deterministic, the fact that we constructed c^* by running A is not a problem: it would make the same mistakes if re-run from scratch on the same sequence and same target. Therefore, A makes d mistakes on this σ and c^*. ∎

5.8.2 The Halving Algorithm

If we are not concerned with running time, a simple algorithm that guarantees to make at most $\log_2(|\mathcal{H}|)$ mistakes for a target belonging to any given class $\mathcal{H}$ is called the *halving algorithm.* This algorithm simply maintains the *version space* $\mathcal{V} \subseteq \mathcal{H}$ consisting of all $h \in \mathcal{H}$ consistent with the labels on every example seen so far, and predicts based on majority vote over these functions. Each mistake is guaranteed to reduce the size of the version space $\mathcal{V}$ by at least half (hence the name), thus the total number of mistakes is at most $\log_2(|\mathcal{H}|)$. Note that this can be viewed as the number of bits needed to write a function in $\mathcal{H}$ down.

5.8.3 The Perceptron Algorithm

Earlier we described the Perceptron algorithm as a method for finding a linear separator consistent with a given training set S. However, the Perceptron algorithm also operates naturally in the online setting as well.

Recall that the basic assumption of the Perceptron algorithm is that the target function can be described by a vector $\mathbf{w}^*$ such that for each positive example $\mathbf{x}$ we have $\mathbf{x}^T\mathbf{w}^* \geq 1$ and for each negative example $\mathbf{x}$ we have $\mathbf{x}^T\mathbf{w}^* \leq -1$. Recall also that we can interpret $\mathbf{x}^T\mathbf{w}^*/|\mathbf{w}^*|$ as the distance of $\mathbf{x}$ to the hyperplane $\mathbf{x}^T\mathbf{w}^* = 0$. Thus, we can view our assumption as stating that there exists a linear separator through the origin with all positive examples on one side, all negative examples on the other side, and all examples at distance at least $\gamma = 1/|\mathbf{w}^*|$ from the separator, where γ is called the margin of separation.

The guarantee of the Perceptron algorithm will be that the total number of mistakes is at most $(R/\gamma)^2$ where $R = \max_t |\mathbf{x}_t|$ over all examples $\mathbf{x}_t$ seen so far. Thus, if there exists a hyperplane through the origin that correctly separates the positive examples from the negative examples by a large margin relative to the radius of the smallest ball enclosing the data, then the total number of mistakes will be small. The algorithm, restated in the online setting, is as follows.

The Perceptron Algorithm: Start with the all-zeroes weight vector $\mathbf{w} = \mathbf{0}$. Then, for $t = 1, 2, \ldots$ do:

1. Given example $\mathbf{x}_t$, predict $\mathrm{sgn}(\mathbf{x}_t^T\mathbf{w})$.
2. If the prediction was a mistake, then update:
   (a) If $\mathbf{x}_t$ was a positive example, let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$.
   (b) If $\mathbf{x}_t$ was a negative example, let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$.

The Perceptron algorithm enjoys the following guarantee on its total number of mistakes.

Theorem 5.10 *On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, \ldots$, if there exists a vector $\mathbf{w}^*$ such that $\mathbf{x}_t^T\mathbf{w}^* \geq 1$ for the positive examples and $\mathbf{x}_t^T\mathbf{w}^* \leq -1$ for the negative examples (i.e., a linear separator of margin $\gamma = 1/|\mathbf{w}^*|$), then the Perceptron algorithm makes at most $R^2|\mathbf{w}^*|^2$ mistakes, where $R = \max_t |\mathbf{x}_t|$.*

Proof: Fix some consistent $\mathbf{w}^*$. We will keep track of two quantities, $\mathbf{w}^T\mathbf{w}^*$ and $|\mathbf{w}|^2$. First of all, each time we make a mistake, $\mathbf{w}^T\mathbf{w}^*$ increases by at least 1. That is because if $\mathbf{x}_t$ is a positive example, then

$$(\mathbf{w} + \mathbf{x}_t)^T\mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* + \mathbf{x}_t^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1,$$

by definition of $\mathbf{w}^*$. Similarly, if $\mathbf{x}_t$ is a negative example, then

$$(\mathbf{w} - \mathbf{x}_t)^T\mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* - \mathbf{x}_t^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1.$$

