1 From Statistical Efficiency To Algorithmic Efficiency

Last week we only addressed statistical learning problems: given a state space $X$, target space $Y = \{0, 1\}$ and class of mapping $H = \{h : X \to Y\}$, how approximately how many samples do we need to learn an $\epsilon$-accurate concept with probability $1 - \delta$.

In this lecture, we redirect our attention to computationally efficient learning algorithms. In some instances, we can come up with polynomial algorithms by reduction to schema like linear programming, and as we will demonstrate, there are certain conditions under which we can develop linear or even sublinear time algorithms to solve learning problems.

2 Learning Linear Classifiers

In this lecture, we return to the problem of learning separating hyperplanes.

Definition 2.1 (Separating Hyperplane). The Separating Hyperplane problem is defined by the sample and target spaces:

$$X = \mathbb{R}^d \quad \text{and} \quad Y = \{-1, 1\}$$

We choose the hypothesis class

$$H = \{h_{a,b} : a \in \mathbb{R}^d, b \in \mathbb{R}\} \quad \text{where} \quad h_{a,b}(x) := \text{sign}(a^T x - b)$$

Given a data set $S = \{(x_i, y_i)\}$, we say the data are linearly separable if there exists an $h \in H$ such that $h(x_i) = y_i$ for all $i$.

Remark 2.2. It is possible that $a^T x - b = 0$, in which case we assume the sign is 1. Note that if the data are drawn from a continuous distribution, the $h_{a,b}$ returns labels in $Y = \{-1, 1\}$ almost surely.

Remark 2.3 (Identifiability). Not that, for any positive constant $c > 0$, the hypothesis $h_{a,c,b,c}(x) = h_{a,b}(x)$, so the concepts are really only identifiable up to constants. Thus, we may assume that $\|a, b\| = 1$, where $[\cdot, \cdot]$ denotes concatenation (this is excluding the degenerate hyperplane $a = b = 0$). It is easy to show that, as long as $[a, b] \neq c[a', b']$ for any $c > 0$, then there exists an $x$ such that $h_{a,b}(x) \neq h_{a',b'}(x)$, so the hypothesis in $H$ are identifiable precisely up to positive constants.

$H$ is frequently chosen not only because it is simple, but because in practice it seems to correspond well to observed data. For example, spam filters rely on learning separating hyperplanes. Moreover, it is a simple space: it has VC dimension $d + 1$.

The following exercise will be given as homework.

Exercise 2.4 (VC Dimension of Hyperplanes). Prove that the VC dimension of $H$ is $d + 1$.

And applying of the Fundamental Theorem of Statistical Machine Learning yields the following proposition, which we state loosely:
Proposition 2.5. In the PAC learning setting, we can learn a concept \( h \in \mathcal{H} \) with sample complexity

\[
m(\epsilon, \delta) = O\left( \frac{d}{\epsilon} \log(1/\delta) \right)
\]

In the agnostic setting, we require

\[
m(\epsilon, \delta) = O\left( \frac{d}{\epsilon^2} \log \frac{1}{\epsilon \delta} \right).
\]

Example 2.6 (Detecting Spam Email). Suppose we are trying to separate spam from meaningful emails using by learning a separating hyperplane. If view emails as bags of words - or vectors of word frequencies - then we need to learn a separating hyperplane in \( \mathbb{R}^{|V|} \), where \( V \) is the vocabulary of words that might appear. Obviously, it is too naive to assume that spam and non-spam emails are not perfectly separable by a hyperplane. Even still, we can appeal to the agnostic case of the preceding proposition to conclude that, to get an \( \epsilon \) error compared to the optimal linear classifier with probability \( 1 - \delta \), we only need around \( |V| \epsilon^2 \log(1/\delta) \) samples. Thus, the number of emails we need scales linearly with vocabulary size.

2.1 Learning Separating Hyperplanes Algorithmically

Let’s now tackle the algorithmic considerations in learning a separating hyperplane. First, we consider the PAC learning setting. In what follows, the input is

\[
S = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}\}
\]

are our goal is compute a vector \( w \in \mathbb{R}^d \) which is consistent with the data. That is

\[
w : \text{sign} (w^T x_i) = y_i \quad \forall i
\]

You might ask why we dropped the intercept term. This is for simplicity, and the follows remark explains why there is no loss of generality in doing so.

Remark 2.7. Note that can write \( a^T x - b = [a, b]^T [x, 1] \) where \([\cdot, \cdot]\) is the concatenation of vectors. Hence, by padding \( x \) with a 1 at the end, we may assume without loss of generality that all hyperplanes pass through the origin.

Question 2.1. How (computationally) difficult is it to compute a consistent \( w \)?

