8. Online learning

COMP0078: Supervised Learning

Carlo Ciliberto

(Slides thanks to Mark Herbster)

University College London
Department of Computer Science

Batch versus Online learning

Batch

Model: There exists **training** data set (sampled **IID**)

Aim: To build a classifier from the training data that predicts well on new data (*from same distribution*)

Evaluation metric: *Generalization error*

Online

Model: There exists an **online sequence** of data (*usually no distributional assumptions*)

Aim: To sequentially predict and update a classifier to predict well on the sequence (i.e. there is no training and test set distinction)

Evaluation metric: *Cumulative error*

Note

There are a variety of models for online learning. Here we focus on the so-called *worst-case* model. Alternately distributional assumption may be made on the data sequence. Also sometimes the phrase "online learning" is used to refer to "online optimisation" that is to use online learning type algorithms as a *training* method for a batch classifier.

Why online learning?

Pragmatically

- “Often” fast algorithms
- “Often” small memory footprint
- “Often” no “statistical” assumptions required e.g. IID-ness
- As a *training* method for **“BIG DATA”** batch classifiers

Theoretically (learning performance guarantees)

- Non-asymptotic
- No statistical assumptions
- There exist techniques to convert *cumulative error* guarantees to *generalisation error* guarantees

Today

Our focus today is on three foundational online “hypotheses” classes.

- Learning with experts
    1. Halving algorithm
    2. Weighted Majority algorithm
    3. Refining and generalising the experts model
- Learning linear classifiers
    - Perceptron

Experts

Part I

Learning with Expert Advice

On-Line Learning with expert advice (1) [V90,LW94,HKW98]

Model: There exists an **online sequence** of data

$$S = S_m = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \qquad \text{with} \qquad (x, y) \in \{0,1\}^n \times \{0,1\}.$$

Interpretation: The vector $\mathbf{x}_t$ is the set of predictions from n experts about an outcome y_t, where expert i predicts $x_{t,i} \in \{0,1\}$ at time t. Each expert at time t is aiming to predict y_t.

What is an "expert"? Example: human experts or the predictions of n separate algorithms.

	experts						
	E_1	E_2	E_3	E_n	prediction	true label	loss
day 1	1	1	0	0	0	1	1
day 2	1	0	1	0	1	0	1
day 3	0	1	1	1	1	1	0
day t	$x_{t,1}$	$x_{t,2}$	$x_{t,3}$	$x_{t,n}$	$\hat{y}_t$	y_t	$\lvert y_t - \hat{y}_t \rvert$

Goal: Find a *"Master"* algorithm to combine the predictions $\mathbf{x}_t$ of the n experts (based on past perf.) to predict $\hat{y}_t$ an estimate of y_t.

On-Line Learning with experts (2)

Protocol of the Master Algorithm

```
For $t = 1, \ldots, m$:
    Get instance              $\mathbf{x}_t \in \{0,1\}^n$
    Predict                   $\hat{y}_t \in \{0,1\}$
    Get label                 $y_t \in \{0,1\}$
    Incur loss (mistakes)     $|y_t - \hat{y}_t|$
```

Evaluation metric: The loss (mistakes) of Master Algorithm A on sequence S is just

$$L_A(S) := \sum_{t=1}^{m} |y_t - \hat{y}_t|$$

where $\hat{y}_t = A(S_{t-1})(x_t)$ is the output of the online algorithm A trained (online) on S_{t-1} and evaluated on x_t

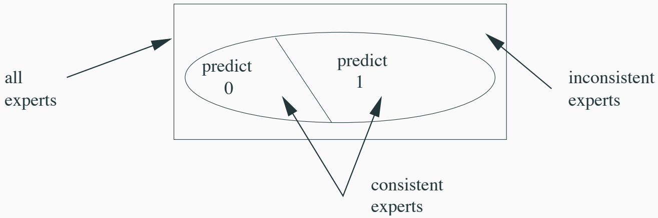
Our Goal: Design master algorithms with "small loss".

Special Case: The (Unknown) “Perfect” Expert

Let’s consider the special setting where there exists at least one expert that is never wrong...

...how “fast” could we find them?

A Solution : Halving Algorithm



The master algorithm:

- Keeps track of only the consistent experts (those that never made a mistake so far)
- Predicts according to the majority vote
- Eliminates wrong experts after each prediction.

Question: How many mistakes does it make, at most?

A run of the Halving Algorithm

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	$\hat{y}$	y	loss
1	1	0	0	1	1	0	0	1	0	1
x	x	0	1	x	x	1	1	1	1	0
x	x	x	1	x	x	0	0	0	1	1
x	x	x	↑	x	x	x	x			
			consistent							

Question: How many mistakes does it make, at most?

A run of the Halving Algorithm

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	$\hat{y}$	y	loss
1	1	0	0	1	1	0	0	1	0	1
x	x	0	1	x	x	1	1	1	1	0
x	x	x	1	x	x	0	0	0	1	1
x	x	x	↑	x	x	x	x			
			consistent							

Question: How many mistakes does it make, at most?

Answer: For any sequence with a consistent expert, the Halving Algorithm makes $\leq \log_2 n$ mistakes.

Exercise: Prove this!

What if no expert is consistent?

Notation

- Recall $L_A(S) := \sum_{t=1}^{m} |y_t - \hat{y}_t|$ is the loss of algorithm A on S
- Denote the loss of i-th expert E_i as

$$L_i(S) := \sum_{t=1}^{m} |y_t - x_{t,i}|$$

Aim

Bounds of the form:

$$\forall S : \quad L_A(S) \; \leq \; a \; \underbrace{\min_i L_i(S)}_{\text{Best Expert!}} + b \; \log(n)$$

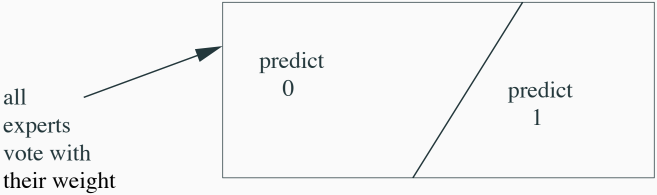
where a, b are “small” constants

Comment: These are known as “Regret” or “Worst-case” loss bounds, i.e., bounds that hold in any case (even the “worst-case”).

