Recent advances in Complexity CIS 6930/CIS 4930 November 26, 2002

Lecture 27

Lecturer: Dr. Meera Sitharam Scribe: Srijit Kamath

1 Proofs with *LOGSPACE* Statistical Tests

In the last lecture we briefly introduced the classification of $LOGSPACE$ statistical tests given by Sivakumar (2001). We now look at some examples.

1.1 Chernoff-Hoeffding Lemma/Bound

Problem 1 *Given m sets $A_1, A_2, \ldots, A_m \subseteq \{1, \ldots, n\}$, the goal is to compute a set Q such that $max_i \Delta_i = ||A_i \cap Q| - |A_i \setminus Q||$ is minimized.*

A randomized algorithm that obtains a Q that is a good approximation to the desired Q is as follows: It picks each element of $\{1, 2, \ldots, n\}$ with probability 1/2. to get Q.
The proof of correctness (that uses the C-H bound) shows that when Q is picked according to this rule, then for some $C > 0$, $\forall i\ P[\Delta_i > C\sqrt{|A_i| logm}] < 1/(2m)$. This means that the probability that the algorithms fails to give a good approximation is bounded above by $1/(2m)$. In other words there is an n bit string whose randomness is tested by some statistical test during this proof. The statistical test can be thought of as an algorithm that either accepts or rejects a string depending on the outcome of the test. The proof that the random string passes this test is the C-H bound. Next we describe the statistical test.

Think of the statistical test as a Turing Machine with the following tapes:

1. Input Tape: It holds n random bits $q_1, q_2, \ldots, q_n$. $q_i = 1$ iff $i \in Q$.
2. Active tape: Has sets $A_1, \ldots, A_m$ and the values $C^2|A|logm$. Note that the contents of the tape are independent of the input itself and depend only on the input length.
3. Index tape: Contain $i \in \{1, \ldots, m\}$.
4. Work tape: Integer variables j, a, b that will take values in the interval $(0, \ldots, n)$. a will count $|A_i \cap Q|$ and b will count $|A_i\ Q|$.

The algorithm is as follows:

```
Initialize $a$ and $b$ to zero.
  for $j = 1$ to $n$ do
    if $j \in A_i$ then
```

```
            if $q_j = 1$
                then $a = a + b$
                else $b = b + 1$
            end if
        end if
    end for
if $(a - b)^2 \leq C^2|A_i|logm$
    then accept
    else reject
```

The statistical test we described is a $LOGSPACE$ computation. It is this test that a random string has to pass in order for the C-H bound to go through. Now that we have a $LOGSPACE$ statistical test we need to give a pseudorandom generator that fools this test (and in fact any $LOGSPACE$ test).

Theorem 1 *$\exists$ constants $A > 0$, $B > 0$, such that $\forall \epsilon > 0$ given a LOGSPACE machine/algorithm/statistical test M and $t \geq A(logn^{O(1)} + log(1/\epsilon))$ it is possible to construct a sample space (set of pseudorandom strings + distribution) $S \subseteq \{0,1\}^{tn}$ such that $|S| = (n/\epsilon)^{O(1)}$ so that for any value I on M's index tapes, $|P_{r\in\{0,1\}^{tn}}(M(I)$ accepts $r) - P_{r\in S}(M(I)$ accepts $r))| < \epsilon$.*

The C-H bound proves that when $q \in \{0,1\}^n$ is chosen uniformly at random $\forall i$, then $D(i)$ rejects with probability at most $1/(2m)$. Applying the pseudorandom construction we get sample space $(n/\epsilon)^C = (n+m)^{O(1)}$ (taking $\epsilon = 1/(2m)$), for some fixed constant C. If q is chosen according to S then $\forall i$ $D(i)$ rejects with probability $< (1/(2m) + \epsilon) = 1/m$. Thus probability $D(i)$ rejects for some i is $< \Sigma(1/m) < 1$. The probability that $D(i)$ accepts for all i is strictly greater than zero. This means that $\exists$ atleast 1 pseudorandom string $q \in S$ such that $D(i)$ accepts for every i.

1.2 Johnson-Lindenstrauss Lemma

This J-L lemma is frquently applied to problems in combinatorial geometry. Typically problems get exponentially complex with increase in dimensions, i.e., complexity is $O(n^d)$. So we would like to try to reduce a problem instance into one with fewer dimensions while preserving the geometric invariant. The following problem serves as an example.

