Advanced Topics in ML and AGT **Fall Semester, 2018/9**

Lecture 5: November 12

Lecturer: Yishay Mansour *Scribe: Yaniv Benny, Yuval Varkel, Or Perel* [1]

This lecture we see how to bound the regret of the Perceptron and Winnow algorithms using Online Gradient Descent (OGD) and Entropic Exponentiated Gradient respectively.

5.1 Perceptron Regret bound using OGD

5.1.1 Perceptron

We want to find a separating hyperplane which separates the inputs in $\mathbb{R}^d$ according to their labels: $Y = \{-1, 1\}$ (more convenient than $\{0, 1\}$).

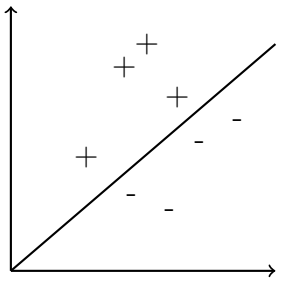

Figure 5.1: The separating hyperplane

Online Algorithm: the algorithm maintains a hyperplane w_t that enables it to classify the inputs.
On round t:

- The learner receives a vector $x_t \in \mathbb{R}^d$
- The learner maintains a weight vector $w_t \in \mathbb{R}^d$
- The learner predicts $p_t = sign(x_t^T w_t)$
- The learner receives $y_t \in Y$
- The learner suffers loss of 1 if $p_t \neq y_t$ and 0 otherwise (0-1 loss)
- The learner updates w_t to w_{t+1}

[1] First revision is due to Vered Zilberstein, Oria Ratzon, Lee Cohen – Fall Semester, 2017/8.

Our goal is to bound the number of errors that the algorithm makes. However, as we can see in *figure 5.2*, it's not possible to bound the number of mistakes even for one dimensional inputs - $x_t \in \mathbb{R}$ in the realizable case.

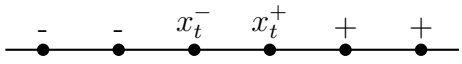

Figure 5.2: Let x_t^- and x_t^+ are the closest points in $\mathbb{R}$ with different labels seen so far. The point x_t^- is the highest point that is labeled - and x_t^+ is the lowest point labeled +. We can't be certain for all the points in-between, and thus we're probably wrong for some sub-interval between them. The area of uncertainty is $[x_t^-, x_t^+]$. Consider $x_{t+1} = \frac{x_t^- + x_t^+}{2}$, and let the learner predict p_{t+1}. The adversary can return the label $y_{t+1} = 1 - p_{t+1}$, forcing the learner to err. This can occur at any time-step.

To overcome this, we use hinge loss as a surrogate convex loss.
Recall we can define 0-1 loss by using an indicator function ($\mathbb{I}$):
$l(w, (x, y)) = \mathbb{I}\{y(w^T x) \leq 0\}$.
We define a set M, that contains all the times in which we make a mistake:
$M = \{t : p_t \neq y_t\}$. The surrogate loss function will be:

$$f_t(w) = [1 - y_t(w^T x_t)]_+$$

Where $[a]_+ = max\{a, 0\}$.
This loss function is the hinge loss. The loss increases as the distance from the separator increases. We modify the loss that $f_t(w) = 0$ if $t \notin M$.

Attributes:

1. f_t is convex, as it is a maximum of two linear functions.
2. If w^* is the optimal solution, it classifies all points with margin=1, $\forall t : (x_t^T w^*) y_t \geq 1$, then $\forall t : f_t(w^*) = l(w^*, (x_t, y_t)) = 0$.
3. $\forall w$: $f_t(w) \geq l(w, (x_t, y_t))$.
   By definition: $f_t(w) \geq 0$.
   If $\lambda = y_t(x_t^T w) \leq 0$ then $l(w, (x_t, y_t)) = 1 \leq f_t(w) = [1 - \lambda]_+ = 1 + |\lambda|$.
   If $\lambda = y_t(x_t^T w) > 0$ then $l(w, (x_t, y_t)) = 0 \leq f_t(w)$ by definition.
   For an arbitrary u we clearly have $\sum_{t=1}^{T} f_t(u) \geq \sum_{t \in M} f_t(u)$

