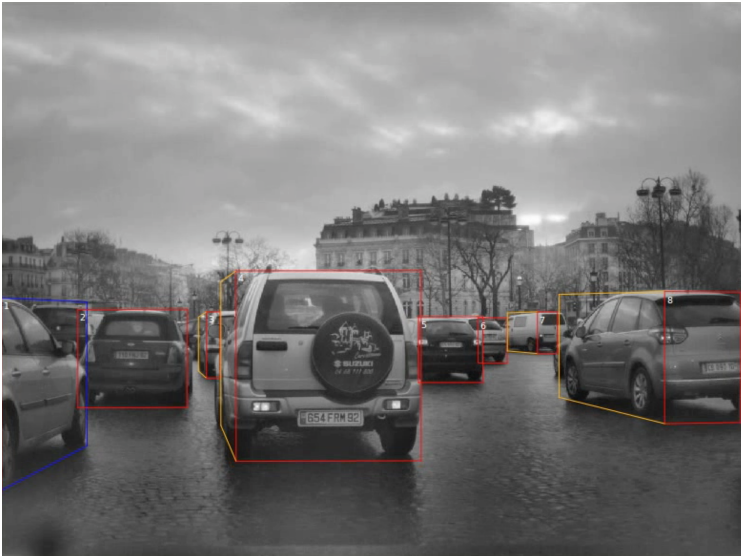
# Minimizing the Maximal Loss: Why and How?

Shai Shalev-Shwartz and Yonatan Wexler

The Hebrew University of Jerusalem
and
OrCam

ICML 2016

# Typical vs. Rare Cases



# Typical vs. Rare Cases

# PAC Learning with Train/Test Mismatch

PAC learning

- $\mathcal{D}$ is a distribution over $\mathcal{X}$
- A target labeling function $h^* \in \mathcal{H}$
- Training set is sampled i.i.d. from $\mathcal{D}$
- Goal: find $h$ s.t. $L_{\mathcal{D}}(h) < \epsilon$ where $L_D(h) = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq h^*(x)]$

# PAC Learning with Train/Test Mismatch

## PAC learning

- $\mathcal{D}$ is a distribution over $\mathcal{X}$
- A target labeling function $h^* \in \mathcal{H}$
- Training set is sampled i.i.d. from $\mathcal{D}$
- Goal: find $h$ s.t. $L_{\mathcal{D}}(h) < \epsilon$ where $L_D(h) = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq h^*(x)]$

## PAC Learning with Train/Test Mismatch

- $\mathcal{D}_1, \mathcal{D}_2$ are two distributions over $\mathcal{X}$
- A target labeling function $h^* \in \mathcal{H}$
- Training set is sampled i.i.d. from $\mathcal{D} = \lambda_1 \mathcal{D}_1 + \lambda_2 \mathcal{D}_2$, $\lambda_1 \gg \lambda_2$
- Goal: find $h$ s.t. both $L_{\mathcal{D}_1}(h) < \epsilon$ and $L_{\mathcal{D}_2}(h) < \epsilon$
- Note: Learner can only sample from $\mathcal{D}$

# How to learn ?

- **Most popular approach**: Minimize the average error to accuracy $\epsilon$

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} 1[h_w(x_i) \neq y_i]$$

# How to learn ?

- **Most popular approach**: Minimize the average error to accuracy $\epsilon$

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} 1[h_w(x_i) \neq y_i]$$

- **Intuitively**: this won't work if $\epsilon > \lambda_2$

# How to learn ?

- **Most popular approach**: Minimize the average error to accuracy $\epsilon$

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} 1[h_w(x_i) \neq y_i]$$

- **Intuitively**: this won't work if $\epsilon > \lambda_2$
- **Sample complexity:** what if we solve the ERM, i.e., find $w$ for which $L_S(w) = 0$ ?

