Online Algorithm in Machine Learning

Avrim Blum

Carnegie Mellon University

Presented by: Zhi Zhang, Nov 30, 2010
Motivation

- **Online Algorithm**: deals with inputs coming over time; no future information available.

- **Machine Learning**: evolves by learning from data observed so far.
Motivation

- **Online Algorithm**: deals with inputs coming over time; no future information available.

- **Machine Learning**: evolves by learning from data observed so far.

**Common Interests**: problems of making decisions about the present based only on knowledge of the past.

**Goal**: gives a sense of some of the interesting ideas and problems in *Machine Learning* area that have an “Online Algorithms” feel to them.
1 Introduction

2 Predicting from Expert Advice
   - A simple algorithm
   - A better algorithm (randomized)

3 Online Learning from Examples
   - A simple algorithm
   - The Winnow algorithm

4 Conclusions
Learning to predict:

1. study the data/information observed so far;
2. make a prediction based on some rules;
3. given the true value, adjust those rules.

Objective: makes as few mistakes as possible.
Learning to predict:

1. study the data/information observed so far;
2. make a prediction based on some rules;
3. given the true value, adjust those rules.

Objective: makes as few mistakes as possible.

How to analyze?
Can we bound the number of mistakes made by our algorithm against some well-performed algorithm with extra information? (similar to competitive analysis for Online algorithm)
Learning to predict:
1. study the data/information observed so far;
2. make a prediction based on some rules;
3. given the true value, adjust those rules.

Objective: makes as few mistakes as possible.

How to analyze?
Can we bound the number of mistakes made by our algorithm against some well-performed algorithm with extra information? (similar to competitive analysis for Online algorithm)

YES!
1 Introduction

2 Predicting from Expert Advice
   - A simple algorithm
   - A better algorithm (randomized)

3 Online Learning from Examples
   - A simple algorithm
   - The Winnow algorithm

4 Conclusions
An example

- A learning algorithm: predicts rain – Y/N
- A group of experts: give advices – Y N N Y ...

<table>
<thead>
<tr>
<th>time</th>
<th>exp₁</th>
<th>...</th>
<th>expₙ</th>
<th>prediction</th>
<th>reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>Y</td>
<td>...</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Day T</td>
<td>Y</td>
<td>...</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
A learning algorithm: predicts rain – Y/N
A group of experts: give advices – Y N N Y ...

<table>
<thead>
<tr>
<th>time</th>
<th>exp₁</th>
<th>...</th>
<th>expₙ</th>
<th>prediction</th>
<th>reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>Y</td>
<td>...</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Day T</td>
<td>Y</td>
<td>...</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Learning Steps (a trial):
1. receives the predictions of the experts;
2. makes its own prediction;
3. is told the correct answer.
An example

**Note:** No assumption about the quality or independence of the experts.

**Goal:** performs nearly as well as the best expert so far, i.e., being *competitive* with respect to the best single expert.
Weighted Majority Algorithm (simple version)

1. Initialize the weights $w_1, \ldots, w_n$ of all experts to 1.
2. Given a set of predictions $\{x_1, \ldots, x_n\}$ by the experts, output the prediction with the highest total weight. That is, output 1 if

$$\sum_{i: x_i = 1} w_i \geq \sum_{i: x_i = 0} w_i$$

and output 0 otherwise.

3. When the correct answer $l$ is received, penalize each mistaken expert by multiplying its weight by $1/2$. That is,

- if $x_i \neq l$, then $w_i \leftarrow w_i/2$;
- if $x_i = l$, then $w_i$ is not modified.

Goto 2.
Weighted Majority Algorithm (simple version)

Theorem

The number of mistakes $M$ made by the Weighted Majority algorithm is never more than $2.41(m \lg n)$, where $m$ is the number of mistakes made by the best expert so far.
Weighted Majority Algorithm (simple version)

Theorem

The number of mistakes $M$ made by the Weighted Majority algorithm is never more than $2.41(m \lg n)$, where $m$ is the number of mistakes made by the best expert so far.

Proof.

Let $W = \sum_i w_i$. Initially, $W = n$.

- If make a mistake, i.e., at least $W/2$ weight of experts predicted incorrectly. Then $W$ is reduced by at least a factor of $1/4$.
- If makes $M$ mistakes, we have:

$$W \leq n(3/4)^M. \quad (1)$$

- The best expert makes $m$ mistakes, then its weight is $1/2^m$.

