Lower Bounds for Majority Voting and Pseudorandom choice

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

March 8, 2013

Abstract

In this article philosophical ramifications of the P(good) equation are described and circuits for the LHS and RHS of the P(good) equation are modelled and their lower bounds are analyzed.

1 Some circuit models for majority voting

In a majority voting milieu, each voter has a criterion to judge a candidate based on which the voter decides to vote for or against. Since satisfying the criterion is akin to a SAT problem, a voter's criterion can be specified as a CNF where each clause of the CNF is a sub-criterion rule that is part of the criterion the voter has and every rule clause is dependent on some variables. The probability that the voter makes a good decision can be defined with conditional probability as follows,

```
Pr(Voter makes good decision) =
Pr(Criterion CNF is good) *
Pr(CNF is populated with correct inputs/ Criterion CNF is good)

Pr[Criterion CNF is good] = p = c1*c2*c3*..ck
where each ci gives the probability that i(th) clause is good.

Pr[CNF is populated with correct inputs] = q = q1*q2*q3*...ql
where each qi gives the probability that i(th) variable
is correctly populated.

From the above, Pr(Voter makes a good decision) = p*q
```

2 Voter as an Oracle Turing Machine

Voter with CNFSAT having bounded error can be modelled as a Polytime Turing machine with bounded error. The Voter Turing machine accepts a CNFSAT criterion and variable assignments to the CNFSAT in its input tape and computes the CNFSAT on those assignments on a worktape and writes the output to an output tape. This computation takes only polynomial time as it is only a verification step. But this computation can have an error as described previously âĂŞ the CNFSAT might output 1 when it has to output 0 and viceversa. Such a Voter Turing machine with error can be found in real life applications. Since it is in polynomial time with bounded error Voter Turing machine is in BPP. This BPP Voter turing machine can be used as an Oracle to other Turing machines and circuits. The CNFSAT criterion can be probabilistic also and thus it could be in Probabilistic SAT(PSAT) - Probabilistic SAT has to satisify probability distribution constraints

on the clauses of the PSAT. But PSAT does not involve bounded error of a voter. Thus BPP model which allows bounded error of a voter is a better oracle. Moreover the bounded error in BPP can be brought down (derandomized in a sense) to a required threshold by repeated runs.

3 Error probability derivation for majority voting

```
For even number of voters(2n),
Pr(majority decision is good) =
(2n)!/(4^n) [ 1/(n+1)!(n-1)! +1/(n-2)!(n+2)! + ... + 1/(2n)!]

For odd number of voters(m)
Pr(majority decision is good) =
m!/(2^m) [ 1/x!(m-x)! + ... + 1/m! ]
where x is the ceiling(m/2)
```

Assumption here is that a good majority decision with high probability also results in a good outcome in the majority voting (rather it is trivial and is stated without proof).

4 Ways to simulate majority voting

Majority can be computed by non-uniform NC1 circuit or through a sorting network of polynomial size (Ajtai et al) or by Valiant's non-constructive majority circuit. As described previously, each voter has a criterion while voting which can be modelled as a CNF SAT with error probability. Thus a majority gate computing majority can be given access to oracle BPP Voter Turing machine as described in previous section. So such a majority voting circuit is in the non-uniform circuit complexity class $NC1^{BPP}$. NC1 is in P which inturn is in BPP. Thus majority voting with error by voter nodes can be computed in $NC1^{BPP}$. The error for this circuit is defined by the RHS of the P(good) equation. Alternatively, instead of having a BPP oracle access, the majority voting circuit can also be designed as a non-uniform Bounded Error NC1 circuit placing it in complexity class BPNC.

5 Ways to simulate junta or a random choice

LHS of the P(good) is a junta which is not the result of a majority vote and it arises by itself. This random choice can be one among the voters. Since there is no perfect randomness, only a pseudorandom generator has to be resorted to for choosing a junta. Pseudorandom generators fool a statistical test into believing as though it is truly random with high probability and form the basis of cryptographic key generators. Hypothesis of existence of pseudorandom generators is yet to be proved. Existence of one-way functions (hard to invert) imply pseudorandom generators. There are prominent PRGs like Nisan which run in polynomial time. Thus a junta can be output by following algorithm:

1. Source for a PRG which runs in poytime
2. Query the PRG to get the (pseudo)random choice

Junta output by the above algorithm can be either good or bad with a probability distribution and thus has a bounded error. This algorithm runs in polynomial time but with bounded error described by LHS of P(good) equation and thus in BPP.

6 Consequences of above bounds for pseudorandom choice and majority voting

As described above pseudorandom choice is in BPP and majority voting with error is in non-uniform NC1 with oracle access to BPP. Both a pseudorandom choice and majority voted choice have the same error probability of being correct as derived by P(good) equation. Though this is against conventional wisdom, the P(good) equation is not necessarily a correct representation of realworld majority voting where individual voters can have differing judgemental abilities with unequal error probability which is a result of varying human intelligence amongst the population and hence is a mootpoint. Thus LHS of the P(good) equation may be more or less compared to the RHS and as a result either a pseudorandom choice junta or a majority voting may be favorable depending on individual judging error probabilities in the population. Thus in real world this error probability derivation may have no relevance at all. But in a theoretical world, a distributed computing system where all nodes are equally intelligent(or have same algorithm running in them), then LHS of P(good) is always equal to RHS since there is zero bias because of uniform distribution. Application of the above separation result for future complexity class separations could be interesting.

7 Acknowledgement

I dedicate this article to God.

8 Bibliography

References

[1] Few Algorithms for ascertaining merit of a document - http://arxiv.org/pdf/1006.4458.pdf

[2] TAC 2010 Update summarization by Interview Algorithm (*http* : *//www.nist.gov/tac/publications/*2010*/appendices/Summarization/guided/*$CMI_I IT.pdf$)

[3] Interview algorithm is in IP=PSPACE

[4] Nisan PRG

[5] Complexity - Arora Barak

[6] Leader Election Algorithms in distributed systems

[7] Majority in non-uniform NC1 - Barrington