The Sample-Computational Tradeoff

Shai Shalev-Shwartz

School of Computer Science and Engineering
The Hebrew University of Jerusalem

NIPS Multi-Trade-Offs Workshop,
December 2012

Based on joint work with:

- Satyen Kale and Elad Hazan (COLT'2012)
- Aharon Birnbaum (NIPS'2012)
- Amit Daniely and Nati Linial (on arxiv)
- Ohad Shamir and Eran Tromer (AISTATS'2012)

Agnostic PAC Learning

- Hypothesis class $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$
- Loss function: $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$
- $\mathcal{D}$ - unknown distribution over $\mathcal{X} \times \mathcal{Y}$
- True risk: $L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h,(x,y))]$
- Training set: $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \overset{\text{i.i.d.}}{\sim} \mathcal{D}^m$
- Goal: use S to find h_S s.t. with high probability,

$$L_{\mathcal{D}}(h_S) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

- ERM rule:

$$\mathrm{ERM}(S) \in \operatorname*{argmin}_{h\in\mathcal{H}} L_S(h) := \frac{1}{m}\sum_{i=1}^{m} \ell(h,(x_i,y_i))$$

Error Decomposition

$$h^\star = \operatorname*{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \quad ; \quad \mathrm{ERM}(S) = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$$

$$L_{\mathcal{D}}(h_S) = \underbrace{L_{\mathcal{D}}(h^\star)}_{\text{approximation}} + \underbrace{L_{\mathcal{D}}(\mathrm{ERM}(S)) - L_{\mathcal{D}}(h^\star)}_{\text{estimation}}$$

- Bias-Complexity tradeoff: Larger $\mathcal{H}$ decreases approximation error but increases estimation error

3-term Error Decomposition (Bottou & Bousquet' 08)

$$h^\star = \operatorname*{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \quad ; \quad \mathrm{ERM}(S) = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$$

$$L_{\mathcal{D}}(h_S) = \underbrace{L_{\mathcal{D}}(h^\star)}_{\text{approximation}} + \underbrace{L_{\mathcal{D}}(\mathrm{ERM}(S)) - L_{\mathcal{D}}(h^\star)}_{\text{estimation}} + \underbrace{L_{\mathcal{D}}(h_S) - L_{\mathcal{D}}(\mathrm{ERM}(S))}_{\text{optimization}}$$

- Bias-Complexity tradeoff: Larger $\mathcal{H}$ decreases approximation error but increases estimation error
- What about optimization error ?
  - Two resources: samples and runtime
  - Sample-Computational complexity (Decatur, Goldreich, Ron '98)

Joint Time-Sample Complexity

Goal:

$$L_{\mathcal{D}}(h_S) \;\leq\; \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \;+\; \epsilon$$

Joint Time-Sample Complexity

Goal:

$$L_{\mathcal{D}}(h_S) \ \le \ \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \ + \ \epsilon$$

- Sample complexity: How many examples are needed ?
- Time complexity: How much time is needed ?

Joint Time-Sample Complexity

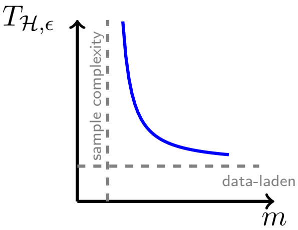
Goal:

$$L_{\mathcal{D}}(h_S) \;\leq\; \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \;+\; \epsilon$$

- Sample complexity: How many examples are needed ?
- Time complexity: How much time is needed ?

Time-sample complexity

$T_{\mathcal{H},\epsilon}(m)$ = how much time is needed when $|S| = m$?



Outline

The Sample-Computational tradeoff:

- Agnostic learning of preferences
- Learning margin-based halfspaces
- Formally establishing the tradeoff

Agnostic learning Preferences

The Learning Problem:

- $\mathcal{X} = [d] \times [d]$, $\mathcal{Y} = \{0, 1\}$
- Given $(i, j) \in \mathcal{X}$ predict if i is preferable over j
- $\mathcal{H}$ is all permutations over $[d]$
- Loss function $=$ zero-one loss

Agnostic learning Preferences

The Learning Problem:

- $\mathcal{X} = [d] \times [d]$, $\mathcal{Y} = \{0, 1\}$
- Given $(i, j) \in \mathcal{X}$ predict if i is preferable over j
- $\mathcal{H}$ is all permutations over $[d]$
- Loss function $=$ zero-one loss

