An Easy Statistical Theory for Highly Scalable Learning Algorithms

Claudio Gentile
DICOM, Universita’ dell’Insubria, Italy
claudio.gentile@uninsubria.it

ICANN 2006
Content of this tutorial/1

Part 1
Intro to on-line learning problems, methods, relative loss bounds:

- On-line learning setting, examples
- Learning with expert advice (Bayes voting), relative loss bounds
- On-line learning linear-threshold functions
- Learning regression functions

focus on
BINARY
classification
Content of this tutorial/2

Part 2
Intro to statistical Pattern Recognition problem, martingales, on-line to batch conversions:

- The statistical Pattern Recognition problem
- Types of error bounds (data/algorithm-independent/dependent)
- Reduction on-line pointwise → i.i.d.
  - Expectation analysis
  - Data-dependent analysis
- Final comments and conclusions
(Worst-case) on-Line Learning

\[ E_1 \quad E_2 \quad E_3 \quad \ldots \quad E_n \quad \text{pred.} \quad \text{true lab.} \quad \text{loss} \]

<table>
<thead>
<tr>
<th>Day 1</th>
<th>1</th>
<th>1</th>
<th>-1</th>
<th>\ldots</th>
<th>-1</th>
<th>-1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 2</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>\ldots</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>Day 3</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>\ldots</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Day t</td>
<td>(z_{t,1})</td>
<td>(z_{t,2})</td>
<td>(z_{t,3})</td>
<td>\ldots</td>
<td>(z_{t,n})</td>
<td>(\hat{y}_t)</td>
<td>(y_t)</td>
<td>(\frac{1}{2}</td>
</tr>
</tbody>
</table>

On-line protocol

For \(t = 1, \ldots, T\) do:

- Get vector \(z_t \in \{-1, 1\}^n\)
- Predict \(\hat{y}_t \in \{-1, 1\}\)
- Get label \(y_t \in \{-1, 1\}\)
- Incur loss \(\frac{1}{2}|y_t - \hat{y}_t| \in \{0, 1\}\)
Halving Algorithm

- Predicts with majority
- If mistake is made then number of consistent Experts is (at least) halved
### A run of the Halving Algorithm (HA)

<table>
<thead>
<tr>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
<th>$E_6$</th>
<th>$E_7$</th>
<th>$E_8$</th>
<th>majority</th>
<th>true label</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

↑

consistent

\[ \forall \text{ sequence with } k \text{ consistent experts (out of } n) \]

HA makes $m \leq \log_2(n/k)$ mistakes: $n/2^m \geq k$
Learning with expert advice/1

What if no expert $E_i$ is consistent?

Sequence of examples $S = (z_1, y_1), \ldots, (z_T, y_T)$

- $L_A(S)$ be total loss of alg. $A$ on sequence $S$
- $L_i(S)$ be total loss of $i$-th expert $E_i$ on $S$

Want bounds of the form:

$$\forall S : \quad L_A(S) \leq a \min_i L_i(S) + b \log(n)$$

where $a, b$ are constants

Bounds loss of algorithm relative to loss of best expert
Learning with expert advice/2

Can’t wipe out experts!
Keep one weight per expert

The Weighted Majority Algorithm

\[ [L89a, LW94] \]

- Predicts with larger side
- Weights of wrong experts are slashed by a factor \( \beta \in [0, 1) \)
Learning with expert advice/3
Number of mistakes of the WM algorithm/1

\[ L_{i,t} = \# \text{ of mistakes of } E_i \text{ before time step } t \]

\[ w_{t,i} = \beta^{L_{i,t}} \text{ weight of } E_i \text{ at beginning of time step } t \]

\[ W_t = \sum_{i=1}^{n} w_{i,t} \quad \text{total weight at time step } t \]

Minority \( \leq \frac{1}{2} W_t \quad \text{Majority} \geq \frac{1}{2} W_t \)

If no mistake then minority multiplied by \( \beta \): \( W_{t+1} \leq 1 \ W_t \)

