Contents

Wiki page: `https://simons-institute.github.io/pseudorandomness/groups/modelstructure.html`

1 Big picture

I'll talk about things which have different names in different fields. You probably know them, but don't know they are the same or related ideas that come up in different fields.

	Boosting	Hard-core lemma	Dense model theorem	Weak regularity	?
Area	ML	CC, Derandomization	Additive combinatorics, CC	Graph theory	
Credit	Shapiro, Freund-Schapire	Impagliazzo, Holenstein	Green-Tao, Barak-Shaltiel-Wigderson	Szemeredi, Frieze-Kannan	
Get	Circuit computing f $1-\delta$ of the time	”	Proof that set isn't δ-dense	”	
Unless	Weak learner fails on distribution of density $\Omega(\delta)$	Hard-core distribution	$\Omega(\delta)$-dense “model” indistinguishable from set	A model succinctly describing set	
Algorithm needed	Weak learner	”	Distinguisher	”	

Implications are due to...

1. Boosting $\implies$ Hard-core: Klivans and Servedio.
2. Hard-core $\implies$ Dense model: Impagliazzo
3. Dense model $\implies$ Weak regularity: Trevisan-Tulsiani-Vadhan, Reingold-Trevisan-Tulsiani-Vadhan
4. Weak regularity $\implies$ boosting: Trevisan-Tulsiani-Vadhan

If set looks like might be dense, either succinct reason (proof) not δ-dense, or dense model (reason why it looks big to certain class of sets)

Hard-core lemma implies dense model theorem.

FK simpler argument with better quantitative bounds, weaker conclusion. Generalized to sparse graphs without giveaway cut (cut prove not dense). Either you get a proof that the cut isn't dense, or get a model that succinctly describes all cuts.

Take a bunch of theorems we know are true and showing implications between them. Hope a lot to gain by looking at connections between them:

1. Versatility. Retrofit. Ex. there are many boosting algorithms. When you follow this progression, you get different quantitative and qualitative versions of dense model theorem and regularity.

   Boosting is a particular proof for the min-max theorems.

2. Algorithmic and constructive:

   There are nonconstructive versions using the min-max theorem for boosting, hard-core lemma, dense model theorem.

   I care about algorithmic versions.

   In ML we care about getting a function that computes a function much of the time.

   On the other side, we're really after the distribution where the weak learner fails, the model that succinctly describes the set. We pay attention to do the reductions in an algorithmic, not just an existential way.

3. Dense model theorem $\rightarrow$ learning. Take boosting technique and use it in an unsupervised way.

4. Generality: some things seem to be specific to a setting (density of graphs). Actually weak regularity doesn't have anything to do with graphs being dense. We can relativize it to subgraphs of any graph. You can look at subgraphs of expanders, bipartite graphs, etc. and plug it in the same machineary. If you want to look at spectral norms rather than cuts, you can plug into the same machinery.

Almost all arrows are trivial! Simple lines of high-school algebra.

Weak regularity does imply dense model theorem and hard-core lemma.

1.1 Boosting

First we discuss the PAC learning model.

Let U be the universe—big set of possible inputs.There is a boolean function f and we get points $(x, f(x))$ where $x \sim U$. The algorithm can request any number of such points. Our goal is to find a hypothesis h such that

$$\mathbb{P}_{x\sim U}[h(x) = f(x)] \geq 1 - \delta.$$

How do we get a hypothesis h? We assume we have a weak learner, which receives the same kind of input, but perhaps from some other distribution μ, and gets something correlated with the right answer.

$$\mathbb{P}_{x\sim\mu}[h_i(x) = f(x)] \ge \frac{1}{2} + \varepsilon.$$

(It has to work on any distribution μ satisfying some assumptions.)

How do we use the weak learner? Give a routine that subsamples the distribution U to pass to μ. We let $\mu : U \to [0,1]$. Keep something in U and keeping it with probability in $[0,1]$, else throw it back (rejection sampling). This induces a probability distribution

$$D_\mu(x) = \frac{\mu(x)}{\sum_{x'\in U}\mu(x')}.$$

Let $|\mu| = \mathbb{E}_{x\in U}\,\mu(x)$. Use this as a notion of density. Condition:

1. We will only run the weak learner on distributions of about the same density, $|\mu| \ge \Omega(\delta)$.

   We don't want to run it on something too puny, because the time to simulate the distribution is inversely proportional to the density.

2. Each $h_i \in T$. Ex. T is linear separators.

   $$F_kT = \{f(h_1,\ldots,h_k) : f \in F, h_1,\ldots,h_k \in J\}.$$

   It turns out that $\mu \in F_KT$: it's basically describable in the same class of functions. $h \in F_KT$.

   Pick one large sample, run on it as much as you'd like. Then the issues of what is μ disappears.

   If you just care about ML we don't care about this details, but to go further, we do.