Next, on each mistake, we claim that $|\mathbf{w}|^2$ increases by at most R^2. Let us first consider mistakes on positive examples. If we make a mistake on a positive example $\mathbf{x}_t$ then we have

$$(\mathbf{w} + \mathbf{x}_t)^T(\mathbf{w} + \mathbf{x}_t) = |\mathbf{w}|^2 + 2\mathbf{x}_t^T\mathbf{w} + |\mathbf{x}_t|^2 \leq |\mathbf{w}|^2 + |\mathbf{x}_t|^2 \leq |\mathbf{w}|^2 + R^2,$$

where the middle inequality comes from the fact that we made a mistake, which means that $\mathbf{x}_t^T\mathbf{w} \leq 0$. Similarly, if we make a mistake on a negative example $\mathbf{x}_t$ then we have

$$(\mathbf{w} - \mathbf{x}_t)^T(\mathbf{w} - \mathbf{x}_t) = |\mathbf{w}|^2 - 2\mathbf{x}_t^T\mathbf{w} + |\mathbf{x}_t|^2 \leq |\mathbf{w}|^2 + |\mathbf{x}_t|^2 \leq |\mathbf{w}|^2 + R^2.$$

Note that it is important here that we only update on a mistake.

So, if we make M mistakes, then $\mathbf{w}^T\mathbf{w}^* \geq M$, and $|\mathbf{w}|^2 \leq MR^2$, or equivalently, $|\mathbf{w}| \leq R\sqrt{M}$. Finally, we use the fact that $\mathbf{w}^T\mathbf{w}^*/|\mathbf{w}^*| \leq |\mathbf{w}|$ which is just saying that the projection of $\mathbf{w}$ in the direction of $\mathbf{w}^*$ cannot be larger than the length of $\mathbf{w}$. This gives us:

$$\begin{aligned} M/|\mathbf{w}^*| &\leq R\sqrt{M} \\ \sqrt{M} &\leq R|\mathbf{w}^*| \\ M &\leq R^2|\mathbf{w}^*|^2 \end{aligned}$$

as desired. ■

5.8.4 Extensions: Inseparable Data and Hinge Loss

We assumed above that there exists a perfect $\mathbf{w}^*$ that correctly classifies all the examples, e.g., correctly classifies all the emails into important versus non-important. This is rarely the case in real-life data. What if even the best $\mathbf{w}^*$ isn't quite perfect? We can see what this does to the above proof: if there is an example that $\mathbf{w}^*$ doesn't correctly classify, then while the second part of the proof still holds, the first part (the dot product of $\mathbf{w}$ with $\mathbf{w}^*$ increasing) breaks down. However, if this doesn't happen too often, and also $\mathbf{x}_t^T\mathbf{w}^*$ is just a "little bit wrong" then we will only make a few more mistakes.

To make this formal, define the *hinge-loss* of $\mathbf{w}^*$ on a positive example $\mathbf{x}_t$ as $\max(0, 1 - \mathbf{x}_t^T\mathbf{w}^*)$. In other words, if $\mathbf{x}_t^T\mathbf{w}^* \geq 1$ as desired then the hinge-loss is zero; else, the hinge-loss is the amount the LHS is less than the RHS.[21] Similarly, the hinge-loss of $\mathbf{w}^*$ on a negative example $\mathbf{x}_t$ is $\max(0, 1 + \mathbf{x}_t^T\mathbf{w}^*)$. Given a sequence of labeled examples S, define the total hinge-loss $L_{hinge}(\mathbf{w}^*, S)$ as the sum of hinge-losses of $\mathbf{w}^*$ on all examples in S. We now get the following extended theorem.

Theorem 5.11 *On any sequence of examples $S = \mathbf{x}_1, \mathbf{x}_2, \ldots$, the Perceptron algorithm makes at most*

$$\min_{\mathbf{w}^*}\left(R^2|\mathbf{w}^*|^2 + 2L_{hinge}(\mathbf{w}^*, S)\right)$$

mistakes, where $R = \max_t |\mathbf{x}_t|$.

Proof: As before, each update of the Perceptron algorithm increases $|\mathbf{w}|^2$ by at most R^2, so if the algorithm makes M mistakes, we have $|\mathbf{w}|^2 \leq MR^2$.