If mistake then majority multiplied by \( \beta \):

\[ W_{t+1} \leq 1 \ \text{Minority} + \beta \ \text{Majority} \leq \frac{1+\beta}{2} \ W_t \]
Learning with expert advice / 3
Number of mistakes of the WM algorithm / 2

Hence: \[ W_{T+1} \leq \left( \frac{1 + \beta}{2} \right)^{L_{WMA}(S)} W_1 \]

total final weight

\[ W_{T+1} = \sum_{j=1}^{n} w_{T+1,j} = \sum_{j=1}^{n} \beta L_j(S) \geq \beta L_i(S) \]

We got: \[ \left( \frac{1 + \beta}{2} \right)^{L_{WMA}(S)} W_1 \geq \beta L_i(S) \]

Solving for \( L_{WMA}(S) \): \[ L_{WMA}(S) \leq \frac{\ln 1/\beta}{\ln \frac{2}{1+\beta}} L_i(S) + \frac{1}{\ln \frac{2}{1+\beta}} \ln n \]

\[ L_{WMA}(S) \leq 2.63 \min_i L_i(S) + 2.63 \ln n \]

\( \beta = 1/e \)
Learning with expert advice/5

- Weighted Majority is just a Bayes voting scheme
- Easy to combine good experts (algorithms) so that prediction alg. is almost as good as best expert
- Bounds are logarithmic in \# of experts

So far:
Learning relative to best expert/component

From now on:
Learning relative to best (thresholded) linear combination of experts(components)
**A more general setting**

<table>
<thead>
<tr>
<th>Instance</th>
<th>Prediction of alg $A$</th>
<th>Label $y$</th>
<th>Loss of alg $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$\hat{y}_1$</td>
<td>$y_1$</td>
<td>$L(y_1, \hat{y}_1)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$x_t$</td>
<td>$\hat{y}_t$</td>
<td>$y_t$</td>
<td>$L(y_t, \hat{y}_t)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$x_T$</td>
<td>$\hat{y}_T$</td>
<td>$y_T$</td>
<td>$L(y_T, \hat{y}_T)$</td>
</tr>
</tbody>
</table>

Total Loss $L_A(S)$

Sequence of examples $S = (x_1, y_1), \ldots, (x_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

Comparison class $\{u\}$

Relative loss $L_A(S) - \inf_{\{u\}} Loss u (S)$

**Goal:** Bound relative loss for arbitrary sequence $S$
Learning linear-threshold functions/1
Another run of the Halving Algorithm/1

Sequence of examples $S = (x_1, y_1), \ldots, (x_T, y_T) \in \mathbb{R}^2 \times \{-1, 1\}$

$S$ is lin. separated by $u \in \mathbb{R}^2: \|u\|_2 = 1$ with margin

$0 < \gamma \leq y_t u^T x_t \forall t \quad R = \max_t \|x_t\|_2$

Experts:
- $n$ (large) linear–threshold functions evenly spread over unit circle
- Expert $i$ predicts $z_{it} = \text{sgn}(u_i^T x_t)$

Feed experts with $x_1$ and get expert prediction vector $z_1$
Learning linear-threshold functions

Another run of the Halving Algorithm

Get true label $y_1 = 1$ (mistake)
version space gets halved

Feed experts with $x_2$
and get expert prediction
vector $z_2$

Get true label $y_2 = 1$ (mistake)
version space gets (at least) halved

...at the end
consistent experts ("margin")

$m_{HA} \leq \log_2(n/k) = O(\log(R/\gamma))$ for large $n$
Learning linear-threshold functions/1
Another run of the Halving Algorithm/3

[HG02, GBNT04, ...]