Clearly,

$$W \geq 1/2^m. \quad (2)$$

Combining (1) and (2), we will get:

$$M \leq 2.41(m + \lg n).$$
Weighted Majority Algorithm (randomized version)

Randomized Weighted Majority Algorithm

1. Initialize the weights $w_1, \ldots, w_n$ of all experts to 1.
2. Given a set of predictions $\{x_1, \ldots, x_n\}$ by the experts, output $x_i$ with probability $w_i/W$, where $W = \sum_i w_i$.
3. When the correct answer $l$ is received, penalize each mistaken expert by multiplying its weight by $\beta$. Goto 2.
Weighted Majority Algorithm (randomized version)

Randomized Weighted Majority Algorithm

1. Initialize the weights $w_1, \ldots, w_n$ of all experts to 1.

2. Given a set of predictions $\{x_1, \ldots, x_n\}$ by the experts, output $x_i$ with probability $w_i/W$, where $W = \sum_i w_i$.

3. When the correct answer $l$ is received, penalize each mistaken expert by multiplying its weight by $\beta$. Goto 2.

Advantages:

- dilutes the worst case.
- applied when predictions are sorts of things that cannot easily be combined together.
Theorem

On any sequence of trials, the expected number of mistakes $M$ made by the Randomized Weighted Majority algorithm satisfies:

$$M \leq \frac{m \ln(1/\beta) + \ln n}{1 - \beta}$$

where $m$ is the number of mistakes made by the best expert so far.
Weighted Majority Algorithm (randomized version)

Theorem

On any sequence of trials, the expected number of mistakes $M$ made by the Randomized Weighted Majority algorithm satisfies:

$$M \leq \frac{m \ln(1/\beta) + \ln n}{1 - \beta}$$

where $m$ is the number of mistakes made by the best expert so far.

Examples:

- $\beta = 1/2$, $M \leq 1.39m + 2 \ln n$.
- $\beta = 3/4$, $M \leq 1.15m + 4 \ln n$.
- ....
Weighted Majority Algorithm (randomized version)

**Theorem**

On any sequence of trials, the expected number of mistakes $M$ made by the Randomized Weighted Majority algorithm satisfies:

$$M \leq \frac{m \ln(1/\beta) + \ln n}{1 - \beta}$$

where $m$ is the number of mistakes made by the best expert so far.

**Examples:**

- $\beta = 1/2$, $M \leq 1.39m + 2 \ln n$.
- $\beta = 3/4$, $M \leq 1.15m + 4 \ln n$.
- . . .

**Observation:** By adjusting $\beta$, we can make the “competitive ratio” as close to 1 as desired, plus an increase in the additive constant.
Weighted Majority Algorithm (randomized version)

Proof.

$F_i$: the fraction of the total weight on the *wrong* answers at the $i^{th}$ trial.

$M$: the expected number of mistakes so far. $m$: the number of mistakes of the best expert so far.

After seeing $t$ examples, $M = \sum_{i=1}^{t} F_i$.

On the $i^{th}$ example, the total weight changes according to:

$$W \leftarrow \beta F_i W + (1 - F_i) W = W(1 - (1 - \beta)F_i)$$

Hence, the final weight is:

$$W = n \prod_{i=1}^{t} (1 - (1 - \beta)F_i)$$

Using the fact that the total weight must be at least as large as the weight on the best expert, we have:

$$n \prod_{i=1}^{t} (1 - (1 - \beta)F_i) \geq \beta^m \quad (3)$$

Taking the natural log of both sides of (3), we get

$$M \leq \frac{m \ln(1/\beta) + \ln n}{1 - \beta}$$
1 Introduction

2 Predicting from Expert Advice
   - A simple algorithm
   - A better algorithm (randomized)

3 Online Learning from Examples
   - A simple algorithm
   - The Winnow algorithm

4 Conclusions
Mistake Bound Learning Model

Definitions:

- example space: $\mathcal{X} = \{0, 1\}^n$.
- example: $x \in \mathcal{X}$.
- concept class: a set of boolean functions $\mathcal{C}$ over the domain $\mathcal{X}$.
- concept: a boolean function $c \in \mathcal{C}$.
Mistake Bound Learning Model

Definitions:

- example space: \( \mathcal{X} = \{0, 1\}^n \).
- example: \( x \in \mathcal{X} \).
- concept class: a set of boolean functions \( C \) over the domain \( \mathcal{X} \).
- concept: a boolean function \( c \in C \).

Learning Steps (a trial):

1. an example is presented to the learning algorithm.
2. the algorithm predicts either 1 or 0.
3. the algorithm is told the true label \( l \in \{0, 1\} \).
4. the algorithm is penalized for each mistake made.

Goal: make as few mistakes as possible.
An example

Objective: learning monotone disjunctions with target function $x_{i1} \lor \ldots \lor x_{ir}$. 
Objective: learning monotone disjunctions with target function $x_{i1} \lor \ldots \lor x_{ir}$.

Algorithm:

1. Begin with a hypothesis $h = x_1 \lor x_2 \lor \ldots \lor x_n$.
2. Each time a mistake is made on a negative example $x$, remove from $h$ all the variables set to 1 by $x$. 
An example

Objective: learning monotone disjunctions with target function $x_{i1} \lor \ldots \lor x_{ir}$.