   The first boosting algorithm I'll give is totally ridiculous from the ML point of view. For people who work on weak regularity on graphs this is the natural version and leads to the standard versions of results.

   Suppose you just look at things information-theoretically. If I learn these hypotheses are relevant, how should I use them to predict the function? What's the best function that composes k of these things and gives a predictor for f?

   h_1 is either T or F, etc. This divides into 2^n buckets. Once you are in a bucket what should you do? When given the next x, all you know is which bucket it's in.

   To optimize, you predict the majority. Let p_B be the probability of the majority; then $1 - p_B$ is the minority. Let $H_i(x)$ be the majority answer in B. B is distributed according to $B(x)$. It takes 2^i time to give a good estimate for the majority answer.

Let $\delta_i = \mathbb{E}_{B\in B[x],x\in U}[1-p_B]$. If the majority $\to 1$ in every bucket, then our error probability is small. While it's big, we still need to learn.

On the same biased distribution, get the same h_1. We want to make sure not boolean combination of previous h_i's is a good predictor for the next distribution. If $f(x) = \mathsf{Maj}_B$, then $\mu(x) = \frac{1-p_B}{p_B}$ for $x \in B$.

What is $|\mu|$?

$$\begin{aligned}\mathop{\mathbb{E}}_x |\mu| = \mathop{\mathbb{E}}_x \mu(x) = \mathop{\mathbb{E}}_B [\mathop{\mathbb{E}}_x[\mu(x)|x \in B]] \qquad (1)\\ = \mathbb{E}[1-p_B+1-p_B] = 2\mathop{\mathbb{E}}_B[1-p_B] = 2\delta_i. \qquad (2)\end{aligned}$$

If measure small, then small failure probability. If haven't succeeded, then μ is large enough so we can continue learning on μ.

Given previous functions, define candidate function. Either it has good success, or distribution has good density, and we continue. I need to show this process terminates within a reasonable amount of time.

We'll look at a potential function that measures how close we are to success. Our goal is to drive all p_B's to 1.

Our potential function is

Flip a coin with expected probability.

$$\begin{aligned}\varphi_i = \mathop{\mathbb{E}}_{p_\varphi} p_1(1-p_1) = \mathop{\mathbb{E}}_B p_1 - p_1^2 \qquad (3)\\ = \mathop{\mathbb{E}}_B p_{B,1}(1-p_{B,1}) \qquad (4)\end{aligned}$$

Land in B_1 with probabilty q, B_0 with probabilty $1-q$. $f=1$, $p_{B,1}+\alpha_+$, $p_{B,1}-\alpha_-$. $q\alpha_t = (1-q)\alpha$.

???

$$q(p_{B_1}+\alpha_+)^2 + (1-q)(p_{N,1}-\alpha_-)^2 = (q+1-q)p_{B,1}^2 + (\alpha_t q - (1-q)\alpha - 1)p_{B,1} + q\alpha_t^2 + (1-q)\alpha_{-i}^2. \qquad (5)$$

$$\begin{aligned}\mathop{\mathbb{E}}_x[\mu(x)(-1)^{f(x)\oplus h_{i+1}(x)}] \geq \varepsilon|\mu| \qquad (6)\\ = \mathop{\mathbb{E}}_{B\in B(x)} \mathop{\mathbb{E}}_{x \in B} \mu(x)(-1)^{f(x)\oplus h_{i+1}(x)} \qquad (7)\\ q\left(\frac{1-p_B}{p_B}(p_B+\alpha_t) - (1-p_B+\alpha_-)\right) \qquad (8)\\ (1-q)\left(\frac{1-p_B}{p_B}(1-p_B+\alpha_-) - (p_B+...)\right) \qquad (9)\end{aligned}$$

Prob h on B is q. For $f=1$, $q+\alpha_+$. For $f=-1$, $q-\alpha_-$. Then

$$p\alpha_+ = (1-p)\alpha_-.$$

Then

$$\begin{aligned}\mathop{\mathbb{E}}_\mu \mu(x)(-1)^{f(x)\oplus h(x)} = \varepsilon_B = p\frac{1-p}{p}(q+\alpha_+ - (1-q-\alpha_+))... \qquad (10)\\ (1-q-\alpha_-) - (1-\alpha_-) \qquad (11)\\ (1-q)(2\alpha_+ + 2\alpha_-). \qquad (12)\end{aligned}$$

On the other hand the expectation

$$p_{B,1} = q\left(\frac{p(q+\alpha_+)}{q}\right)^2 - (1-q)\left(\frac{p(1-q-\alpha)}{1-q}\right)^2 \tag{13}$$

$$= p^2\left(q + \frac{2\alpha_+}{q} + \frac{\alpha_+^2}{q}\right)() \tag{14}$$

Try to make an energy increment argument.