For $d$-dim vectors:

$$m_{HA} \leq \log_2 \frac{1}{\mu(\text{consistent}(S))}$$

$$= O(d \log(R/\gamma)), \quad R = \max_t \|x_t\|_2$$

Proof: $y_t u^\top x_t \geq \gamma$ and $\|u - u'\|_2 < \gamma/R$

$\implies y_t (u')^\top x_t > 0$

$\implies \exists$ ball $B$ of radius $\gamma/2R$: $B \subseteq \text{consistent}(S)$,

$\mu(B) = (\gamma/2R)^{d-1}\mu(\text{surface of } d\text{-dim unit sphere})$

Drawbacks: Linear dependence on dimension $d$

Looks too time-consuming (linear dependence on $d$?)
Learning linear-threshold functions/2
The (first-order) Perceptron algorithm \[\text{[Ro62, ...]}\]

Keep weight vector \( w_t \in \mathbb{R}^n \)

In trial \( t \):

- Get instance \( x_t \in \mathbb{R}^n \)
- Predict with \( \hat{y}_t = \text{SGN}(w_t^\top x_t) \in \{-1, 1\} \)
- Get label \( y_t \in \{-1, 1\} \)
- **If mistake** \( (y_t w_t^\top x_t \leq 0) \) **then** update \( w_{t+1} := w_t + y_t x_t \)
Learning linear-threshold functions/3
Perceptron convergence theorem/1

[Bl62,No62,...]

Arbitrary sequence $S = (x_1, y_1), \ldots, (x_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

$\# \text{ of mistakes} \leq \inf_{\gamma > 0, \|u\|_2 = 1} \left( D_\gamma(u; S) + \frac{\sqrt{\sum_{t \in M} \|x_t\|^2}}{\gamma} \right)$

$D_\gamma(u; S) = \sum_{t \in M} \max\{0, 1 - y_t u^\top x_t / \gamma\}$

$M$ is set of mistaken trials $t$,

$D_\gamma(u; S) = \sum_{t \in M} \max\{0, 1 - y_t u^\top x_t / \gamma\}$
Learning linear-threshold functions/3
Perceptron convergence theorem/2

When $S$ is separated by $u : \|u\|_2 = 1$ with margin
$\gamma \leq y_t u^\top x_t \quad \forall t$
gets
$\# \text{ of mistakes} \leq \frac{R^2}{\gamma^2}$,

$\|x_t\| \leq R$

Pointwise bound:
Depends on radius $R$ and margin $\gamma$
Learning linear-threshold functions/4

The second-order Perceptron algorithm [CBCG05]

Keep weight vector $\mathbf{w}_t \in \mathbb{R}^n$ and matrix $S_t$

In trial $t$:

- Get instance $\mathbf{x}_t \in \mathbb{R}^n$
- Predict with $\hat{y}_t = \text{SGN}(\mathbf{w}_t^\top (aI + S_t)^{-1} \mathbf{x}_t) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$
- **If mistake then** update
  - $\mathbf{w}_{t+1} := \mathbf{w}_t + y_t \hat{\mathbf{x}}_t$
  - $S_{t+1} = S_t + \hat{\mathbf{x}}_t \hat{\mathbf{x}}_t^\top$, \hspace{1cm} $\hat{\mathbf{x}}_t = \mathbf{x}_t / \|\mathbf{x}_t\|$

Turns to first-order when $a \to \infty$
Learning linear-threshold functions
Second-order convergence theorem [Ge04]

When \( S = (\hat{x}_1, y_1), \ldots, (\hat{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\} \)
is separated by \( u \) with margin \( \gamma \leq y_t u^\top \hat{x}_t, \quad ||\hat{x}_t|| \leq 1 \ \forall t \)
gets

\[
\text{# of mistakes} \leq \frac{a + \sum_{i=1}^{n} \ln(1 + \frac{\lambda_i}{a})}{\gamma}
\]

More complicated bound in the nonseparable case

Pointwise bound:
Depends on eigenstructure \( \{\lambda_i\} \) of Gram matrix \( [\hat{x}_j^\top \hat{x}_k]_{j,k \in \mathcal{M}} \)
and linearly on inverse margin \( \gamma \)
Learning linear-threshold functions

Kernel Perceptron [FS98,...]

