Week 8. PAC learning, VC dimension

Lecturer: Prof. Santosh Vempala Scribe: Ezgi Karabulut, Kyle Xu

1 Mistake-bound Learning

Learning can be described as the following process: Assume that we are given the points x in some space (i.e. $\mathbb{R}^n, \{0,1\}^n$), and a label for each point $l(x) \in \{1, -1\}$ the task is to find a function $\tilde{l}(\cdot)$ that best mimics $l(\cdot)$.

Batch learning (supervised learning), given a series of training examples $(x^1, l(x^1)), \ldots, (x^m, l(x^m))$, and a new point x, predicts $l(x)$. The question in batch learning is the number of training example necessary to learn the function.

Another variant is called online learning. In online learning, the example points are shown one by one. Given x, you predict $\tilde{l}(x)$. If the prediction is wrong, the true label $l(x)$ is revealed. The goal in mistake-bound learning is to make only a finite number of mistakes before learning the function.

Generally in learning you assume that the labelling function has a certain structure, and you try to find the function that is the best fit for your data and has the desired structure. The class of functions with the desired structure that you are testing for are called a hypothesis class.

Theorem 1.1. *Knowing a hypothesis class $\mathcal{H}$, there exists an algorithm that learns l by making $\leq \log_2 |\mathcal{H}|$ mistakes.*

Proof. The following algorithm satisfies that bound: Given x, compute $h(x)$ for all $h \in \mathcal{H}$, and predict $l(x)$ to be the majority of the $h(x)$ values, i.e. the majority of the predictions in $\mathcal{H}$. Whenever the prediction is wrong, the majority of the functions are eliminated, hence the algorithm makes $\leq \log_2 |\mathcal{H}|$ mistakes. □

2 Learning Conjunctions

The class of conjunctions is a concept class where the label of x is determined by an AND function of a subset of its literals.

To learn conjunctions, you start with an initial conjunction that consists of all literals and their negations. Whenever the conjunction predicts incorrectly on an example x with $l(x) = 1$, the literals of x that have value 0 are removed from the conjunction. See Table 1 for an illustration of the algorithm.

The algorithm starts with $2n$ literals, eliminates n literals in the first mistake, and at least 1 literal in the remaining mistakes; and using this argument, we can conclude that the number of mistakes is at most n.

This result is for mistake bound model. Now consider the following model: Assume that the data x is drawn independently from an unknown distribution $\mathcal{D}$. We are given a sample

TABLE 1: Illustration of the mistake-bound learning algorithm of conjunctions

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$l(x)$	Conjunction
x^1:	1	0	1	0	1	1	1	1	$x_1 \wedge x_3 \wedge x_5 \wedge x_6 \wedge x_7 \wedge \bar{x}_2 \wedge \bar{x}_4$
x^2:	0	1	1	0	1	0	1	0	$x_1 \wedge x_3 \wedge x_5 \wedge x_6 \wedge x_7 \wedge \bar{x}_2 \wedge \bar{x}_4$
x^3:	1	0	0	1	1	1	0	1	$x_1 \wedge x_5 \wedge x_6 \wedge \bar{x}_2$
x^4:	0	0	1	1	1	1	1	0	$x_1 \wedge x_5 \wedge x_6 \wedge \bar{x}_2$
x^5:	1	1	1	1	0	0	0	0	$x_1 \wedge x_5 \wedge x_6 \wedge \bar{x}_2$
x^6:	1	0	0	0	1	1	1	1	$x_1 \wedge x_5 \wedge x_6 \wedge \bar{x}_2$
				$\vdots$					

S that consists of $(x, l(x))$ pairs, and $|S| = m$. The question is how many examples from $\mathcal{D}$ are needed to ensure $\mathsf{P}_{x\in\mathcal{D}}(\tilde{l}(x) \neq l(x)) \leq \epsilon$.

Let $\tilde{l}$ be such that $\mathsf{P}_{\mathcal{D}}(\tilde{l}(x) \neq l(x)) > \epsilon$, and it is consistent with our sample of size m, i.e. $\forall x \in S\ \tilde{l}(x) = l(x)$.

