Presenting: 'Cryptographic primitives based on hard learning problems: Blum, Furst, Kearns, Lipton'

Vishvas Vasuki

April 19, 2009

Outline

What to look out for?

What to look out for?

- A new definition for hardness of learning.

What to look out for?

- A new definition for hardness of learning.
- A pseudorandom bit generator using hard to learn class of functions.

Outline

The binary classification problem

The binary classification problem

- Teach the computer difference between male face and female face by giving examples.

The binary classification problem

- Teach the computer difference between male face and female face by giving examples.

- Ask computer: Is  a female face?

The binary classification problem

- Teach the computer difference between male face and female face by giving examples.

- Ask computer: Is 

a female face?
- Computer wins if it succeeds with good probability.

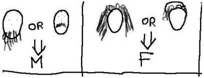
The binary classification problem formalized

The binary classification problem formalized

- Known set of n features. Eg: Hairstyle, facial hair, moustache present. 

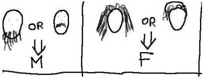
The binary classification problem formalized

- Known set of n features. Eg: Hairstyle, facial hair, moustache present. 

- Unknown classification function (concept) $c : \{0, 1\}^n \rightarrow \{M, F\}$. Eg: 

The binary classification problem formalized

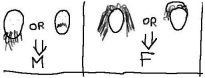
- Known set of n features. Eg: Hairstyle, facial hair, moustache present. 

- Unknown classification function (concept) $c : \{0,1\}^n \rightarrow \{M, F\}$. Eg: 

- c belongs to a known set of functions, C_n. Eg: Polynomial sized DNF over n variables, Halfspaces etc..

The binary classification problem formalized

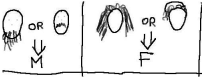
- Known set of n features. Eg: Hairstyle, facial hair, moustache present. 

- Unknown classification function (concept) $c : \{0,1\}^n \rightarrow \{M, F\}$. Eg: 

- c belongs to a known set of functions, C_n. Eg: Polynomial sized DNF over n variables, Halfspaces etc..
- See m(n) examples: $\{(s_1, c(s_1)), (s_2, c(s_2))..\} = (S, c(S))$.

The binary classification problem formalized

- Known set of n features. Eg: Hairstyle, facial hair, moustache present. 

- Unknown classification function (concept) $c : \{0,1\}^n \rightarrow \{M, F\}$. Eg: 

- c belongs to a known set of functions, C_n. Eg: Polynomial sized DNF over n variables, Halfspaces etc..
- See m(n) examples: $\{(s_1, c(s_1)), (s_2, c(s_2))..\} = (S, c(S))$.
- Now, classify test set: $\{s'_1, s'_2, ..\}$. 

Some notation

Some notation

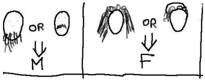
- C_n: a set of classifiction functions $\{0,1\}^n \rightarrow \{0,1\}$. Their ensemble $C = \{C_n\}$.

Some notation

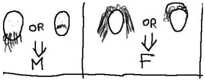
- C_n: a set of classifiction functions $\{0,1\}^n \to \{0,1\}$. Their ensemble $C = \{C_n\}$.
- D_n: A distribution over inputs: $\{0,1\}^n$. Their ensemble: $D = \{D_i\}$.

L learns a set of functions C wrt D

L learns a set of functions C wrt D

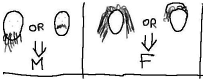
- Fix some $c \in C_n$. 

L learns a set of functions C wrt D

- Fix some $c \in C_n$. 

- Pick $S \sim D_n^{m(n)}$. 

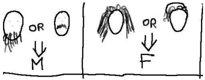
L learns a set of functions C wrt D

- Fix some $c \in C_n$. 

- Pick $S \sim D_n^{m(n)}$. 

- Alg L, upon studying (S, c(S)) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.

L learns a set of functions C wrt D

- Fix some $c \in C_n$. 

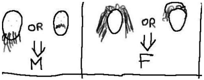
- Pick $S \sim D_n^{m(n)}$. 

- Alg L, upon studying (S, c(S)) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.
- $$Pr_{S \sim D_n^{m(n)}, x \sim D_n}(L(S, c(S), x) = c(x)) \geq 1 - \epsilon$$

  in time $poly(n, \frac{1}{\epsilon})$. 

L learns a set of functions C wrt D

- Fix some $c \in C_n$.

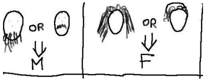


- Pick $S \sim D_n^{m(n)}$.

- Alg L, upon studying (S, c(S)) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.
-
$$Pr_{S \sim D_n^{m(n)}, x \sim D_n}(L(S, c(S), x) = c(x)) \geq 1 - \epsilon$$
in time $poly(n, \frac{1}{\epsilon})$.

- L learns C if you can say this $\forall c \in C_n, D_n, \forall n$.