A Solution: Weighted Majority Algorithm [LW94]

The Master algorithm:

- Can't eliminate experts!
- Keeps track of how reliable each espert is
  (By keeping track of a weight w_i for each expert)



- Predicts according to the larger (weighted) vote
- Weights of wrong experts are multiplied by $\beta \in [0, 1)$

Number of mistakes of the WM algorithm

$$\begin{aligned} M &= \#\text{ mistakes of master algorithm at the "end"} \\ M_{t,i} &= \#\text{ mistakes of expert } E_i \text{ at the start of trial } t \\ M_i = M_{m+1,i} &= \#\text{ of total mistakes of expert } E_i \\ w_{t,i} &= \beta^{M_{t,i}} \text{ weight of } E_i \text{ at beginning of trial } t\ (w_{1,i} = 1) \\ W_t &= \sum_{i=1}^{n} w_{t,i} \text{ total sum of weights at the start of trial } t \end{aligned}$$

For each trial... (Minority) $\leq \frac{1}{2} W_t$ (Majority) $\geq \frac{1}{2} W_t$

If the Master algorithm:

- ...is right, then the weights of the "minority" experts are multiplied by β:

$$W_t = Minority + Majority \geq \beta \cdot Minority + Majority = W_{t+1}$$

- ...makes a mistake, then majority multiplied by β:

$$W_{t+1} \leq \underbrace{\tfrac{1}{2} W_t}_{\text{minority}} + \beta \underbrace{\tfrac{1}{2} W_t}_{\text{majority}} \qquad (why?)$$

Number of mistakes of the WM algorithm – Continued-1

Hence, $W_{t+1} \leq \frac{1+\beta}{2} W_t$.

Number of mistakes of the WM algorithm – Continued-1

Hence, $W_{t+1} \leq \frac{1+\beta}{2} W_t$. Therefore, we have

$$\underbrace{W_{m+1}}_{\substack{\text{total final}\\ \text{weight}}} \leq \left(\frac{1+\beta}{2}\right)^M \underbrace{W_1}_{\substack{n\\ \text{\# of experts!}}}$$

At the same time,

$$W_{m+1} = \sum_{j=1}^{n} w_{m+1,j} = \sum_{j=1}^{n} \beta^{M_j} \geq \beta^{M_i}$$

Combining the upper and lower bounds...

$$\left(\frac{1+\beta}{2}\right)^M n \geq \beta^{M_i}$$

Number of mistakes of the WM algorithm – Continued-2

Taking the log and solving for M...

$$M \quad \leq \quad \frac{\ln \frac{1}{\beta}}{\ln \frac{2}{1+\beta}} \, M_i + \frac{1}{\ln \frac{2}{1+\beta}} \, \ln n$$

For example, choosing $\beta = 1/e$

$$M \leq \underbrace{2.63}_{a} \, \min_i M_i \; + \; \underbrace{2.63}_{b} \ln n$$

For all sequences, the loss of master algorithm is comparable to the loss of the best expert.

Refining and generalising the experts model – 1

More generally we would like to obtain *regret* bounds for arbitrary loss functions $L : \mathcal{Y} \times \hat{\mathcal{Y}} \to [0, +\infty]$. Making our notion of regret more precise we would like guarantees of the form,

$$L_A(S) - \min_{i \in [n]} L_i(S) \leq o(m)\,,$$

where the right-hand side is termed regret since it is how much we "regret" predicting with the algorithm as opposed to the optimal predictor on the data sequence.

Here $o(m)$ denotes some function sublinear in m (the number of examples in S) and possibly dependent on other parameters.

Why $o(m)$?

Refining and generalizing the experts model - 2

Remember that $L_A(S)$ is the cumulative sum of the errors incurred by A.

Therefore $\frac{1}{m}L_A(S)$ is the average error incurred **(so far)** by A.

If the sublinear Regret bound holds,

$$\frac{L_A(S) - \min_{i \in [n]} L_i(S)}{m} \leq \frac{o(m)}{m} \, ,$$

Since $\frac{o(m)}{m} \to 0$ for $m \to \infty$, this implies that asymptotically our algorithm incurs in the same average loss as the average loss of the best expert.

Refining and generalising the experts model – 3

In the following we will show two example of regret bounds generalising the analysis of the weighted majority algorithm.

1. For a loss function $L : \{0,1\} \times [0,1] \to [0,+\infty]$ the *entropic loss*

$$L(y,\hat{y}) = y \ln \frac{y}{\hat{y}} + (1-y) \ln \frac{1-y}{1-\hat{y}}$$

2. For an arbitrary bounded loss function $L : \mathcal{Y} \times \hat{\mathcal{Y}} \to [0,B]$.

For the first the regret will be the small constant $\log(n)$ for the second the regret will be $O(\sqrt{m \log n})$.

A regret bound for the entropic (log) loss

Unlike in the case of the weighted majority we will now:

- allow predictions in $[0, 1]$ rather than just $\{0, 1\}$, and
- predict with the *weighted average* rather than the "majority".

At trial t, the expert i will have weight $w_{t,i} = \beta^{\,L_{t,i}} = e^{-\eta\, L_{t,i}}$ with:

- where $L_{t,i}$ is the cumulative loss of E_i at the start of trial t
- and η is a positive learning rate

The Master predicts with the weighted average (dot product)

$$\hat{y}_t \quad = \quad \sum_{i=1}^{n} \underbrace{\frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}}}_{\substack{v_{t,i} \\ \text{normalized} \\ \text{weights}}} \; x_{t,i} = \boldsymbol{v}_t \cdot \mathbf{x}_t$$

where $x_{t,i}$ is the prediction of E_i in trial t

Set the initial weights $\mathbf{w}_1 = (1, \ldots, 1)$ and thus $\mathbf{v}_1 == (\frac{1}{n}, \ldots, \frac{1}{n})$.

