67577 – Intro. to Machine Learning **Fall semester, 2004/5**

Lecture 3: Linear Separators I: Perceptron

Lecturer: Amnon Shashua *Scribe: Amnon Shashua* [1]

In the previous lectures we focused on the boolean concept family with the purpose of studying formal aspects of learning models such as mistake bounds and accuracy and confidence parameters associated with the PAC learning model. In this lecture (and some of those that follow) we will begin to consider a widely popular concept class which consists of linear decision surfaces (a hyperplane in R^n) separating positive and negative examples. The concept class is popular for a number of reasons: (i) in many interesting and practical situations one can obtain a fairly good classifier using linear separating functions, (ii) the analysis of such classifiers is well understood, and (iii) with the algorithms we will study in this and next few lectures one can easily embed the instance space into a high dimensional space thereby effectively extending the concept class to include non-linear decision surfaces.

In this lecture we will start with an online algorithm called the Perceptron (Rozenblatt, 1957) for finding a consistent hypothesis (a separating hyperplane). Recall that in an online setting we observe a sequence of instance-label pairs. Unlike batch-learning where one can expect a training set to be given in advance, here we are concerned with situations where examples are given one at a time in an incremental fashion and the learner's task is to update its hypothesis in order to minimize the prediction error. In the mistake-bound mode, the prediction error is simply the total number of mistakes made on the training set of length m. Like with batch-learning, we consider the "realizable" case where a target concept exists, and the unrealizable case where there is no guarantee that there exists a hypothesis which is consistent with all the examples provided.

In the realizable case, the learner will desire to reach a point where the number of mistakes is fixed, i.e., there is an upper-bound on the number of mistakes. In the conjunction learning task with n literals we saw that the upper bound does not depend on the length of the training set or its complexity — it takes at most n mistakes for the best on-line algorithm on the worst training set. In more typical on-line tasks, such as the Perceptron learning discussed here, the upper-bound depends on how "complex" the training set is but not on the length of the training set and in some cases not on the dimension of the input space X. We will see that the upper-bound depends only on the training data and not on the particular (consistent) hypothesis. We will discuss the unrealizable case later in the lecture.

We will focus today on the case where each instance $\mathbf{x}_i$ is in R^n. For convenience, each we use set $Y = \{-1, +1\}$ (rather than $Y = \{0, 1\}$. Thus, the hypotheses we will consider are functions $h(\mathbf{x})$ that map instances from R^n into one of the two possible labels in Y. We will focus on a particular hypotheses class that consists of all possible hyperplanes. Each hypothesis in the class is of the form,

$$h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \quad \text{where} \quad \mathbf{w} \in \mathbb{R}^n \quad .$$

Therefore, each hypothesis in the class corresponds to a hyperplane. A slightly more general class of hyperplanes is obtained if we let the hyperplane $\mathbf{w}$ pass through a general point in $\mathbb{R}^n$. The

[1] Sources: this handout evolved from Yoram Singer notes of IML'01. This year there were contributions from Shai Schwartz as well.

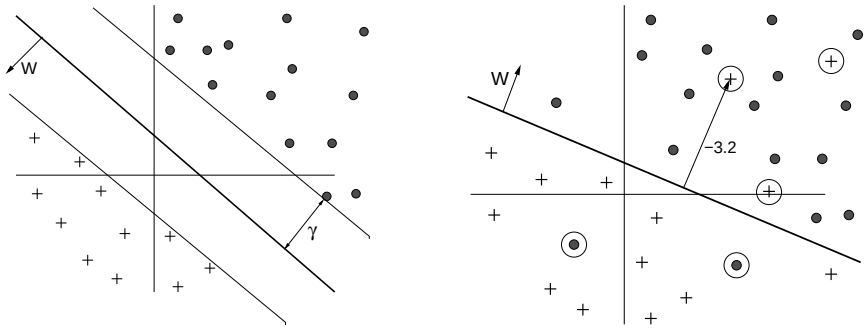

Figure 3.1: $\mathbf{w}$ linearly separates the sample on the left with a minimal margin value of γ while the sample on the right is not separable by $\mathbf{w}$.

hypothesis class in this case is

$$C = \{h(\mathbf{x}) \mid h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \ ; \ \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}\} \quad .$$