It matters that the weak learner only looks at distributions with density $\geq 2\delta$. From boosting to hard-core:

Lemma 1.1. *Let f be any function on U and T any class of tests. Either there exists $H \in F_K T$, $H(x) = f(x)$ with probability $1-\delta$, or there exists μ, $|\mu| \geq 2\delta$, $\forall h \in T$,*

$$\mathbb{P}_{x\in\mu}[h(x) = f(x)] \leq \frac{1}{2} + \varepsilon,$$

$k = O\left(\frac{1}{\varepsilon^2\delta^2}\right)$.

Proof. Let weak learner be exhaustive search. get sequence of distribution, all of $\geq 2\delta$. See if any of them have a good bias. If we always get something with bias $\geq \frac{1}{2}$, continue in the strong learner, until we get $H(x) = f(x)$ with probability $1-\delta$. If at some point our exhaustive search algorithm gets stuck, we have a distribution that's hard core. □

Region where very hard, indistinguishable from random.

Why have to keep track: we want size of hard-core to be what it should be. 2δ is what it should be.

A lot of times we want to compute one or the other. Then we need to use the boosting algorithm. By using the boosting algorithm we get something stronger. We get that the hardcore distribution doesn't have unlimited complexity. It's describable by the function we want to compute, $\mu \in F_{k+1}(J \cup \{f\})$.

Let $S \subseteq U$. We have some tests T that don't reveal that it's small. If big, then δ fraction of whole thing.

Definition 1.2: Let $S \subseteq U$. Let $\mathcal{T}$ be a class of tests. For $T \in \mathcal{T}$, $\mathbb{E}_{x\in S} T(x)$. S is (ε, δ)-pseudo-dense for $\mathcal{T}$ if for all $T \in J$, $T(U) \geq \delta T(S) - \varepsilon$. This is written as $S \sim_{\varepsilon,\delta} U$.

One way to be pseudo-dense it to be dense. Another, one step removed, is that there's R that's indistinguishable from the whole distribution, and it's dense in R. and $T(U)$ and $T(R)$ are within δ fraction of each other.

Theorem 1.3 (Dense model theorem)**.** *There exists μ, $|\mu| \geq \delta$ and $S \sim_{\varepsilon,,\delta} D_\mu$.*

If S is (ε, δ)-pseudodense vs $F_k\mathcal{T}$, $k = O\left(\frac{1}{\varepsilon^2\delta^2}\right)$ then there exists μ, $\mu \in F_k\mathcal{T}$, $|\mu| \geq \delta - \varepsilon$, $D_\mu \sim_{\mathcal{T}} S$ to within $O(\varepsilon/\delta)$.

By going through boosting and hard-core, we get characterization of μ, describable using original tests.

Hard function should be membership in S. This happens almost never, so we have to blow up membership to make it comparable to what it should be.

Proof. Let U' be: δ' fraction is $(0,x), x \in S$, while $1-\delta'$ is $(1,x), x \in U$. Let $\delta' = \frac{\delta}{1+\delta}$. Let $f(b,x) = b$, $T((b,x)) = f(b,x)$ with probability $1-\delta'$.

The tests are just applied to the x part.

$$\begin{aligned} \delta' T(S) + (1-\delta')(1-T(U)) &= 1-\delta' + \delta'(T(S)) - (1-\delta')T(U) && (15) \\ &= 1-\delta' + \frac{1}{1+\delta}(\delta T(S) - T(U)) \le 1-\delta'+\varepsilon. && (16) \end{aligned}$$

This distribution has a hardcore subdistribution on which is as hard as a random coin flip. With δ' probability., sample from S, $f=1$. with $1-\delta'$ probability, sample from U, $f=0$. The subset has size $2\delta'$, in order to be hard, half comes from S (all of it), half from U. f hard to predict means the 2 sides indistinguishable.

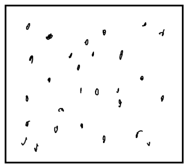
The model depends on k and f. Magically when we apply it we only want the part of the model where $f=0$. It's definable with k things from $\mathcal{T}$. No membership oracle required. □

2

I'll start with motivation: one thing we could hope to get from this.

1. Let X be a set, e.g. a distribution of points in the square.

   Let S be some distribution on points in X.

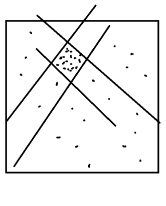
   

   Let $\mathcal{T}$ be a set of classifiers, ex. a set of half-planes.

   Let $\mathcal{F}_K\mathcal{T}$ be boolean functions on K functions in $\mathcal{T}$; here, partitions into polygonal regions by k half-planes.

   We want to pre-process the distribution to be able to answer queries in $\mathcal{F}_K\mathcal{T}$.