Method I:

- $\mathrm{ERM}_{\mathcal{H}}$
- Sample complexity is $\frac{d}{\epsilon^2}$

Agnostic learning Preferences

The Learning Problem:

- $\mathcal{X} = [d] \times [d]$, $\mathcal{Y} = \{0, 1\}$
- Given $(i, j) \in \mathcal{X}$ predict if i is preferable over j
- $\mathcal{H}$ is all permutations over $[d]$
- Loss function $=$ zero-one loss

Method I:

- $\text{ERM}_{\mathcal{H}}$
- Sample complexity is $\frac{d}{\epsilon^2}$
- Varun Kanade and Thomas Steinke (2011): If RP$\neq$NP, it is not possible to efficiently find an ϵ-accurate permutation

Agnostic learning Preferences

The Learning Problem:

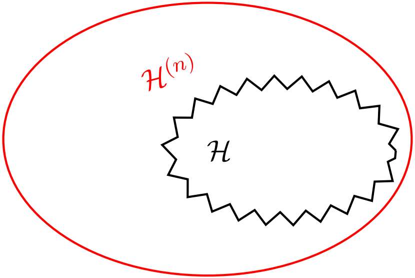
- $\mathcal{X} = [d] \times [d]$, $\mathcal{Y} = \{0, 1\}$
- Given $(i, j) \in \mathcal{X}$ predict if i is preferable over j
- $\mathcal{H}$ is all permutations over $[d]$
- Loss function $=$ zero-one loss

Method I:

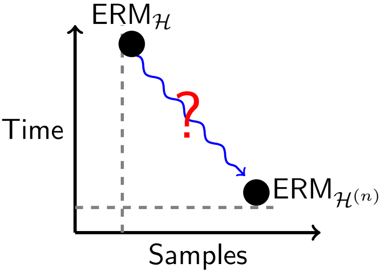
- $\text{ERM}_{\mathcal{H}}$
- Sample complexity is $\frac{d}{\epsilon^2}$
- Varun Kanade and Thomas Steinke (2011): If RP$\neq$NP, it is not possible to efficiently find an ϵ-accurate permutation
- Claim: If $m \geq d^2/\epsilon^2$ it is possible to find a predictor with error $\leq \epsilon$ in polynomial time

Agnostic learning Preferences

- Let $\mathcal{H}^{(n)}$ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$
- $\text{ERM}_{\mathcal{H}^{(n)}}$ can be computed efficiently
- Sample complexity: $VC(\mathcal{H}^{(n)})/\epsilon^2 = d^2/\epsilon^2$
- Improper learning



Sample-Computational Tradeoff



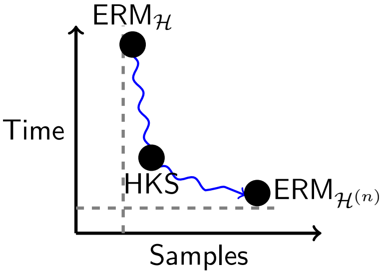
	Samples	Time
$\text{ERM}_{\mathcal{H}}$	d	$d!$
$\text{ERM}_{\mathcal{H}^{(n)}}$	d^2	d^2

Is this the best we can do?

- Analysis is based on upper bounds
- Is it possible to (improperly) learn efficiently with $d\log(d)$ examples ? Posed as an open problem by:
  - Jake Abernathy (COLT'10)
  - Kleinberg, Niculescu-Mizil, Sharma (Machine Learning 2010)

Is this the best we can do?

- Analysis is based on upper bounds
- Is it possible to (improperly) learn efficiently with $d\log(d)$ examples ? Posed as an open problem by:
  - Jake Abernathy (COLT'10)
  - Kleinberg, Niculescu-Mizil, Sharma (Machine Learning 2010)
- Hazan, Kale, S. (COLT'12):
  - Can learn *efficiently* with $\frac{d\log^3(d)}{\epsilon^2}$ examples



	Samples	Time
ERM$_{\mathcal{H}}$	d	$d!$
HKS	$d\log^3(d)$	$d^4\log^3(d)$
ERM$_{\mathcal{H}^{(n)}}$	d^2	d^2