Algorithm:

1. Begin with a hypothesis $h = x_1 \lor x_2 \lor \ldots \lor x_n$.
2. Each time a mistake is made on a negative example $x$, remove from $h$ all the variables set to 1 by $x$.

Analysis:

1. We only remove variables that are guaranteed to not be in the target function, so we never make a mistake on a positive example.
2. Since each mistake removes at least one variable from $h$, the algorithm makes at most $n$ mistakes.
The Winnow Algorithm

**Objective:** learning **monotone** disjunctions with target function $x_{i1} \lor \ldots \lor x_{ir}$.

### The Winnow Algorithm

1. Initialize the weights $w_1, \ldots, w_n$ of the variables to 1.
2. Given an example $x = \{x_1, \ldots, x_n\}$, output 1 if
   \[
   w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \geq n
   \]
   and output 0 otherwise.
3. If the algorithm makes a mistake:
   1. If the algorithm predicts negative on a positive example, then for each $x_i$ equal to 1, double the value of $w_i$.
   2. If the algorithm predicts positive on a negative example, then for each $x_i$ equal to 1, cut the value of $w_i$ in half.
The Winnow algorithm

Theorem

The Winnow Algorithm learns the class of disjunctions in the Mistake Bound model, making at most $2 + 3r(1 + \lg n)$ mistakes when the target concept is a disjunction of $r$ variables.
The Winnow algorithm

Theorem

The Winnow Algorithm learns the class of disjunctions in the Mistake Bound model, making at most $2 + 3r(1 + \lg n)$ mistakes when the target concept is a disjunction of $r$ variables.

Property: The Winnow algorithm is designed for learning with especially few mistakes when the number of relevant variables $r$ is much less than the total number of variables $n$. 
The Winnow algorithm

Proof.

1. Bound the number of mistakes that will be made on positive examples.

   - Any mistake made on a positive example must double at least one of the weights in the target function.
   - Any mistake made on a negative example will not halve any of these weights.
   - Each of these weights can be doubled at most \(1 + \lg n\).

   Therefore, Winnow makes at most \(r(1 + \lg n)\) mistakes on positive examples.
Proof.

1. Bound the number of mistakes that will be made on positive examples.
   - Any mistake made on a positive example must double at least one of the weights in the target function.
   - Any mistake made on a negative example will not halve any of these weights.
   - Each of these weights can be doubled at most $1 + \lg n$.

   Therefore, Winnow makes at most $r(1 + \lg n)$ mistakes on positive examples.

2. Bound the number of mistakes made on negative examples.
   - Each mistakes made on a positive example increases the total weight by at most $n$.
   - Each mistakes made on a negative example decreases the total weight by at least $n/2$.
   - The total weight never drops below zero.

   Therefore, Winnow makes at most $2 + 2r(1 + \lg n)$ mistakes on positive examples.
The Winnow algorithm

Proof.

1. Bound the number of mistakes that will be made on positive examples.
   - Any mistake made on a positive example must double at least one of the weights in the target function.
   - Any mistake made on a negative example will not halve any of these weights.
   - Each of these weights can be doubled at most $1 + \lg n$.

   Therefore, Winnow makes at most $r(1 + \lg n)$ mistakes on positive examples.

2. Bound the number of mistakes made on negative examples.
   - Each mistakes made on a positive example increases the total weight by at most $n$.
   - Each mistakes made on a negative example decreases the total weight by at least $n/2$.
   - The total weight never drops below zero.

   Therefore, Winnow makes at most $2 + 2r(1 + \lg n)$ mistakes on positive examples.

The number of total mistakes is bounded by $2 + 3r(1 + \lg n)$. □
1. Introduction

2. Predicting from Expert Advice
   - A simple algorithm
   - A better algorithm (randomized)

3. Online Learning from Examples
   - A simple algorithm
   - The Winnow algorithm

4. Conclusions
There are a group of algorithms in *Computational Learning Theory* that look particularly interesting from the point of view of *Online Algorithms*. 
There are a group of algorithms in *Computational Learning Theory* that look particularly interesting from the point of view of *Online Algorithms*.

1. Algorithms for combining the advice of experts.
   - Weighted Majority Algorithm – $2.41(m + \lg n)$
   - Randomized Weighted Majority Algorithm ($\beta$) – $\frac{m \ln(1+\beta)+\ln n}{1-\beta}$
There are a group of algorithms in *Computational Learning Theory* that look particularly interesting from the point of view of *Online Algorithms*.

Algorithms for combining the advice of experts.

1. Weighted Majority Algorithm – $2.41(m + \lg n)$
2. Randomized Weighted Majority Algorithm ($\beta$) – $\frac{m \ln(1+\beta) + \ln n}{1-\beta}$

The model of online mistake bound learning.

1. The Winnow Algorithm – $2 + 3r(1 + \lg n)$
Questions?