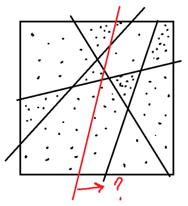
2. A violation of pseudo-density in this setting means there is a polygonal region with many more points from S than its volume, a "hot spot".

$$\text{Area}\,(\text{region}) < \delta \mathbb{P}_S\,(\text{region}) - \varepsilon.$$

3. A model is a partition into polygonal regions, with a probability distribution on regions. A simple model is defined by at most k lines.

   The property of a model is that we can estimate half-space probabilities ("given any half-space, what proportion of points of S are on one side of it?") by treating the points as if uniform within regions.



4. The algorithmic requirement in order to process the points to answer queries in $\mathcal{F}_K\mathcal{T}$ is: given a set of points sampled from S, and a set of points sampled from U, find a half-space that approximately maximizes the difference in probabilities for these two sets. The equivalent in boosting is a distinguishing algorithm.

Setting	Boosting	Hard-core measure	DMT/transference principle	Weak regularity
	WL: $\|\mu_i\| \geq 2\delta$, $\mu_i = g(h_1, \ldots, h_i, f)$, $h_{i+1} \in \mathcal{T}$, k iterations	Hardcore measure: $\mu_k = g(h_1, \ldots, h_k, f)$, $\|\mu_k\| \geq 2\delta$	Model: $\mu_k = g(h_1, \ldots, h_k, o)$, $\|\mu_k\| \geq \delta$	
	SL: $H = G(h_1, \ldots, h_k)$, $\mathbb{P}[H = f] \geq 1 - \delta$	Violation of hardness: $H = G(h_1, \ldots, h_k)$, $\mathbb{P}[H = f] > 1 - \delta$	Violation of pseudo-density $H = G(h_1, \ldots, h_k)$, $H(U) \leq \delta H(S) - \varepsilon$	
Assumption	WL never fails	Violation is impossible	Violation of pseudo-density is impossible	Actually dense
Conclusion	SL works	Hard-core measure exists, with same k, G, g	Model exists	Model exists
Algorithmic	Weak learner requirement	Approximately optimal weak learner	Approximately optimal distinguisher	

Weak regularity: Same as above
Assumptions: Actually dense
Conclusion: model exists
Some comments:

1. Boosting: Note it's important that the δ here is the same; many boosting algorithms meet this criterion.

   The theorem says that "either weak learner fails or strong learner works."

   In boosting, we think of weak learner as never failing.

2. Hard-core measure lemma: The lemma says that either we can find hard-core measure, on which no function can compute the function f more than $\frac{1}{2} + \delta$ of time; or find a function computing f $1 - \delta$ of the time.

   Here, we want to come up with the measure. Although the logical format is the same as boosting, here we assume that the violations never happen (there is no strong learner).

   Every boosting algorithm gives hard-core measure lemma with the same parameters, and with exactly the same way of "gluing" the functions.

   Sometime you care about computational complexity of G but not of g, or vice versa.

3. We can convert the hard-core measure theorem into the dense model theorem/transference principle (Tao and Ziegler).

   Here, we have a distribution we're trying to model. Either the distribution has pseudo-density property— there isn't a violation that's definable from k different properties from hypothesis class, where violation means that the expected value is much smaller on U than on S—or we get a model of density $\geq \delta$.

   Assuming that violation of pseudo-density does not happen, we get a model.

4. Weak regularity is just DMT except the distribution actually is dense. It's not so interesting that it has a dense model.

   What we get is that the dense model you get is simple, definable in terms of a small number of basic hypotheses.

   Sometimes we care about simplicity in the model, and sometimes simplicity in G.

5. Note the k is the same throughout. Reductions preserves k, and the functions h_i, G.

   We don't only have the fact that boosting implies hard-core lemma implies regularity lemma. We have the stronger result that whatever boosting algorithm you give me, I get a hard-core lemma and regularity lemma with the same parameters and algorithm. Thus we can pick the boosting algorithm that gives the best results for our application.

Boosting with oblivious decision trees.

Let

- f be function we want to learn, $f : \chi \to \{0, 1\}$.
- $\mathcal{T}$ class of Boolean functions we use as basic hypotheses.
- $F_k\mathcal{T}$ compositions of $\leq k$ functions in $\mathcal{T}$.

Measure $\mu : X \to [0, 1]$, $|\mu| = \mathbb{E}_{x \in X}\, \mu(x)$, $D_\mu(x) = \frac{\mu(x)}{|\mu||X|}$ induced distribution: pick x, keep with probability $\mu(x)$, else repeat.

- Weak learner: If $\mu \in F_k\mathcal{T}$ and $|\mu| \geq 2\delta$, whp returns $h \in \mathcal{J}$ so that $\mathbb{P}_{x\sim D_\mu}[h(x) = f(x)] \geq \frac{1}{2} + \varepsilon$.
- Strong learner: whp, finds $H \in F_k\mathcal{T}$, $\mathbb{P}_{x\in X}(H(x) = f(x)) \geq 1 - \delta$.