HKS: Proof idea

- Each permutation π can be written as a matrix, s.t.,

$$W(i,j) = \begin{cases} 1 & \text{if } \pi(i) < \pi(j) \\ 0 & \text{o.w.} \end{cases}$$

- Definition: matrix is (β, τ) decomposable if its symmetrization can be written as $P - N$ where P, N are PSD, have trace bounded by τ, and diagonal entries bounded by β
- Theorem: There's an online algorithm with regret of $\sqrt{\tau \beta \log(d) T}$ for predicting the elements of (β, τ)-decomposable matrices
- Lemma: Permutation matrices are $(\log(d), d \log(d))$ decomposable.

Outline

The Sample-Computational tradeoff:

- Agnostic learning of preferences ✓
- Learning margin-based halfspaces
- Formally establishing the tradeoff

Learning Margin-Based Halfspaces

- Goal: Find $h_S : \mathcal{X} \to \{\pm 1\}$ such that

$$\mathbb{P}[h_S(x) \neq y] \;\leq\; (1+\alpha) \min_{w:\|w\|=1} \mathbb{P}[y\langle w, x\rangle \leq \gamma] + \epsilon$$

Learning Margin-Based Halfspaces

- Goal: Find $h_S : \mathcal{X} \to \{\pm 1\}$ such that

$$\mathbb{P}[h_S(x) \neq y] \;\leq\; (1+\alpha) \min_{w:\|w\|=1} \mathbb{P}[y\langle w, x\rangle \leq \gamma] + \epsilon$$

- Known results:

	α	Samples	Time
Ben-David and Simon	0	$\frac{1}{\gamma^2\,\epsilon^2}$	$\exp(1/\gamma^2)$
SVM (Hinge-loss)	$\frac{1}{\gamma}$	$\frac{1}{\gamma^2\,\epsilon^2}$	$\mathrm{poly}(1/\gamma)$

- Trading approximation factor for runtime
- What if $\alpha \in (0, 1/\gamma)$?

Learning Margin-Based Halfspaces

Theorem (Birnbaum and S., NIPS'12)

Can achieve α-approximation using time and sample complexity of

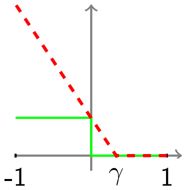
$$\mathrm{poly}(1/\gamma) \cdot \exp\left(\frac{4}{(\gamma\,\alpha)^2}\right)$$

Corollary

Can achieve $\alpha = \frac{1}{\gamma\sqrt{\log(1/\gamma)}}$ *in polynomial time*

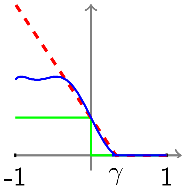
Proof Idea

- SVM relies on the hinge-loss as a convex surrogate:
  $\ell(w,(x,y)) = \left[1 - y\frac{\langle w,x\rangle}{\gamma}\right]_+$



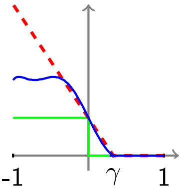
Proof Idea

- SVM relies on the hinge-loss as a convex surrogate: $\ell(w,(x,y)) = \left[1 - y\frac{\langle w,x\rangle}{\gamma}\right]_+$
- Compose the hinge-loss over a polynomial $[1 - yp(\langle w,x\rangle)]_+$



Proof Idea

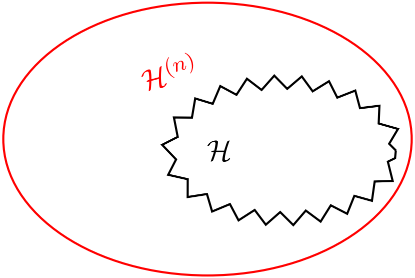
- SVM relies on the hinge-loss as a convex surrogate:
$\ell(w,(x,y)) = \left[1 - y\frac{\langle w,x\rangle}{\gamma}\right]_+$
- Compose the hinge-loss over a polynomial $[1 - yp(\langle w,x\rangle)]_+$



- But now the loss function is non convex ...

Proof Idea (Cont.)