As in the previous class we assumed that there exists $c_t \in C$ that labels correctly all the instances. In the case of hyperplanes this assumption amounts to the existence of a pair $(\mathbf{w}^\star, b^\star)$. For each example (instance-label pair) $(\mathbf{x}_i, y_i)$ we get that

$$c_t(\mathbf{x}_i) = y_i \ \Leftrightarrow \ \text{sign}(\mathbf{w}^\star \cdot \mathbf{x}_i + b^\star) = y_i \ \Leftrightarrow \ y_i(\mathbf{w}^\star \cdot \mathbf{x}_i + b^\star) > 0 \quad .$$

The term $y_i(\mathbf{w}^\star \cdot \mathbf{x}_i + b^\star)$ is called the (signed) margin of example i. On the left hand side of Fig. 3.1 we show an illustration of the notion of margin. In the figure all the examples from the negative class ($y_i = -1$) are denoted with circles and the positive examples with pluses. (Try to identify b for the left hand side figure.) In the left plot, we see that the hyperplane $\mathbf{w}$ separates $\mathbb{R}^2$ into two subsets where each subset contains examples from a *single* class. We therefore say that $\mathbf{w}^\star, b^\star$ is a *linear separator* for the the sample $Z = \{(\mathbf{x}_i, y_i)\}$.

Let us assume for a while that $b^\star = 0$ and we thus omit b. Toward the end of this class we will discuss how to learn linear separators for which $b \neq 0$. Note that if $\mathbf{w}^\star$ is a linear separator so is $\alpha\mathbf{w}^\star$ for any $\alpha > 0$. We can therefore normalize $\mathbf{w}^\star$ such that $\|\mathbf{w}^\star\| = 1$. (Recall the l_2 norm of a vector v, denoted $\|v\|_2$, or in short $\|v\|$, is $\sqrt{\sum_{i=1}^n v_i^2}$.) The margin of an example $(\mathbf{x}_i, y_i)$ with respect to a normalized linear separator $\mathbf{w}^\star$ caries a geometrical meaning. The value of the margin, $y_i(\mathbf{w}^\star \cdot \mathbf{x}_i)$, is the distance of $\mathbf{x}_i$ from $\mathbf{w}^\star$. If we take however an arbitrary hyperplane $\mathbf{w}$ which does not necessarily linearly separate the sample Z then we *might* obtain *negative* margin values, that there might exists an example $(\mathbf{x}_i, y_i)$ such that $y_i(\mathbf{w} \cdot \mathbf{x}_i) < 0$. A negative margin value designates the fact a point $\mathbf{x}_i$ lies on the "wrong" of the half-space induces by a linear separator. On the right hand side of plot of Fig. 3.1 we show a sample that is not linearly separable with respect to the hyperplane $\mathbf{w}$ in the plot. All the examples which obtain a negative margin are designated with circle around them.

Given the geometrical interpretation above we can say that the goal of an online learning algorithms is to learn a hyperplane such that the learned hyperplane will eventually obtain a positive margin on all the examples. Put another way, since the class of all possible hyperplanes is not finite (it's not even countable) the margin will be our tool for analyzing the the mistake bounds the learning algorithm we describe in the sequel.

The algorithm for learning linear separators is called the Perceptron algorithm. Its roots go back to 1957 and it was invented by Rosenblatt and was analyzed by numerous researchers from

different perspectives. It was quite popular at the time as it can be thought as an engineering simplification of how a real neuron works. A simplified model of how a real neuron operates is that it takes n inputs, multiplies by some weight vector $\mathbf{w}$, compares the result with some threshold b (or applies some logistic function which approximates a hard threshold decision) and outputs either +1 or -1. Perceptrons do fairly well in practice, they can learn many functions of interest and are easy to train. They can also model high-order (non-linear) decision surfaces (but that later in the course) easily and they are optimal Bayesian classifiers in some cases. The period of excitement from Perceptrons have passed, but the the main building blocks and analysis will serve as a useful introductory material for subsequent lectures.

The Perceptron algorithm maintains a single hyperplane. We denote the hyperplane used for prediction on round t by $\mathbf{w}^t$. This hyperplane is modified whenever the Perceptron algorithm made a prediction mistake. The pseudo-code of the algorithm is given in Fig. 3.2. For convenience, we therefore set $\mathbf{w}^{t+1}$ to be $\mathbf{w}^t$ if the Perceptron algorithm predicted the label correctly. If there was a prediction error then $\mathbf{w}^{t+1}$ is set to be $\mathbf{w}^t + y_t\mathbf{x}_t$. It can easily verified that one of the effects of this update is that the margin obtained $\mathbf{w}^{t+1}$ on $\mathbf{x}_t$ is larger than the (negative) margin of $\mathbf{w}^t$ on $\mathbf{x}_t$:

$$y_t(\mathbf{w}^{t+1} \cdot \mathbf{x}_t) = y_t(\mathbf{w}^t + y_t\mathbf{x}_t) \cdot \mathbf{x}_t = y_t(\mathbf{w}^t \cdot \mathbf{x}_t) + \|\mathbf{x}_t\|^2,$$