The Weighted Average Algorithm – Summary

```
WA Algorithm
Initialise : $\mathbf{v}_1 = (\frac{1}{n}, \ldots, \frac{1}{n})$, $L_{\text{WA}} := 0$, $\mathbf{L} := \mathbf{0}$,
Input:   $\eta \in (0, \infty)$, Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

For $t = 1, \ldots, m$ Do
    Receive instance  $\mathbf{x}_t \in [0,1]^n$
    Predict           $\hat{y}_t := \mathbf{v}_t \cdot \mathbf{x}_t$
    Receive label     $y_t \in [0,1]$
    Incur loss        $L_{\text{WA}} := L_{\text{WA}} + L(y_t, \hat{y}_t)$,
                      $L_i := L_i + L(y_t, x_{t,i})$ $(i \in [n])$
    Update            $v_{t+1,i} := \frac{v_{t,i} e^{-\eta L(y_t, x_{t,i})}}{\sum_{j=1}^{n} v_{t,j} e^{-\eta L(y_t, x_{t,j})}}$ for $i \in [n]$.
```

Weighted Average Algorithm - Theorem

Theorem

For all sequences of examples

$$S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \qquad \text{with} \qquad (x, y) \in [0,1]^n \times [0,1]$$

the regret of the *weighted average* WA algorithm is

$$L_{\mathrm{WA}}(S) - \min_i L_i(S) \leq \underbrace{1/\eta}_{b} \ln(n)$$

with square and entropic loss for $\eta = 1/2$ and $\eta = 1$ respectively.

Constant b as dependent on loss function

Loss		$b = 1/\eta$
entropic	$L_{\mathrm{en}}(y, \hat{y}) = y \ln \frac{y}{\hat{y}} + (1-y) \ln \frac{1-y}{1-\hat{y}}$	1
square	$L_{\mathrm{sq}}(y, \hat{y}) = (y - \hat{y})^2$	2

- For simplicity, we will prove only for entropic loss when $\mathcal{Y} := \{0, 1\}$ and $\hat{\mathcal{Y}} := [0, 1]$. The result holds for many loss function (sufficient smoothness and convexity with different η and b). See [KW99] for proof.

Weighted Average Algorithm - Proof

Notation: $\Delta_n := \{\mathbf{x} \in [0,1]^n : \sum_{i=1}^n x_i = 1\}$. Let $d : \Delta_n \times \Delta_n \to [0,\infty]$ be the rel. entropy $d(\mathbf{u},\mathbf{v}) := \sum_{i=1}^n u_i \ln \frac{u_i}{v_i}$. Note: $L_{\text{en}}(y,\hat{y}) = d((y,1-y),(\hat{y},1-\hat{y}))$.

Weighted Average Algorithm - Proof

Notation: $\Delta_n := \{\mathbf{x} \in [0,1]^n : \sum_{i=1}^n x_i = 1\}$. Let $d : \Delta_n \times \Delta_n \to [0,\infty]$ be the rel. entropy $d(\mathbf{u},\mathbf{v}) := \sum_{i=1}^n u_i \ln \frac{u_i}{v_i}$. Note: $L_{\mathrm{en}}(y,\hat{y}) = d((y,1-y),(\hat{y},1-\hat{y}))$.

Proof – 1

We first prove the following "progress versus regret" equality.

$$L_{\mathrm{en}}(y_t,\hat{y}_t) - \sum_{i=1}^n u_i L_{\mathrm{en}}(y_t,x_{t,i}) = d(\mathbf{u},\mathbf{v}_t) - d(\mathbf{u},\mathbf{v}_{t+1}) \text{ for all } \mathbf{u} \in \Delta_n\,. \qquad (1)$$

Weighted Average Algorithm - Proof

Notation: $\Delta_n := \{\mathbf{x} \in [0,1]^n : \sum_{i=1}^n x_i = 1\}$. Let $d : \Delta_n \times \Delta_n \to [0,\infty]$ be the rel. entropy $d(\mathbf{u},\mathbf{v}) := \sum_{i=1}^n u_i \ln \frac{u_i}{v_i}$. Note: $L_{\text{en}}(y,\hat{y}) = d((y,1-y),(\hat{y},1-\hat{y}))$.

Proof – 1

We first prove the following "progress versus regret" equality.

$$L_{\text{en}}(y_t,\hat{y}_t) - \sum_{i=1}^n u_i L_{\text{en}}(y_t,x_{t,i}) = d(\mathbf{u},\mathbf{v}_t) - d(\mathbf{u},\mathbf{v}_{t+1}) \text{ for all } \mathbf{u} \in \Delta_n\,. \qquad (1)$$

Observe that

$$d(\mathbf{u},\mathbf{v}_t) - d(\mathbf{u},\mathbf{v}_{t+1}) = \sum_{i=1}^n u_i \ln \frac{v_{t+1,i}}{v_{t,i}}$$

Let $y_t = 1$. Then (using $L_{\text{en}}(1,x) = -\ln x$)

$$\sum_{i=1}^n u_i \ln \frac{v_{t+1,i}}{v_{t,i}} = \sum_{i=1}^n u_i \ln \frac{\frac{v_{t,i} e^{-L_{\text{en}}(1,x_{t,i})}}{\sum_{j=1}^n v_{t,j} e^{-L_{\text{en}}(1,x_{t,j})}}}{v_{t,i}} = \sum_{i=1}^n u_i \ln \frac{\frac{v_{t,i} x_{t,i}}{\sum_{j=1}^n v_{t,j} x_{t,j}}}{v_{t,i}} = \sum_{i=1}^n u_i \ln \frac{x_{t,i}}{\hat{y}_t}$$

$$= \left(\sum_{i=1}^n u_i \ln x_{t,i}\right) - \ln(\hat{y}_t) = L_{\text{en}}(y_t,\hat{y}_t) - \sum_{i=1}^n u_i L_{\text{en}}(y_t,x_{t,i})$$

by symmetry we also have the case $y = 0$ demonstrating (1).

