Algoritmos e Incerteza **(PUC-Rio INF2979, 2018.2)**

Lecture 5: Applications to Abstract MWU

September 11, 2018

Lecturer: Marco Molinaro *Scribe: Luisa Rosa*

1 Prediction with Experts: Abstract game

As discussed in the previous lecture, we have the following setup for prediction with experts in the abstract game:

- There are m actions or objects (e.g. experts)
- At time t the algorithm chooses $p^t = (p_1^t, ..., p_m^t) \in \Delta^m$, where $p_i^t \in [0, 1]$ and $\sum_{i=1}^m p_i^t = 1$
- Next, the loss vector $l^t = (l_1^t, ..., l_m^t)$ is revealed, in which $l^t \in [0, 1]^m$
- Then, the algorithm pays the loss l^t associated with the chosen distribution p^t

$$\langle p^t, l^t \rangle = \sum_{i=1}^{m} p_i^t l_i^t$$

- **Goal:** Minimizing the expected total loss compared to the loss of the best action or object ($\min_i \sum_t l_i^t$). The algorithm $AbstractMWU$, already studied, satisfies:

$$\sum_t \langle p^t, l^t \rangle \leq \min_i \sum_t l_i^t + 2\sqrt{T \log m}$$

There is also the **maximization version** of the abstract game where instead of losses l^t we have rewards r^t for having chosen the action or object i. The expected reward is $\sum_{i=1}^m p_i^t r_i^t = \langle p^t, r^t \rangle$. The goal here is maximize the expected reward, and we can compare with the best action, that is, $\max_i \sum_t r_i^t$. The algorithm $AbstractMWU$-max satisfies:

$$\sum_t \langle p^t, r^t \rangle \geq \max_i \sum_t r_i^t - 2\sqrt{T \log m}$$

note that the $AbstractMWU$-max may be better as it plays a probability distribution while the best expert always plays the same information.

2 Applications

Now, we present two applications for $AbstractMWU$ in the maximizing version, boosting and linear classification. These applications are offline, i.e., we know the data before applying the algorithm.

2.1 Boosting

Idea: To combine simple classifiers to form a better classifier. For example, consider the following points in $\mathbb{R}^2$, with binary classes, " $+$ " or " $-$ ".

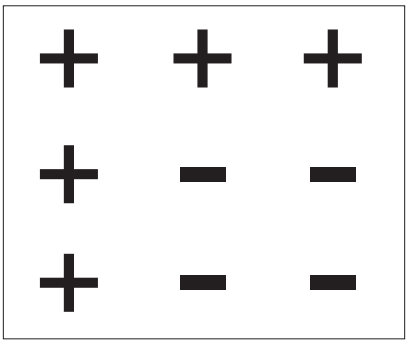

Figure 1: Set points in $\mathbb{R}^2$

Let's consider **decision stumps classifiers** that are, in for this data, classifiers given by a single vertical or horizontal separator. Such classifiers are not perfect because we do not have a single separator that gives us a perfect classification of these points in the plane. So, the idea of boosting is to use a combination of these classifiers.

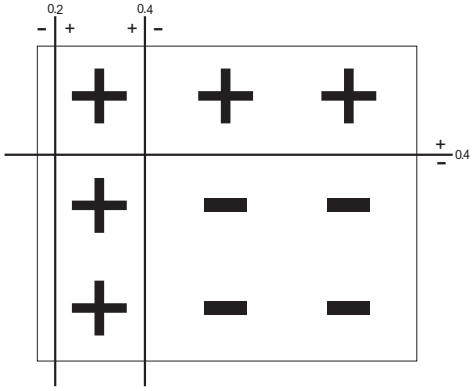

Figure 2: Combination of 3 decision stump classifiers with weights that form a compose perfect classifier

At any point in the plane in Figure 2, if we classify according to the labels "$+$" or "$-$" through the weighted majority of the 3 decision stump classifiers with weights, we always have the correct classification because the resulting classifier is perfect.

Q1: Given a set of base classifiers, how do we find the ideal weights to obtain a perfect classifier? To obtain the answer to this question consider the following boosting setup:

- We have a set of points $x_1, ..., x_n \in \mathbb{R}^d$,
- and the data points' corresponding labels $y_1, ..., y_n \in \{-1, +1\}$.
- We also have a family $\mathcal{C}$ of base classifiers, where a classifier is a function which maps the attributes to a class.

  $$\mathcal{C}\text{: family functions } c\text{: } \mathbb{R}^d \rightarrow \{-1, +1\}$$

  $c(x_i)$ is the label that the classifier gives to the point x.

