My favorite open problems (and a few results)

Avrim Blum

Carnegie Mellon University

[Talk given at 2001 NeuroCOLT Alpine Workshop on Computational Complexity Aspects of Learning]

Big picture

"What do you see as the most basic open questions in computational learning?"

Here are my answers.

Focus on standard model:

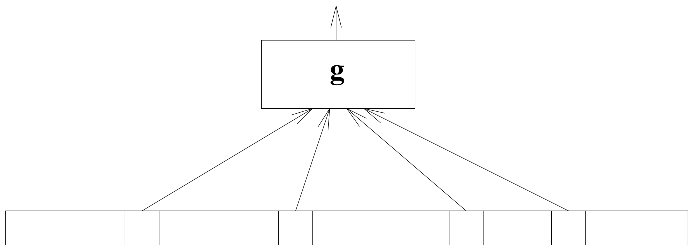
- Learning a boolean function from examples.
- Goal is polynomial time in relevant parameters.

Also, no membership queries.

Problem 1

- Examples in $\{0,1\}^n$. Uniform distribution.
- But, only r of these n bits are relevant.
- Target is arbitrary function of these r.

(Think of $r = \log n$ or $r = \log\log n$)



$f(x) = g(h(x))$. g is a truth-table with 2^r entries.

Q1: Learn in poly time for $r = O(\log n)$?

Note: this is special case of "Are DNF learnable over uniform dist?" and "Are decision trees learnable over uniform dist?"

Problem 1, contd.

- Examples in $\{0,1\}^n$. Uniform distribution.
- But, only r of these n bits are relevant.
- $f(x) = g(h(x))$. g is arbitrary truth table.

Q2: How about $r = \log\log n$?

In this case, there are only $2^{2^{\log\log n}}$ possible truth tables.

Q3: How about running time of $n^{o(r)}$?

Q4: How about just $n^{r/2}$?

(use Fourier spectrum somehow?)

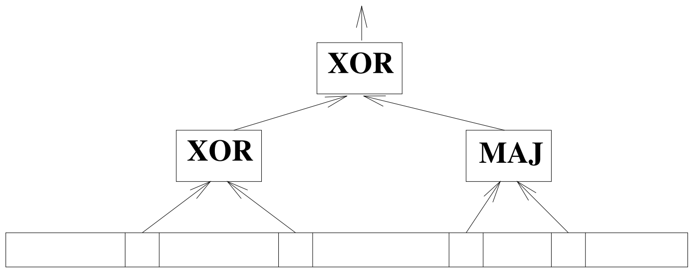
Problem 1, contd, contd

Some facts:

- Easy with MQs.
- If $\Pr_x(f(x) = 1) = \frac{1}{2}$ then weak implies strong.

Can assume relevant bits picked at random, but what do the hard truth-tables look like?

- Random truth tables are easy w.h.p. (Single-variable correlations)
- Example of hard function: pick sets A and B of size $r/2$ each, at random. Do $par(A) \oplus maj(B)$.



But can do these in time $n^{r/4}$.

Note: you can create these yourself, so can use to create hard cases of DNF, DTs.

Segue to Problem 2

Intuition for parity/majority function: majority acts like noise applied to parity. And, parity with noise seems hard.

Next topic: what can be learned in presence of noise? and specific open problems from [BKW00].

Learning with noise

- Labels flipped iid with prob $\eta < 1/2$.
- [Kearns93]: Learnable by Statistical Queries implies Learnable with Noise.
- [BFJKMR94]: Learnable by SQs iff only poly many (nearly) uncorrelated functions.

Question: is SQ-learnable the *same* as learnable with noise?

- [BKW00]: Can learn class of parity functions that depend on only first $k = O(\log n \log\log n)$ bits in the presence of noise.

  Not SQ-learnable since $O(n^{\log\log n})$ uncorrelated functions.

Open Questions

(Let's do open questions first, then do the result)

[BKW00] learns parity functions over first $k = O(\log n \log\log n)$ bits in presence of noise. Equiv: learns general parity with noise in time $2^{O(n/\log n)}$.

Open questions:

- Basic technique could extend to $k = O(\log^2 n)$ [or, $2^{O(\sqrt{n})}$ time for general case] if can solve certain apx problem better.
- Same time but poly samples?
- Dependence on η: Separate SQ from Learnable-with-noise for $\eta = 1/2 - 1/n$? Relates to open question: "can noise-tolerance be boosted?"

Learning Parity functions

- Parity over first $k = O(\log n \log\log n)$ bits.

$$\begin{array}{ll} 10110101 & + \\ 01101011 & - \\ 10011001 & - \\ 11100110 & + \\ & \\ 00000001 & ? \end{array}$$

- Gaussian elimination for zero noise: write $0\ldots01$ as sum of training examples. Then last bit of target vector is the sum of their labels.