Weighted Average Algorithm - Proof continued

Proof – 2

Now observe that (1) is a telescoping equality and we have

$$\sum_{t=1}^{m} L_{\text{en}}(y_t, \hat{y}_t) - \sum_{t=1}^{m} \sum_{i=1}^{n} u_i L_{\text{en}}(y_t, x_{t,i}) = d(\mathbf{u}, \mathbf{v}_1) - d(\mathbf{u}, \mathbf{v}_{m+1})$$

Now since the above holds for any $\mathbf{u} \in \Delta_n$ in particular the unit vectors $(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$ and then if we upper bound by noting that $d(\mathbf{u}, \mathbf{v}_1) \leq \ln n$ and $-d(\mathbf{u}, \mathbf{v}_{m+1}) \leq 0$ we have proved theorem. □

HEDGE **Algorithm**

HEDGE was introduced in [FS97], generalising the weighted majority analysis to the *allocation* setting.

Allocation setting

On each trial the learner plays an allocation $\mathbf{v}_t \in \Delta_n$, then nature returns a loss vector $\boldsymbol{\ell}_t$. I.e., the loss of expert i is $\ell_{t,i}$.

Two models for the learner's play (HA-1,HA-2):

1. [HA-1]:We simply incur loss so that $L_A(t) := \mathbf{v}_t \cdot \boldsymbol{\ell}_t$.
2. [HA-2]:Alternately learner randomly selects an action $\hat{y} \in [n]$ according to the discrete distribution over $[n]$ so that $\text{Prob}(j) := v_{t,j}$. Thus
$$E[L_{HA} \text{ on trial } t] = E_{\hat{y}_t \sim \mathbf{v}_t}[\ell_{t,\hat{y}}] = \mathbf{v}_t \cdot \boldsymbol{\ell}_t \,.$$

- Observe that this setting can *simulate* the setting where we receive side-information $\mathbf{x}_t$ and have a fixed loss function.
- For the randomised setting the "mechanism" generating the loss vectors $\boldsymbol{\ell}_t$ must be oblivious to the learner's selection $\hat{y}$ until trial $t+1$.

The Hedge Algorithm (HA) – Summary

Hedge Algorithm (HA-1)

```
Initialise : $\mathbf{v}_1 := (\frac{1}{n}, \ldots, \frac{1}{n})$, $L_{\text{HA}} := 0$, $\mathbf{L} := \mathbf{0}$ ; Select: $\eta \in (0, \infty)$,
For $t = 1$ To $m$ Do
    Predict $\mathbf{v}_t \in \Delta_n$
    Receive loss $\boldsymbol{\ell}_t \in [0,1]^n$
    Incur loss $L_{\text{HA}} := L_{\text{HA}} + \mathbf{v}_t \cdot \boldsymbol{\ell}_t$, $L_i := L_i + \ell_{t,i}$ $(i \in [n])$
    Update Weights $v_{t+1,i} := \frac{v_{t,i} e^{-\eta \ell_{t,i}}}{\sum_{j=1}^{n} v_{t,j} e^{-\eta \ell_{t,j}}}$ for $i \in [n]$.
```

Hedge Algorithm (HA-2)

```
Initialise : $\mathbf{v}_1 := (\frac{1}{n}, \ldots, \frac{1}{n})$, $L_{\text{HA}} := 0$ $\mathbf{L} := \mathbf{0}$ ; Select: $\eta \in (0, \infty)$,
For $t = 1$ To $m$ Do
    Predict $\mathbf{v}_t \in \Delta_n$ and sample $\hat{y}_t \sim \mathbf{v}_t$
    Receive loss $\boldsymbol{\ell}_t \in [0,1]^n$
    Incur loss $E[L_{\text{HA}}] := E[L_{\text{HA}}] + \mathbf{v}_t \cdot \boldsymbol{\ell}_t$, $L_i := L_i + \ell_{t,i}$ $(i \in [n])$
    Update Weights $v_{t+1,i} := \frac{v_{t,i} e^{-\eta \ell_{t,i}}}{\sum_{j=1}^{n} v_{t,j} e^{-\eta \ell_{t,j}}}$ for $i \in [n]$.
```

HEDGE - Theorem

Theorem Hedge (Bound) [LW94,FS97]

For all sequence of loss vectors

$$S = \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m \in [0,1]^n$$

the regret of the *Hedge* HA-2 algorithm with $\eta = \sqrt{2 \ln n / m}$ is

$$E[L_{\mathrm{HA}}(S)] - \min_i L_i(S) \leq \sqrt{2m \ln n}\,.$$

Hedge Theorem - Proof (1)

Proof – 1

We first prove the following "progress versus regret" inequality.

$$\mathbf{v}_t \cdot \boldsymbol{\ell}_t - \mathbf{u} \cdot \boldsymbol{\ell}_t \leq \frac{1}{\eta}\left(d(\mathbf{u}, \mathbf{v}_t) - d(\mathbf{u}, \mathbf{v}_{t+1})\right) + \frac{\eta}{2} \sum_{i=1}^{n} v_{t,i} \ell_{t,i}^2 \text{ for all } \mathbf{u} \in \Delta_n \,. \tag{2}$$