We run Online Gradient Descent (OGD) on the series of losses f_t:

- Initialization: $w_1 = 0$
- Update: $w_{t+1} = w_t - \eta z_t$, where $z_t \in \partial f_t(w_t)$

Now,

- If $t \notin M$, i.e: $y_t(w_t^T x_t) \geq 0$, then $f_t(w_t) = 0$, set $z_t = 0$
- If $t \in M$, i.e: $y_t(w_t^T x_t) < 0$, then $f_t(w_t) = 1 - y_t(w_t^T x_t) > 0$, set $z_t = -y_t x_t$

Note we'll want to compare with a different w only for the points in time where the algorithm have made a mistake. That should definitely serve as a lower bound on the general error of w.

Overall, the update rule is: $w_{t+1} = \begin{cases} w_t & t \notin M \\ w_t + \eta y_t x_t & t \in M \end{cases}$

Therefore $w_{t+1} = \eta \sum_{i \in M, i \leq t} y_i x_i$,

$p_t = sign(w_t^T x_t) = sign(\eta \sum_{i \in M, i \leq t-1} y_i (x_i^T x_t))$.

Note that as long as $\eta > 0$, the parameter η doesn't affect the result of the $sign(\cdot)$ function. Since η doesn't matter, we might as well choose $\eta = 1$ and produce the Perceptron algorithm:

Algorithm 1 Perceptron

- Initialization: $w_1 = 0$
- For $t \in [1, T]$:
  - receive x_t
  - predict $p_t = sign(w_t^T x_t)$
  - if $y_t(w_t^T x_t) < 0$ then $w_{t+1} = w_t + y_t x_t$
  - else $w_{t+1} = w_t$

In order to analyze the Perceptron we use the regret bounds seen in the last two lectures.

5.1.2 Perceptron analysis using regret bounds

In our case, the sub-gradient $z_t \in \partial f_t(w_t)$ is:

$$z_t = \begin{cases} 0 & t \notin M \\ -y_t x_t & t \in M \end{cases}$$

with the updates being: $w_{t+1} = w_t - \eta z_t$.

OGD bound:

$$Regret_T(u) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u) \leq \frac{1}{2\eta}||u||_2^2 + \frac{\eta}{2}\sum_{t=1}^{T}||z_t||_2^2$$

Since f_t is a surrogate loss function for the 0-1 loss, the mistake bound is: $|M| \leq \sum_{t=1}^{T} f_t(w_t)$

Assume bounded inputs: $\forall t : ||x_t||_2 \leq R$, then:

$$||z_t||_2 = \begin{cases} 0 & t \notin M \\ ||x_t||_2 \leq R & t \in M \end{cases}$$

Using this in the $Regret_T(u)$ bound, we get:

$$|M| - \sum_{t=1}^{T} f_t(u) \leq \frac{1}{2\eta}||u||_2^2 + \frac{\eta}{2}|M|R^2$$

We can set $\eta = \frac{||u||}{R\sqrt{|M|}}$, which is fine because we can choose any $\eta > 0$ we like. This is true because η is not needed during the algorithm- we only use it for analysis, so there is no problem with it being dependent on $|M|$ and $||u||$.
After rearranging, we obtain:

$$|M| - \sum_{t=1}^{T} f_t(u) \leq R||u|\sqrt{|M|}$$

Notice that:

- If there exists u that classifies all the points correctly with margin=1 ($y_t(u^T x_t) \geq 1$) then $\forall t : f_t(u) = 0$. Hence $|M| \leq R||u|\sqrt{|M|}$ and therefore $|M| \leq R^2||u||_2^2$. Such a realizable case is called: *'separability with margin'*.
  We can choose $\gamma = \frac{1}{||u||_2}$ and get $|M| \leq R^2||u||_2^2 = \frac{R^2}{\gamma^2}$, which is exactly the known mistake bound for Perceptron, under the assumption that $y_t(\frac{u^T}{||u||}x_t) \geq \frac{1}{||u||} = \gamma$.