L learns a set of functions C wrt D

- Fix some $c \in C_n$. 

- 

Pick $S \sim D_n^{m(n)}$.
- Alg L, upon studying (S, c(S)) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.
- $$Pr_{S \sim D_n^{m(n)}, x \sim D_n}(L(S, c(S), x) = c(x)) \geq 1 - \epsilon$$
  in time $poly(n, \frac{1}{\epsilon})$. 

- L learns C if you can say this $\forall c \in C_n, D_n, \forall n$.
- Else hard to learn.

About computational learning theory

About computational learning theory

- Can you learn it in polynomial time?

About computational learning theory

- Can you learn it in polynomial time?
- Is it hard to learn it in polynomial time?

Outline

Outline

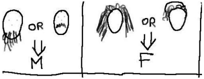
Some notation

Some notation

- C_n: a set of classifiction functions $\{0,1\}^n \rightarrow \{0,1\}$.

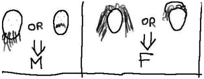
Some notation

- C_n: a set of classifiction functions $\{0,1\}^n \rightarrow \{0,1\}$.
- P_n: A distribution over C_n. Their ensemble: $P = \{P_n\}$.



Some notation

- C_n: a set of classifiction functions $\{0,1\}^n \rightarrow \{0,1\}$.
- P_n: A distribution over C_n. Their ensemble: $P = \{P_n\}$.



- D_n: A distribution over inputs: $\{0,1\}^n$. Their ensemble: $D = \{D_i\}$.

Hardness of learning as a cryptographic assumption

Hardness of learning as a cryptographic assumption

- Want 'C is hard to learn' to be a cryptographic assumption.

Hardness of learning as a cryptographic assumption

- Want 'C is hard to learn' to be a cryptographic assumption.
- Any alg L should learn C only with negligible probability.

Hardness of learning as a cryptographic assumption

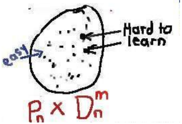
- Want 'C is hard to learn' to be a cryptographic assumption.
- Any alg L should learn C only with negligible probability.
- Take (P_n, D_n). Pick classifier c using P_n. Pick many examples using D_n. Your alg cannot match c(x) with non negligible probability. 

Outline

Problem in hardness definition

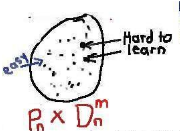
Problem in hardness definition

- Even if you have efficient alg L to learn all but a tiny scattered subset of C, it is no good. C is 'hard to learn'.



Problem in hardness definition

- Even if you have efficient alg L to learn all but a tiny scattered subset of C, it is no good. C is 'hard to learn'.



- If 'Learning C is hard' were a cryptographic assumption, any proof of security built on this assumption would be **worthless**. L is strong enough to break this assumption, by cryptographic standards.

Outline

Learning a concept class C wrt (P, D)

Learning a concept class C wrt (P, D)

- So, weaken hardness of learning defn or strengthen learnability defn.

Learning a concept class C wrt (P, D)

- So, weaken hardness of learning defn or strengthen learnability defn.
- Pick $D_n \in D, P_n \in P, c \in P_n$.

Learning a concept class C wrt (P, D)

- So, weaken hardness of learning defn or strengthen learnability defn.
- Pick $D_n \in D, P_n \in P, c \in P_n$.

- Alg L, upon studying S, c(S) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.

Learning a concept class C wrt (P, D)

- So, weaken hardness of learning defn or strengthen learnability defn.
- Pick $D_n \in D, P_n \in P, c \in P_n$.

- Alg L, upon studying S, c(S) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.
- $$Pr_{S \sim D_n^{m(n)}, x \sim D_n, c \in P_n}(L(S, c(S), x) = c(x)) \geq 1 - \epsilon$$

.

Learning a concept class C wrt (P, D)

- So, weaken hardness of learning defn or strengthen learnability defn.
- Pick $D_n \in D, P_n \in P, c \in P_n$.

- Alg L, upon studying S, c(S) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.
-
$$Pr_{S \sim D_n^{m(n)}, x \sim D_n, c \in P_n}(L(S, c(S), x) = c(x)) \geq 1 - \epsilon$$

.

- L learns C if you can say this $\forall P_n, D_n$.

Learning a concept class C wrt (P, D)

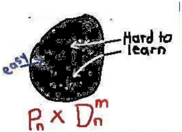
- So, weaken hardness of learning defn or strengthen learnability defn.
- Pick $D_n \in D, P_n \in P, c \in P_n$.

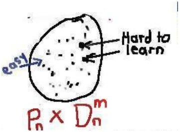
- Alg L, upon studying S, c(S) in time $poly(n, \frac{1}{\epsilon})$, classifies new examples with $\leq \epsilon$ error rate.
- $$Pr_{S \sim D_n^{m(n)}, x \sim D_n, c \in P_n}(L(S, c(S), x) = c(x)) \geq 1 - \epsilon$$

.

