned}$$

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

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

Recall:

$$\begin{aligned} |S|\left|\overline{S}\right| &= 1_S^T L_n 1_{\overline{S}} \\ E\left(S,\overline{S}\right) &= 1_S^T L_G L_{\overline{S}} \end{aligned}$$

but

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

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 $O\left(\sqrt{\log n}\right)$

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 distribtion
- learn paramters describing distribtion
- evaluate quality by *Risk* - expected value of some loss function over the distribtion 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 $\{y_i, x_i\} \epsilon \mathbb{R}$

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 *ie* by multiplying by $(1+\epsilon)$
- ϵ = parameter judges how much confidence to place in expert's prediction/regularization parameter

Discrete Experts:

- set of experts E that makes predictions $f_{E_{i,t}} \epsilon \mathbb{R}^n$
- set of vectors $\{x \epsilon \mathbb{R}^n : \sum_{i=1}^n x_i = 1\}$ = weights on experts
- $l_t(i)$ = loss of expert i at stage t $l(\hat{p}_t, y_t)$ = 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

$$\begin{aligned} W_{t+1,i} &= W_{t,i}(1-\epsilon)^{l_t(i)} \\ &= \\ &\quad (1-\epsilon)^{\sum_{t=1}^T l_t(i)} \\ &= e^{-n\sum_{t=1}^T l_t(i)}, \text{where} \quad n = -\log(1-\epsilon) \end{aligned}$$

Thm:

For any expert E_j j$\epsilon [n]$

$$\sum_{t=1}^T l_t(\hat{p}_t) \leq \frac{\log n}{\epsilon} + \frac{y}{\epsilon}\sum_{t=1}^T l_t(i)$$

Proof

use potential function argument.

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

First, relate potetial function

$$\begin{aligned} W_{t+1} &\geq W_{t+1,i} = (1-\epsilon)^{\sum_{t=1}^T l_t(i)} \\ &= e^{-n\sum_{t=1}^n l_t(i)} \end{aligned}$$

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 $(1-\epsilon)^x \leq 1 - \epsilon x$ for $0 \leq \epsilon \leq 1$

So

$$\begin{aligned} w_{t+1} &\leq \sum_i w_{t,i}\left(1-\epsilon l_t\left(i\right)\right) \\ &= w_t\left(1-\frac{\epsilon}{w_t}\sum_i w_{t,i}l_t\left(i\right)\right) \\ &= w_t\left(1-\epsilon l_t\left(\hat{p}_t\right)\right) \\ &\leq w_t\exp\left(-\epsilon l_t\left(\hat{p}_t\right)\right) \\ &\leq w_t\exp\left(-\epsilon\sum_{t=1}^{T} l_t\left(\hat{p}_t\right)\right) \end{aligned}$$

$$\begin{aligned} e^{-\eta\sum_t l_t(i)} &\leq W_{t+1} \leq ne^{-\epsilon\sum_t l_t(\hat{p}_t)} \\ \frac{\eta}{\epsilon}\sum_t l_t\left(i\right) &\leq \frac{\log n}{\epsilon} - \frac{\epsilon}{\epsilon}\sum_t l_t\left(\hat{p}_t\right) \\ \sum_t l_t\left(\hat{p}_t\right) &\leq \frac{\log n}{\epsilon} + \frac{n}{\epsilon}\sum_t l_t\left(n\right) \\ &\leq \frac{\log n}{\epsilon} + (1+\epsilon)\sum_t l_t\left(i\right) \end{aligned}$$

Define the regret

$$\begin{aligned} R_T &= \sum_{t=1}^{T} l_t\left(p_t\right) - \min_{experts}\sum_t l\left(f_{E,t}\right) \\ &\leq \frac{\log n}{\epsilon} + \epsilon\sum_t l_t\left(i\right) \end{aligned}$$

if $\epsilon = \sqrt{\frac{\log n}{T}}$

$$\leq 2\sqrt{T\log n}$$

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

<u>applications to algorithms</u>

- AHK $\rightarrow$ generalize the losses to matrix losses to solve SDPs$\rightarrow O\left(n^2\right)$ 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 $\left(S, \overline{S}\right)$ and the matching player chooses a perfect matching M across $\left(S, \overline{S}\right)$. then $G_{t+1} \rightarrow G_t + M$.
3. game stops when G is an expander *eg* $l_G \geq \frac{1}{10}$
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 $\left(S_{n/2t}, \overline{S}_{n/2t}\right)$ from the sweep cut. Call flow-based improvement algorithm to get a cut $\left(T_t, \overline{T_t}\right)$ 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 = $E\left[error\right]$.

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