**Introduction to Machine Learning** **Fall Semester, 2013**

# Lecture 5: November 10

*Lecturer: Yishay Mansour* *Scribe: ym*

## 5.1 Introduction - Online Learning Model

Imagine a *robot* that needs to classify oranges (objects) as *"export"* (high-quality) or *"local-market"* (low-quality). We want to do this by letting it work and classify oranges as they arrive (online). After making each of its decisions, an *expert* (i.e., an experienced worker) provides it with the "correct" classification so that if it makes mistakes it will be able to modify its prediction method. One hopes that the robot will converge to a "good" classification method.

In this lecture we study the *online learning* protocol.

***Online model:*** The time steps are divided to stages. At time $t$ the following occur:

1. The algorithm receives an unlabeled example $x_t$.
2. The algorithm predicts a classification $b_t$ for $x_t$. The prediction function $h_t$ in stage $t$ is called "current hypothesis".
3. The algorithm is then told the correct answer, $c^*(x) \in \{-1, +1\}$.

In the online model, the number of time steps are usually unbounded.

Note that the online model is adversarial, i.e., we assume that the provided input is selected in the worst possible way for the learning algorithm. For this reason we do not assume any stochastic assumption regarding how the inputs are generated.

We will call $h_t$ (which is used to perform step (2) to generate the prediction), the algorithm's "current hypothesis" (sometimes also referred to as "concept").

A **mistake** is an incorrect prediction, namely $c^*(x_t) \neq b_t$. (Recall that $b_t = h_t(x_t)$.)

The **goal** is to make a (small) bounded number of mistakes, which is independent of the number of predictions. If we achieve our goal, and had already made the maximum number of mistakes, then from this point onwards our hypothesis will classify all observations correctly (otherwise, the adversary can cause the algorithm more mistakes than the bound). Similarly, the online algorithm will not cycle through hypotheses (otherwise the adversary would be able to force an infinite number of errors).

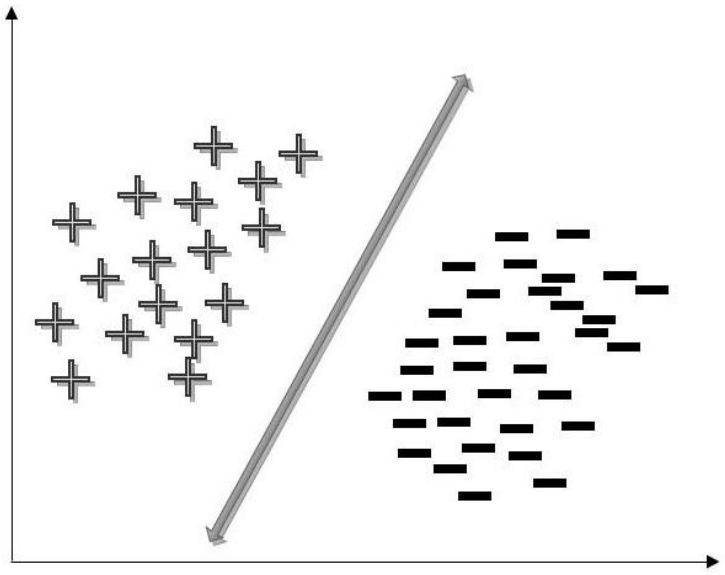

Figure 5.1: A linear separator example

## 5.2 Learning Linear Separators

We will consider the examples as being from $\{0,1\}^n$ or from $\mathbb{R}^n$ (the algorithm will not be sensitive to the difference). In the realizable case, the goal is to find $w_0$ (a threshold) and $\vec{w}$ (a weights vector) defining a hyperplane $\vec{w} \cdot \vec{x} = w_0$ (the notation '$\cdot$' refers to the dot product of two vectors, i.e., $\vec{w} \cdot \vec{x} = \sum_{i=1}^{n} w_i x_i$) such that all positive examples are on one side and all negative examples are on the other. I.e., $\vec{w} \cdot \vec{x} \geq w_0$ for positive $\vec{x}$'s and $\vec{w} \cdot \vec{x} < w_0$ for negative $\vec{x}$'s.

For simplicity, we use a threshold $w_0 = 0$, so we are looking at learning functions like: $\sum_{i=1}^{n} w_i x_i \geq 0$. (We can simulate a nonzero threshold by adding a "dummy" attribute $x_0$ that is always $-1$, and the weight assign to it would be $w_0$.)