- Let $p(x) = \sum_j \beta_j x^j$ be the polynomial
- Original class: $\mathcal{H} = \{x \mapsto p(\langle w, x \rangle) : \|w\| = 1\}$
- Define kernel: $k(x, x') = \sum_j |\beta_j| (\langle x, x' \rangle)^j$
- New class: $\mathcal{H}^{(n)} = \{x \mapsto \langle v, \Psi(x) \rangle : \|v\| \leq B\}$ where Ψ is the mapping corresponds to the kernel
- $\text{ERM}_{\mathcal{H}^{(n)}}$ can be computed efficiently (due to convexity)
- Sample complexity: B^2/ϵ^2



Can we do better ?

Theorem (Daniely, Lineal, S. 2012)

For every kernel, SVM cannot obtain $\alpha < \frac{1}{\gamma \, \mathrm{poly}(\log(\gamma))}$ *with* $\mathrm{poly}(1/\gamma)$ *samples. A similar lower bound holds for any feature-based mapping (not necessarily kernel-based).*

- Open problem: lower bounds for other techniques / any technique ?

Proof ideas

- A one dimensional problem: $\mathcal{D} = (1-\lambda)\mathcal{D}_1 + \lambda\mathcal{D}_2$

- Every low degree polynomial with hinge-loss smaller than 1 must have $p(\gamma) \approx p(-\gamma)$.
- Pull back the distribution to high dimension
- Use a characterization of Hilbert spaces corresponding to symmetric kernels, from which we can write f using Legendre polynomials and reduce to the 1-dim case
- By averaging the kernel over the group of linear isometries of $\mathbb{R}^d$, we relax the assumption that the kernel is symmetric

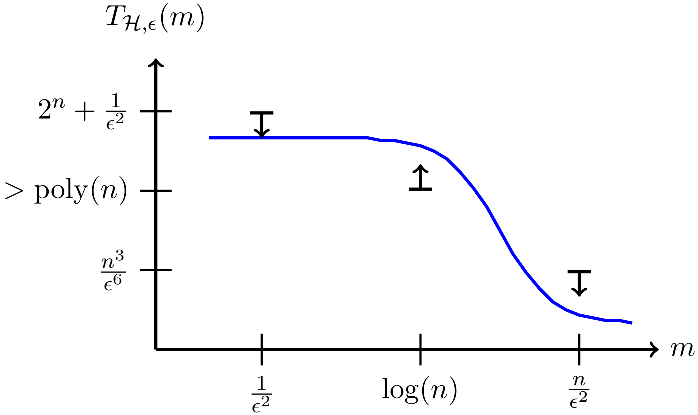
Outline

The Sample-Computational tradeoff:

- Agnostic learning of preferences ✓
- Learning margin-based halfspaces ✓
- Formally establishing the tradeoff

Formal Derivation of Gaps

Theorem (Shamir, S., Tromer 2012): Assume one-way permutations exist, there exists an agnostic learning problem such that:



Proof: One Way Permutations

$P : \{0,1\}^n \to \{0,1\}^n$ is one-way permutation if it's one-to-one and

- It is easy to compute $\mathbf{w} = P(\mathbf{s})$
- It is hard to compute $\mathbf{s} = P^{-1}(\mathbf{w})$

Proof: One Way Permutations

$P : \{0,1\}^n \rightarrow \{0,1\}^n$ is one-way permutation if it's one-to-one and

- It is easy to compute $\mathbf{w} = P(\mathbf{s})$
- It is hard to compute $\mathbf{s} = P^{-1}(\mathbf{w})$

Goldreich-Levin Theorem: If P is one way, then for any algorithm A,

$$\exists \mathbf{w} \text{ s.t. } \mathop{\mathbb{P}}_{\mathbf{r}}[A(\mathbf{r}, P(\mathbf{w})) = \langle \mathbf{r}, \mathbf{w} \rangle] < \frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$$

Proof: The Learning Problem

The Domain

- Let P be a one-way permutation.
- $\mathcal{X} = \{0,1\}^{2n}, \mathcal{Y} = \{0,1\}$
- Domain: $Z \subset \mathcal{X} \times \mathcal{Y}$
  - $((\mathbf{r}, \mathbf{s}), b) \in Z$ iff $\langle P^{-1}(\mathbf{s}), \mathbf{r} \rangle = b$
- (Inner product over GF(2))