$\mathsf{P}(\tilde{l}$ is consistent with m random examples from $\mathcal{D}) \leq (1-\epsilon)^m$

Furthermore, the number of possible $\tilde{l}$'s in the concept class of conjunctions is at most 3^n (each one of the n variables can either appear as itself, not appear at all, or their negation may appear).

Therefore the probability of the existence of an $\tilde{l}$ that is consistent with the m random examples, yet has $err_{\mathcal{D}}(\tilde{l}) > \epsilon$ is bounded from above by $(1-\epsilon)^m 3^n$. We want to bound this probability by δ. $(1-\epsilon)^m 3^n < \delta$ yields $m = O\left(\frac{1}{\epsilon}(n + \log\frac{1}{\delta})\right)$.

3 PAC-Learning

A concept class $\mathcal{H}$ is (ϵ, δ)-PAC *(probably(δ) approximately(ϵ) correct)* learnable if there exists an algorithm that with probability $\geq 1-\delta$ given labelled examples drawn from an unknown distribution $\mathcal{D}$ and hypothesis $h \in \mathcal{H}$, finds a hypothesis $\tilde{h}$ such that $\mathsf{P}_{\mathcal{D}}(\tilde{h}(x) \neq h(x)) \leq \epsilon$ using time and sample complexity bounded by $\text{poly}(\frac{1}{\epsilon}, \log\frac{1}{\delta}, n, < h >)$.

Theorem 3.1. *If you can learn a mistake bound model using $\leq M$ mistakes, then the number of examples to PAC learn is $\leq \frac{M}{\epsilon}\log\frac{M}{\delta}$.*

4 PAC-Learning Halfspaces

Given $x \in \mathbb{R}^n$, the label function in the concept class of halfspaces (or equivalently linear threshold functions) is of the form $l(x) = sign(w^T x - t)$, i.e. $l(x) = 1 \Leftrightarrow w^T x \geq t$ and $l(x) = -1 \Leftrightarrow w^T x < t$.

There might be direct correspondences between concept classes. For instance, conjunctions can be formulated as halfspaces with the following example transformation: $x_1 \wedge \bar{x_2} \wedge x_5 \Rightarrow x_1 + (1 - x_2) + x_5 \geq 3$. Similarly, polynomials can be formulated as halfspaces. $\sum a_{ij} x_i x_j \geq t$, with the definition of a new variable $y_{ij} = x_i x_j$ can be expressed as $\sum a_{ij} y_{ij} \geq t$.

In order to come up with a bound on the sample size for generic PAC-learning to find a

hypothesis consistent with data, we make use of the following inequality.

$$(1-\epsilon)^m|\mathcal{H}| < \delta \text{ implies } m = O(\tfrac{1}{\epsilon}\log|\mathcal{H}| + \log\tfrac{1}{\delta}) \tag{1}$$

The problem in the class of hyperplanes is that $|\mathcal{H}|$ is infinite in halfspaces. However, as long as two hyperplanes provide the same signs for the data, i.e. the same partition, they are equivalent. Therefore, we define $\mathcal{H}(m)$: maximum number of ways to partition m points using hypotheses from $\mathcal{H}$.

Claim 4.1. *For hyperplanes in $\mathbb{R}^n$, $\mathcal{H}(m) = O(m^n)$.*

Proof. In the case of hyperplanes in $\mathbb{R}^2$, a line is defined by 2 points; therefore there are $\binom{m}{2}$ possible choices of lines.

Once the line is determined, it might be slightly perturbed (only changing the sign of the points that are used to define the line) and there are 4 possibilities for which side those defining points will be on.

For hyperplanes in $\mathbb{R}^2$, $\mathcal{H}(m) = 4\binom{m}{2} = O(m^2)$.