We begin by discussing the Perceptron algorithm, an online algorithm for learning linear separators, and one of the oldest algorithms used in machine learning (by Rosenblatt from 1957).

## 5.3 The Perceptron Algorithm

The main idea of this algorithm is that as long as we do not make a mistake, we remain with the same separator. When we do make a mistake - we move the separator towards it.

We scale all examples $\mathbf{x}$ to have Euclidean length 1 (i.e. $\|x\|_2 = 1$), since this doesn't

affect which side of the plane they are on (given our assumption that $w_0 = 0$).

**The Perceptron Algorithm:**

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize $t$ to 1.
2. Given example $\mathbf{x}_t$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x}_t \geq 0$.
3. On a mistake, update as follows: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + c^*(x)\mathbf{x}_t$, namely,
   - Mistake on positive (i.e., $c^*(x) = 1$): $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_t$.
   - Mistake on negative (i.e., $c^*(x) = -1$): $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_t$.

**The intuition:** Suppose we encounter $\mathbf{x}$, (we denote $\mathbf{x}_t$ as simply $\mathbf{x}$ for simplicity). If we make a mistake classifying $\mathbf{x}$, then after the update it follows that

$$\mathbf{w}_{t+1} \cdot \mathbf{x} = (\mathbf{w}_t + c^*(x)\mathbf{x}) \cdot \mathbf{x} = \mathbf{w}_t \cdot \mathbf{x} + c^*(x)\mathbf{x} \cdot \mathbf{x} = \mathbf{w}_t \cdot \mathbf{x} + c^*(x).$$

This implies that if the example was positive $c^*(x) = +1$ we increase the dot product $(\mathbf{w}_t \cdot \mathbf{x})$ and if the example was negative $c^*(x) = -1$ we decrease the dot product $(\mathbf{w}_t \cdot \mathbf{x})$. So the update is always in the "right" direction.

**Theorem 5.1** *Let $\mathcal{S}$ be a sequence of labeled examples consistent with a linear threshold function $\mathbf{w}^* \cdot \mathbf{x} \geq 0$, where $\mathbf{w}^*$ is a unit-length vector. Then the number of mistakes $M$ on $\mathcal{S}$ made by the online Perceptron algorithm is at most $(1/\gamma)^2$, where*

$$\gamma = \min_{\mathbf{x} \in \mathcal{S}} \frac{|\mathbf{w}^* \cdot \mathbf{x}|}{||\mathbf{x}||}.$$

*(I.e., if we scale examples to have Euclidean length 1, then $\gamma$ is the minimum distance of any example to the plane $\mathbf{w}^* \cdot \mathbf{x} = 0$.)*

The parameter "$\gamma$" is often called the "margin" of $\mathbf{w}^*$ (or more formally, the $L_2$ margin because we are scaling by the $L_2$ lengths of the target and examples). The margin $\gamma$ represents the minimal distance of each example $\mathbf{x}$ from $\mathbf{w}^*$, after normalizing both $\mathbf{w}^*$ and the examples. Another way to view the quantity $\frac{\mathbf{w}^* \cdot \mathbf{x}}{||\mathbf{x}||}$ is that it is the cosine of the angle between $\mathbf{x}$ and $\mathbf{w}^*$, so we will also use $\cos(\mathbf{w}^*, \mathbf{x})$ for it.

*Proof of Theorem 5.1.* We are going to consider two quantities: $\mathbf{w}_t \cdot \mathbf{w}^*$ and $||\mathbf{w}_t||$. We will show a claim regarding the change in each of the two quantities. We assume, without loss of generality, that we always make mistakes, since if we do not make a mistake we do not change the weights.

**Claim 1:** $\mathbf{w}_{t+1} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$. That is, every time we make a mistake, the dot-product of our weight vector with the target increases by at least $\gamma$. (Note that since $\mathbf{w}_1 \cdot \mathbf{w}^* = 0$, this implies that $\mathbf{w}_{t+1} \cdot \mathbf{w}^* \geq 0$.

**Proof:** if $\mathbf{x}$ was a positive example, then we get $\mathbf{w}_{t+1} \cdot \mathbf{w}^* = (\mathbf{w}_t + \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w}_t \cdot \mathbf{w}^* + \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$ (by definition of $\gamma$ and since $||\mathbf{x}|| = 1$). Similarly, if $\mathbf{x}$ was a negative example, we get $(\mathbf{w}_t - \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w}_t \cdot \mathbf{w}^* - \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$. (Recall that $\mathbf{x} \cdot \mathbf{w}^* < 0$ when $\mathbf{x}$ is a negative example.)