Keep pool of "support vectors" $M_t$

In trial $t$:

- Get instance $x_t \in \mathbb{R}^n$
- Predict with $\hat{y}_t = \text{sgn}(\sum_{i \in M_t} y_i K(x_i, x_t)) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$
- If mistake then update $M_{t+1} := M_t \cup \{t\}$
Learning linear-threshold functions
Kernel Perceptron convergence theorem

Arbitrary sequence $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

$$\# \text{ of mist.} \leq \inf_{\gamma > 0, f \in H_K, \|f\| = 1} \left( \begin{aligned} D_\gamma(f; S) + & \sqrt{\sum_{t \in \mathcal{M}} K(\mathbf{x}_t, \mathbf{x}_t)} \\ \text{"loss" of } f \\ \gamma \end{aligned} \right)$$

$H_K = \{ f(\cdot) = \sum_{t=1}^T \alpha_t K(\mathbf{x}_t, \cdot) : \alpha_t \in \mathbb{R} \}$,

$\mathcal{M}$ is set of mistaken trials $t$,

$D_\gamma(f; S) = \sum_{t \in \mathcal{M}} \max\{0, 1 - y_t f(\mathbf{x}_t)/\gamma\}$

Separable case:

$\# \text{ of mistakes} \leq R^2/\gamma^2$, \quad $K(\mathbf{x}_t, \mathbf{x}_t) \leq R^2$
Learning linear-threshold functions
Kernel Second-order Perceptron

Keep pool of ”support vectors” $\mathcal{M}_t$

In trial $t$:

- Get instance $\mathbf{x}_t \in \mathbb{R}^n$
- Predict with $\hat{y}_t = \text{sgn} \left( \mathbf{y}_t^\top \left( a \mathbf{I} + \left[ \hat{K}(\mathbf{x}_i, \mathbf{x}_j) \right]_{i,j \in \mathcal{M}_t} \right)^{-1} \mathbf{v}_t \right) \in \{-1, 1\}$,
- Get label $y_t \in \{-1, 1\}$
- If mistake then update $\mathcal{M}_{t+1} := \mathcal{M}_t \cup \{t\}$
Learning linear-threshold functions/9

Kernel Second-order convergence theorem

When $S = (x_1, y_1), \ldots, (x_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$ is separated by $f(\cdot) = \sum_{t=1}^{T} \alpha_t \hat{K}(x_t, \cdot)$, $\alpha_t \in \mathbb{R}$, with margin $\gamma \leq y_t f(x_t)$ $\forall t$ gets

$$\# \text{ of mist.} \leq \frac{a + \sum_i \ln(1 + \frac{\lambda_i}{a})}{\gamma},$$

$\lambda_i$ is $i$-th eigenvalue of (normalized) kernel Gram matrix $[\hat{K}(x_i, x_j)]_{i,j \in \mathcal{M}},$

$\mathcal{M}$ is set of mistaken trials
Learning linear-threshold functions
Second-order Perceptron: computational aspects

Time per trial

**Primal formulation**

compute \((aI + S_t)^{-1}\) based on \((aI + S_{t-1})^{-1}\)

\(O(n^2)\) extra time per trial

**Dual formulation**

compute \(\left(aI + [\hat{G}(\mathbf{x}_i, \mathbf{x}_j)]_{i,j \in \mathcal{M}_t}\right)^{-1}\)

based on \(\left(aI + [\hat{G}(\mathbf{x}_i, \mathbf{x}_j)]_{i,j \in \mathcal{M}_{t-1}}\right)^{-1}\)

\(O(|\mathcal{M}_t|^2)\) extra inner products (kernel evaluations) per trial

# of mistakes so far
Learning linear-threshold functions/11

Additive algorithms

An additive algorithm (e.g. first/second-order Perceptron):

- Relies on linear algebra
- Is rotation invariant (depends on data via angles)
- Can be easily kernelized ($\mathbf{x}_i^\top \mathbf{x}_j \rightarrow K(\mathbf{x}_i, \mathbf{x}_j)$)
- Has no bias for axes-parallel directions (no feature selection)
Learning linear-threshold functions/12
Nonadditive algorithms

- No linear algebra
- No rotation invariance
- Harder to kernelize
- Bias for sparse solutions (built-in feature selection)