- For general case: $|M| - R||u||\sqrt{|M|} - \sum_{t=1}^{T} f_t(u) \leq 0$

  Solve by deriving the roots as $x^2 - bx - c \leq 0$, with $x = \sqrt{|M|}$, $b = R||u||$, $c = \sum_{t=1}^{T} f_t(u)$, we have: $0 \leq x \leq \frac{b+\sqrt{b^2+4c}}{2} \leq \frac{b+\sqrt{b^2}+\sqrt{4c}}{2} = b + \sqrt{c}$, and we get: $x^2 \leq b^2 + c + 2b\sqrt{c}$.
  Resulting in: $|M| \leq R^2||u||_2^2 + \sum_{t=1}^{T} f_t(u) + 2R||u||\sqrt{\sum_{t=1}^{T} f_t(u)}$

  This mistake bound is independent of realizability. Notice it depends on the hinge loss of the separator u.

5.2 Winnow Regret bound using Entropic Exponentiated Weight

5.2.1 Winnow

We want to find a k-monotone disjunctive Boolean function:
$X = \{0,1\}^d$ and the disjunction function is $x[i_1] \vee ... \vee x[i_k]$, for some $k \leq d$.

Define $w^* \in \{0,1\}^d$ as follows:
Let $w^*[i]$ to be 1 in the i_j coordinates and 0 otherwise.
$\exists j : x[i_j] = 1 \Rightarrow 1 \leq (w^*)^T x$
$\forall j : x[i_j] = 0 \Rightarrow 0 = (w^*)^T x$

In fact, every vector $w \in \{0,1\}^d$ has exactly k elements that are equal to 1 (and the rest are 0), then $sign(w^T x - \frac{1}{2})$ is a representation of a k-monotone disjunctive boolean function.
Our hypothesis class is: $H = \{x \to sign(w^T x - \frac{1}{2}), w \in \{0,1\}^d, ||w||_1 = k\}$

Notice that this is not a convex problem, since the set of w vectors are not a convex set and a $0-1$ *loss* function isn't convex.
In order to make this problem convex:

1. Change the w vectors: $w \in \mathbb{R}_+^d$ [instead of $w \in \{0,1\}^d$]

2. Define a surrogate loss function, f_t, as follows.
   Note that when we make an error $sign(w^T x - \frac{1}{2}) \neq y \Leftrightarrow y(w^T x - \frac{1}{2}) \leq 0$.
   The $0-1$ *loss* is $l(w,(x,y)) = \mathbb{I}\{y(w^T x - \frac{1}{2}) \leq 0\}$

We're interested in finding a surrogate function for l and use it to create the Winnow algorithm. For that we define a set- M, that contains all the times in which we make a mistake: $M = \{t : p_t \neq y_t\}$.

Then we choose the surrogate loss, f_t, to be:

$$f_t(w) = [1 - y_t(2w^T x_t - 1)]_+$$

We modify the loss that $f_t(w) = 0$ if $t \notin M$. Observe that:

1. $f_t(w)$ is convex since it is a maximum of linear functions.
2. $\forall w$: $f_t(w) \geq l(w, (x_t, y_t))$:
   By definition: $f_t(w) \geq 0$.
   If w makes an error, i.e., $y_t(2w^T x_t - 1) \leq 0$, then $f_t(w) \geq 1 = l(w, (x_t, y_t))$.
   If w doesn't make an error, i.e., $y_t(2w^T x_t - 1) > 0$, then $l(w, (x_t, y_t)) = 0 \leq f_t(w)$.
3. For the Realizable case, it holds that: $\exists u \in \{0, 1\}^d : ||u||_1 = k, \forall t : f_t(u) = l(u) = 0$