**Claim 2:** $||\mathbf{w}_{t+1}||^2 \leq ||\mathbf{w}_t||^2 + 1$. That is, every time we make a mistake, the length squared of our weight vector increases by at most 1.

**Proof:** if $\mathbf{x}$ was a positive example, we get $||\mathbf{w}_t + \mathbf{x}||^2 = ||\mathbf{w}_t||^2 + 2\mathbf{w}_t \cdot \mathbf{x} + ||\mathbf{x}||^2$. This is less than $||\mathbf{w}_t||^2 + 1$ because $\mathbf{w}_t \cdot \mathbf{x}$ is negative (remember, we made a mistake on $\mathbf{x}$). Same thing (flipping signs) if $\mathbf{x}$ was negative but we predicted positive.

Claim 1 implies that after $M$ mistakes, $\mathbf{w}_{M+1} \cdot \mathbf{w}^* \geq \gamma M$. On the other hand, Claim 2 implies that after $M$ mistakes, $||\mathbf{w}_{M+1}|| \leq \sqrt{M}$. Now, all we need to do is use the fact that $\mathbf{w}_t \cdot \mathbf{w}^* \leq ||\mathbf{w}_t||$: Since $\mathbf{w}^*$ is a unit vector, then a vector maximizing the dot product $\mathbf{w}_t \cdot \mathbf{w}^*$ would be $\frac{\mathbf{w}_t}{||\mathbf{w}_t||}$, which entails that

$$\mathbf{w}_t \cdot \mathbf{w}^* \leq \mathbf{w}_t \cdot \frac{\mathbf{w}_t}{||\mathbf{w}_t||} = \frac{||\mathbf{w}_t||^2}{||\mathbf{w}_t||} = ||\mathbf{w}_t||.$$

Therfore, we get

$$\gamma M \leq \mathbf{w}_{M+1} \cdot \mathbf{w}^* \leq ||\mathbf{w}_{M+1}|| \leq \sqrt{M}$$

and thus $M \leq \frac{1}{\gamma^2}$. □

### 5.3.1 Perceptron - Unrealizable case

**What if there is no perfect separator ($w^*$)?** What if only *most* of the data is separable by a large margin (as seen on Figure 5.2), or what if $\mathbf{w}^*$ is not perfect? We need in this case to reconsider Claim 1. Claim 1 said that we make "$\gamma$ amount of progress" on every mistake. Now it's possible there will be mistakes where we make very little progress, or even negative progress. One thing we can do is bound the total number of mistakes we make in terms of the total distance we would have to move the points to make them actually separable by margin $\gamma$. Let us call that $\text{TD}_\gamma$, where $TD_\gamma = \sum_{t=1}^{T} \max\{0, \gamma - c^*(\mathbf{x}_t)(\mathbf{x}_t \cdot \mathbf{w}^*)\}$. Similar to Claim 1, we get that after $M$ mistakes, $\mathbf{w}_{M+1} \cdot \mathbf{w}^* \geq \gamma M - \text{TD}_\gamma$. So, combining with Claim 2, we get that $\sqrt{M} \geq \gamma M - \text{TD}_\gamma$, which gives an upper bound on $M$.

To compute the upper bound on $M$ we can solve the quadratic equation, but a simple upper bound is $M \leq \frac{1}{\gamma^2} + (\frac{2}{\gamma})\text{TD}_\gamma$. The quantity $\frac{1}{\gamma}\text{TD}_\gamma$ is called the total *hinge-loss* of $\mathbf{w}^*$. The hinge loss of a point $\mathbf{x}$ with respect to $\mathbf{w}^*$ can be define as $\max\{0, 1 - y\}$, where $y = \frac{c^*(x) \cdot \mathbf{x} \cdot \mathbf{w}^*}{\gamma}$ and $c^*(x)$ is the classification of $\mathbf{x}$. The expression for the hinge loss is equivalent to $\frac{1}{\gamma}\max\{0, \gamma - c^*(\mathbf{x})\mathbf{x} \cdot \mathbf{w}^*\}$. The hinge loss is a loss function that begins paying linearly