Let $Z_t := \sum_{i=1}^{n} v_{t,i} \exp(-\eta \ell_{t,i})$, so that $v_{t+1,i} = v_{t,i}/Z_t$. Observe that

$$\begin{aligned}
d(\mathbf{u}, \mathbf{v}_t) - d(\mathbf{u}, \mathbf{v}_{t+1}) &= \sum_{i=1}^{n} u_i \ln \frac{v_{t+1,i}}{v_{t,i}} \\
&= -\eta \sum_{i=1}^{n} u_i \ell_{t,i} - \sum_{i=1}^{n} u_i \ln Z_t \\
&= -\eta \mathbf{u} \cdot \boldsymbol{\ell}_t - \ln \sum_{i=1}^{n} v_{t,i} \exp(-\eta \ell_{t,i}) \\
&\geq -\eta \mathbf{u} \cdot \boldsymbol{\ell}_t - \ln \sum_{i=1}^{n} v_{t,i}(1 - \eta \ell_{t,i} + \frac{1}{2}\eta^2 \ell_{t,i}^2) && (3) \\
&= -\eta \mathbf{u} \cdot \boldsymbol{\ell}_t - \ln(1 - \eta \mathbf{v}_t \cdot \boldsymbol{\ell}_t + \frac{1}{2}\eta^2 \sum_{i=1}^{n} v_{t,i} \ell_{t,i}^2) \\
&\geq \eta(\mathbf{v}_t \cdot \boldsymbol{\ell}_t - \mathbf{u} \cdot \boldsymbol{\ell}_t) - \frac{1}{2}\eta^2 \sum_{i=1}^{n} v_{t,i} \ell_{t,i}^2 && (4)
\end{aligned}$$

Using inequalities $e^{-x} \leq 1 - x + \frac{x^2}{2}$ for $x \geq 0$ and $\ln(1+x) \leq x$ for (3) and (4) demonstrating (2).

Hedge Theorem - Proof (2)

Proof – 2

Summing over t and rearranging we have

$$\sum_{t=1}^{m} (\mathbf{v}_t \cdot \boldsymbol{\ell}_t - \mathbf{u} \cdot \boldsymbol{\ell}_t) \leq \frac{1}{\eta} (d(\mathbf{u}, \mathbf{v}_1) - d(\mathbf{u}, \mathbf{v}_{m+1})) + \frac{\eta}{2} \sum_{t=1}^{m} \sum_{i=1}^{n} v_{t,i} \ell_{t,i}^2$$

$$\leq \frac{\ln n}{\eta} + \frac{\eta}{2} \sum_{t=1}^{m} \sum_{i=1}^{n} v_{t,i} \ell_{t,i}^2 \tag{5}$$

Now since since the above holds for any $\mathbf{u} \in \Delta_n$ it then holds in particular for the unit vectors $(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$ and then we upper bound by noting that $d(\mathbf{u}, \mathbf{v}_1) \leq \ln n$, $-d(\mathbf{u}, \mathbf{v}_{m+1}) \leq 0$, and $\sum_{t=1}^{m} \sum_{i=1}^{n} v_{t,i} \ell_{t,i}^2 \leq m$. Finally we "tune" by choosing $\eta = \sqrt{2 \ln n / m}$ and we have proved theorem. □

Question: how can we the above to prove a theorem if the loss is now in the range $[0, B]$?

Comments

- Easy to combine many pretty good experts (algorithms) so that Master is guaranteed to be almost as good as the best
- Bounds logarithmic in number of experts. Use many experts! Limits only in computational resources.
- Observe updating is multiplicative
- There exist algorithms [V90] with improved (smaller) constants for b with a more complex prediction strategies than weighted average.

Next: So far we have given bounds which grow slowly in the number of experts. The only significant drawback is potentially computational if we wish to work with large classes of experts. With this is mind we may wish to work with *structured* sets of experts for either computational advantages or advantages in bound. We consider now linear combinations of experts (linear classifiers).

Part II

Online learning of linear classifiers

A more general setting (1)

Instance	Prediction of alg A	Label	Loss of alg A	
$\mathbf{x}_1$	$\hat{y}_1$	y_1	$L(y_1, \hat{y}_1)$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\mathbf{x}_t$	$\hat{y}_t$	y_t	$L(y_t, \hat{y}_t)$	Sequence of examples
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\mathbf{x}_m$	$\hat{y}_m$	y_m	$L(y_m, \hat{y}_m)$	
		Total Loss	$L_A(S)$	

$S = (\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$

Comparison class $\mathcal{U} = \{\boldsymbol{u}\}$ (AKA hypothesis space, concept class)

Relative loss (Regret)

$$L_A(S) - \inf_{\{\boldsymbol{u} \in \mathcal{U}\}} Loss_{\boldsymbol{u}}(S)$$

Goal: Bound relative loss for arbitrary sequence S

A more general setting (2)

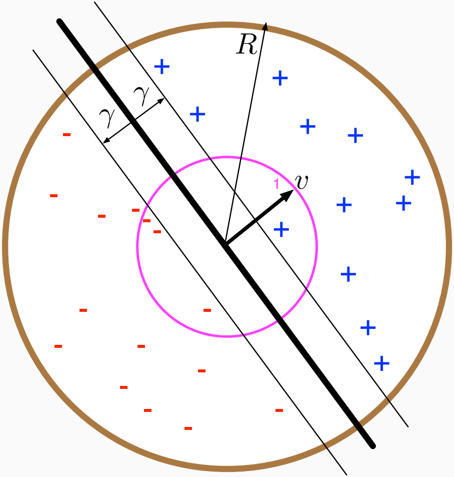
Now

- We consider the case where $\mathcal{U}$ is a set of *linear threshold* function.
- For simplicity we will focus on the case where there exists a $\mathbf{u} \in \mathcal{U}$ s.t. $Loss_{\boldsymbol{u}}(S) = 0$. This is known as *realizable* case. Compare to the previously considered halving algorithm versus weighted majority algorithm.