# How to learn ?

- **Most popular approach**: Minimize the average error to accuracy $\epsilon$

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} 1[h_w(x_i) \neq y_i]$$

- **Intuitively**: this won't work if $\epsilon > \lambda_2$
- **Sample complexity:** what if we solve the ERM, i.e., find $w$ for which $L_S(w) = 0$ ?
- **Intuitively**: still not enough, because if we only see few examples from $\mathcal{D}_2$ we might overfit

# How to learn ?

- **Most popular approach**: Minimize the average error to accuracy $\epsilon$

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} 1[h_w(x_i) \neq y_i]$$

- **Intuitively**: this won't work if $\epsilon > \lambda_2$
- **Sample complexity:** what if we solve the ERM, i.e., find $w$ for which $L_S(w) = 0$ ?
- **Intuitively**: still not enough, because if we only see few examples from $\mathcal{D}_2$ we might overfit
- **Theorem (informally):** under some conditions, many examples from $\mathcal{D}_1$ and a few examples from $\mathcal{D}_2$ suffices to ensure small error on both $\mathcal{D}_1$ and $\mathcal{D}_2$

# Refined Sample Complexity Analysis

## Theorem

*Define*

- $\mathcal{H}_{1,\epsilon} = \{h \in \mathcal{H} : L_{D_1}(h) \le \epsilon\}$
- $c = \max\{c' \in [\epsilon, 1) : \forall h \in \mathcal{H}_{1,\epsilon},\, L_{D_2}(h) \le c' \Rightarrow L_{D_2}(h) \le \epsilon\}$.

*Then, it suffices to sample* $\frac{\mathrm{VC}(\mathcal{H})}{\epsilon}$ *examples from* $\mathcal{D}_1$ *and* $\frac{\mathrm{VC}(\mathcal{H}_{1,\epsilon})}{c}$ *examples from* $\mathcal{D}_2$*.*

Proof idea:

- Think about ERM as two steps: (1) find $\mathcal{H}_{1,\epsilon}$ based on examples from $D_1$ (2) find a hypothesis within $\mathcal{H}_{1,\epsilon}$ that is good on the examples from $D_2$
- "Shell analysis" (Haussler-Kearns-Seung-Tishby'96) for the 2nd step

# Refined Sample Complexity Analysis

## Theorem

*Define*

- $\mathcal{H}_{1,\epsilon} = \{h \in \mathcal{H} : L_{D_1}(h) \le \epsilon\}$
- $c = \max\{c' \in [\epsilon, 1) : \forall h \in \mathcal{H}_{1,\epsilon},\ L_{D_2}(h) \le c' \Rightarrow L_{D_2}(h) \le \epsilon\}$.

*Then, it suffices to sample* $\frac{\mathrm{VC}(\mathcal{H})}{\epsilon}$ *examples from* $\mathcal{D}_1$ *and* $\frac{\mathrm{VC}(\mathcal{H}_{1,\epsilon})}{c}$ *examples from* $\mathcal{D}_2$.

Proof idea:

- Think about ERM as two steps: (1) find $\mathcal{H}_{1,\epsilon}$ based on examples from $D_1$ (2) find a hypothesis within $\mathcal{H}_{1,\epsilon}$ that is good on the examples from $D_2$
- "Shell analysis" (Haussler-Kearns-Seung-Tishby'96) for the 2nd step

Implication: to be good on $\mathcal{D}_2$ we must achieve zero training error

# Two Equivalent Ways to Solve the ERM problem

Minimize average loss to accuracy $< 1/n$:

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} 1[h_w(x_i) \neq y_i]$$

Minimize max loss to accuracy $< 1$:

$$\min_{w \in \mathbb{R}^d} L_S(w) := \max_{i \in [n]} 1[h_w(x_i) \neq y_i]$$

# Oracle Assumption

Assumption: There exists an online learner for $w$ with a mistake bound $C$

# The Mistake Bound Model (Littlestone 1988)

- The Online Game: At each round $t$, learner picks $w_t$, adversary responds with $i_t$, and learner pays $\phi_{i_t}(w_t) = 1[h_{w_t}(x_{i_t}) \neq y_{i_t}]$

# The Mistake Bound Model (Littlestone 1988)

- The Online Game: At each round $t$, learner picks $w_t$, adversary responds with $i_t$, and learner pays $\phi_{i_t}(w_t) = 1[h_{w_t}(x_{i_t}) \neq y_{i_t}]$
- Mistake Bound: The learner enjoys a mistake bound $C$ if for any $T$ and any sequence $i_1, \ldots, i_T$, it makes at most $T$ mistakes