In the previous example, we have $d = 2$ and $\mathcal{C}$: "All vertical or horizontal separators", more formally:

$$\mathcal{C} : \{c^{v,+,a} : a \in \mathbb{R}\} \cup \{c^{v,-,a} : a \in \mathbb{R}\} \cup \{c^{h,+,a} : a \in \mathbb{R}\} \cup \{c^{h,-,a} : a \in \mathbb{R}\}$$

where v and h represents respectively vertical and horizontal separators.

$$c^{v,+,a}(x_i) = \begin{cases} +1 & \text{if } x_i \geq a, \\ -1 & \text{if } x_i < a. \end{cases} \quad c^{v,-,a}(x_i) = \begin{cases} +1 & \text{if } x_i \leq a, \\ -1 & \text{if } x_i > a. \end{cases}$$

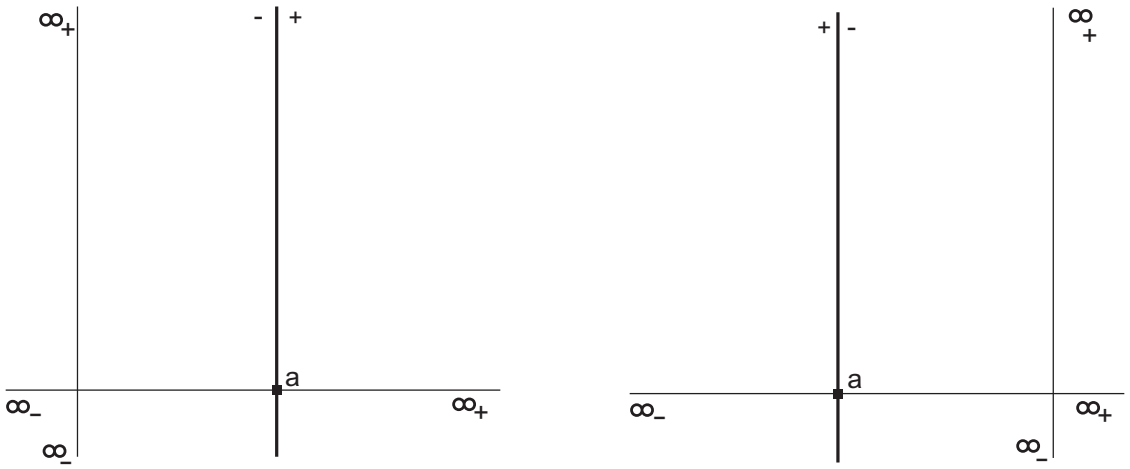

Figure 3: Representation of $c^{v,+,a}$ and $c^{v,-,a}$ in $\mathbb{R}^2$

we can define the horizontal classifiers is analogously.

For us, a composite classifier is one obtained in the following way:

- We take classifiers $c_1, ..., c_k \in \mathcal{C}$ with their associated weights $w_1, ..., w_k \in \mathbb{R}_+$
- The class of x is given by weighted majority, i.e.,the class of x is "+" if $\sum_{i \in c_i(x)=+} w_i > \sum_{i \in c_i(x)=-} w_i$ and the class of x is "-" otherwise.

- **Goal:** To attain the composition that makes as few mistakes as possible, because to find a perfect classifier depends on the quality of the classifiers in $\mathcal{C}$. In particular, a reasonable hypothesis to be considered is that there is at least one classifier better than the random guess, or rather, it correctly classifies 50% of the points "on average".

Definition 2.1. Given a distribution $p = (p_1, ..., p_n)$ over the data points, a classifier c is γ-*weak learning* with respect to p if the correctly classifies a p-fraction at least $\frac{1}{2} + \gamma$ of the data, i.e.,

$$\sum_{i:c(x_i)=y_i} p_i \geq \frac{1}{2} + \gamma.$$

Let us look at a simple example, for $n = 5$ and distribution $p = (0, 0.8, 0.1, 0.1, 0)$

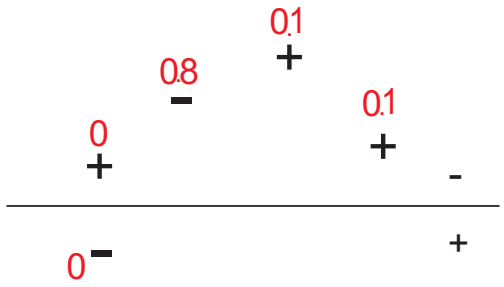

Figure 4: Example from p-fraction of correct class $x_i =$ " $-$ " equals to 0.8, so $\gamma = 0.3$

In Figure 4, we have an classifier that is γ-*weak learning* with $\gamma = 0.3$ to distribution p over data, i.e, the $\sum_{i:c(x_i)=-} p_i \geq \frac{1}{2} + 0.3$.