To answer this we need to take a quick detour into Linear Programming. We will not review the basic details of the theory, only the definition and main result thereof:

Definition 2.8 (Linear Program or LP). A linear program is any objective that can be written in the following form:

\[
\text{maximize } c^T x \quad \text{over } x \in \mathbb{R}^n \quad \text{(7)}
\]

subject to \( Ax \leq b \quad (8)\)

where \( c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n} \) and \( b \in \mathbb{R}^m \) are known coefficients, \( x \) is the variable over which the objective is being maximized, and here \( \leq \) denotes an entrywise inequality. A linear program is feasible if there exists some \( x \) satisfying the constraints and infeasible otherwise.

Linear Programming (LP) is a fundamental algorithmic scheme in computer science, but we will not delve deeper into the theory. Instead, we will simply appeal to the following theorem:
Theorem 2.9. There exists an algorithm to compute the optimal value of a linear program which runs polynomially in the inputs. In particular, given a feasible system $Ax \leq b$ of linear inequalities, there exists a polynomial time algorithm to return a feasible $x$ satisfying those inequalities.

Because of this theorem, we can exhibit a polynomial time algorithm by reducing the problem of learning a consistent hyperplane to an LP.

Theorem 2.10. There exists a linear program equivalent to finding a separating hyperplane consistent with the data.

Proof. Recall that the objective is to find a $w$ such that $\text{sign}(w^T x_i) = y_i$. Equivalently, we want $(w^T x_i)y_i \geq 0$. That is, we want to solve $Aw \geq 0$, where $A \in \mathbb{R}^{|S| \times d}$ has rows $A_i = x_i y_i$, and $0 \in \mathbb{R}^{|S|}$ is the zero vector. This is solvable in polynomial time by Theorem 2.9.

Corollary 2.11. In the PAC setting, a consistent hyperplane can be found in polynomial run time.

2.2 Learning Separating Hyperplanes Aagnostically

The next question to ask is: what is the hyperplane hypothesis does not fit the data exactly? What if there are sources of noise or error? Stated otherwise, can we agnostically learn the best hyperplane (the one that is consistent with the most data points) in polynomial time? Here the ERM algorithm on a set of samples

$S = \{(x_i, y_i)\}$

returns the $w$ which makes the fewest mistakes. That is,

$$w \in \arg \min_{(x_i, y_i)} \sum I(\text{sign}(w^T x_i) \neq y_i)$$

(10)

Statistical learning theory is misleadingly optimistic: recall that it would suggest that the ERM algorithm only needs $O\left(\frac{d \log(1/\delta)}{\epsilon^2}\right)$. However, the algorithmic prospects are decidedly bleaker: computing $w$ in Equation [10] is NP hard, and hence cannot be made efficient unless $P = NP$. In fact, there is even a stronger theorem limiting the hopes of an agnostic hyperplane learning algorithm:

Theorem 2.12 ([1]). The problem of finding a $w$ such that $\text{err}_S(w) < 1/2 - \epsilon$ for any fixed $\epsilon > 0$ is NP hard in general.

Remark 2.13. Feldman et. al.’s approximation theory is tight, in the sense that an error $1/2$ algorithm is trivial: pick $w : \|w\| \leq 1$ uniformly at random from the $d - 1$ sphere. Then

$$Pr \left( \text{sign}(w^T x) = 1 \right) = Pr \left( \text{sign}(w^T x) = -1 \right) = \frac{1}{2}$$

(11)

Instead of solving the NP hard problem, we can attempt to solve a “Convex Relaxation” of the problem. For example, instead of minimizing the real errors, you can minimize the sum of real errors. For example, the well-known SVM minimizes the sum of the distances of misclassified examples from the hyperplane (there are many ways to do this).

But, even the PAC learning case is not too promising. As suggested above, one can think of an LP as systems of “inequalities”. Whereas systems of equalities are relatively easy to solve using Gaussian elimination, it was only until 1979 that the preceding theorem was proven. Moreover, even though linear programs are polynomial time algorithms, they run super-linearly in the number of arguments, and are hence often impractical to solve in large-data applications.
2.3 Practically Efficient Algorithms for the Separating Hyperplane Problem

Given the computational difficulties associated with learning separating hyperplanes in general, we will describe a very simple algorithm that runs efficiently under a nice simplifying assumption. The algorithm is known as the “Perceptron”, and dates back to the earliest research into artificial intelligence conducted during the 1950’s. In fact, the “Perceptron” is the neural network, a model of the activation of a single neuron.

Like the algorithms we’ve examined thus far, the Perceptron is learns a separating hyperplane. Here the input is a data set

\[ S = \{(x_i, y_i)\} \in (\mathbb{R}^d \times \{-1, 1\})^n \]  

The algorithm is as follows:

1. Initialize \( w_0 = 0 \)
2. At each step \( t \), find a sample \((x_{it}, y_{it})\) such that \( \text{sign}(w_t^T x_{it}) \neq y_{it} \). In practice, we often chose \( x_{it} = \arg \min( w_t^T x_i y_i ) \).
3. Set \( w_{t+1} \leftarrow w_t + x_{it} \text{sign}(y_{it}) \).