Since the term $\|\mathbf{x}_t\|^2$ is non-negative, the resulting update step pushes the dot product to become more positive (it may take more than a single error to push the dot product over the threshold). Note that without further assumptions this algorithm may never converge. For example, an adversarial choice of examples would generate the magnitudes $\|\mathbf{x}\|$ of subsequent examples to decrease geometrically such that any given example contributes more weight than the remainder of the series of examples.

The assumption we will need to make is that there is some *interval* of consistent solutions and then the upper bound of the number of errors would depend on the size of that interval. For example, take the simplest concept task (of this nature) where one wishes to find a scalar $z \in [0, 1]$ that separates positive and negative examples given in the interval $[0, 1]$. Such a concept cannot be learned by a finite number of examples because potentially the specification of z requires an infinite amount of information. An adversary will not choose z in advance. Instead, the learner at every stage has a certain region of uncertainty (which gets smaller after each mistake) for z and the adversary can then present the learner with a point in the uncertainty region and tell the learner that his guess was wrong. Since the uncertainty region is infinite divisible this process can go on forever.

The assumption we will make is that:

$$\exists \gamma > 0, \mathbf{w}^* \quad s.t. \quad \forall (\mathbf{x}_i, y_i) \in S, \ (y_i \mathbf{w}^* \cdot \mathbf{x}_i) > \gamma.$$

In other words, we have assumed the existence of a *gap* so that the learner does not need to learn $\mathbf{w}^*$ perfectly but just well enough to correctly classify outside of the gap. The gap γ is the smallest margin over all the examples in S with respect to $\mathbf{w}^*$. We will refer to this gap as the "margin obtained by $\mathbf{w}^\star$." The proof below shows that the number of prediction errors that the Perceptron algorithm makes is bounded and inversely proportional to the margin obtained by $\mathbf{w}^\star$.

Theorem 1 *Let $(\bar{x}^1, y^1), \ldots, (\bar{x}^T, y^T)$ be an input sequence for the Perceptron algorithm described in Fig. 3.2 where $\mathbf{x}_t \in \mathbb{R}^n$ and $y^t \in \{-1, +1\}$. Denote by $R^2 = \max_t \|\mathbf{x}_t\|^2$. Assume that there is a vector $\mathbf{w}^t*$ of a unit norm, $\|\mathbf{w}^t*\| = 1$, that classifies the entire sequence correctly with margin*

$$\gamma = \min_t \{y^t(\mathbf{w}^t* \cdot \mathbf{x}_t)\} > 0 \ .$$

Initialize:

- Set $\mathbf{w}^1 = 0$ $(\mathbf{w}^1 \in \mathbb{R}^n)$.

Loop: For $t = 1, 2, \ldots, T$

- Get a new instance $\mathbf{x}_t \in \mathbb{R}^n$.
- Predict $\hat{y}_t = \text{sign}(\mathbf{w}^t \cdot \mathbf{x}_t)$.
- Get a new label y_t.
- If $y_t \neq \hat{y}_t$ update $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + y_t \mathbf{x}_t$ (otherwise $\mathbf{w}^{t+1} = \mathbf{w}^t$).

Output : $h(\mathbf{x}) = \text{sign}(\mathbf{w}^{T+1} \cdot \mathbf{x})$.

Figure 3.2: Pseudo-code of the Perceptron algorithm.

Then, the number of mistakes that the Perceptron algorithm makes is at most

$$\frac{R^2}{\gamma^2} \ .$$

Proof:
We prove the theorem by showing that the angle between $\mathbf{w}^t$ and $\mathbf{w}^\star$ decreases as a function of t. To do so we examine $\phi^t = \frac{\mathbf{w}^t \cdot \mathbf{w}^\star}{\|\mathbf{w}^t\|}$. Let us first look at $\mathbf{w}^{t+1} \cdot \mathbf{w}^\star$. If on round t there was a prediction mistake then

$$\begin{aligned} \mathbf{w}^{t+1} \cdot \mathbf{w}^\star &= (\mathbf{w}^t + y_t \mathbf{x}_t) \cdot \mathbf{w}^\star \\ &= \mathbf{w}^t \cdot \mathbf{w}^\star + y_t (\mathbf{x}_t \cdot \mathbf{w}^\star) \\ &\geq \mathbf{w}^t \cdot \mathbf{w}^\star + \gamma \ . \end{aligned} \tag{3.1}$$