Now, see that, we have two weights, $w_1, ..., w_k$ assigned to the classifiers and $p_1, ..., p_n$ to the data. We associate the p distribution to the data so that we can focus on the most "important" data, that is, to focus on the most challenging examples and learn how to deal with them. For this we use, as already mentioned, the AbstractMWU-max, increasing the weights of the data with the highest number of mistakes.

Theorem 2.2. *If the family of classifiers $\mathcal{C}$ has a γ-weak learning for every distribution p over the data, then the the MWU Boosting algorithm returns a composite classifier $\bar{c}$ that is perfect, as long as $T > \frac{4 \log n}{\gamma^2}$.*

Proof. If we consider, one more time, the guarantee of the algorithm AbstractMWU-max, we have

$$\sum_t \langle p^t, r^t \rangle \geq \max_i \sum_t r_i^t - 2\sqrt{T \log m}.$$

And, for all data point i, the number mistakes of classifiers c^t's is given by $\sum_t r_i^t$, and from the above inequality, we can say that

Algorithm 1 MWU Boosting

Initialization: Set the initial distribution over the data to be $p_1 = (\frac{1}{n}, ..., \frac{1}{n})$

for t = 1 to T **do**
  Find classifier $c^t \in \mathcal{C}$ that is a γ-*weak learner* for distribution p^t
  Define reward r^t:
  if c^t wrong in data i **then**
    $r_i^t = 1$
  else
    $r_i^t = 0$
  end if
  Send the reward $r_t \in \mathbb{R}^n$ to MWU, and get new distribution $p_i^{t+1} \in \mathbb{R}^n$:
end for
return the composed classifier $\overline{c}$ that is the majority of $c_1, c_2, ..., c_T$

$$\sum_t r_i^t \leq \sum_t \langle p^t, r^t \rangle + 2\sqrt{T \log m}$$

Since c^t is a γ-*weak learner* with respect to p^t, we have that the p^t-fraction of data in which c^t is wrong is $\langle p^t, r^t \rangle = \sum_{i:c(x_i) \neq y_i} p_i \leq \frac{1}{2} - \gamma$. Adding over all t, we get

$$\sum_t \langle p^t, r^t \rangle \leq T(\frac{1}{2} - \gamma).$$

We can organize and obtain that the number of classifiers c^t's wrong at x_i is:

$$\sum_t r_i^t < T(\frac{1}{2} - \gamma) + 2\sqrt{T \log m}.$$

Then, when $T > \frac{4 \log n}{\gamma^2}$,we have

$$\forall i \sum_t r_i^t < \frac{T}{2}$$

Since the majority of the classifiers c_i's predicts correctly i, then, $\overline{c}$ is always correct.

□

Observation: There are better proceeding of boosting, a few including generalization, for example, the algorithm Ada Boost [2].

2.2 Linear Classification

Consider the following problem: Given vectors $a_1, ..., a_k \in \mathbb{R}^n$, find a solution for the system

$$\forall i \ \langle a_i, p \rangle > 0$$

where $p \in \Delta^n$.

To motivate this, consider the following classification setup in which,

- We have the set of vectors $x_1, ..., x_k \in \mathbb{R}^n$ and a set points $y_1, ..., y_k \in \{-1, +1\}$,
- where we want found a classifier given by a array $p \in \mathbb{R}^n$, such that, $\langle x_i, p\rangle > 0$ if $y_i = 1$ and $\langle x_i, p\rangle < 0$ if $y_i = 1$. So that, we can consider the classifier as hyperplane that passes through the origin correctly classifying.

We can assume that p belongs to a distribution Δ:

1. Make $p \geq 0$ introducing the "negation" of each coordinate, for example:

$$p_1 = (0.5, 0.2) \rightarrow (0.5, -0.5, 0.2, -0, 2)$$

2. Normalize so $\sum_j p_j = 1$.

Now that we have p as a distribution, can rearrange this setup to obtain the problem initially considered. Notice $y_i\langle x_i, p\rangle > 0$ iff data i is classified correctly and, using the properties of the inner product, we arrived at $\langle y_i x_i, p\rangle > 0$. Making $y_i x_i$ equal a_i, we get the initially problem, where $\langle a_i, p\rangle > 0$.