Now, we can almost run the Perceptron algorithm. The only thing left to do is to expand the input so we can implement the threshold.
$\phi(x) : x \to (2x, -1)$
$\phi(w) : w \to (w, 1)$
Remember that the mistake bound of the Perceptron is: $|M| \leq R^2 ||u||_2^2$.
After the transformation:
$||\phi(u)||_2^2 = k + 1$
$||\phi(x)||_2^2 = 4d + 1$
If we choose $R^2 = 4d + 1$, we'll get:
$|M| \leq R^2 ||u||_2^2 \leq (k + 1)(4d + 1) = O(kd)$

We would like to reduce the linear dependency on d so it would be a logarithmic one (i.e., $|M| = O(klogd)$). For that we can use unnormalized exponentiated gradient.

5.2.2 Unnormalized Exponentiated Gradient

Parameter: $\eta > 0$
Initialization: $w_1 = \overrightarrow{(1/d)}$
Update: $w_{t+1}[i] = w_t[i] e^{-\eta z_t[i]}$
The gradient: $z_t = \begin{cases} z_t = 0 & t \notin M \\ z_t = 2y_t x_t & t \in M, y_t(2w_t^T x_t - 1) < 0 \end{cases}$

Combing the last two we get the update rule:
$w_{t+1}[i] = \begin{cases} w_t[i] & t \notin M \\ w_t[i]e^{-2\eta y_t x_t t[i]} & t \in M \end{cases}$

Notice that unlike the Normalized Exponentiated Gradient algorithm we've seen in Lecture 4, the w_t vectors here aren't normalized, since probability vectors aren't necessary for the Winnow algorithm.

$$Regret \leq \sum_{t=1}^{T}(w_t - u)^T z_t \leq \frac{1 + (\log(d) - 1)||u||_1 + \sum u_i \log u_i}{\eta} + \eta \sum_{t=1}^{T}\sum_{i=1}^{d} w_t[i]{z_t}^2[i]$$

The second inequality is due to *theorem 4.9* from lecture 4 (actually, here we have unnormalized weights, but the proof is similar).

Since $||u||_1 \leq k+1$, we can bound: $\frac{1+(\log(d)-1)||u||_1}{\eta} \leq \frac{1+(\log d-1)(k+1)}{\eta} \leq \frac{(\log d)(k+1)}{\eta}$.

In addition, $\sum u_i \log u_i \leq 0$, and we will later see in *Lemma 5.1* that: $\sum_{i=1}^{d} w_t[i]{z_t}^2[i] \leq 2f_t(w_t)$

Overall we get:

$$Regret = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(u) \leq \frac{(k+1)\log d}{\eta} + 2\eta \sum_{t=1}^{T} f_t(w_t)$$

$$|M| \leq \sum_{t=1}^{T} f_t(w_t) \leq \frac{1}{1-2\eta}[\frac{(k+1)\log d}{\eta} + \sum_{t=1}^{T} f_t(u)]$$

In the Realizable case $\sum_{t=1}^{T} f_t(u) = 0$, we can choose $\eta = \frac{1}{4}$, to get the wanted bound: $|M| \leq 2[4(k+1)\log d] = 2[4(k+1)\log d] = O(k \log d)$.

Lemma 5.1 $\forall t : \sum_{i=1}^{d} w_t[i]{z_t}^2[i] \leq 2f_t(w_t)$

Proof:

- For $t \notin M : f_t(w_t) = 0$, therefore $z_t = 0$.
- For $t \in M, y_t = 1 : f_t(w_t) = 2(1 - w_t^T x_t)_+$ since $x_t \in \{0,1\}^d : {x_t}^2[i] = x_t[i]$,
  so $\sum_i w_t[i](2x_t[i])^2 = 4w_t^T x_t \leq 2 \leq 2f_t(w_t)$. The last inequality is due to the definition of $f_t(w_t)$ - we know that if $f_t(w_t) \geq 1$ and $y_t = 1$, then $2w_t^T x \leq 1$.
- For $t \in M, y_t = -1 : f_t(w_t) = 2w_t^T x_t$ and again using $f_t(w_t)$ definition $2w_t^T x_t \geq 1$, therefore $\sum_i w_t[i]z_t^2[i] = 4w_t^T x_t \leq 2f_t(w_t)$.