What we can no longer say is that each update of the algorithm increases $\mathbf{w}^T\mathbf{w}^*$ by at least 1. Instead, on a positive example we are "increasing" $\mathbf{w}^T\mathbf{w}^*$ by $\mathbf{x}_t^T\mathbf{w}^*$ (it could be negative), which is at least $1 - L_{hinge}(\mathbf{w}^*, \mathbf{x}_t)$. Similarly, on a negative example we "increase" $\mathbf{w}^T\mathbf{w}^*$ by $-\mathbf{x}_t^T\mathbf{w}^*$, which is also at least $1 - L_{hinge}(\mathbf{w}^*, \mathbf{x}_t)$. If we sum this up over all mistakes, we get that at the end we have $\mathbf{w}^T\mathbf{w}^* \geq M - L_{hinge}(\mathbf{w}^*, S)$, where we are using here the fact that hinge-loss is never negative so summing over all of S is only larger than summing over the mistakes that $\mathbf{w}$ made.

[21] This is called "hinge-loss" because as a function of $\mathbf{x}_t^T\mathbf{w}^*$ it looks like a hinge.

Finally, we just do some algebra. Let $L = L_{hinge}(\mathbf{w}^*, S)$. So we have:

$$\begin{array}{rcl}
\mathbf{w}^T\mathbf{w}^*/|\mathbf{w}^*| & \leq & |\mathbf{w}| \\
(\mathbf{w}^T\mathbf{w}^*)^2 & \leq & |\mathbf{w}|^2|\mathbf{w}^*|^2 \\
(M-L)^2 & \leq & MR^2|\mathbf{w}^*|^2 \\
M^2 - 2ML + L^2 & \leq & MR^2|\mathbf{w}^*|^2 \\
M - 2L + L^2/M & \leq & R^2|\mathbf{w}^*|^2 \\
M & \leq & R^2|\mathbf{w}^*|^2 + 2L - L^2/M \ \leq \ R^2|\mathbf{w}^*|^2 + 2L
\end{array}$$

as desired. ■

5.9 Online to Batch Conversion

Suppose we have an online algorithm with a good mistake bound, such as the Perceptron algorithm. Can we use it to get a guarantee in the distributional (batch) learning setting? Intuitively, the answer should be yes since the online setting is only harder. Indeed, this intuition is correct. We present here two natural approaches for such online to batch conversion.

Conversion procedure 1: Random Stopping. Suppose we have an online algorithm $\mathcal{A}$ with mistake-bound M. Say we run the algorithm in a single pass on a sample S of size M/ϵ. Let X_t be the indicator random variable for the event that $\mathcal{A}$ makes a mistake on example $\mathbf{x}_t$. Since $\sum_{t=1}^{|S|} X_t \leq M$ for *any* set S, we certainly have that $\mathbf{E}[\sum_{t=1}^{|S|} X_t] \leq M$ where the expectation is taken over the random draw of S from $\mathcal{D}^{|S|}$. By linearity of expectation, and dividing both sides by $|S|$ we therefore have:

$$\frac{1}{|S|}\sum_{t=1}^{|S|}\mathbf{E}[X_t] \ \leq \ M/|S| = \epsilon. \tag{5.1}$$

Let h_t denote the hypothesis used by algorithm $\mathcal{A}$ to predict on the tth example. Since the tth example was randomly drawn from $\mathcal{D}$, we have $\mathbf{E}[err_{\mathcal{D}}(h_t)] = \mathbf{E}[X_t]$. This means that if we choose t at random from 1 to $|S|$, i.e., stop the algorithm at a random time, the expected error of the resulting prediction rule, taken over the randomness in the draw of S and the choice of t, is at most ϵ as given by equation (5.1). Thus we have:

Theorem 5.12 (Online to Batch via Random Stopping) *If an online algorithm $\mathcal{A}$ with mistake-bound M is run on a sample S of size M/ϵ and stopped at a random time between 1 and $|S|$, the expected error of the hypothesis h produced satisfies $\mathbf{E}[err_{\mathcal{D}}(h)] \leq \epsilon$.*

Conversion procedure 2: Controlled Testing. A second natural approach to using an online learning algorithm $\mathcal{A}$ in the distributional setting is to just run a series of controlled tests. Specifically, suppose that the initial hypothesis produced by algorithm $\mathcal{A}$ is h_1. Define $\delta_i = \delta/(i+2)^2$ so we have $\sum_{i=0}^{\infty}\delta_i = (\frac{\pi^2}{6} - 1)\delta \leq \delta$. We draw a set of