Example: $p$-norm algorithms
Learning linear-threshold functions/13

$p$-norm algs [GLS01, GL99, Ge03]

Keep weight vector $\mathbf{w}_t \in \mathbb{R}^n$

In trial $t$:

- Get instance $\mathbf{x}_t \in \mathbb{R}^n$
- $\mathbf{f}(\cdot) = \nabla \frac{1}{2} \| \cdot \|_p^2$, $p \geq 2$
- Predict $\hat{y}_t = \text{SGN}(\mathbf{f}(\mathbf{w}_t)^\top \mathbf{x}_t) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$

- If mistake then update $\mathbf{w}_{t+1} := \mathbf{w}_t + y_t \mathbf{x}_t$

Notice:

- $p = 2$ gets (first-order) Perceptron
- $p = O(\ln n)$ gets Weighted Majority/Winnow [L88, LW94]
- $2 < p < O(\ln n)$ interpolates between the two extremes
Learning linear-threshold functions/14

$p$-norm Perceptron convergence theorem/1

[GLS01,GL99,Ge03]

Arbitrary sequence $S = (x_1, y_1), \ldots, (x_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

\[ \# \text{ mistakes} \leq \inf_{\gamma > 0, \|u\|_q = 1} \left( D_\gamma(u; S) + \frac{\sqrt{(p - 1) \sum_{t \in M} \|x_t\|^2_p}}{\gamma} \right) \]

$M$ is set of mistaken trials $t$,

\[ D_\gamma(u; S) = \sum_{t \in M} \max\{0, 1 - y_t u^\top x_t / \gamma\} \]
Learning linear-threshold functions/14

$p$-norm Perceptron convergence theorem/2

When $S$ is separated by $u : ||u||_q = 1$ with margin
\[ \gamma \leq y_t u^T x_t \forall t \quad \left( \frac{1}{p} + \frac{1}{q} = 1 \right) \]

dual norms

gets
\[ \# \text{ of mistakes} \leq (p - 1) \frac{R^2}{\gamma^2} \]

\[ ||x_t|| \leq R \]

Pointwise bound:
Depends on $p$-norm radius $R$
and $(q$-norm$)$ margin $\gamma$
Batch algorithm run on-line

\[ S = (x_1, y_1), ..., (x_T, y_T) \in \mathbb{R}^n \times \{-1, +1\} \]

\( A \) is generic \textbf{batch} classification alg.

In trial \( t \):

- train \( A \) on prefix \( S_{t-1} = (x_1, y_1), ..., (x_{t-1}, y_{t-1}) \)
- Get \( h_t(x) = A_{S_{t-1}}(x) \)
- Mistake if \( h_t(x_t) \neq y_t \)

Can count \# mistakes on sequence \( S \)
End of Part 1
Generalization bounds/1

Given

- class $\mathcal{H}$ of $\pm 1$ functions
- i.i.d. sequence $S = (X_1, Y_1), \ldots, (X_T, Y_T)$ over $\mathbb{R}^n \times \{-1, 1\}$,

want to compute hypothesis $\hat{H} = \hat{H}_S$ with small risk $\text{risk}(\hat{H}) = \mathbb{E}_{X,Y} \left[ \text{loss}(Y, \hat{H}(X)) \right]$

\[
\mathbb{P} \left( \text{risk}(\hat{H}) \leq \inf_{h \in \mathcal{H}} \text{risk}(h) + \epsilon \right) \geq 1 - \delta
\]
Generalization bounds/2: VC Uniform conv. [VC71]

Key quantity is empirical risk

\[ \text{risk}_{\text{emp}}(h) = \frac{1}{T} \sum_{t=1}^{T} \text{loss}(Y_t, h(X_t)) \]