When generalized to hyperplanes in $\mathbb{R}^n$, $\mathcal{H}(m) = 2^n\binom{m}{n} = O(m^n)$. □

Theorem 4.2. *$\forall\mathcal{H}$. $|S| \geq \frac{2}{\epsilon}(\log\mathcal{H}(2m) + \log\frac{1}{\delta})$ examples suffice to PAC-learn $\mathcal{H}$.*

Theorem 4.3. *$\forall\mathcal{H}$. $|S| \geq \frac{8}{\epsilon^2}(\log\mathcal{H}(2m) + \log\frac{1}{\delta})$ examples suffice to ensure that $\forall h \in \mathcal{H}$, $|err_{\mathcal{D}}(h) - err_S(h)| \leq \epsilon$.*

Corollary 4.4 (Theorem 4.2). *For halfspaces, substituting $O(m^n)$ for $\mathcal{H}(m)$ gives*
$m \geq \frac{2}{\epsilon}(\log(2m)^n + \log\frac{1}{\delta}) \Rightarrow m \geq \frac{c}{\epsilon}(n\log\frac{1}{\epsilon} + \log\frac{1}{\delta})$

Proof of [. Theorem 4.2] Let S be a sample of size m. We define event A: $\exists h \in \mathcal{H}$ s.t. $err_S(h) = 0$ and $err_{\mathcal{D}}(h) > \epsilon$. We want to bound the probability of event A, $\mathsf{P}(A)$. We choose another set of size m, S'. Similarly we define event B: $\exists h \in \mathcal{H}$ s.t. $err_S(h) = 0$ and $err_{S'}(h) > \epsilon/2$. We know that

$$\mathsf{P}(B) \geq \mathsf{P}(A \cap B) = \mathsf{P}(A)\mathsf{P}(B|A) \tag{2}$$

holds. We will use (2) to bound $Pr(A)$. Assuming A, since $err_{\mathcal{D}}(h) > \epsilon$, $\mathsf{E}[err_{S'}(h)|A] > \epsilon$ holds as well.

$$\begin{aligned}
\mathsf{P}(\bar{B}|A) &= \mathsf{P}(err_{S'}(h) > \epsilon/2|A) \\
&\leq \mathsf{P}(err_{S'}(h) > \frac{1}{2}\mathsf{E}[err_{S'}(h)|A]|A) \\
&\leq e^{-\left(\frac{1}{2}\right)^2\frac{\epsilon m}{2}} \qquad (3) \\
&\leq \frac{1}{e} \qquad (4)
\end{aligned}$$

Notice that (3) is the result of Chernoff bound, and plugging $m \geq 8/\epsilon$ gives (4). (4) implies $\mathsf{P}(B|A) \geq 1/2$. Plugging this in to (2) gives $\mathsf{P}(A) \leq 2\mathsf{P}(B)$.

The next step is to show that $\mathsf{P}(B)$ is small. Let S'' be a sample of size $2m$. We partition the elements of S'' into S and S' and ask the same question: what is the probability that $err_S(h) = 0$ and $err_{S'}(h) > \epsilon/2$. We do this by first partitioning S'' into pairs $(a_1, b_1), \ldots (a_m, b_m)$ randomly, and then determining by a coin-toss which element in each pair goes to S, and which to S'. Since $err_{S'}(h) > \epsilon/2$ by our assumption, there must be $r \geq \epsilon m/2$ mistakes in S''. The probability of all of those r mistakes to go to S' and not to S in the random partition is $\leq \left(\frac{1}{2}\right)^{\epsilon m/2}$

$$\mathsf{P}(B) \leq \mathcal{H}(2m)\left(\frac{1}{2}\right)^{\epsilon m/2} < \frac{\delta}{2} \implies m > \frac{2}{\epsilon}(\log \mathcal{H}(2m) + \log \frac{1}{\delta}) \tag{5}$$

suffices to ensure $\mathsf{P}(A) < \delta$. □

5 VC Dimension

Definition: The Vapnik-Chervonenkis dimension of a concept class $\mathcal{H}$ is the maximum size of a set whose all m examples can be labeled in all 2^m possible ways using concepts $h \in \mathcal{H}$.

Theorem 5.1. *The VC dimension of general linear classifiers in $\mathbb{R}^n$ is $n + 1$.*

Proof. First, we show that there exists a set of $n + 1$ points in $\mathbb{R}^n$ whose examples can be labeled in all 2^n possible ways using linear classifiers: $y = sign(w^T x + b)$. Consider the set X below with with labeling $y_i = \{1, -1\}$,

$$X = \begin{bmatrix} -x_1^T- \\ -x_2^T- \\ -x_3^T- \\ \vdots \\ -x_{n+1}^T- \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{6}$$

then the linear classifier can be written as $sign(w^T x + b) = sign(A\theta)$, where:

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 1 \\ & & & \ddots & & \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}, \theta = \begin{bmatrix} w \\ b \end{bmatrix} \tag{7}$$

Apparently A is invertible and therefore there exists $\theta = A^{-1}y$ for any 2^n possible labelings.