Let μ_0 be constant 1.
While $|\mu_i| \geq 2\delta$ do

- $h_{i+1} \hookleftarrow WL(\mu_i)$
- Partition X according to values of $h_1, \ldots, h_i$ into "blocks" B, Maj_B most likely value in δ.
- $H_{i+1} =$ output $\mathsf{Maj}_{B(x)}$,

$$\mu_{i+1} = \begin{cases} \frac{1-p_{\mathsf{Maj}}}{p_{\mathsf{Maj}}}, & \text{if } f(x) = \mathsf{Maj} \\ 1, & \text{otherwise} \end{cases}$$

cost exponential in k in most circumstances. under what circumstances not exp in k? If VC dimension small.

number of boolean comb. Grows at most order k^d.

Induce new distribution, make f unbiased within each block.

Total probability: on $1 - p_{\mathsf{Maj}}$, $f = 1$, make p_{Maj}. $f = 0$, $1 - p_{\mathsf{Maj}}$.

(Last time: if $|\mu_i| \leq 2\delta$, $\mathbb{P}_{x\in X}[H_i = f] \geq 1 - \delta$.)

We need to show this method terminates.

Consider the potential function

$$\varphi = \mathbb{E}_{x\in_U X, B\leftarrow B(x)}[(\mathbb{P}[f = 1|B])^2]$$

Closer to 1, closer it is to boolean. This is maximized if every bucket/block is either constantly 1 or 0.

We need to show every iteration increases this potential function.

WTS: In every iteration φ increases by $\Omega((\varepsilon\delta)^2)$.

Fix B.

$$\begin{aligned}
p &= \mathbb{P}[f = 1|B] && (17)\\
q &= \mathbb{P}[h_{i+1} = 1|B] && (18)\\
q + \alpha_+ &= \mathbb{P}[h_{i+1} = 1|B, f = 1] && (19)\\
q - \alpha_- &= \mathbb{P}[h_{i+1} = 1|B, f = 0] && (20)\\
\alpha_+ p &= \alpha_-(1-p) \text{ by conservation} && (21)\\
B &\hookleftarrow \alpha_-(1-p) && (22)\\
p_0 &= \mathbb{P}[f = 1|B_0] = \frac{\mathbb{P}[f = 1 \wedge h = 0]}{\mathbb{P}[h = 0]} = \frac{p(1-q-\alpha_+)}{1-q} && (23)\\
p_1 &= \mathbb{P}[f = 1|B_1] = \frac{\mathbb{P}[f = 1 \wedge h = 1]}{\mathbb{P}[h = 1]} = \frac{p(q+\alpha_+)}{q} && (24)\\
(\text{Value of potential}) &= qp_1^2 + (1-q)p_0^2 = p^2\left(\frac{(q+\alpha_+)^2}{q} + \frac{(1-q-\alpha_+)^2}{1-q}\right) && (25)\\
&= p^2\left(\left(q + 2\alpha_+ + \frac{\alpha_+^2}{q}\right) + \left(1 - q - 2\alpha_+ + \frac{\alpha_+^2}{1-q}\right)\right) && (26)\\
&= p^2\left(1 + \frac{\alpha_+^2}{q} + \frac{\alpha_+^2}{1-q}\right) && (27)\\
&\geq p^2 + 4p^2\alpha_+^2 \geq \alpha_+^2 && (28)
\end{aligned}$$

if $\mathsf{Maj} = 1$.

Assume $\mathsf{Maj}_B = 1$.

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\in B}[\mu(x)((-1)^{(h(x)\neq f(x))})] = \quad & (f=1)p\left(\frac{1-p}{p}\right)[(q+\alpha_+) - (1-q-\alpha_+)] && (29)\\
& + (f=0)(1-p)1[1-(1-\alpha_-) - (q-\alpha_-)] && (30)\\
&= (1-p)[\cancel{2\eta - 1 + 1 - 2\eta} + 2\alpha_+ + 2\alpha_-] + 2\alpha_+(1-p) + 2\alpha_+ p = 2\alpha_+ && (31)
\end{aligned}$$

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\in B} \cdots &= \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{B=B(x)} \mu(x)(-1)^{(f(x)\neq h_{i+1}(x))} && (32)\\
&= \mathop{\mathbb{E}}_{B} 2\alpha_+ \geq \varepsilon|\mu| = 2\delta\varepsilon && (33)\\
\Delta\varphi &= \mathop{\mathbb{E}}_{B} \alpha_t^2 \geq (\delta\varepsilon)^2. && (34)
\end{aligned}$$

Every step don't terminate, potential function increases by $(\delta\varepsilon)^2$. It's a number in $[0,1]$, so the number of iterations is at most $k \leq O\left(\frac{1}{(\delta\varepsilon)^2}\right)$. At most polynomial in parameters.