To derived the last inequality we used the fact that $\mathbf{w}^\star$ separates the examples with a margin of at least γ. Furthermore, if after t rounds the Perceptron algorithm made m mistakes then by unraveling successively Eq. (3.1) we get,

$$\mathbf{w}^t \cdot \mathbf{w}^\star \geq \mathbf{w}^1 \cdot \mathbf{w}^\star + m\gamma = m\gamma \ . \tag{3.2}$$

Examining the norm of $\mathbf{w}^{t+1}$ assuming there was a prediction error on round t we get,

$$\begin{aligned} \|\mathbf{w}^{t+1}\|_2^2 \ = \ \mathbf{w}^{t+1} \cdot \mathbf{w}^{t+1} &= (\mathbf{w}^t + y_t \mathbf{x}_t) \cdot (\mathbf{w}^t + y_t \mathbf{x}_t) \\ &= \|\mathbf{w}^t\|_2^2 + \underbrace{2 y_t (\mathbf{w}^t \cdot \mathbf{x}_t)}_{<0} + \underbrace{\|\mathbf{x}_t\|_2^2}_{\leq R^2} \\ &\leq \|\mathbf{w}^t\|_2^2 + R^2 \ . \end{aligned} \tag{3.3}$$

If we again unravel the inequality above, assuming m errors in t rounds, get,

$$\|\mathbf{w}^t\|_2^2 \leq m R^2 \ . \tag{3.4}$$

To derive the inequality above we used the fact the there was a prediction error and hence the margin of $\mathbf{w}^t$ on $(\mathbf{x}_t, y_t)$ is negative and the fact that the norm of all the instances is bounded above

by R. We now use Cauchy's Inequality

$$\sum_{i=1}^{n} a_i b_i \leq \sqrt{\left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right)} \ ,$$

and combine it with Eq. (3.3) and Eq. (3.4) to get that,

$$m\gamma \leq \mathbf{w}^t \cdot \mathbf{w}^\star \leq \|\mathbf{w}^t\| \leq \sqrt{mR^2} \ .$$

The above inequality implies that $\sqrt{m} \leq \frac{R}{\gamma}$ and thus $m \leq \frac{R^2}{\gamma^2}$. ■

We see form the proof that the angle between the hypothesis $\mathbf{w}^t$ and the target hypothesis $\mathbf{w}^*$ is bounded below by $m\gamma$ and above by $\sqrt{m}R$. Any consistent hypothesis will fall in this loose interval, therefore the mistake bound applies to any consistent hypothesis including ones who have an arbitrarily small margin. If the hypothesis returned by the Perceptron makes a number of mistakes very close to the bound, then the associated hyperplane has a large margin. On the other hand if the number of mistakes is much smaller than the bound, then the separating hyperplane may be quite close to the instance points, i.e., have a small margin. In other words, the bound on the number of mistakes is a property of the complexity (i.e., the margin size γ) of the training set Z and not the particular hypothesis to which the Perceptron converges to.

It remains to describe how to use the Perceptron when $b^\star \neq 0$. Rather than developing a generalized algorithm we can use the Perceptron algorithm as is and modify the input instances as follows. Instead of using the original instances which are vectors in $\mathbb{R}^n$ we use vectors in $\mathbb{R}^{n+1}$. Denote by $\bar{\mathbf{x}}_t$ the expanded instance used on round t. We set the first n components of $\bar{\mathbf{x}}_t$ to be $\mathbf{x}_t$ and set the last component to 1. We now run the Perceptron algorithm with input vectors in $\mathbb{R}^{n+1}$. If $\mathbf{w}^\star, b^\star$ separates the sample with a margin of γ, then $\tilde{\mathbf{w}}^\star$ separates the expanded instances as well. Note that the norm of $\tilde{\mathbf{w}}^\star$ is $\sqrt{\|\mathbf{w}^\star\|_2^2 + b^{*2}} = \sqrt{1 + b^{*2}}$. Hence, the margin obtained by $\tilde{w}^\star$ on the expanded sample is $\gamma/\sqrt{1 + b^{*2}}$ (but the geometric interpretation remains the same — the distance of the closest point to the hyperplane).