VC-bound:

\[
\mathbb{P} \left( \sup_{h \in \mathcal{H}} |\text{risk}_{\text{emp}}(h) - \text{risk}(h)| \geq C \sqrt{\frac{d + \ln 1/\delta}{T}} \right) \leq \delta
\]

\[ \implies \hat{H} = \arg\inf_{h \in \mathcal{H}} \text{risk}_{\text{emp}}(h) \text{ is s.t.} \]

\[
\mathbb{P} \left( \text{risk}(\hat{H}) \leq \inf_{h \in \mathcal{H}} \text{risk}(h) + 2C \sqrt{\frac{d + \ln 2/\delta}{T}} \right) \geq 1 - \delta
\]
Generalization bounds/3:
Data-dep. uniform conv./1

\[ [\text{Ba98, BLM00, WSTSS99, BM02, ...}] \]

\[
\sqrt{\frac{d + \ln 2/\delta}{T}} \rightarrow C_T(S) + \sqrt{\frac{\ln 1/\delta}{T}}
\]

\( C_T(S) = C_T(S, \mathcal{H}) \)

is sample statistic:

\begin{align*}
&\text{empirical VC-entropy [BLM00, WSTSS99]} \\
&\text{Rademacher complexity [BM02]} \\
&\text{Maximum discrepancy [BLM00]} \\
&\ldots
\end{align*}

Stronger than VC since \( C_T(S) \approx \mathbb{E}[C_T(S)] \ll \sqrt{d/T} \)
Generalization bounds/3: 
Data-dep. uniform conv./2

Others (e.g., margin-based bounds for linear-threshold functions)  
[AKLL02, KP02, LSM01, SFBL98, ...]

\[ \mathbb{P} \left( \forall h \in \mathcal{H} : \text{risk}(h) \leq \text{risk}_{\text{emp}}(h) + C_T(h, S) + c \sqrt{\frac{\ln 1/\delta}{T}} \right) \geq 1 - \delta \]

Leave algorithmic problem of computing \( h \in \mathcal{H} \) optimizing trade-off

\( \text{risk}_{\text{emp}}(h) \) vs \( C_T(h, S) \)
Digression: martingales/1

Coin-tossing game:
$X_1, X_2, ..., X_t$ are $\pm 1$ i.i.d. variables with $P(X_t = +1) = 1/2$

Gambler’s strategy:
$L_t = L_t(X_1, X_2, ..., X_t)$ is gain (or loss) at time $t$
$L_1, L_2, ..., L_t$ are no longer independent

Game is fair if

$$E[L_{t+1} | X_1, ..., X_t] = 0 \quad \text{(w.p.1) } \forall t$$

- Sequence $L_1, L_2, ..., \text{ is martingale difference sequence} \quad \text{(w.r.t. } X_1, X_2, ...,)$

- Partial sums $S_t = L_1 + L_2 + ... + L_t$ is martingale \quad \text{(w.r.t. } X_1, X_2, ...,)
Digression: martingales/2

Laws of large numbers
(empirical average concentrates around mean)
extend to martingales

<table>
<thead>
<tr>
<th>Independent variables</th>
<th>Dependent variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Zero-mean) independent r.v.</td>
<td>Martingale diff. sequence</td>
</tr>
<tr>
<td>Sum of (zero-mean) indep. r.v.</td>
<td>Martingale</td>
</tr>
</tbody>
</table>

(Hoeffding-Azuma) If $L_1, L_2, ..., L_T$ is martingale difference sequence with bounded $L_t$

$$\frac{L_1 + L_2 + ... + L_T}{T} \approx 0$$ (with high probability)
On-line pointwise bounds

\[ S = (x_1, y_1), \ldots, (x_T, y_T) \]

Pointwise bounds so far:

Total \# mistakes \( A(S) \leq \text{some function}(S) \)

- \( n, R, \gamma \) (Halving)
- \( R, \gamma \) (1st Perc)
- \( \lambda_i, \gamma \) (2nd Perc)
- \( R, \gamma \) (dual)
- \( R, \gamma \) (p-norm)
On-line pointwise → i.i.d. data-dependent/1

Sweep through sequence of examples $S$ just once!

