1 Last time and today

1.1 Last time

- Finished proof of Siegenthaler’s Theorem.
- Started learning parity under noise.

1.2 Today

- Not covered in these notes. Finish proof for learning parities under noise.
- Learning with membership queries (MQ) under the uniform distribution.
- Learning juntas with MQ.
- Learning sparse boolean functions with MQ.

1.3 Relevant Readings

Learning with Membership Queries

We will introduce a learning protocol that gives more power to the learner as compared to previously studied models. The game between the learner and nature is as follows: nature chooses a target hypothesis. The learner chooses a particular input \( x \) and asks nature what its label is with nature returning the correct label for \( x \). This interaction lasts for a finite number of epochs. The game stops with the learner returning a hypothesis. The learner is considered successful if the returned hypothesis is “close” (with respect to sampling instances uniformly from the instance space) to the target hypothesis with high probability.

It is easy to see that the model is easier as compared to the standard PAC Learning under the Uniform Distribution model, since the learner can simulate the latter model by sampling instances uniformly from the input space.

Learning Juntas in the MQ Model

We will first show that we can learn efficiently \( r \)-juntas in the MQ model. Let’s consider \( f \) to be a Boolean function \( \{-1, 1\}^n \rightarrow \{-1, 1\} \) that only depends on \( r \in [n] \) of its variables (so \( f \) can’t be constant). The algorithm for finding the relevant variables is simple:

1. Find two inputs \( x \) and \( y \) such that \( f(x) \neq f(y) \).
2. Search for an input \( u \) that is ‘between’ \( x \) and \( y \) such that \( f(u) \neq f(u^{(i)}) \) (where \( u^{(i)} \) is identical to \( u \) except its \( i \)-th bit is flipped, for some \( i \in [n] \)).
3. If \( r > 1 \), apply the above process recursively on \( f_{|i \rightarrow 1} \) and \( f_{|i \rightarrow 0} \) (Boolean functions restricted to \( n - 1 \) bits) which are \( r - 1 \) juntas.

For any pair \( x \) and \( y \), there is a minimal sequence of bit flips of size at most \( n \) that can transform \( x \) into \( y \) or vice-versa. In step 2 above, ‘between’ simply means an intermediate input obtained at any step in such a transformation. Such an input obviously exists since there must be an input in the minimal sequence of bit flips that transforms \( x \) into \( y \) such that the function changes. This input has the desired property. Doing a linear or binary search for such an input will find it in time at most \( O(n) \) for any given \( x, y \) pair and will implicitly determine one relevant variable (the flipped bit in \( u \)). The power of doing membership queries allows us to actually do the search, which is not possible in the classic random sampling model.
Step 1 above assumes that we are given two inputs with different values. To find such a pair, we can randomly sample an input $x$, query its value and then search for another input $y$ with the opposite value. The search is done by randomly sampling new inputs until such an input is found. The concentration of inputs that are evaluated to $-1$ is at least $\frac{1}{2^r}$ (and the same holds for inputs evaluated to $1$). The reason is simple: if we look at function $f$ being defined only on the relevant $r$ variables, since $f$ is non-constant, each output value must appear for at least one of the possible $2^r$ inputs, and thus, if we sample inputs uniformly at random, it will take (on expectation) at most $\frac{1}{2^r}$ samples to find a $1$ or a $-1$. So in $\text{poly}(\frac{1}{2^r})$ samples, with high probability, we can obtain such a pair of inputs.

Putting the two observations together, the total sampling cost for finding the $r$ relevant variables is $\text{poly}(\frac{1}{2^r}, \log(n))$ and the time complexity is $\text{poly}(\frac{1}{2^r}, n)$.

4 Learning Sparse Boolean Functions in the MQ Model

The additional power given by the ability to do queries proves essential for efficiently learning Boolean functions with a small number of dominant Fourier coefficients, irrespective of whether the coefficients are on sets which are 'small' or 'large'. This characteristic is the main advantage of learning with membership queries as compared to the Low Degree algorithm that is efficient only when the dominant Fourier coefficients are on 'small' sets.

Given a Boolean function $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$, the goal for the first part of this section will be to find the Fourier coefficients of arbitrary sets $S$ under the assumption that the coefficient $\hat{f}(S)$ is large.