Proof: The Learning Problem

The Hypothesis Class

- $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in \{0,1\}^n\}$ where $h_{\mathbf{w}} : \mathcal{X} \to [0,1]$ is

$$h_{\mathbf{w}}(\mathbf{r},\mathbf{s}) = \begin{cases} \langle \mathbf{w}, \mathbf{r} \rangle & \text{if } \mathbf{s} = P(\mathbf{w}) \\ 1/2 & \text{o.w.} \end{cases}$$

The Loss Function:

- Absolute loss (= expected 0-1)

$$\ell(h, ((\mathbf{r},\mathbf{s}), b)) = |h(\mathbf{r},\mathbf{s}) - b|$$

Proof: The Learning Problem

The Hypothesis Class

- $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in \{0,1\}^n\}$ where $h_{\mathbf{w}} : \mathcal{X} \to [0,1]$ is

$$h_{\mathbf{w}}(\mathbf{r},\mathbf{s}) = \begin{cases} \langle \mathbf{w}, \mathbf{r} \rangle & \text{if } \mathbf{s} = P(\mathbf{w}) \\ 1/2 & \text{o.w.} \end{cases}$$

The Loss Function:

- Absolute loss (= expected 0-1)

$$\ell(h, ((\mathbf{r},\mathbf{s}), b)) = |h(\mathbf{r},\mathbf{s}) - b| = \begin{cases} 0 & \text{if } \mathbf{s} = P(\mathbf{w}) \\ 1/2 & \text{o.w.} \end{cases}$$

- Note: $L_{\mathcal{D}}(h_{\mathbf{w}}) = \mathbb{P}[\mathbf{s} \neq P(\mathbf{w})] \cdot \frac{1}{2}$

Proof: The Learning Problem

The Hypothesis Class

- $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in \{0,1\}^n\}$ where $h_{\mathbf{w}} : \mathcal{X} \to [0,1]$ is

$$h_{\mathbf{w}}(\mathbf{r},\mathbf{s}) = \begin{cases} \langle \mathbf{w}, \mathbf{r} \rangle & \text{if } \mathbf{s} = P(\mathbf{w}) \\ 1/2 & \text{o.w.} \end{cases}$$

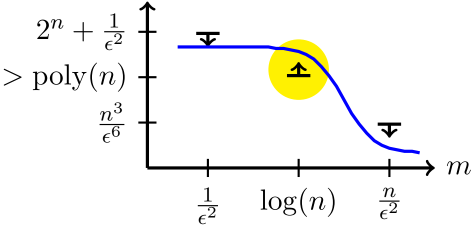
The Loss Function:

- Absolute loss (= expected 0-1)

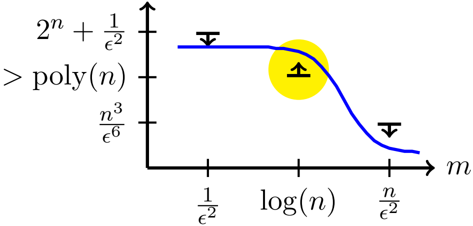
$$\ell(h, ((\mathbf{r},\mathbf{s}), b)) = |h(\mathbf{r},\mathbf{s}) - b| = \begin{cases} 0 & \text{if } \mathbf{s} = P(\mathbf{w}) \\ 1/2 & \text{o.w.} \end{cases}$$

- Note: $L_{\mathcal{D}}(h_{\mathbf{w}}) = \mathbb{P}[\mathbf{s} \neq P(\mathbf{w})] \cdot \frac{1}{2}$
- Agnostic: $L_{\mathcal{D}}(h_{\mathbf{w}}) = 0$ only if $\mathbb{P}[\mathbf{s} = P(\mathbf{w})] = 1$