To summarize the realizable case, the Perceptron will converge to some consistent hypothesis with a number of mistakes below a bound which depends only on the *largest* margin of the training data. The hypothesis returned by the Perceptron is not necessarily the one with maximal margin, though. Nevertheless, the mistake bound depends on the maximal margin associated with the training set, rather than on the margin associated with the hypothesis returned by the Perceptron. Another worthwhile point to note is that the dimension of the input space is not a factor in the mistake bound. This can be exploited by adding additional variables in order to handle non-linear decision surfaces without significantly changing the error bounds. This will be further elaborated in subsequent lectures.

3.1 Unrealizable Case

Recall that in batch-learning, the error associated with a hypothesis $h(\mathbf{x})$ is bounded by above $Opt(C)$ plus a measure which depends on the class complexity, the length of training set and the confidence measure: $err(h) \leq Opt(C) + \epsilon$. The measure $Opt(C)$ is the error obtained by the optimal hypothesis (one which has the smallest error). The situation with on-line learning in the unrealizable case is similar. The number of mistakes made by a hypothesis is bounded form above by a combination of two terms. The first is a term which depends on the complexity of the hypothesis on the training data (like in previous section) and the second term is the number of

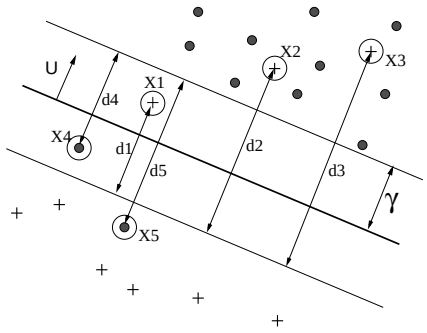

Figure 3.3: Illustration of the slack variables added to each example that does not achieve a margin of γ in the analysis of the Perceptron algorithm for the non-separable case.

mistakes made by the *best* hypothesis on the training set. The combination provides a tradeoff, as the best hypothesis (makes the smallest number of mistakes) may be associated with a high complexity measure (say a very small margin). The on-line learner needs to find a hypothesis which would optimize this tradeoff. We will get back to this type of tradeoff when we discuss the SVM learning algorithm in Lecture 4.

Back to the Perceptron, in case the training set Z is not linearly separable one can still obtain a bound on the number of mistakes made by the Perceptron as a function of the margin and number of *margin errors* made by the hypothesis. The bound on the number of mistakes would be determined by the optimal tradeoff between the two, as described below.

Let $\mathbf{u}$ be any vector in $\mathbb{R}^n$ and set γ to any positive value. For each example $(\mathbf{x}_i, y_i)$ in a finite sample Z we now define the following variable

$$d_i = \max\{0, \gamma - y_i(\mathbf{u} \cdot \mathbf{x}_i)\} \quad .$$

In words, d_i is 0 if $(\mathbf{x}_i, y_i)$ attains a margin of *at least* γ with respect to $\mathbf{u}$ and otherwise it is equal to the amount we need to add in order to obtain a virtual margin of γ. These slack amounts are illustrated in Fig. 3.3.

Given these slack variables it is possible to prove the following mistake bound.

Theorem 2 (Fruend & Schapire, 1998) *Let $Z = \{(\mathbf{x}_i, y_i)\}$ be an input sequence for the Perceptron algorithm described in Fig. 3.2 where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$. Denote by $R^2 = \max_i \|\mathbf{x}_i\|^2$. Let $\mathbf{u}$ be any vector with $\|\mathbf{u}\| = 1$ and $\gamma > 0$. Define the deviation for each example as $d_i = \max\{0, \gamma - y_i(\mathbf{u} \cdot \mathbf{x}_i)\}$ and define $D = \sqrt{\sum_{i=1}^{T} d_i^2}$. Then, the number of mistakes that the Perceptron algorithm makes is at most*

$$\left(\frac{R + D}{\gamma}\right)^2 \quad .$$

Note the interesting fact that the theorem holds for any $\mathbf{u}$. What happens if we choose an arbitrary $\mathbf{u}$ which induces a random partition? In this case the deviation variables are going to be large and the bound on the number of mistakes is likely to be larger than T and thus the bound becomes vacuous. However, if there exists a hyperplane $\mathbf{u}$ such that most of the examples do obtain a margin of at least γ then the Perceptron algorithm will make a relatively small number of mistakes. Different hyperplanes and margin values will attain different mistake bounds. Thm. 2 tacitly implies

that a good $\mathbf{u}$ should make a trade-off between the margin γ and the number of examples for which d_i is not zero. We will discuss in the next classes learning algorithms that take this trade-off into account.

Last, we would like to note that the bound of Thm. 2 reduces to the bound Perceptron's first mistake bound (Thm. 1) when the sample is separable and thus $D = 0$.