We will use a notational trick and partition the $n$ variables in two pieces: $S_1$ containing the first $k$ variables and $T_2$ containing the last $n-k$ variables for some fixed $k \in 0, ..., n$ (where 1 and 2 in the subscripts for $S_1$ and $T_2$ are meant as a mnemonic for their order in the partition). Then, for a fixed $k$ and a fixed $S_1 \subseteq [k]$ we define a partial Boolean function, $f_{k,S_1} : \{-1, 1\}^{n-k} \rightarrow \mathbb{R}$:

$$f_{k,S_1}(x) = \sum_{T_2 \subseteq [k+1,n]} \hat{f}(S_1 \cup T_2) \chi_{T_2}(x), \forall x \in \{-1, 1\}^{n-k}$$ (1)

To gain intuition, it is worth stating the values of $f_{k,S_1}$ for extreme values of $k$:
4 LEARNING SPARSE BOOLEAN FUNCTIONS IN THE MQ MODEL

\[ f_{0, \emptyset}(x) = \sum_{T_2 \subseteq [1, n]} \hat{f}(T_2) \chi_{T_2}(x) = f(x) \] (2)

\[ f_{n, S_1}(x) = \hat{f}(S_1) \chi_{\emptyset}(x) = \hat{f}(S_1) \] (3)

The intuition for the role of the two pieces of the partition is that the various \( S_1 \)'s partition the set of \( 2^n \) Fourier coefficients in \( 2^k \) pieces, with each piece containing \( 2^{n-k} \) Fourier coefficients (corresponding to the various choices of \( T_2 \)).

Now, the main insight as to why using membership queries offers an advantage over uniform sampling from the instance space is that we can approximately compute \( f_{k, S_1}(x) \) for some fixed and known \( k, S_1 \) and \( x \) by fixing the values of the 'last' \( n - k \) variables and querying only inputs with those variable fixed. This is impossible in the uniform sampling model. We will soon see why this fact is important, but first let's make the above statement precise:

**Lemma 1.** For a fixed \( k \in \{0, \ldots, n\} \), a fixed \( S_1 \subseteq [k] \) and a fixed \( x \in \{-1, 1\}^{n-k} \),

\[ f_{k, S_1} = \mathbb{E}_{y \in \{-1, 1\}^k} [f(yx) \chi_{S_1}(y)]. \]

**Proof.** We write \( f(yx) \) using the Fourier coefficients for \( f \):

\[ f(yx) = \sum_{T \subseteq [n]} \hat{f}(T) \chi_T(yx) \] (4)

Let’s partition \( T \) in two pieces: \( T_1 = T \cap [k] \) and \( T_2 = T \cap [k + 1, n] \). Since \( \chi_T \) computes parity over \( T \), \( \chi_T(yx) = \chi_{T_1}(y) \chi_{T_2}(x) \) and thus:

\[ f(yx) = \sum_{T \subseteq [n]} \hat{f}(T_1 \cup T_2) \chi_{T_1}(y) \chi_{T_2}(x) \] (5)

\[ = \sum_{T_1 \subseteq [k]} \sum_{T_2 \subseteq [k+1, n]} (\hat{f}(T_1 \cup T_2) \chi_{T_2}(x)) \chi_{T_1}(y) \] (6)

where the second equality follows from \( T_1 \) and \( T_2 \) being disjoint. Then:

\[ \mathbb{E}_{y \in \{-1, 1\}^k} [f(yx) \chi_{S_1}(y)] = \mathbb{E}_{y} [\sum_{T_1 \subseteq [k]} \sum_{T_2 \subseteq [k+1, n]} \hat{f}(T_1 \cup T_2) \chi_{T_2}(x) \chi_{T_1}(y) \chi_{S_1}(y)] \] (7)

\[ = \sum_{T_1 \subseteq [k]} \sum_{T_2 \subseteq [k+1, n]} \hat{f}(T_1 \cup T_2) \chi_{T_2}(x) \mathbb{E}_{y} [\chi_{T_1}(y) \chi_{S_1}(y)] \] (8)