Perceptron

The Perceptron set-up

Assumption: Data is linearly separable by some margin γ. Hence exists a hyperplane with normal vector $\mathbf{v}$ such that



1. $\|\mathbf{v}\| = 1$
2. All examples $(\mathbf{x}_t, y_t)$
   - $\forall y_t \;\; y_t \in \{-1, +1\}$.
   - $\forall \mathbf{x}_t, \;\; \|\mathbf{x}_t\| \leq R$.
3. $\forall (\mathbf{x}_t, y_t), y_t(\mathbf{x}_t \cdot \mathbf{v}) \geq \gamma$

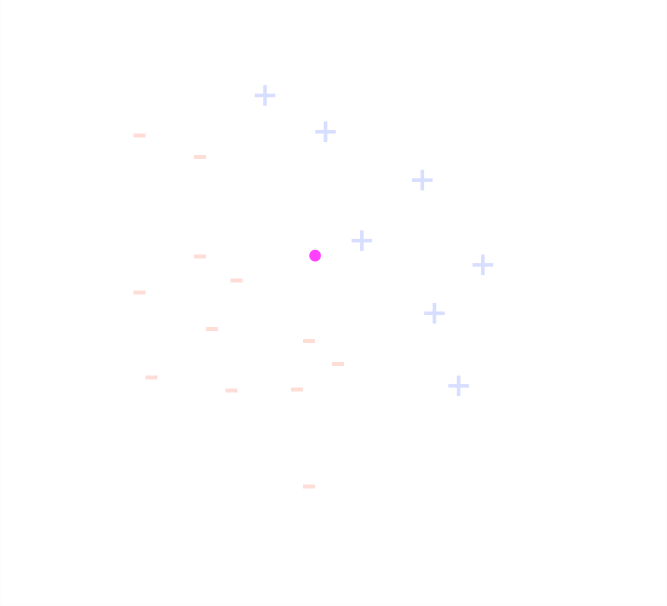
The Perceptron learning algorithm

Perceptron Algorithm

```
Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$    with    $(x, y) \in \mathbb{R}^n \times \{-1, 1\}$

1. Initialise $\mathbf{w}_1 = \vec{0}$; $M_1 = 0$.
2. For $t = 1$ to $m$ do
3. Receive pattern: $\mathbf{x}_t \in \mathbb{R}^n$
4. Predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
5. Receive label: $y_t$
6. If mistake $(\hat{y}_t y_t \leq 0)$
      • Then Update $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$; $M_{t+1} = M_t + 1$
7. Else $\mathbf{w}_{t+1} = \mathbf{w}_t$; $M_{t+1} = M_t$.
8. End For
```

Example: trace for the Perceptron algorithm



Bound on number of mistakes

- The number of mistakes that the perceptron algorithm can make is at most $\left(\frac{R}{\gamma}\right)^2$.
- Proof by combining upper and lower bounds on $\|\mathbf{w}\|$.

Pythagorean Lemma

On trials where "mistakes" occur we have the following inequality,

Lemma: **If** $(\mathbf{w}_t \cdot \mathbf{x}_t)y_t < 0$ **then** $\left\|\mathbf{w}_{t+1}\right\|^2 \leq \|\mathbf{w}_t\|^2 + \|x_t\|^2$

Proof:

$$\begin{aligned}\left\|\mathbf{w}_{t+1}\right\|^2 &= \|\mathbf{w}_t + y_t\mathbf{x}_t\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2(\mathbf{w}_t \cdot \mathbf{x}_t)y_t + \|\mathbf{x}_t\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + \|\mathbf{x}_t\|^2\end{aligned}$$

Upper bound on $\|\mathbf{w}_t\|$

Lemma: $\|\mathbf{w}_t\|^2 \leq M_t R^2$

Proof: By induction

- Claim: $\|\mathbf{w}_t\|^2 \leq M_t R^2$
- Base: $M_1 = 0$, $\|\mathbf{w}_1\|^2 = 0$
- Induction step (assume for t and prove for $t+1$) when we have a mistake on trial t:

$$\|\mathbf{w}_{t+1}\|^2 \leq \|\mathbf{w}_t\|^2 + \|\mathbf{x}_t\|^2 \leq \|\mathbf{w}_t\|^2 + R^2 \leq (M_{t+1})R^2$$

Here we used the Pythagorean lemma. If mistake $M_{t+1} = M_t + 1$ else there is no mistake, then trivially $\mathbf{w}_{t+1} = \mathbf{w}_t$ and $M_{t+1} = M_t$.

Lower bound on $\|\mathbf{w}_t\|$

Lemma: $M_t\gamma \leq \|\mathbf{w}_t\|$

Observe: $\|\mathbf{w}_t\| \geq \mathbf{w}_t \cdot \mathbf{v}$ because $\|\mathbf{v}\| = 1$. (Cauchy-Schwarz)

We prove a lower bound on $\mathbf{w}_t \cdot \mathbf{v}$ using induction over t

- Claim: $\mathbf{w}_t \cdot \mathbf{v} \geq M_t\gamma$
- Base: $t = 1$, $\mathbf{w}_1 \cdot \mathbf{v} = 0$
- Induction step (assume for t and prove for $t + 1$):
  If mistake ($M_{t+1} = M_t + 1$) then

$$\begin{aligned}
\mathbf{w}_{t+1} \cdot \mathbf{v} &= (\mathbf{x}_t y_t) \cdot \mathbf{v} \\
&= \mathbf{w}_t \cdot \mathbf{v} + y_t \mathbf{x}_t \cdot \mathbf{v} \\
&\geq M_t\gamma + \gamma \\
&= (M_t + 1)\gamma
\end{aligned}$$

Combining the upper and lower bounds

Let $M := M_{m+1}$ denote the total number of updates ("mistakes") then

$$(M\gamma)^2 \leq \|\mathbf{w}_{m+1}\|^2 \leq MR^2$$

Thus simplifying we have the famous ...

Theorem (Perceptron Bound [Novikoff])

For all sequences of examples

$$S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \qquad \text{with} \qquad (\mathbf{x}, y) \in \mathbb{R}^n \times \{-1, 1\}$$

the mistakes of the Perceptron algorithm is bounded by

$$M \leq \left(\frac{R}{\gamma}\right)^2,$$

with $R := \max_t \|\mathbf{x}_t\|$ when there exists a vector $\mathbf{v}$ with $\|\mathbf{v}\| = 1$ and constant γ such that $(\mathbf{v} \cdot \mathbf{x}_t)y_t \geq \gamma$ for all t.