Get sequence of hypotheses
$H_0, H_1, H_2, ..., H_T$: $H_t = H_t((x_1, y_1), ..., (x_t, y_t))$

**Goal:** Extract one with small risk
Early ref: [L] (separate test set)
On-line pointwise $\to$ i.i.d. data-dependent/2

Which one?

1. **Last** one: $H_T$ (back to uniform convergence ...)

2. **Average** one: $\overline{H} = \frac{1}{T} \sum_{t=0}^{T} H_t \in [0, 1]$
   (convex upper bound on 0-1 loss)

3. **Best penalized** one:

   $$\text{risk}_{\text{emp}}(H_t, t + 1) = \frac{1}{T - t} \sum_{i=t+1}^{T} \text{loss}(Y_i, H_t(X_i))$$

   $$\hat{H} = \arg\min_{t=0...T-1} \left( \text{risk}_{\text{emp}}(H_t, t + 1) + \sqrt{\frac{1}{T - t} \ln \frac{T}{\delta}} \right)$$

   ![Diagram showing past and future time periods for $H_t$]
On-line pointwise $\rightarrow$ i.i.d. data-dependent/3

Proof technique/1

\[
\begin{align*}
H_0 &\quad (X_1, Y_1) \quad H_0 \\
H_1 &\quad (X_2, Y_2) \quad H_1 = H_1((X_1, Y_1)) \\
H_2 &\quad (X_3, Y_3) \quad H_2 = H_2((X_1, Y_1), (X_2, Y_2)) \\
\vdots &\quad \vdots \\
\end{align*}
\]

Build martingale kind of process

\[
L_t = L_t((X_1, Y_1), \ldots, (X_t, Y_t)) \\
= \text{loss}(Y_t, H_{t-1}(X_t)) - \text{risk}(H_{t-1}) \\
\mathbb{E}[\text{loss}(Y_t, H_{t-1}(X_t)) \mid (X_1, Y_1), \ldots, (X_{t-1}, Y_{t-1})]
\]
On-line pointwise $\rightarrow$ i.i.d. data-dependent/3
Proof technique/2

From the very definition of $L_t$
$L_1, L_2, \ldots, L_T$ is bounded ($|L_t| \leq 1$)
martingale difference sequence
w.r.t. $(X_1, Y_1), \ldots, (X_T, Y_T)$

Hoeffding-Azuma:

$$\frac{1}{T} \sum_{t=1}^{T} \left[ \text{loss}(Y_t, H_{t-1}(X_t)) - \text{risk}(H_{t-1}) \right] \approx 0$$
On-line pointwise $\rightarrow$ i.i.d. data-dependent/3

Proof technique/3

$$
\begin{align*}
\frac{1}{T} \sum_{t=1}^{T} \text{loss}(Y_t, H_{t-1}(X_t)) & \approx \frac{1}{T} \sum_{t=1}^{T} \text{risk}(H_{t-1}) \\
\text{MT} & \\
\text{# of mistakes} & \\
\end{align*}
$$

(*) (Hoeffding-Azuma) \hspace{1cm} [DGL96]

(**) bounded and convex (Jensen)

(***) general bounded (Chernoff-Hoeffding) \hspace{1cm} [DGL96]
On-line pointwise $\rightarrow$ i.i.d. data-dependent/4

Simplest bounds

Convex:
$$\mathbb{P} \left( \text{risk}(\overline{H}) \leq M_T + L \sqrt{\frac{2}{T} \ln \frac{2}{\delta}} \right) \geq 1 - \delta$$

bound on range of convex loss

More general:
$$\mathbb{P} \left( \text{risk}(\hat{H}) \leq M_T + 6 \sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta$$
On-line pointwise $\rightarrow$ i.i.d. data-dependent/5
Some applications: plug and play/1

Recall bound on Halving Algorithm for separable case:

$$M_T \leq \frac{1}{T} O \left( d \log(R/\gamma) \right)$$

Just plug back into

$$P \left( \text{risk}(\hat{H}) \leq M_T + 6 \sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta$$

Gets

$$P \left( \text{risk}(\hat{H}) \leq \frac{1}{T} O \left( n \log(R/\gamma) \right) + 6 \sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta$$