All you get from this is a decision tree. Brute force. Complexity is exponential in k. This is a bug, not a feature.

Can always convert sign into decision tree. On hard-core, DMT give same thing. In-complexity terms, we don't get good hard-core, because circuit size 2^k circuit size. Better boosting algorithm makes G have smaller complexity. If stopping point is HCL, not boosting you want. In dense model theorem, fine because all you care about is size of k not complexity of G.

This potential function matches this boosting algorithm. Other boosting algorithms you can analyze with other potential functions.

Why I want to do this way with this potential function:

- Like potential function used most in graph theory. Property: you can't make negative progress, you always go forwards.
- Translated get usual Sz, except don't get equal size partitions.

We have different batches/blocks that we're partitioning to. Suppose we get stuck: no function that correlates globally, but there are many blocks where we can find functions that correlate with the function inside that block. If ε fraction of blocks find functions that correlate, query them all. That's going to be our next round.

2^k functions correlating. Now 2^{2^k} buckets, but increased potential function by poly in terms of ε, δ. This is a familiar argument, we can only go $\frac{1}{\varepsilon}$ iterations before we terminate.

We've increased everything a tower depending on ε. ε had better be a big constant. We have a hard-core defined in terms of k things. Hard-core looks like: partition. $1-\delta$ fraction where have answer correct. Boolean combinations of k things you've learned. Except for ε fraction, each part: no better method than predict constant function.

Terminate: on most blocks we can't beat majority answer.

Bounds: you need a tower. No avoidance of tower in some contexts. Graphs are not the same context.

strong hardcore, or strong dense model. I don't know the formulation let alone the proof. I have the proof I need the formuation.

Take any boolean function. Any class of tests. Partition, $\forall \varepsilon$ exists k, partition into k, such that in almost all partitions you can't do better than constant function in predicting it.

Fix set of vertices V of set n. Let U be edges in complete graph on V. (Interesting if U is not the complete graph, ex. U is the edges in d-regular expander on V.)

Let τ be the set of cuts defined by $A, B \subseteq V$. WLOG $A \cap B = \phi$.

Directed edge is $e = \{a, b\}$. $T(e) = 1$ iff $a \in A$ and $b \in B$. Apply this chain to this class of tests.

What does $F_K T$ look like? You have a bunch of sets A, B and you're defined by which sets they go between. Or think of it as: for every test you have 3 regions: those in A, B, neither. When you look at all of k things, split vertices into 3^k regions. A little overkill, but all the tests, can split into 3^k. Knowing region of endpoints tells you which subset of tests are true.

This looks not quite quivalent. Here test sets are combinatorial rectangles. Sz: partition of left and right such that every rectangle works. That's a more structured version that what you want? One gives you the other?

If you use this kind of boosting, you don't get much better than this. If you use other types, you get more structured.

Plugging into the generic regularity lemma, $|E| \geq \delta\binom{n}{2}$. This says there exists $\mu = G(T_1, \ldots, T_k)$, where $k = O\left(\frac{1}{\varepsilon^2\delta^2}\right)$ good predictor for any cut in the graph. This is the weak regularity of Kannan-Frieze. For Szemeredi we need the stronger boosting lemma.

Use the T's to divide vertices into 3^k subsets such that μ is a constant on every pair of subsets. Which sets you're in dictate which is true.

3^k. If you want to know how many edges between A and B, this is

$$\frac{E_G(A,B)}{|E_G|} \approx_\varepsilon \sum_{i,j} \mu_{ij} \frac{|A \cap A_i||B \cap B_i|}{|V|^2}.$$

Constantly much information. You can't recover original set of edges, but can approximate any cut by knowing how big the intersections are relatively.

If we do this with G a subset of an expander, we get the same thing except

$$\frac{E_G(A,B)}{|E_{G'}|} =$$

Use expander mixing, with error term of expander mixing, get same thing.

3 Improved boosting algorithms and implications

Based on Holenstein, simplified.

We had a generic reduction from boosting algorithms to dense model theorems to regularity lemmas. Today I'll discuss how to get different intermediate results from boosting algorithms. I'll sample the variety of boosting results and see what kind of dense model theorems/regularity lemmas they give. I'll talk about recursive uses of dense model theorems to give decompositions.

I'll start with the standard algorithms and then use a modification from Holenstein necessary for the dense model theorem. The modification is fairly generic; it's about one parameterthat is not important for boosting but important for dense model theorems.

A weak learner is given μ_t and produces h_t that has some advantage on the subdistribution (normalize by expectation of the weight μ_i to get the probability) [1]

$$\mathbb{E}_{x\sim_D U}(-1)^{h_t(x)=f(x)}\mu_t(x) \geq \varepsilon|\mu_i| \tag{35}$$

$$\iff \mathbb{P}_{x\sim D_{\mu_i}}[h_t(x)=f(x)] \geq \frac{1}{2}+\varepsilon \tag{36}$$

Here

$$|\mu_t| = \mathbb{E}_x|\mu_t(x)| \tag{37}$$

$$N_t(x) = \sum_{t'=1}^{t}(-1)^{h_{t'}(x)=f(x)} \tag{38}$$

$$I = N_t(x). \tag{39}$$

If I is negative (we're not doing so well), we weight x more; the better we're doing, the less important it is to continue doing well on the data point x.