Proof of Second Claim

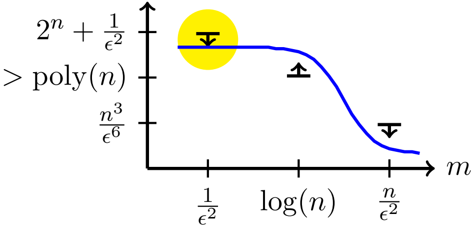


Proof of Second Claim



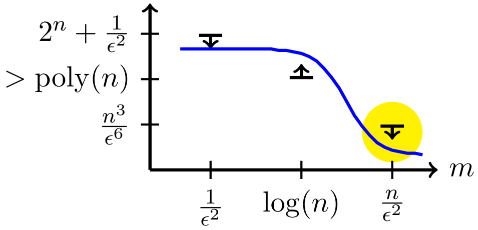
- Suppose we can learn with $m = O(\log(n))$ examples
- $\forall \mathbf{w}$, define $\mathcal{D}_{\mathbf{w}}$ s.t. $\mathbf{r}$ is uniform, $\mathbf{s} = P(\mathbf{w})$, and $b = \langle \mathbf{r}, \mathbf{w} \rangle$
- To generate an i.i.d. training set from $\mathcal{D}_{\mathbf{w}}$:
  - Pick $\mathbf{r}_1, \ldots, \mathbf{r}_m$ and $b_1, \ldots, b_m$ at random
  - If $b_i = \langle \mathbf{r}_i, \mathbf{w} \rangle$ for all i we're done
  - This happens w.p. $1/2^m = 1/\text{poly}(n)$
- Feed the training set to the learner, to get $h_{\mathbf{w}'}(\mathbf{r}, P(\mathbf{w})) \approx \langle \mathbf{r}, \mathbf{w} \rangle$
- Goldreich-Levin theorem $\Rightarrow$ contradiction

Proof of First Claim

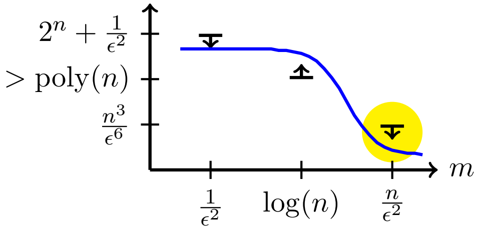


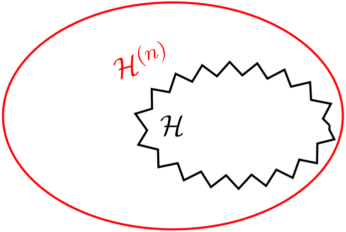
- Recall: $L_{\mathcal{D}}(h_{\mathbf{w}}) = \mathbb{P}[\mathbf{s} \neq P(\mathbf{w})] \cdot \frac{1}{2} = \mathbb{P}[P^{-1}(\mathbf{s}) \neq \mathbf{w}] \cdot \frac{1}{2}$
- Problem reduces to *multiclass* prediction with hypothesis class of constant predictors
- Sample complexity is $1/\epsilon^2$

Proof of Third Claim

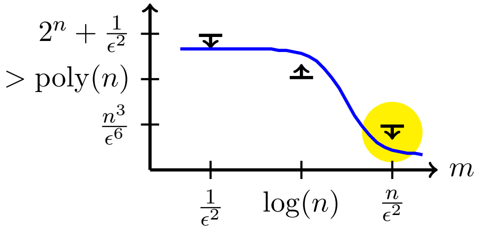


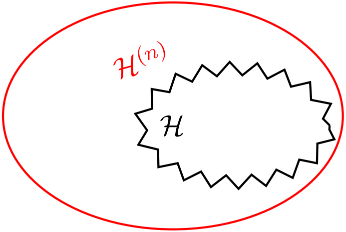
Proof of Third Claim





Proof of Third Claim





- Original class:

$$h_{\mathbf{w}}(\mathbf{r}, \mathbf{s}) = \begin{cases} \langle \mathbf{w}, \mathbf{r} \rangle & \text{if } \mathbf{s} = P(\mathbf{w}) \\ 1/2 & \text{o.w.} \end{cases}$$

- New class:

$$h_{((\mathbf{r}_1, \mathbf{s}'), b_1), \ldots, ((\mathbf{r}_{m'}, \mathbf{s}'), b_{m'})}(\mathbf{r}, \mathbf{s}) = \begin{cases} \sum_i \alpha_i b_i & \text{if } \mathbf{r} = \sum_i \alpha_i \mathbf{r}_i \wedge \mathbf{s} = \mathbf{s}' \\ 1/2 & \text{o.w.} \end{cases}$$

- New class is efficiently learnable with $m = n/\epsilon^2$

Summary

- The Bias-Variance tradeoff is well understood
- We study the Sample-Computational tradeoff
- More data can reduce runtime

Open Questions

- Other techniques to control the tradeoff
- Stronger lower bounds for real-world problems