Next, we show that for any set containing $d + 2$ points, there exist one labeling that can not be achieved by linear classifier. In this case, there are more rows than columns in A, and there must be some row a_j that can be written as a linear combination of other rows: $a_j = \sum_{i \neq j} k_i a_i$, where not all $k_i = 0$. We now assign $y_i = sign(k_i)$ for all $k_i \neq 0$ and find a θ correctly predicts these labels. Then $\langle a_j, \theta \rangle = \sum k_i sign(k_i) |\langle a_i, \theta \rangle| > 0$, and none of these θs can label $y_j = -1$.

Combining the two steps above, we have proved theorem 5.1. □

Lemma 5.2. *Let the VC-dimension of $\mathcal{H}$ be d, then for any set containing m examples, we have $\mathcal{H}(m) \leq \sum_{i=0}^{d} \binom{m}{i} \leq m^d$*

Proof. First, we prove $\mathcal{H}(m) \leq \sum_{i=0}^{d} \binom{m}{i}$ by induction.

For $d = 0, m \geq 1$, the inequality holds as both sides equal to 1. For $m = 1, d \geq 1$, the inequality holds as both sides equal to 2. Now assume that it holds for $(m-1, d-1)$ and for $(m-1, d)$, we will prove that it also holds for (m, d).

For any set containing m examples, consider all it's possible labelings

$$\mathcal{H}_{x^m} = \{h(x1), \ldots, h(x_m) | h \in \mathcal{H}\} \tag{8}$$

and all possible labelings of its subsets containing $m-1$ examples

$$\mathcal{H}_{x^{m-1}} = \{h(x1), \ldots, h(x_{m-1}) | h \in \mathcal{H}\}. \tag{9}$$

All sequences in $\mathcal{H}_{x^{m-1}}$ must appear either once or twice in $\mathcal{H}_{x^m}$, and the ones that appear twice must be differentiated by $h(x_m)$ in $\mathcal{H}_{x^m}$. Let $\mathcal{H}_2$ be the set of all sequences in $\mathcal{H}_{x^{m-1}}$ that appear twice in $\mathcal{H}_{x^m}$:

$$\begin{aligned} \mathcal{H}_2 = \{(l_1, \ldots, l_{m-1}) \in \mathcal{H}_{x^{m-1}} | \; \exists h, h' \in \mathcal{H} \; s.t. \; & h(x_i) = h'(x_i) \forall i = 1, \ldots, m-1 \\ & and \; h(x_m) \neq h'(x_m)\}. \end{aligned} \tag{10}$$

Then it is clear that

$$|\mathcal{H}_{x^m}| = |\mathcal{H}_{x^{m-1}}| + |\mathcal{H}_2|; \tag{11}$$

Now $\mathcal{H}_{x^{m1}}$ is a restriction to $m1$ points labeled by the concept class $\mathcal{H}$ of VC-dimension d. Moreover, $\mathcal{H}_2$ can be viewed as a restriction to $m1$ points labeled by the concept class of VC-dimension at most $d1$. With our induction assumption, we have $|\mathcal{H}_{x^{m-1}}| \leq \sum_{i=0}^{d} \binom{m-1}{i}$ and $|\mathcal{H}_2| \leq \sum_{i=0}^{d-1} \binom{m-1}{i}$. Therefore:

$$\begin{aligned} \mathcal{H}(m) &= |\mathcal{H}_{x^m}| = |\mathcal{H}_{x^{m-1}}| + |\mathcal{H}_2| \\ &\leq \sum_{i=0}^{d} \binom{m-1}{i} + \sum_{i=0}^{d-1} \binom{m-1}{i} \\ &= 1 + \sum_{i=1}^{d} \binom{m-1}{i} + \sum_{i=1}^{d} \binom{m-1}{i-1} \\ &= 1 + \sum_{i=1}^{d} \binom{m}{i} \\ &= \sum_{i=0}^{d} \binom{m}{i} \end{aligned} \tag{12}$$

Next, we prove $\sum_{i=0}^{d} \binom{m}{i} \leq m^d$ by induction on d.