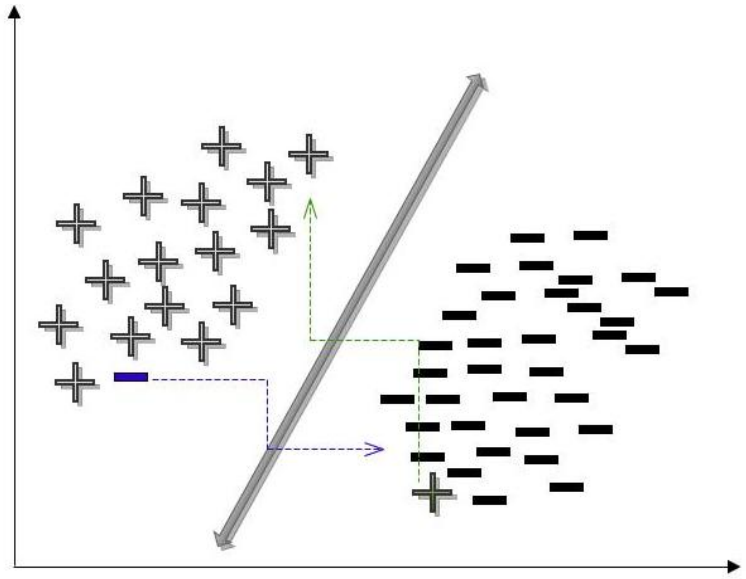

Figure 5.2: A case in which no perfect separator exists

as it approaches the hyperplane after the margin parameter $\gamma$ (see Figure 5.3). Note that the maximum contribution of an error is $1 + \gamma$, since the examples are normalized to a unit length.

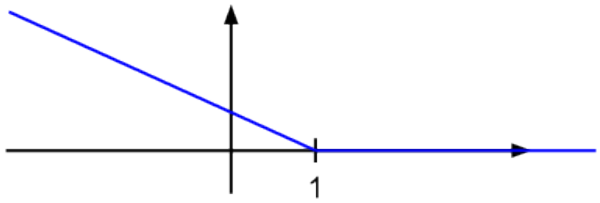


Figure 5.3: Illustrating hinge loss

This is partially good news: we cannot necessarily say that we are making only a number of mistakes proportional to that which $\mathbf{w}^*$ does (in fact, the problem of finding an approximately-optimal separator is NP-hard), but we can say we are doing well in terms of the "total distance" parameter or the hinge loss of $\mathbf{w}^*$.

## 5.4 WINNOW2

We now describe an alternative algorithm for learning hyperplanes. Here it will be more convenient to assume that $\mathbf{x} \in \{0,1\}^n$ rather than $\mathbf{x} \in [0,1]^n$, but the algorithm extends to the latter case as well. Also we will discuss the case where the coefficients are only positive.

Suppose we had a separation between the positive examples and the negative examples by the hyperplane $(\mu_1, \ldots, \mu_n)$ where

$$\sum_{i=1}^{n} \mu_i \cdot x_i \ \geq \ \theta \ \Leftrightarrow c^*(\mathbf{x}) = 1$$

$$\sum_{i=1}^{n} \mu_i \cdot x_i \ \leq \ \theta - \delta \ \Leftrightarrow c^*(\mathbf{x}) = 0$$

($\delta$ will influence the algorithm's mistake complexity.)

We initialize the weight vector to 1, i.e., $w_i = 1$. As before, we update only when the algorithm makes an error. WINNOW has a parameter $\beta > 1$. We distinguish between two error types. For False-Negative errors, we multiply the weights by the factor $\beta$, and call this a *promotion step*. For False-Positive errors, we divide the weights by $\beta$, and call this a *demotion* step). We use $\beta = 1 + \frac{\delta}{2}$ to derive our bounds.

**The algorithm's response to mistakes:**

| Name | Update Scheme | target | prediction |
|---|---|---|---|
| Demotion | $\forall x_i = 1 \ \ set \ \ w_i = w_i/\beta$ | 0 | 1 |
| Promotion | $\forall x_i = 1 \ \ set \ \ w_i = \beta \cdot w_i$ | 1 | 0 |

Figure 5.4: The update scheme used by WINNOW2

**Theorem 5.2** *If there exists a hyperplane $(\mu_1, \ldots, \mu_n)$ with a separation of $0 < \delta \leq 1$, and if we run* WINNOW2 *with $\beta = 1 + \frac{\delta}{2}$ and $\theta \geq 1$, then the number of mistakes is bounded by:*

$$O(\frac{1}{\delta^2} \cdot \frac{n}{\theta} \ + \ (\frac{1}{\delta} \ + \ \frac{1 \cdot ln\theta}{\delta^2}) \cdot \sum_{i=1}^{n} \mu_i)$$

In order to prove the theorem we first prove three lemmas. We start by distinguishing between the mistakes according to their cause. Let $\mathbf{u}$ – the number of promotion steps, and $\mathbf{v}$ – the number of demotion steps.