- But, if noise rate $1/2 - \epsilon/2$ then adding t examples gives noise rate $1/2 - \epsilon^t/2$.

  $t = 2: \quad 2(\frac{1}{2} - \frac{\epsilon}{2})(\frac{1}{2} + \frac{\epsilon}{2}) = \frac{1}{2} - \frac{\epsilon^2}{2}.$

- So, can add only $O(\log n)$ examples if we want noticeable bias.

Problem: Gaussian Elimination uses k examples.

contd

- Parity over first $k = O(\log n \log\log n)$ bits.

$$\begin{array}{ll} 10110101 & + \\ 01101011 & - \\ 10011001 & - \\ 11100110 & + \\ & \\ 00000001 & ? \end{array}$$

Idea (think of $\eta = 1/4$):

- Gaussian elimination needs $O(k)$ examples to write $0 \ldots 01$ as sum of k training examples. Get bias $1/2^k$.
- Alternatively, could just draw examples until see $0 \ldots 01$. But this takes 2^k examples.
- Can we do something in-between?

Intuition

What can we say information-theoretically?

- $O(n)$ examples of width $k = O(\log n \log\log n)$.

$$\begin{array}{ll} 10110101 & + \\ 01101011 & - \\ 10011001 & - \\ 11100110 & + \\ 00000001 & ? \end{array}$$

- $\binom{cn}{t}$ subsets of t examples. Pairwise indep.
- Expect one of these will sum to $0\ldots01$ so long as $n^t \approx 2^k$, i.e., $t \approx k/\log n$.

So...

- If we could *find* smallest subset t, we could learn for $k = O(\log^2 n)$.
- We'll find subset of size 2^t.

Basic idea, contd

Goal: Learn for $k = O(\log n \log\log n)$.

- View examples as strings of length $\lg\lg n$ over a $2^{O(\log n)}$-letter alphabet.
- Do "Gaussian Elimination" over this alphabet.

A	.	.	.	.	.	(1 example)
B	.	.	.	.	.	(1 example)
Z	.	.	.	.	.	(1 example)
0	A	.	.	.	.	(2 examples)
0	B	.	.	.	.	(2 examples)
0	0	A	.	.	.	(4 examples)
0	0	B	.	.	.	(4 examples)
0	0	0	A	.	.	(8 examples)

Technical statement: Learn length-k parity, for $k = ab$ in time, examples $poly((\frac{1}{1-2\eta})^{2^a}, 2^b)$.

Another way to view the apx problem

Here's something that seems really hard:

> Given a set of random examples labeled by a parity function, find the approximately shortest consistent parity. Similar to: can parity functions be learned attribute-efficiently?

Here's what we need:

> Given a set of random examples with labels chosen independently from the examples, find the approximately shortest consistent parity function.

How about one more, just for fun?

Decision lists over string-valued attributes

E.g., If $x_2 =$ "sunny" then $+$, else if $x_5 =$ "cold" then $-$, else $+$.

- n attributes.
- DL of length ℓ. (ℓ could be larger than n)
- Each attribute can take on arbitrary n-bit string as value.

Goal: Efficient alg with mistake bound $poly(n, \ell)$.

Note: for boolean features, standard algorithm is $O(n\ell)$ and WinnowII is $O(2^{2\ell} \log n)$. If could get $\ell^a n^b$ for $a > 0, b < 1$, that would solve the problem.

Application of computational learning theory to other areas...

Splay trees [Sleator,Tarjan]

- Binary search trees are good ways of storing data for lookup.

```
        H
      /   \
    C       P
           / \
          M   Z
```

- Can rotate to move data, preserving order.

```
        H                          P
      /   \                      /   \
    C       P      <===>       H       Z
           / \                / \
          M   Z              C   M
```

- Splay trees: self-adjusting bst. Rotates items accesssed towards the top.

  Satisfies *static optimality* property: on any sequence of requests, cost is within $O(1)$ of optimal static tree in hindsight.

Adaptive search trees contd

Model: we pay 1 unit per pointer followed. We pay 1 unit per rotation.

What if we don't care about computation time in deciding which rotation to do?

Then can get static optimality with expert-advice algorithm:

- One expert per bst. (Roughly 2^n)
- Access causes each expert to incur loss based on depth.
- Movement cost between experts at most $4n$.

Total cost $(1+\epsilon)$OPT $+O(n^2/\epsilon)$.

Can this help at all for dynamic optimality conjecture?

Other directions

Use simple-but-hard learning problems for human-doable challenge-response authentication?

Experiments with coke machine....

Questions like:

- How to present examples to user?
- Can you get security parameter k more simply than k repetitions?