The inequality holds for $(d = 0, m \geq 0)$ as both side equal to 1. We assume that it holds for $d = k - 1$ and will prove that it also holds for $d = k$:

$$\begin{aligned}\sum_{i=0}^{k} \binom{m}{i} &= \sum_{i=0}^{k-1} \binom{m}{i} + \binom{m}{k} \\ &\leq m^{k-1} + \frac{m(m-1)\cdots(m-k+1)}{k!} \\ &\leq m^{k-1} + (m-1)\frac{m^{k-1}}{k!} \\ &\leq m^{k-1} + (m-1)m^{k-1} \\ &= m^k\end{aligned} \tag{13}$$

□

Recall that from **Theorem 4.2**, $|S| \geq \frac{2}{\epsilon}(\log \mathcal{H}(2m) + \log \frac{1}{\delta})$ examples suffice to PAC-learn $\mathcal{H}$. Use the m^d as the upper bond for $\mathcal{H}(m)$, we have $|S| \geq \frac{4}{\epsilon}(d \log m + \log \frac{1}{\delta})$ examples suffice to PAC-learn $\mathcal{H}$.

6 Two Learning Algorithms

Theorem 6.1. *The Perceptron algorithm starting with $w^0 = 0$ and using the update $w^j \leftarrow w^{j-1} + l(x_i)x_i$ on a wrongly labeled example x_i, terminates within*

$$\frac{||w^*||_2^2 \ \max ||x_i||_2^2}{\gamma^2} \quad \left(= \frac{1}{\gamma^2} \ for \ x \in B_2^n, \ ||w^*|| = 1\right) \tag{14}$$

iterations where $\gamma = min \ |\langle w, x\rangle|$, and w^* is a concept that correctly labels all the examples.

Proof. Suppose that w^* defines a separating hyperplane and is normalized so that $\gamma = min \ |\langle w, x\rangle|$ calculates the distance from the hyperplane defined by w^* to the closest x_i in the training data. Since the $l(x) \in \{1, -1\}$, we must have

$$\min \ l(x)\langle w, x\rangle = \min \ |\langle w, x\rangle| = \gamma > 0. \tag{15}$$

From (15) and the updating rule for w^j, we conclude that:

$$\langle w^j, w^*\rangle \geq \langle w^{j-1}, w^*\rangle + \gamma. \tag{16}$$

And by induction and the fact that $w^0 = 0$:

$$\langle w^j, w^*\rangle \geq j\gamma \tag{17}$$

Also from the updating rule and the fact that x_i is misclassified by w^{j-1}, we have:

$$\begin{aligned}||w^j||_2^2 &= ||w^{j-1}||_2^2 + ||x_i||_2^2 + l(x_i)\langle w^{j-1}, x_i\rangle \\ &\leq ||w^{j-1}||_2^2 + ||x_i||_2^2 \\ &\leq ||w^{j-1}||_2^2 + \max ||x_i||_2^2\end{aligned} \tag{18}$$

And by induction and the fact that $w^0 = 0$:

$$||w^j||_2^2 \leq j \ \max ||x_i||_2^2 \tag{19}$$

Combine (17), (19), and Cauchy-Schwartz inequality, we have:

$$\begin{aligned} j^2 &\leq \frac{\langle w^j, w^* \rangle^2}{\gamma^2} \leq \frac{||w^j||_2^2 \ ||w^*||_2^2}{\gamma^2} \leq \frac{j \ \max ||x_i||_2^2 \ ||w^*||_2^2}{\gamma^2} \\ j &\leq \frac{||w^*||_2^2 \ \max ||x_i||_2^2}{\gamma^2} \end{aligned} \tag{20}$$

This means that if at the jth iteration, there exists a wrongly classified example, then j must satisfy (20). In other words, if there exist some separating hyperplanes that classify all the examples correctly, then perceptron learning algorithm will find a separating hyperplane within j iterations bounded by (20). □