# The Mistake Bound Model (Littlestone 1988)

- The Online Game: At each round $t$, learner picks $w_t$, adversary responds with $i_t$, and learner pays $\phi_{i_t}(w_t) = 1[h_{w_t}(x_{i_t}) \neq y_{i_t}]$
- Mistake Bound: The learner enjoys a mistake bound $C$ if for any $T$ and any sequence $i_1, \ldots, i_T$, it makes at most $T$ mistakes
- Example: The Perceptron (Rosenblatt 1958):
  - $h_w(x) = \mathrm{sign}(\langle w, x \rangle)$, $y \in \{\pm 1\}$
  - The Perceptron rule: $w_{t+1} = w_t + \phi_{i_t}(w_t)\, x_{i_t} / \|x_{i_t}\|$
  - Theorem (Agmon 1954, Minsky, Papert 1969):
    If exists $w^*$ s.t. for every $i$, $y_i \langle w^*, x_i \rangle / \|x_i\| \geq 1$, then Perceptron's mistake bound is $C = \|w^*\|^2$

# Back to the ERM problem

Minimize average loss to accuracy $< 1/n$:

$$\min_{w \in \mathbb{R}^d} L_S(w) := \frac{1}{n} \sum_{i=1}^{n} \phi_i(w)$$

Minimize max loss to accuracy $< 1$:

$$\min_{w \in \mathbb{R}^d} L_S(w) := \max_{i \in [n]} \phi_i(w)$$

# Naive Approaches

**Minimize average loss to accuracy** $< 1/n$

- Apply the online learner with random examples from $[n]$
- Runtime to achieve zero error: Need $C/T < 1/n$ so $T > n\,C$ and total time $> n\,C\,d$

# Naive Approaches

**Minimize** **average** **loss to accuracy** $< 1/n$

- Apply the online learner with random examples from $[n]$
- Runtime to achieve zero error: Need $C/T < 1/n$ so $T > n\,C$ and total time $> n\,C\,d$

**Minimize** **max** **loss to accuracy** $< 1$**:**

- Apply the online learner while feeding it with the worst example at each iteration
- Runtime for zero error: $C$ iterations, each cost $d\,n$, so total time $> n\,C\,d$

# Naive Approaches

**Minimize** **average** **loss to accuracy** $< 1/n$

- Apply the online learner with random examples from $[n]$
- Runtime to achieve zero error: Need $C/T < 1/n$ so $T > n\,C$ and total time $> n\,C\,d$

**Minimize** **max** **loss to accuracy** $< 1$**:**

- Apply the online learner while feeding it with the worst example at each iteration
- Runtime for zero error: $C$ iterations, each cost $d\,n$, so total time $> n\,C\,d$

Our approach: runtime is $\tilde{O}((n + C)\,d)$

# Our Approach: Focused Online Learning

Rewrite the Max-Loss problem:

$$\min_{w} \max_{i\in[n]} \phi_i(w) \quad = \quad \min_{w} \max_{p\in\mathbb{S}_n} \sum_{i=1}^{n} p_i \phi_i(w)$$

- Zero-sum game between $w$ player and $p$ player
- Use the online learner for the $w$ player
- Use a variant of EXP3 (Auer, Cesa-Bianchi, Freund, Schapire, 2002) for the $p$ player
- Our variant explores w.p. $1/2$: this leads to low-variance, and crucial for the analysis

# Our Approach: Focused Online Learning

- Initialize: $q = (1/n, \ldots, 1/n)$
- For $t = 1, 2, \ldots, T$
  - Sample $i_t$ according to $p = 0.5\, q + 0.5\, (1/n, \ldots, 1/n)$
  - Feed $i_t$ to the online learner
  - Update $q_{i_t} = q_{i_t}\ \exp(\phi_{i_t}(w_t)\, /(2np_{i_t}))$ and normalize

# Our Approach: Focused Online Learning

- Initialize: $q = (1/n, \ldots, 1/n)$
- For $t = 1, 2, \ldots, T$
  - Sample $i_t$ according to $p = 0.5\, q + 0.5\, (1/n, \ldots, 1/n)$
  - Feed $i_t$ to the online learner
  - Update $q_{i_t} = q_{i_t} \, \exp(\phi_{i_t}(w_t) \, / (2np_{i_t}))$ and normalize

Observe: Using tree data-structure, each iteration costs $O(\log(n))$ plus the online learner time