Similar to [HG02]
On-line pointwise → i.i.d. data-dependent

Some applications: plug and play

Recall bound on Kernel Perceptron:

$$M_T \leq \inf_{\gamma > 0, \ f \in \mathcal{H}_K, \ |f| = 1} \frac{1}{T} \left( D_\gamma(f; S) + \frac{\sqrt{\sum_{t \in \mathcal{M}} K(x_t, x_t)}}{\gamma} \right)$$

Separable case:

$$M_T \leq \frac{1}{T} \max_{t \in \mathcal{M}} K(x_t, x_t)$$

Plug back into

$$\mathbb{P} \left( \text{risk}(\hat{H}) \leq M_T + 6 \sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta$$

Similar to [BM02] for SVM
On-line pointwise $\rightarrow$ i.i.d. data-dependent/5
Some applications: plug and play/3

Recall bound on Kernel Second-order Perceptron
(separable case)

\[ M_T \leq \frac{1}{T} \alpha + \sum_i \ln(1 + \frac{\lambda_i}{\alpha}), \]

Plug into

\[ \mathbb{P} \left( \text{risk}(\hat{H}) \leq M_T + 6 \sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta \]

Similar to [WSTSS99] for SVM
On-line pointwise $\rightarrow$ i.i.d. data-dependent

Some applications: plug and play

$A$ is your favourite batch classification alg.

Run it in on-line fashion and count $\#$ of mistakes

Still get

$$
P \left( \text{risk}(\hat{H}) \leq M_T + 6 \sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta
$$

No direct mention of, e.g., “complexity of function classes”

Try it yourself with other specific alg.s.
On-line pointwise $\rightarrow$ i.i.d. data-dependent/6

Remarks

These bounds:

- are algorithm-specific (NO uniform convergence arguments, closer in spirit to algorithmic stability/luckiness) \[BE02,HeWi02,\ldots\]
- proven by simple large deviation on martingales
- refer to efficient algs (on-line, one sweep)
- are tight (I believe ...)
- Are widely applicable (in principle)
On-line pointwise → i.i.d. data-dependent
Refinements

\[ \mathbb{P} \left( \text{risk}(\hat{H}) \leq \min_{t=0\ldots T-1} \left( M_{t,T} + 6 \sqrt{\frac{1}{T-t} \ln \frac{T}{\delta}} \right) \right) \geq 1 - \delta, \]

where \( M_{t,T} = \frac{1}{T-t} \sum_{i=t+1}^{T} \text{loss}(Y_i, H_{i-1}(X_i)) \) (loss on suffix)

Basically “\( \min_{t=0\ldots T-1} \)” replaces “\( t = 0 \)”
On-line pointwise → i.i.d. data-dependent

Refinements

\[
\mathbb{P} \left( \text{risk}(\hat{H}) \leq M_T + O \left( \frac{1}{T} \ln \frac{T}{\delta} + \sqrt{\frac{M_T}{T} \ln \frac{T}{\delta}} \right) \right) \geq 1 - \delta,
\]

\[
\hat{H} = \arg\min_{t=0 \ldots T-1} \left( \text{risk}_{emp} + \frac{1}{T-t} \ln \frac{T}{\delta} + \sqrt{\frac{\text{risk}_{emp}}{T-t} \ln \frac{T}{\delta}} \right)
\]

\[
\text{risk}_{emp} = \text{risk}_{emp}(H_t, t + 1) = \frac{1}{T-t} \sum_{i=t+1}^{T} \text{loss}(Y_i, H_t(X_i))
\]

(Uses Bernstein-type inequalities for martingales) \[\text{[F75,DZ01]}\]

Can be combined with “Refinement/1”
Conclusions

- Pointwise bounds for on-line algorithms directly turn to (tight) data-dependent i.i.d. bounds
- Easy plug and play
- Resulting algs. are still as efficient as on-line (one epoch over training sequence)
- Simple proofs, algorithm-specific, no uniform convergence
- Can be immediately extended to regression frameworks