Theorem 6.2. *The Winnow algorithm starting with $w_i = 1$ for every $i \in \{1, \ldots, n\}$ with the update rule (on mistakes):*
$w_i = (1 + \epsilon) w_i$ for each i s.t. $x_i = 1$ on an example that should be labeled positive, and $w_i = w_i/(1 + \epsilon)$ on an example that should be labeled negative,
has a mistake bound of

$$\mathcal{O}\left(\frac{||w^*||_1^2 \ \max ||x^*||_\infty^2 \log n}{\gamma^2}\right), \tag{21}$$

where w^* is consistent with all examples and $\gamma = min|w^{*T} x|$. In the case of learning k-out-of-r-majorities, this is a mistake bound of $\mathcal{O}(rk \log n)$

Before proving this theorem, we consider a simpler version:
Winnow2 algorithm: Consider binary classification for examples $x \in 0, 1^n$ which have r out of n feature elements. For an example, if any one of these r elements is 1, then this example belongs to class $+1$. Otherwise, the example belongs to class -1.
To training a classifier for these examples is to find which rs are feature elements:
Define: $W = \sum_{i=1}^n w_i > 0$
Initialize $w_i = 1 \ \forall i$
Prediction: $sign(w^T x - n)$
Update(+): when a positive example is predicted as negative, $w_i \leftarrow 2w_i \ \forall i : x_i = 1$
Update(-): when a negative example is predicted as positive, $w_i \leftarrow \frac{1}{2} w_i \ \forall i : x_i = 1$

Theorem 6.3. *The Winnow2 algorithm makes $\mathcal{O}(r \log_2 n)$ mistakes before finding a classifier for examples stated above.*

Proof. To bound the total number of mistakes, we first relate the total number of mistakes that can be made on the positive and negative examples.
Define:

M_+ = the total number of positive example mistakes (positive examples predicted to be negative) made before finding the classifier.

M_- = the total number of negative example mistakes (negative examples predicted to be positive) made before finding the classifier.

To relate the maximum numbers of positive and negative mistakes, consider the total value W of the weight vector.
For a positive example mistake, $\sum_{i:\ x_i=1} W_i \leq n$,

$$\Delta W_+ = \sum_{i:\ x_i=1} 2W_i - W_i = \sum_{i:\ x_i=1} W_i \leq n \tag{22}$$

For a negative example mistake, $\sum_{i:\ x_i=1} W_i \geq n$,

$$\Delta W_- = \sum_{i:\ x_i=1} \frac{1}{2}W_i - W_i = -\frac{1}{2}\sum_{i:\ x_i=1} W_i \leq -\frac{1}{2}n \tag{23}$$

Since the initially $W = n$ and the final W must be positive as well given our updating rule:

$$W = n + M_+\Delta W_+ + M_-\Delta W_+ \geq 0 \tag{24}$$

Plug equation (22) and (23) in (24), we have

$$M_- \leq 2M_+ + 2 \tag{25}$$

By assumption, there are r active elements in x. Given w_i are initialized to 1, each w_i can be doubled $\log_2 n$ times, and there are r of them, so

$$M_+ \leq r \log_2 n \tag{26}$$

Combine (25) and (26), we have

$$M_+ + M_- \leq 3M_+ + 2 = 3r \log_2 n + 2 = \mathcal{O}(r \log_2 n) \tag{27}$$

□

Now we proceed to the proof of **Theorem 6.2**. We will first prove the the case for k-out-of-r majorities, and then generalize it to subspace learning.

Proof. Consider binary classification for examples $x \in 0, 1^n$ which have r out of n feature elements. For an example, if any k of these r elements is 1, then this example belongs to class $+1$. Otherwise, the example belongs to class -1.

To training a classifier for these examples is to find which rs are feature elements and what k is a good threshold:
Define: $W = \sum_{i=1}^{n} w_i > 0$
Initialize $w_i = 1\ \forall i$
Prediction: $sign(w^T x - n)$
Update(+): when a positive example is predicted as negative, $w_i \leftarrow (1+\epsilon)w_i\ \forall i : x_i = 1$
Update(-): when a negative example is predicted as positive, $w_i \leftarrow \frac{1}{(1+\epsilon)}w_i\ \forall i : x_i = 1$

Similarly to the Winnow2 proof, we define:
M_+ = the total number of positive example mistakes (positive examples predicted to be negative) made before finding the classifier.
M_- = the total number of negative example mistakes (negative examples predicted to be positive) made before finding the classifier.