□

Remark - Let us now compare the bounds of Perceptron and Winnow algorithms:
The Perceptron's mistake bound is $\frac{||u||_2^2||x||_2^2}{\gamma^2}$, a function of $||u||_2^2$ and $||x||_2^2$
The Winnow's mistake bound is $\frac{||u||_1^2||x||_\infty^2}{\gamma^2}$, a function of $||u||_1^2$ and $||x||_\infty^2$
Since $||u||_1^2 \geq |u||_2^2$ and $||x||_\infty^2 \leq ||x||_2^2$, the bounds are not comparable in general.

5.3 Online Algorithm to Batch Conversions

We would like to use online algorithm as a black box when solving a "normal" learning problem.

Reminder - *General Learning Algorithm*:

- Input: Sample: S=$\{(x_i,y_i)|1 \leq i \leq m\}$
- Cost function: $c(w,(x,y))$, where $w \in H$
- Output: $w^* = argmin_{w \in H}(\frac{1}{m}\sum_{i=1}^{m} c(w,(x_i,y_i))$

The output is the result of an Empirical Risk Minimization (ERM) algorithm, for which we know that in order to get a good generalization error,
Let $e_P(w) = \mathbb{E}_{(x,y)\sim P}[c(w,(x,y))]$ and $e_S(w) = \frac{1}{|S|}\sum_{(x_t,y_t)\sim S} c(w,(x_t,y_t))$. Then for any distribution P, and any $w \in H$ we have $P[|e_p(w) - e_S(w)| \geq \epsilon] \leq \delta$, when the sample size is $|S| \geq O(\frac{VCdim(H)}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}))$.

We shall now write a general ERM algorithm that uses an existing online algorithm:

Algorithm 2 General ERM Algorithm with an online algorithm black box

For every $(x_i, y_i) \in S$:

- Feed the algorithm with x_i
- Receive w_i
- Feed the algorithm with y_i

Now all we have to do is to choose w^* out of all the $\{w_1, ...w_m\}$ we received.
Since we do not have a guarantee that the regret is monotonically decreasing, choosing w_m

isn't necessary a good idea.
Instead, we suggest two options for w^* that are based on $\{w_1, ...w_T\}$.
Define: $C(\bar{w}) = \mathbb{E}_{(x,y)\sim D}[c(\bar{w}, (x, y))]$, the expected loss.

Option 1: Averaging the $w's$: $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$:

$$\mathbb{E}[C(\bar{w})] = \mathbb{E}[C(\tfrac{1}{T}\sum_{t=1}^{T} w_t)] \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[C(w_t)] \text{ [assuming } C \text{ is convex]}$$

$$\forall u : \mathbb{E}[C(\bar{w})] - C(u) \leq \tfrac{1}{T}\sum_{t=1}^{T}\mathbb{E}[C(w_t)] - C(u) \leq E[Regret(u)]$$

Option 2: randomizing the $w's$: choose I at random from $[1, T]$, i.e. $P[I = t] = \frac{1}{T}$ for $t \in [1, T]$. Let $\bar{w} = w_I$:

$$\mathbb{E}_{S,I}[C(\bar{w})] = \tfrac{1}{T}\sum_{t=1}^{T}\mathbb{E}[C(w_t)] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[f_t(w_t)]$$

$$\forall u : \mathbb{E}[C(\bar{w})] - C(u) = \mathbb{E}_S[\tfrac{1}{T}\sum_{t=1}^{T} f_t(w_t) - f_t(u)] \leq E[Regret(u)]$$

So the expected value of the regret for the w we've picked is similar to the online algorithm's cumulative Regret.