Comments

Comments

- It is often convenient to express the bound in the following form. Here define $\mathbf{u} := \frac{\mathbf{v}}{\gamma}$ then

$$M \leq R^2 \|\mathbf{u}\|^2 \quad (\forall \mathbf{u} : (\mathbf{u} \cdot \mathbf{x}_t) y_t \geq 1)$$

- Suppose we have *linearly separable* data set S. Questions:
  1. Observe that $\mathbf{w}_{m+1}$ does not necessarily linearly separate S. Why?
  2. How can we use the PERCEPTRON to find a vector $\mathbf{w}$ that separates S?
  3. How long will this computation take?
- There are variants on the PERCEPTRON that operate on a single example at a time that converge to the “SVM” max-margin linear separator.

Regret Bounds for Linear Separation

Going Deeper : Regret Bounds for Linear Separation

Recall the regularisation approach to supervised learning.

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{t=1}^{m} L(y_t, h(\mathbf{x}_t)) + \lambda \text{penalty}(h)$$

Example: Ridge Regression

$$\arg\min_{\mathbf{w} \in \mathbb{R}^n} \sum_{t=1}^{m} L(y_t, \mathbf{w} \cdot \mathbf{x}_t) + \lambda \|\mathbf{w}\|^2$$

Example: Soft Margin SVM

$$\arg\min_{\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}} \sum_{t=1}^{m} L_{\text{hi}}(y_t, \mathbf{w} \cdot \mathbf{x}_t + b) + \lambda \|\mathbf{w}\|^2$$

with $L_{\text{hi}}(y, \hat{y}) := \max(0, 1 - y\hat{y})$.

Online Approach

Recall the regularisation approach to “BATCH” supervised learning.

$$\arg\min_{h \in \mathcal{H}} \sum_{t=1}^{m} L(y_t, h(\mathbf{x}_t)) + \lambda \text{penalty}(h)$$

Question: how can we approach it **online?**

Online Approach

Recall the regularisation approach to "BATCH" supervised learning.

$$\underset{h\in\mathcal{H}}{\arg\min}\sum_{t=1}^{m}L(y_t,h(\mathbf{x}_t))+\lambda\text{penalty}(h)$$

Question: how can we approach it **online?**

A possible strategy is, every time we see a new sample (x_{t+1},y_{t+1}) to produce a new h_{t+1} such that

- It **fits** the new data point well
- It is not "too different" from the previous h_t

$$h_{t+1}=\underset{h\in\mathcal{H}}{\arg\min}\,L(y_t,h(\mathbf{x}_t))+\lambda\text{penalty}(h,h_t)$$

Online Gradient Descent with Hinge Loss and $\|\cdot\|_2^2$ penalty

Let's consider SVMs:

- Hinge loss: $L_{\mathrm{hi}}(y, \hat{y}) = \max(0, 1 - y\hat{y})$.
- Linear hypotheses: $h(\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$.

Then, the online update becomes

$$\mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathbb{R}^n}{\arg\min}\, L_{\mathrm{hi}}(y_t, \mathbf{w} \cdot \mathbf{x}_t) + \lambda \|\mathbf{w} - \mathbf{w}_t\|^2$$

Solving for the update (taking the "derivative" and set to zero) corresponds to choosing $\mathbf{w}_{t+1}$ as follows:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t & y_t(\mathbf{w} \cdot \mathbf{x}_t) > 1 \\ \mathbf{w}_t + \frac{y_t \mathbf{x}_t}{2\lambda} & y_t(\mathbf{w} \cdot \mathbf{x}_t) < 1 \end{cases}$$

Note : If $y_t(\mathbf{w} \cdot \mathbf{x}_t) = 1$ then we may choose either.

OGD with Hinge Loss and $\|\cdot\|_2^2$ penalty

OGD Algorithm

```
Initialise : $\mathbf{w}_1 := \mathbf{0}$, $L_{\mathrm{OGD}} := 0$
Select:   $\eta \in (0, \infty)$

For $t = 1$ To $m$ Do
    Receive instance  $\mathbf{x}_t \in \mathbb{R}^n$
    Predict           $\hat{y}_t := \mathbf{w}_t \cdot \mathbf{x}_t$
    Receive label     $y_t \in \{-1, 1\}$
    Incur loss        $L_{\mathrm{OGD}} := L_{\mathrm{OGD}} + L_{\mathsf{hi}}(y_t, \hat{y}_t)$
    Update weights $\mathbf{w}_{t+1} := \mathbf{w}_t + \mathbf{1}_{\{y_t \hat{y}_1 < 1\}} \eta y_t \mathbf{x}_t$.
```

How does the above differ from the perceptron?

Regret Bound for OGD

Theorem (Based on [G03])

Let $R = \max_t \|\mathbf{x}_t\|$ and $\eta := \frac{U}{R\sqrt{m}}$. Then, for any vector $\mathbf{u}$, such that $\|\mathbf{u}\| \leq U$, the sequence produced by OGD, satisfies

$$\sum_{t=1}^{m} L_{\mathrm{hi}}(y_t, \hat{y}_t) - L_{\mathrm{hi}}(y_t, \mathbf{u} \cdot \mathbf{x}_t) \leq \sqrt{U^2 R^2 m}\,, \tag{6}$$

Regret Bound for OGD – Proof (1)

Proof

Using the convexity of the hinge loss (wrt its 2nd argument), we have

$$L_{\mathsf{hi}}(y_t, \hat{y}_t) - L_{\mathsf{hi}}(y_t, \mathbf{u} \cdot \mathbf{x}_t) \leq (\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{z}_t \, , \tag{7}$$

where

$$\mathbf{z}_t := -y_t \mathbf{x}_t \mathbf{1}_{\{y_t(\mathbf{w}_t \cdot \mathbf{x}_t) < 1\}} \in \underbrace{\partial_{\mathbf{w}} L_{\mathsf{hi}}(y_t, \mathbf{w} \cdot \mathbf{x}_t)}_{\text{subdifferential!}} \, .$$

From the update we have,

$$\|\mathbf{w}_{t+1} - \mathbf{u}\|^2 = \|\mathbf{w}_t - \eta \mathbf{z}_t - \mathbf{u}\|^2 = \|\mathbf{w}_t - \mathbf{u}\|^2 - 2\eta(\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{z}_t\rangle + \eta^2 \|\mathbf{z}_t\|^2 \, .$$