For a positive example mistake, $\sum_{i:\, x_i=1} W_i \leq n$,

$$\Delta W_+ = \sum_{i:\, x_i=1} (1+\epsilon)W_i - W_i = \epsilon \sum_{i:\, x_i=1} W_i \leq \epsilon n \tag{28}$$

For a negative example mistake, $\sum_{i:\, x_i=1} W_i \geq n$,

$$\Delta W_- = \sum_{i:\, x_i=1} \frac{1}{1+\epsilon} W_i - W_i = -\frac{\epsilon}{1+\epsilon} \sum_{i:\, x_i=1} W_i \leq -\frac{\epsilon}{1+\epsilon} n \tag{29}$$

Since the initially $W = n$ and the final W must be positive as well given our updating rule:

$$W = n + M_+ \Delta W_+ + M_- \Delta W_+ \geq 0 \tag{30}$$

Plug equation (28) and (29) in (30), we have

$$\frac{\epsilon}{1+\epsilon} M_- \leq \epsilon M_+ + 1 \tag{31}$$

For all r weights on the feature elements, When a positive mistake is made, at least k of the relevant variables will be increased by $(1+\epsilon)$. When a negative mistake is made, at most $(k-1)$ feature elements fired, and each was decreased by $(1+\epsilon)$. Similar to the proof in Winnow2, the total number of net increase we need on the r feature elements is no more than $r \log_{1+\epsilon} n$, and therefore:

$$kM_+ - (k-1)M_- \leq r \log_{1+\epsilon} n \tag{32}$$

Combine (31) and (32), we have

$$(k - (k-1)(1+\epsilon))M_+ \leq r \log_{1+\epsilon} n + (k-1)\frac{(1+\epsilon)}{\epsilon} \tag{33}$$

and using the approximation $\log_{1+\epsilon} n \approx \frac{1}{\epsilon} \log n$

$$(1+\epsilon)M_+ \leq \frac{r}{\epsilon} \log n + (k-1)\frac{1+\epsilon}{\epsilon}$$

$$M_+ \leq \frac{r}{\epsilon(1+\epsilon)} \log n + \frac{1}{\epsilon}$$

Choosing $\epsilon = \frac{1}{2k}$,

$$M_+ \leq \frac{r}{\epsilon(1+\epsilon)} \log n + \frac{1}{\epsilon}$$

$$\begin{aligned} M = M_+ + M_- &\leq M_+ + (1+\epsilon)M_+ + \frac{1+\epsilon}{\epsilon} \\ &\leq M_+(2+\epsilon) + \frac{1+\epsilon}{\epsilon} \\ &\leq (\frac{r}{\epsilon(1+\epsilon)} \log n + \frac{1}{\epsilon})(2+\epsilon) + \frac{1+\epsilon}{\epsilon} \\ &\leq \frac{(2+\epsilon)}{(1+\epsilon)} \frac{1}{\epsilon} r \log n + \frac{(3+2\epsilon)}{\epsilon} \\ &\leq c_1 kr \log_2 n + c_2 \end{aligned}$$

Therefore:

$$M = \mathcal{O}(rk \log_2 n) \tag{34}$$

In the general half space representation,

$$\sum w_i^* x_i \geq w_o^*$$

we can assume $w_i^* > 0$, because if a $w_i^* < 0$, we replace the corresponding x_i by $y_i = 1 - x_i$. Furthermore, we can assume the x_i are integers by scaling up the real values and rounding, and then representing these integers $[w_i x_i]$ by repeating each x_i a total of w_i^* times.

In this form, we have a k out of r Majority Function where $k = w_0^* \leq r$ and $r = \sum w_i^* = |W^*||_1$.

The guarantee is therefore:

$$\mathcal{O}\left(\frac{||W^*||_1^2 \, ||x||_\infty}{\gamma^2} \log_2 n \right)$$

where $\gamma = \min_{x_i}(w^T x_i)$ and $||x||_\infty$ are added for scaling general half space learning to k-out-of-r-majorities learning. □