(9)
due to the linearity of expectation, and then:

\[
\mathbb{E}_{y \in \{-1,1\}^k}[f(y)xS_1(y)] = \sum_{T_2 \subseteq [k+1,n]} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x)\mathbb{E}_y[\chi_{S_1}(y)\chi_{S_1}(y)] \tag{10}
\]

\[
+ \sum_{T_1 \subseteq [k], T_1 \neq S_1} \sum_{T_2 \subseteq [k+1,n]} \hat{f}(T_1 \cup T_2)\chi_{T_2}(x)\mathbb{E}_y[\chi_{T_1}(y)\chi_{S_1}(y)]
\tag{11}
\]

\[
= \sum_{T_2 \subseteq [k+1,n]} \hat{f}(S_1 \cup T_2)\chi_{T_2}(x) \tag{12}
\]

\[
= f_{k,S_1} \tag{13}
\]

where the first equality follows from partitioning \([k]\) and the second equality from the known fact that \(\mathbb{E}_x[\chi_S(x)\chi_T(x)] = 1\) if \(S = T\) and 0 otherwise.

A nice characteristic of the above relation is that it can be approximated via random samples from \(\{-1,1\}^k\), by fixing the rest of the variables from \([k+1,n]\)—which is allowed in the membership query model. We can use the Chernoff bound to get a value of the empirical average close to the expectation from the formula for \(f_{k,S_1}\) using a number of samples that only depends on the desired accuracy and success probability (and independent of the number of variables \(k\)). This in turn allows us to approximate (with high probability) the value of \(f_{k,S_1}\). This thus leads to the fact:

**Fact 2.** For a fixed \(k, S_1\) and \(x\), a given an accuracy parameter \(\gamma\) and a limit on the acceptable failure probability \(\delta\), we can approximate \(f_{k,S_1}\) by a value \(v\) such that \(|v - f_{k,S_1}(x)| \leq \gamma\) w.p. at least \(1 - \delta\) using at most \(O(\frac{1}{\gamma^2} \log(\frac{1}{\delta}))\) random samples from \(\{-1,1\}^k\).

The time complexity is determined only by the need to compute the empirical average for \(O(\frac{1}{\gamma^2} \log(\frac{1}{\delta}))\) \(n\)-bit numbers and is thus \(O(\frac{n}{\gamma^2} \log(\frac{1}{\delta}))\).

We will now make a simplifying assumption (and discuss at the end how to address it) and consider that we can compute \(f_{k,S_1}(x)\) exactly. We will introduce three observations that will be used in the main theorem. For all three we will consider \(f : \{-1,1\}^n \rightarrow \{-1,1\}\) and a real constant \(\theta > 0\).

**Observation 3.** At most \(\frac{1}{\theta^2}\) sets \(S\) have \(|\hat{f}(S)| \geq \theta\).

*Proof.* Assume more than \(\frac{1}{\theta^2}\) sets have \(|\hat{f}(S)| \geq \theta\). Then \(\sum_{S \subseteq [n]} \hat{f}(S)^2 \geq \sum_{S \subseteq [n], |\hat{f}(S)| \geq \theta} \hat{f}(S)^2 > \theta^2 \frac{1}{\theta^2} = 1 = \sum_{S \subseteq [n]} \hat{f}(S)^2\), which is a contradiction (with the last equality due to the fact that \(f\) takes values in \(\{-1,1\}\)).
Observation 4. For a fixed \( k \in \{0,\ldots,n\} \), at most \( \frac{1}{2^k} \) of the \( 2^k \) functions \( f_{k,S} \) can have 
\[ \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] \geq \theta^2. \]

Proof. Since \( f_{k,S} \) is a Boolean function, we can use Parseval to get
\[ \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] = \sum_{T_2 \subseteq [k+1,n]} \hat{f}(S_1 \cup T_2)^2. \]
Then \( \sum_{S_i \subseteq [k]} \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] = \sum_{S_i \subseteq [k]} \sum_{T_2 \subseteq [k+1,n]} \hat{f}(S_1 \cup T_2)^2 = \sum_{S \subseteq [n]} \hat{f}(S)^2 = 1. \) So we can use Observation 3 to get the desired result. \( \square \)