Thus

$$(\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{z}_t = \frac{1}{2\eta} \left(\|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2 + \eta^2 \|\mathbf{z}_t\|^2 \right) \, . \tag{8}$$

Regret Bound for OGD – Proof (2)

Proof – Continued

From (8) we have

$$\begin{aligned}\sum_{t=1}^{m}(\mathbf{w}_t-\mathbf{u})\cdot\mathbf{z}_t &= \sum_{t=1}^{m}\frac{1}{2\eta}\left(\|\mathbf{w}_t-\mathbf{u}\|^2-\|\mathbf{w}_{t+1}-\mathbf{u}\|^2+\eta^2\|\mathbf{z}_t\|^2\right)\\ &\leq \frac{1}{2\eta}\left(\|\mathbf{u}\|^2+\eta^2\sum_{t=1}^{m}\|\mathbf{z}_t\|^2\right)\\ &= \frac{1}{2\eta}\|\mathbf{u}\|^2+\frac{\eta}{2}\sum_{t=1}^{m}\|\mathbf{x}_t\|^2\mathbf{1}_{\{y_t(\mathbf{w}_t\cdot\mathbf{x}_t)<1\}}\\ &\leq \frac{1}{2\eta}U^2+\frac{\eta}{2}mR^2\\ &= \sqrt{U^2R^2m} \quad (\text{recall } \eta := \frac{U}{R\sqrt{m}})\end{aligned}$$

Lower bounding the L.H.S. with (7) and we are done. ■

Compare to the proof of Hedge.

Deriving the perceptron algorithm/bound from OGD

The perceptron bound can be arrived by an analysis of OGD.

1. Observe that equation (6) implies,

$$\sum_{t=1}^{m}[y_t \neq \mathrm{sign}(\hat{y}_t)] - L_{\mathrm{hi}}(y_t, \mathbf{u}\cdot\mathbf{x}_t) \leq \sqrt{U^2R^2m}\,.$$

2. Now assume there exists a linear classifier $\mathbf{u}$ such that $y_t(\mathbf{u}\cdot\mathbf{x}_t) \geq 1$ for all $t = 1, \ldots, m$. Thus,

$$\sum_{t=1}^{m}[y_t \neq \mathrm{sign}(\hat{y}_t)] \leq \sqrt{U^2R^2m}\,.$$

3. Now make OGD conservative that is we only update when $y_t\hat{y}_t \leq 0$ versus $y_t\hat{y}_t \leq 1$ i.e., trials in which a mistake is made.
4. Thus with respect to the bound we can ignore the trials where a mistake is not made so that we can set $m = M := \sum_{t=1}^{m}[y_t \neq \mathrm{sign}(\hat{y}_t)]$ which implies

$$M \leq \sqrt{U^2R^2M} \longrightarrow M \leq U^2R^2$$

5. Finally note that the sign of the predictions is unchanged (why?) for any value of $\eta > 0$ thus we may set $\eta = 1$.

Wrapping Up

- We have considered a supervised learning setting where no randomness in the data is assumed (it could even be adversarial!)
- We have identified a different goal from the stochastic setting: having a cumulative error that is close to the one of the best model in the class.
- We first studied the case where we want to leverage the recommendations of experts.
- We then considered the case of "transforming" our previous stochastic approaches to supervised learning (e.g. Tikhonov regularization) to online settings.

Problems – 1

1. Suppose $\mathcal{X} = \{\textsc{True}, \textsc{False}\}^n$. Give a polynomial time algorithm $\mathcal{A}$ with a mistake bound $M(\mathcal{A}) \leq O(n^2)$ for any example sequence which is consistent with a k-literal conjunction. Your answer should contain an argument that $M(\mathcal{A}) \leq O(n^2)$.
2. State the perceptron convergence theorem [Novikoff] explaining the relation with the hard margin support vector machine solution.
3. **[Hard]:** Define the c-regret of learning algorithm as

   $$c\text{−regret}(m) = L_A(S) - c \min_{i \in [n]} L_i(S)$$

   thus the usual regret is the 1-regret.

   3.1 Argue for the weighted majority set-up argue that without randomised prediction it is impossible for all training sequences to obtain sublinear c-regret for $c < 2$.

   3.2 Show how to select β to obtain sublinear 2-regret.
4. Consider binary prediction with expert advice, with a perfect expert. Prove that any algorithm makes at least $\Omega(\min(m, \log_2 n))$ mistakes in the worst case.

Problems – 2

1. Recall that by tuning the weighted majority we achieved a bound

$$M \leq 2.63 \min_i M_i + 2.63 \ln n$$

Now by using randomisation in the prediction, design an algorithm with a bound that has the property

$$\frac{M}{m} \leq \min_{i \in [n]} \frac{M_i}{m} \text{ as } m \to \infty\,,$$

for the weighted majority setting (i.e., the mean prediction error of the algorithm is bounded by the mean prediction error of the "best" expert). Recalling that m is the number of examples (and the "tuning" of the algorithm may depend on m). For contrast compare this to problem 3.1 above.

Useful references

1. Nicolò Cesa-Bianchi and Gábor Lugosi, *Prediction, learning, and games.*, (2006), Note this is a book.
2. N. Littlestone, *Learning quickly when irrelevant attributes abound: a new linear threshold algorithm*, (1988).
3. N. Littlestone and M. K. Warmuth. *The weighted majority algorithm*, (1994)
4. V. Vovk, *Aggregating strategies*, (1990).
5. Y. Freund and R. Schapire, *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, (1997)
6. Haussler, D., Kivinen, J. and Warmuth, M.K. *Sequential Prediction of Individual Sequences Under General Loss Functions,* (1998)
7. M. Herbster and M. Warmuth, *Tracking the Best Expert*, (1998)
8. J. Kivinen and M. Warmuth, *Averaging Expert predictions*, (1999)
9. S. Shalev-Schwartz, *Online Learning and Online Convex Optimization*, (2011)