- L learns C if you can say this $\forall P_n, D_n$.
- Else hard to learn.

The new defintion, pictorially

'Hard to learn': 

'Learnable': 

Outline

Pseudorandom bit generator (PRBG)

Pseudorandom bit generator (PRBG)

- G_n takes n bits input, makes $g(n) > n$ bits output.

Pseudorandom bit generator (PRBG)

- G_n takes n bits input, makes $g(n) > n$ bits output.
- Any polynomial time alg T does not behave noticably differently on $y \sim U(\{0,1\}^{g(n)})$ and $g(x)|x \sim U(\{0,1\}^n)$.

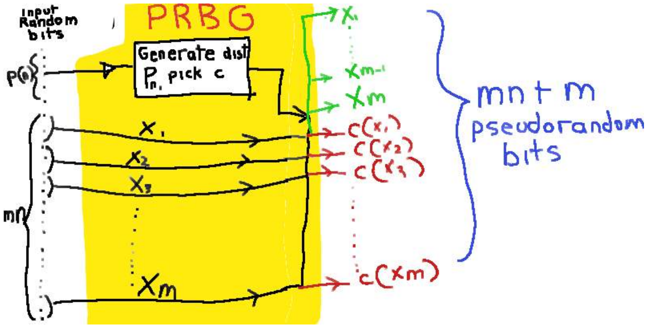
Pseudorandom bit generator (PRBG)

- G_n takes n bits input, makes $g(n) > n$ bits output.
- Any polynomial time alg T does not behave noticably differently on $y \sim U(\{0,1\}^{g(n)})$ and $g(x)|x \sim U(\{0,1\}^n)$.
- $G = \{G_n\}$.

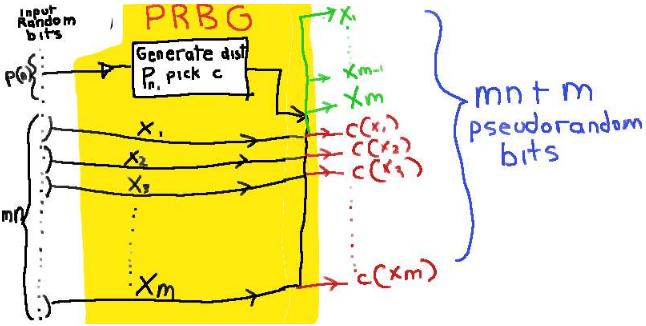
Pseudorandom bit generator (PRBG)

- G_n takes n bits input, makes $g(n) > n$ bits output.
- Any polynomial time alg T does not behave noticably differently on $y \sim U(\{0,1\}^{g(n)})$ and $g(x)|x \sim U(\{0,1\}^n)$.
- $G = \{G_n\}$.
- Now, construct a PRBG.

Construct PRBG from hard to learn P_n over C_n



Construct PRBG from hard to learn P_n over C_n



- Proof by contradiction: If you could break this PRBG, C_n not hard to learn wrt $(P_n, U(\{0,1\}^n)$.
- 11100111001111000001011001... *'Can I predict the next bit?'*

Outline

Some other results in the paper: Skip

Some other results in the paper: Skip

- Things you can make from hard to learn set of functions: One way functions, A private key cryptosystem.

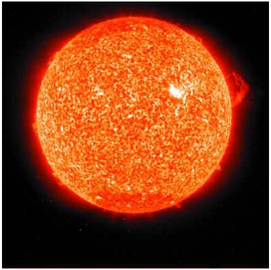
Some other results in the paper: Skip

- Things you can make from hard to learn set of functions: One way functions, A private key cryptosystem.
- A pseudo random generator based on hardness of learning parity functions in the presence of noise.

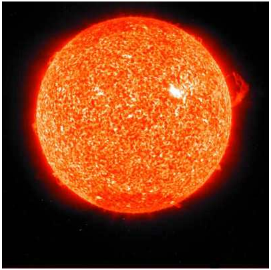
Some other results in the paper: Skip

- Things you can make from hard to learn set of functions: One way functions, A private key cryptosystem.
- A pseudo random generator based on hardness of learning parity functions in the presence of noise.
- They take more pains to relate the circuit size and depth required to evaluate functions in hard to learn C_n with the circuit depth and size of the primitives generated.

The take home message

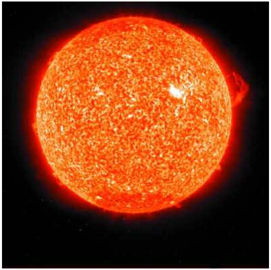


The take home message



- Can use hardness of learning, properly defined, as a cryptographic assumption.

The take home message



- Can use hardness of learning, properly defined, as a cryptographic assumption.
- Can generically make pseudorandom bit generator from hard to learn but easy to evaluate classes of functions.

Bye!