Observation 5. If \( |\hat{f}(S_1 \cup T_2)| \geq \theta \), then \( \mathbb{E}[f_{k,S}(x)^2] \geq \theta^2. \)

Proof. In the same way as in Observation 4, \( \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] = \sum_{T_2 \subseteq [k+1,n]} \hat{f}(S_1 \cup T_2)^2 \geq (\hat{f}(S_1 \cup T_2))^2 \geq \theta^2. \) \( \square \)

We are now in a position to state the main claim. But before doing so, let’s make an even stronger assumption than before and assume that for any fixed \( k, S_1, x \) we can compute exactly and in unit time \( \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] \). Then:

Claim 6. For any function \( f : \{-1,1\}^n \rightarrow \{-1,1\} \), if the above assumption holds, we can exactly identify all sets \( S \subseteq [n] \) such that \( |\hat{f}(S)| \geq \theta \) in time \( \text{poly}(n, \frac{1}{\theta}) \).

Proof (sketch). The idea of the proof is to build a binary tree starting with \( S_1 = \emptyset \) at the root and expanding \( S_1 \) with a new variable at each new layer. Thus, at layer \( k \), each node \( i \) corresponds to the function \( f_{k,S_1^{(i)}} \) where \( S_1^{(i)}, i \in [2^k] \) is the subset of variables that have fixed values and were considered at the previous layers (we will label \( S_1^{(i)} \) as \( S_1 \) unless it is confusing).

We will label a node as active if its associated \( f_{k,S} \) has \( \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] \geq \theta^2 \) and inactive otherwise.

We operate at each layer by expanding all active nodes with a new variable and then select among the generated nodes only those that are active. We apply this process starting at the root (which is active with a value for \( \mathbb{E}[f_{k,\emptyset}^2] = 1 \)) and continue recursively until each \( S_1 \) contains all variables. Due to Observation 4, it can be shown inductively that at each level there are at most \( \frac{1}{2^k} \) active nodes as the sum of the coefficients of \( \mathbb{E}[f_{k,S_1^{(i)}}^2] \) is 1 (as in the proof for Observation 4). But this is true even at the level of the leaves, so all active nodes at the level of the leaves are the desired Fourier coefficients.

Since the tree has height \( n \) and since at each layer the effort is equal to the amount of active nodes, the total time to compute the desired Fourier coefficients is \( \text{poly}(n, \frac{1}{\theta}) \). \( \square \)

We will now address the problem of removing the assumption that for any fixed \( k, S_1, x \) we can compute \( \mathbb{E}_{x \in \{-1,1\}^{n-k}}[f_{k,S}(x)^2] \) exactly and in unit time. We know from
Fact 2 that, for any fixed $k, S_1, x$ we can approximate within an accuracy $\gamma f_{k,S_1}(x)$ using $O(\frac{1}{\gamma^2})$ samples (ignoring log factors). Then we simply get enough random samples (given by the Chernoff bound again) of $x \in \{−1, 1\}^{n−k}$, for each such value $x$ we get the number of samples mentioned above to approximate $f_{k,S_1}(x)$ accurately, and then average all to get an empirical estimate of $\mathbb{E}_{x \in \{−1, 1\}^{n−k}}[f_{k,S_1}(x)^2]$ that will be accurate with high probability. It can then be shown that:

**Theorem 7.** For a Boolean function $f : \{−1, 1\} \rightarrow \{−1, 1\}$, given a threshold $\theta > 0$ and query access to the inputs, the algorithm from Claim 6 can efficiently (in time $\text{poly}(n, \frac{1}{\theta}, \log(\frac{1}{\delta}))$) and with probability at least $1 − \delta$ return a set of subsets $S_i \subseteq [n], i \in [t]$, for some $t$, with $|\hat{f}(S_i)| \geq \frac{\theta}{2}$ and with the property that any $S \subseteq [n]$ with $|\hat{f}(S)| \geq \theta$ with be among the $t$ returned subsets.

The above result offers a way to efficiently locate all significant Fourier coefficients, under the assumption that most of the Fourier concentration is on a 'small' number of coefficients, and thus extends the standard Low Degree Algorithm.