The following lemma shows that the number of demotion steps cannot be much larger than the number of promotion steps.

**Lemma 5.3** *For any* $\mathbf{u}$ *and* $\mathbf{v}$*,*

$$\mathbf{v} \;\leq\; \frac{\beta}{\beta-1}\cdot\frac{n}{\theta} \;+\; \beta\cdot\mathbf{u}$$

**Proof:** Again consider $\sum_{i=1}^{n} w_i$. A promotion step increases the sum $\sum w_i$ by at most $(\beta-1)\cdot\theta$. A demotion step decreases the sum $\sum w_i$ by at least $(1-\frac{1}{\beta})\cdot\theta$.
Combining the two,

$$0 \;\leq\; \sum_{i=1}^{n} w_i \;\leq\; n \;+\; (\beta-1)\cdot\theta\cdot\mathbf{u} \;-\; \mathbf{v}\cdot\frac{\beta-1}{\beta}\cdot\theta$$

But since the weights are never negative, we have:

$$n \;+\; (\beta-1)\cdot\theta\cdot\mathbf{u} \;-\; \mathbf{v}\cdot\frac{\beta-1}{\beta}\cdot\theta \;\geq\; 0$$

Thus,

$$\mathbf{v}\cdot\frac{\beta-1}{\beta} \;\leq\; \frac{n}{\theta} \;+\; \mathbf{u}\cdot(\beta-1),$$

which implies that:

$$\mathbf{v} \;\leq\; \frac{\beta}{\beta-1}\cdot\frac{n}{\theta} \;+\; \beta\cdot\mathbf{u}.$$

□

The following lemma shows that any individual weight can not be too large.

**Lemma 5.4** *For all* $i$*,* $w_i \;\leq\; \beta\cdot\theta$*.*

**Proof:** Since $\theta \geq 1$ and $\beta > 1$, all the weights are initially $1 \leq \beta\cdot\theta$. For any $j$, the value of $w_j$ is only promoted during a trial in which $x_j = 1$ and $\sum_{i=1}^{n} w_i\cdot x_i \;\leq\; \theta$. These conditions can only occur if $w_j \leq \theta$ immediately prior to the promotion. Thus $w_j \leq \beta\cdot\theta$ after the promotion. □

**Lemma 5.5** *After* $\mathbf{u}$ *promotion steps and* $\mathbf{v}$ *demotion steps: there exists an* $i$ *for which*

$$log w_i \;\geq\; \frac{\theta\cdot\mathbf{u} - (\theta-\delta)\cdot\mathbf{v}}{\sum_{i=1}^{n}\mu_i}\cdot log\beta$$

**Proof:** We consider the function

$$\Phi = \Pi_{i=1}^{n} w_i^{\mu_i}$$

**Promotion Step** $\sum_{i=1}^{n} \mu_i \cdot x_i \geq \theta$. Let $w_i'$ be the weights after the promotion step. For each $x_i = 1$, $w_i' = w_i \cdot \beta$.

$$\Phi' = \Phi \cdot \Pi_{i=1}^{n} (\beta^{x_i})^{\mu_i} = \Phi \cdot \beta^{\sum_{i=1}^{n} x_i \mu_i} \geq \Phi \cdot \beta^{\theta}$$

**Demotion Step** $\sum_{i=1}^{n} \mu_i \cdot x_i < \theta - \delta$. Let $w_i'$ be the weights after the demotion step. For each $x_i = 1$, $w_i' = \frac{w_i}{\beta}$.

$$\Phi' = \Phi \cdot \Pi_{i=1}^{n} (\frac{1}{\beta_i^x})^{\mu_i} = \Phi \cdot \beta^{-\sum_{i=1}^{n} x_i \mu_i} \leq \Phi \cdot \beta^{-(\theta-\delta)}$$