Theorem 2.3. *Suppose that there are $a_1, a_2, ..., a_k \in [-1, 1]^n$, $p* \in \Delta^n$, and , $\forall i \langle a_i, p*\rangle \geq \alpha$ (think of as a margin of classification, i.e, we can made linear classification with a margin with slack α). Then, there exists a algorithm, Winnow, that find finds a feasible p for the problem in $T = \frac{8 \log n}{\alpha^2}$ iterations.*

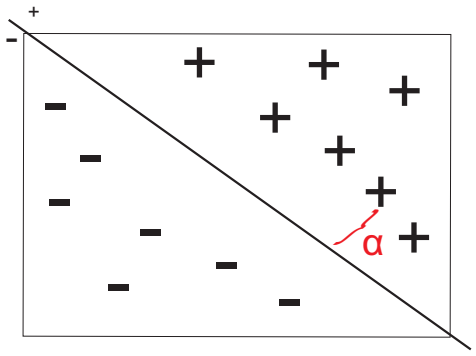

Figure 5: Example of a hyperplane with a margin of classification that classifies correctly

Note that the more difficult the instance I is, the higher the number of iterations we need, because if the margin is small, which makes α low, the larger is the size of T for the algorithm to

find $p*$ feasible. The algorithm Winnow uses a generalization of AbstractMWU-max, where we have $r_t \in [-1, 1]^n$. We still have the following guarantee [1]:

$$\sum_t \langle p^t, r^t \rangle \geq \max_i \sum_{=1}^{T} r_i^t - 2\sqrt{T \log n}$$

and we can rewrites the above equation how,

$$\sum_t \langle p^t, r^t \rangle = \max_{p \in \Delta^n} \langle p, \sum_t r^t \rangle - 2\sqrt{T \log n}$$

Since p on put all weight in the best coordinate. To connect the problem with AbstractMWU- max, we can set r^t as a_i, whereas in the algorithm we have the inner product of p with r^t.

Algorithm 2 Winnow

Initialization: Set the initial distribution $p_1 = (\frac{1}{n}, ..., \frac{1}{n})$

1: **for** `t = 1 to T` **do**
2: MWU returns p^t
3: **if** $\langle a_i, p^t \rangle < 0$ for any i **then**
4: $r^t = a_i$
5: **end if**
6: Send r^t to MWU
7: receive from MWU p^{t+1}
8: **end for**
9: **return** p^T

Proof. [of Theorem 2.3] If the algorithm ends before of iteration T, it returns a feasible solution. Let's prove by contradiction this always happens:

- Suppose that the algorithm does not end before T interactions,
- By construction of r^t, $\langle r^t, p^t \rangle < 0$. So,

$$\sum_{t=1}^{T} \langle r^t, p^t \rangle < 0$$

- From the guarantee of MWU,

$$\sum_{t=1}^{T} \langle r^t, p^t \rangle \geq \max_{p \in \Delta^n} \langle p, \sum_t r^t \rangle - 2\sqrt{T \log n}$$

Since $p*$ is a distribution $\in \Delta^m$, we get,

$$\begin{aligned}\sum_{=1}^{T}\langle r^t, p^t\rangle &\geq \langle p*, \sum_t r^t\rangle - 2\sqrt{T\log n} \\ &= \sum_t \langle p*, r^t\rangle - 2\sqrt{T\log n}\end{aligned}$$

Since by assumption $\langle p*, a_i\rangle \geq \alpha$ for all i (and the r^t's are such a_i's),

$$\sum_t \langle p*, r^t\rangle - \sqrt{T\log n} \geq T\alpha - 2\sqrt{T\log n}.$$

As T is large, or more precisely, $T = \frac{8\log n}{\alpha^2}$, the RHS is non-negative. Thus, $\sum_t \langle r^t, p^t\rangle \geq 0$. However we assumed $\sum_t \langle r^t, p^t\rangle < 0$ at the beginning of our proof, reaching the desired contradiction.

□

References

[1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8:121–164, 2012.

[2] R. E. Schapire and Y. Freund. *Boosting: Foundations and algorithms*. MIT press, 2012.