# Our Approach: Focused Online Learning

- Initialize: $q = (1/n, \ldots, 1/n)$
- For $t = 1, 2, \ldots, T$
  - Sample $i_t$ according to $p = 0.5\, q + 0.5\,(1/n, \ldots, 1/n)$
  - Feed $i_t$ to the online learner
  - Update $q_{i_t} = q_{i_t}\, \exp(\phi_{i_t}(w_t)\,/(2np_{i_t}))$ and normalize

Observe: Using tree data-structure, each iteration costs $O(\log(n))$ plus the online learner time

**Theorem**

*If $T \geq \tilde{\Omega}\,(n + C)$, and $k = \Omega(\log(n))$, and $t_1, \ldots, t_k$ are sampled at random from $[T]$, then with high probability*

$$\forall i, \quad \phi_i\,(\text{Majority}(w_{t_1}, \ldots, w_{t_k})) = 0$$

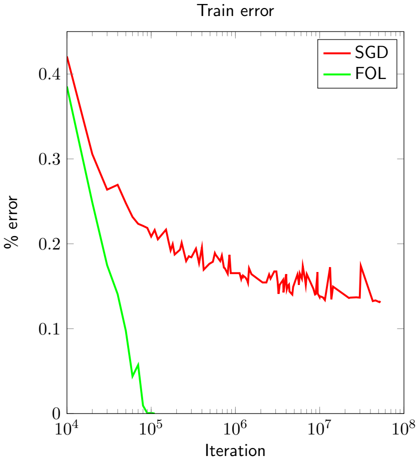
# Proof Sketch

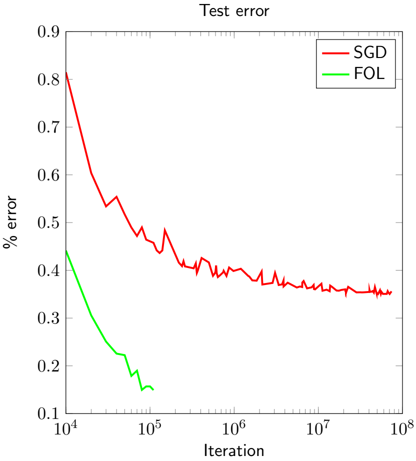
- The vector $z_t = \frac{\phi_{i_t}(w_t)}{p_{i_t}} e_{i_t}$ is an unbiased estimate of the gradient $(\phi_1(w_t), \ldots, \phi_n(w_t))$
- The update of $q$ is Mirror Descent w.r.t. Entropic regularization with $z_t$
- A certain generalized definition of variance of $z_t$ is bounded by $2n$ because of the strong exploration
- A Bernstein's type inequality for Martingales leads to strong concentration
- Union bound over every $i$ concludes the proof

# Related Work

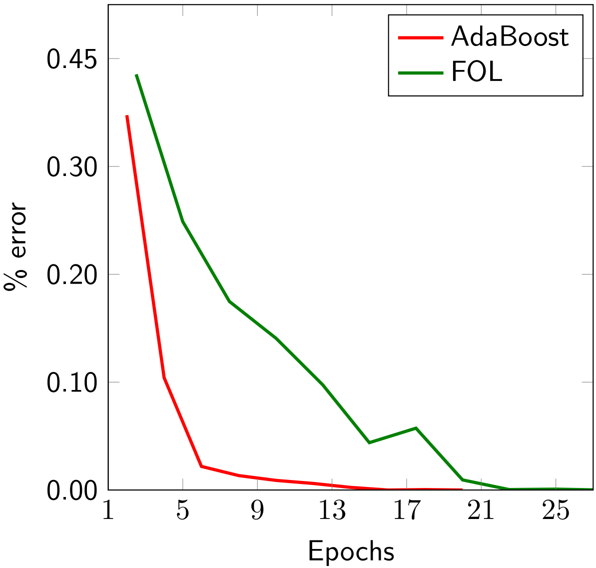
- Auer et al 2002: The main idea is there, but EXP3.P.1 costs $\Omega(n)$ per iteration
- Hazan, Clarckson, Woodruff 2012, Hazan, Koren, Srebro 2011: Only for linear classifiers, rate of $(n + d)C$.
  (Our rate is $(n + C)d$)
- AdaBoost (Freund & Schapire 1995): Only for binary classification, batch nature, similar rate.
  In practice: AdaBoost's predictor is an ensemble while ours is a single classifier

# Illustration



# FOL vs. AdaBoost



# Summary

- Some applications call for 100% success
- Focused Learning means faster learning !