Initially, $\prod_{i=1}^{n} w_i^{\mu_i} = 1$. After **u** promotion steps and v demotion steps, we have
From the above facts we can conclude that :

$$\prod_{i=1}^{n} w_i^{\mu_i} \geq \beta^{\theta \cdot \mathbf{u}} \cdot \beta^{-(\theta-\delta)\cdot \mathbf{v}}$$

We can take the log of both sides

$$\sum_{i=1}^{n} \mu_i log w_i \geq [\mathbf{u} - (\theta - \delta) \cdot \mathbf{v}] \, log\beta$$

Which means that there exists a $i$ for which:

$$log w_i \geq \frac{[\theta \cdot \mathbf{u} - (\theta - \delta) \cdot \mathbf{v}]}{\sum_{i=1}^{n} \mu_i} \cdot log\beta$$

□

**Proof of theorem 2.5**: From Lemmas 5.3 – 5.5 we have:

$$\frac{\theta \cdot \mathbf{u} - (\theta - \delta) \cdot \mathbf{v}}{\sum \mu_i} \cdot log\beta \;\leq\; log(\beta \cdot \theta) = log\beta \;+\; log\theta.$$

Recall that the total number of mistakes is equal to **u**+**v**.
Since $\beta > 1$ and $\mu_i \geq 0$ we get:

$$\theta \cdot \mathbf{u} - (\theta - \delta) \cdot \mathbf{v} \;\leq\; (1 + \frac{log\theta}{log\beta}) \cdot \sum_{i=1}^{n} \mu_i$$

and by using Lemma 5.3 it is sufficient to gurantee that,

$$\theta \cdot \mathbf{u} - (\theta - \delta) \cdot (\frac{\beta}{\beta - 1} \cdot \frac{n}{\theta} \ + \ \beta \cdot \mathbf{u}) \ \leq \ (1 + \frac{log\theta}{log\beta}) \cdot \sum_{i=1}^{n} \mu_i.$$

We can now use the fact that $\beta = 1 + \frac{\delta}{2}$. The above requirement can be reduced to,

$$\mathbf{u} \leq (\frac{1}{\delta^2} + \frac{2 \cdot log\theta}{\delta^2}) \cdot \sum_{i=1}^{n} \mu_i + 2 \cdot \frac{n}{\theta} \cdot \frac{\theta - \delta}{\delta^2} \cdot (1 + \frac{\delta}{2})$$

From Lemma 5.5 we have that,

$$\mathbf{u} + \mathbf{v} \leq \mathbf{u} + \frac{\beta}{\beta - 1} \cdot \frac{n}{\Theta} + \beta \cdot \mathbf{u} = (2 + \frac{\delta}{2}) + (\frac{1 + \frac{\delta}{2}}{\frac{\delta}{2}}) \cdot \frac{n}{\theta}$$

Which concludes the proof. ■

## 5.5 Winnow versus Perceptron:

One can generalize the basic analysis we did for Winnow to the case of learning linear separators; the guarantee depends on the $L_1, L_\infty$ margin of the target. In particular, if the target vector $w^*$ is a linear separator such that $w^* \cdot x > c$ on positives and $w^* \cdot x < c - \alpha$ on negatives, then the mistake bound of Winnow is:

$$O\left(\left(\frac{L_1(w^*)L_\infty(X)}{\alpha}\right)^2 \log(n)\right),$$

And the mistake bound of Perceptron is:

$$O\left(\left(\frac{L_2(w^*)L_2(X)}{\alpha}\right)^2\right).$$

The quantity $\gamma = \frac{\alpha}{L_1(w^*)L_\infty(X)}$ is called the "$L_1, L_\infty$" margin of the separator, and our bound is $O(\frac{1}{\gamma^2} \cdot \log(n))$. On the other hand, the Perceptron algorithm has a mistake bound of $O(\frac{1}{\gamma^2})$ where $\gamma = \frac{\alpha}{L_2(w^*)L_2(X)}$ (this called the "$L_2, L_2$" margin of the separator).

One thing that is lost using Winnow, is obviously the $\log(n)$, but the norms are also different.

Intuitively, if $n$ is large but most features are irrelevant (i.e., target is sparse but examples are dense), the Winnow is better because adding irrelevant features increases $L_2(X)$ but not $L_\infty(X)$. On the other hand, if the target is dense and examples are sparse, then Perceptron is better.