[1] I think of U as uniform but it doesn't matter.

M_I is a function that starts with 1, and goes to 0 gradually, for example, linear.

$$M_I = \begin{cases} 1, & \text{if } I \le 0 \\ 1-\gamma I, & 0 \le I \le \frac{1}{\gamma} \\ 0, & \text{if } I \ge \frac{1}{\gamma} \end{cases} \tag{40}$$

$$\gamma = \frac{1}{4}\varepsilon\delta. \tag{41}$$

AdaBoost works better:

$$M_I = \begin{cases} 1, & \text{if } I \le 0 \\ (1+\gamma)^{-I}, & I > 0 \end{cases} \tag{42}$$

$$\gamma = \frac{1}{4t}. \tag{43}$$

We need γ small because we need the slope to be small relative to the value.

Let $A_I = \sum_{J \ge I} M_J$.

$$A_t = \mathop{\mathbb{E}}_{x \sim U} A_{N_t(x)}$$

is our generic potential function.

$$\Delta_I = M_{I-1} - M_I \tag{44}$$

$$\Delta_t = \mathop{\mathbb{E}}_{x \sim U} \Delta_{N_t(x)} \tag{45}$$

How we picked $1+\gamma$: to make slope proportional to value with small constant of proportionality.

$$\Delta_I = \begin{cases} 0 & \text{if } I \le 0 \\ \gamma & \text{if } 0 \le I \le \frac{1}{\gamma} \\ 0 & \text{if } I > \frac{1}{\gamma} \end{cases} \tag{46}$$

$$\Delta_I = \begin{cases} 0 & \text{if } I < 0 \\ \gamma M_I & \text{if } I > 0. \end{cases} \tag{47}$$

"Move things from the negative zone to the safe region." Potential function is how many credit we could get ourselves in the future. if nothing to give credit left to, done.

Algorithm 3.1 (Basic boosting)**:**

- $\mu_0 = 1$
- Repeat until $|\mu_t| \le 2\delta$.
  - $h_{t+1} \hookleftarrow WL(\mu_t)$
  - Increase when right, decrease when wrong: $w_{t+1}(x) = M_{N_{t+1}(x)}$. ($N_t(x)$ increases or decreases by 1 depending on whether it is right at t.)
- Return $H \in \mathsf{Maj}(h_1, \ldots, h_t)$, $1 - 2\delta$.

We'll modify this later to get $1-\delta$. ($1-2\delta$ is not good enough because of the 2. For learning purposes, you just replace δ by $\frac{\delta}{2}$, but with the 2, it doesn't imply the hard-core set lemma or dense model theorem.)

$$N_t(x) = \sum_{t'=1}^{t} (-1)^{h_{t'}(x)=f(x)} \tag{48}$$

The average value is at most 2δ. Everything where majority incorrect has μ-value 1.

We had

$$\mathop{\mathbb{E}}_{x\sim U} (-1)^{h_{t+1}(x)=f(x)}\mu_t(x) \geq \varepsilon|\mu_t|. \tag{49}$$

Think of this as saying: We expect most of the time to be taking a step forward rather than a step back.

If $f(x) = h_{t+1}(x)$, then

$$A_{N_{t+1}(x)} = A_{N_t(x)} - \mu_t(x) \tag{50}$$

If $f(x) \neq h_{t+1}(x)$, then

$$\begin{aligned} A_{N_{t+1}(x)} &= A_{N_t(x)} + M_{N_t(x)-1} && (51)\\ &= A_{N_t(x)} + M_{N_t(x)} + M_{N_t(x)} + \Delta_{N_t(x)} && (52)\end{aligned}$$

For the two choices of M_I, $\Delta_{N_t(x)}$ is bounded by γ, $\gamma M_{N_t(x)}$, respectively.

We get

$$\Delta A \leq -\varepsilon|\mu_t| + \mathbb{E}_x \Delta_{N_t(x)} \tag{53}$$

$$\Delta A \leq \begin{cases} -\varepsilon|\mu_t| + \gamma \\ -\varepsilon|\mu_t| + \gamma|\mu_t| \end{cases} \tag{54}$$

Using either example, we know a bound on Δ. In example, 1, decrease of $2\delta\varepsilon$, so setting for $\geq$, for example 2, $\delta \ll \varepsilon$ to make progress.