Problem 2 *Given n points $u_1, \ldots, u_n \in \Re^d$ (where d is large) we want $\Phi : \Re^d \longrightarrow \Re^k$ ($k << d$) such that for each $i, j, 1 \leq i < j \leq n$, $(1-\epsilon)||u_i - u_j||^2 \leq ||\Phi(u_i) - \Phi(u_j)||^2$*

One proposed solution is to project the given vectors into a random $k-$ dimensional subspace and it can be shown that with high probabilty the test is passed. However, it is hard to show that the test $\in LOGSPACE$. So we need to convert to an equivalent lemma whose statistical test $\in LOGSPACE$. Such a convertion is given in Achlioptus (2001). Φ is constructed by taking a $k \times d$ matrix (call it A_Φ) such that the rows of the matrix are the basis of the $k-$ dimensional subspace desired and whose entries $\in \{-1, +1\}$. Now we have $\Phi(u_i) = A_\Phi u_i$. The statistical test takes this random matrix entries as input (with u_is on the advice tape) and tests the statement. In this case we have a $LOGSPACE$ test.

2 Physics, Information and Computation

In the remainder of this lecture and the next lecture we will study the relationship between physics, information and computation. We begin by giving an introduction to Kolmogorov complexity.

2.1 Kolmogorov Complexity

Suppose that we want to measure the amount of information contained in a finite binary string. Consider, for example, the string $111\ldots1$, where the number of 1s is $10,000$. Intuitively, this string contains little information.
Observation: The following program outputs the string:

```
for $i = 1$ to $10,000$
print 1
```

Likewise, consider the number $\pi = 3.1415\ldots$, which is an infinite sequence of digits. There is also a short program that computes the consecutive digits of π. These observations motivate the following definition.

Definition 1 *The amount of information in a finite string is the size of the shortest program that without additional data, computes the string and terminates.*

It appears that the amount of information in a string depends on the programming language used. However it has been proved that all reasonable choices of programming languages lead to the quantification of information that is invariant up to an additive constant.This quantity is called the *Kolmogorov Complexity* of the string. Some strings such as those in the examples above have shorter descriptions than themselves and are said to be *compressible.*

2.2 Information Theory

A parallel development was the study of information theory and coding by Shannon, which we briefly review.
Suppose we want to assign a quantity of information to an ensemble of possible messages. All messages in the ensemble being equally probable, this quantity is the number of bits needed to count all possibilities. Note that in this case we ignore the meaning of the message itself. Rather the interest lies in communicateing a message between a sender and a receiver. Further we assume that the set of all possible messages is known to both the sender and the receiver. The set of messages from which the selection takes place is called an *ensemble.* We will only consisder countable ensembles.

Definition 2 *The entropy of a random variable X with outcomes in an ensemble S and all messages in the ensemble equally likely is the quantity $H(X) = logd(S)$, where $d(S)$ is the cardinality of S.*

By choosing a particular message 'a' from S, we remove the entropy from X by the assignment $X = a$ and produce or transmit information $I = logd(S)$ by our selection of a. Expressing in bits, $\lceil I' = logd(S) \rceil$.
Note that if we have K independent variables $X_1, X_2, \ldots, X_k$ each can take n_i values $i = 1, 2, \ldots, k$ the number of combinations possible is $n_1 n_2, \ldots n_k$ and the entropy (when the messages are equally likely) is $H(X_1, X_2, \ldots, X_k) =$

$logn_1 + logn_2 + \ldots + logn_k = logn.$

What is the entropy when the messages occur with given probabilities?

Definition 3 *The entropy of a random variable X with given probabilities $P(X = a_i) = p_i$ is $H(X) = -\Sigma p_i log p_i$.*

If we want to encode a sequence $x_1 x_2 \ldots x_n$, $x_i \in N$, call $x_1 x_2 \ldots x_n$ the *source sequence* and $y_1 y_2 \ldots y_n$ with $y_i = E(x_i)$ the *code sequence.* A code is *uniquely decodable* if for each source sequence of finite length, the code sequence corresponding to that source sequence is different from the code sequence corresponding to any other source sequence. A code is a prefix code if the set of code words is *prefix free*, i.e., no code used is a prefix of another code word.