In the two cases

$$\begin{aligned} A_0 &= \frac{1}{2\gamma} && t \leq O\left(\frac{1}{\varepsilon^2\delta^2}\right) && (55)\\ A_0 &= \frac{1+\gamma}{\gamma} && t \leq O\left(\frac{1}{\varepsilon^2\delta}\right). && (56)\end{aligned}$$

How to get rid of the factor of 2? Have a moving target for where we draw the line. As time goes on, it's harder to be in the safe zone; move things to the danger zone.

Algorithm 3.2:

- $\mu_0 = 1$, $s = 0$.
- Repeat until $|A_t| < 2\delta s$.
  - $h_{t+1} \hookleftarrow WL(\mu_t)$

  - Increase when right, decrease when wrong: $\mu_{t+1}(x) = M_{N_{t+1}(x)-s}$.
  - If $|\mu_{t+1}| < 2\delta$, $s \hookleftarrow s+1$. (Move everything 1 step to the left.)
  - $\mu_{t+1}(x) M_{N_{t+1}} - s$.
  - Note that the invariant $\mu_t > 2\delta$ is maintained in this loop.

- Return $H_t(x)$: if $\left|\sum_{i=1}^t (-1)^{h_i}\right| \geq s$ use $\mathsf{Maj}(h_1(x), \ldots, h_t(x))$ (if there are s more 1's than 0, say 1; if there are s more 0's than 1, say 0); otherwise, flip a coin with bias $\frac{1}{2} + \frac{\sum_{i=1}^t (-1)^{h_i(s)}}{2s}$ and output 1 if heads. (Use preponderance of 0's and 1's as a coin flip.)

We saw that the change in A due to updating N_{t+1} is

$$\Delta A = \varepsilon|\mu| - \mathbb{E}\Delta_t.$$

When s is set to $s+1$, the change in A is

$$\Delta A = |\mu| + \mathbb{E}\Delta_t \tag{57}$$
$$\Delta A = -\varepsilon|\mu| + |\mu| + 2\mathbb{E}\Delta_t \tag{58}$$
$$\leq 2\delta - \Omega(\varepsilon\delta) \tag{59}$$
$$\Delta(A - 2\delta s) < -\Omega(\varepsilon\delta). \tag{60}$$

This is the same decrease as before, so the bound on the step is the same as before.

Why is this good? Holenstein uses a rounding trick, also in TTB.

$$A_t(x) \geq s_t - N_t(x) + A_0, \qquad \text{for } N_t(x) < s \tag{61}$$

Suppose that $N_t(x) \geq s$. Then we're always correct. Otherwise, we're incorrect with probability at most $\frac{s-N_t(x)}{2s}$. Thus

$$\mathbb{E}\,(\text{fraction incorrect}) = \frac{1}{2s}\mathbb{E}[s - N_t(x)] \tag{62}$$
$$\leq \mathbb{E}\frac{A_t(x)}{2s} = \frac{2\delta_S}{2s} = \delta. \tag{63}$$

Is this the same as: Pick a threshold at random and use that threshold?

What did we get? We gave 2 algorithms with $k = O\left(\frac{1}{\varepsilon^2\delta^2}\right)$ and $k = O\left(\frac{1}{\varepsilon^2\delta}\right)$. Before, we had to allow any boolean function on k variables, a large class; here, we take a majority function. This is in any complexity class that can compute majority.

$$H = g(h_1, \ldots, h_k) = \text{sign}(\sum_i h_i - s) \tag{64}$$
$$\mu = G(h_1, \ldots, h_k, f) = G'(\sum (-1)^{h_i(x)=f(x)}). \tag{65}$$

Both are much simpler than what we had before.

What kind of results do we get from this? Having g be in a small complexity class is important for using the hard-core set lemma for moderate complexity classes like TC^0. I'll go all the way to regularity and see what we get.

Before, we got a model of a dense graph $|E_G| \geq \delta|V|^2$. We have a model that looks like a partition into 3^k sets. Think of this as approximating the adjacency matrix of a graph by a matrix with big blocks of constants, $A_G = M$, $\text{rank}(M) = 3^k = 3^{O\left(\frac{1}{\varepsilon^2\delta^2}\right)}$. We can replace δ^2 by δ. We get a bunch of cuts in a graph, not necessarily disjoint, just $\text{poly}\left(\frac{1}{\varepsilon^2\delta^2}\right)$. Weigh edges by how many of the cuts they're in. Each cut is a rank 1 matrix. If you want to know the number of cuts you're in, write as a matrix of rank k rather than 3^k. The one thing we have to do: that's the crude score, we're actually interested in a piecewise linear function or exponentially decreasing function of that score. Componentwise we have to apply this function to M.

Frieze and Kannan look at a spectral approximation. Instead of boolean tests, they use tests defined by prob distributions. $b^T A_G a$. They get a similar result without a truncation operation.

Different boosting have different combining functions which give different structures of model which give qualitatively and quantitatively different regularity lemmas.