One can show that, according to some relaxed version of error, the Perceptron is the best classifier of order \( \epsilon \). In this lecture, we show that the Perceptron behaves very well under a more limiting a assumption: when the data have a moderate “margin”.

**Definition 2.14** (Margin). Let \( w \) be a hyperplane \( \|w\| = 1 \). We define the margin of a data point to \( w \) by \( (w^T x_i) y_i \), which is the sign-corrected distance of a point to \( w \). We define margin of the data to \( w \) by \( \min_i (w^T x_i y_i) \). Finally, we define margin of the data as the distance of the closest point to the largest margin hyperplane on the data, and this is denoted by

\[ \sigma := \max_{\|w\| = 1} \min_i (w^T x_i) y_i \]  

**Theorem 2.15** (Novikoff, 1962). Suppose that all \( \|x_i\| = 1 \), and that the data are linearly separable with margin \( \sigma > 0 \). Then the Perceptron algorithm returns a \( w \) such that \( \text{sign}(w^T x_i) = y_i \) after \( \frac{1}{\sigma^2} \) iterations. Moreover, if we modify Step 2 of the algorithm to choose a point of distance no more than \( \sigma \) to \( w_t \), then the Perceptron returns a hyperplane \( w \) which separates the data with margin \( \sigma \) in no more than \( \frac{2}{\sigma^2} \) iterations.

**Remark 2.16.** The stipulation \( \|x_i\| = 1 \) is for simplicity and can be imposed without loss of generality by normalizing the data points and replacing \( \sigma \) by \( \frac{\sigma}{\max \|x_i\|} \).

**Proof.** The following proof is due to Novikoff. For simplicity and without loss of generality, we assume that \( y_i = 1 \) always because we can flip the sign of the \( x_i \). We show that in \( 1/\sigma^2 \) iteration, we return a hyperplane that correctly classifiers our data. Suppose that, at step \( t + 1 \), we have still not classified all the data correctly. Then the following

**Claim 2.17.** \( \|w_t\|^2 \leq t \)

**Proof.** Since \( x_i \) is misclassified by \( w_t \), \( w_t \cdot x_i < 0 \). Hence,

\[ \|w_{t+1}\|^2 = \|w_t + x_i^{(t)}\|^2 = \|w_t\|^2 + 2w_t \cdot x_i + \|x_i\|^2 \leq \|w_t\|^2 + 1 \leq t + 1 \]  

**Claim 2.18.** \( w_t \cdot w^* \geq \sigma t \)
Proof. For all $x_i$, $x_i \cdot w^* \geq w_t$. Hence

$$w_{t+1} \cdot w^* = (w_t + x_i) \cdot w^* \geq w_t \cdot w^* + \sigma \geq \sigma (t + 1)$$

With these two claims, we conclude

$$\sigma t \leq w_t \cdot w^* \leq \|w_t\| \leq \sqrt{t} \tag{15}$$

Thus, we must have $t \leq \frac{1}{\sigma^2}$.

Remark 2.19. Note that we don’t need the practical heuristic in Step 2 which chooses $x_i, y_i = \arg \min_i w^T x_i y_i \tag{16}$

However, in a relaxed sense (not addressed in these notes), updating with the minimum margin data point can be shown to satisfy stronger performance bounds.

Now let’s compute the runtime of the Perceptron algorithm.

**Proposition 2.20.** Suppose the data $x_i \in \mathbb{R}^d$ are normalized, and that the margin is $\sigma > 0$. Finally, suppose we have $\Theta(d/\epsilon^2)$ samples (motivated by the Agnostic Learning setting). Then, the Perceptron runs in $O \left( \frac{d^2}{\sigma^2 \epsilon^2} \right)$. In particular, for fixed $\sigma$, the Perceptron Algorithm is linear in the size of the data and quadratic in dimension.

**Proof.** Implemented naively, each iteration requires a pass through $O(d/\epsilon^2)$ data points, performing vector operations which take $O(d)$ time. This yields a runtime of $\Theta(d^2/\epsilon^2)$ per iteration. As there are only $O \left( \frac{1}{\sigma^2} \right)$, we recover the desired bound. Note that the data consists of $\Theta(d/\epsilon^2)$ vectors of length $d$. Then our data takes up $\Theta(d^2/\epsilon^2)$ space, and hence our algorithm runs linearly in the data size.

In fact, a more efficient, randomized algorithm exists, called the “sublinear perceptron”, for which:

**Theorem 2.21.** Suppose the data $x_i \in \mathbb{R}^d$ are normalized, and that the margin is $\sigma > 0$. Then, there exists a randomized algorithm which computes a separating hyperplane $w$ from $\Theta(d/\epsilon^2)$ samples in $O \left( \frac{d}{\epsilon \sigma^2} \right)$ time with high probability. In particular, this algorithm runs sublinearly in the input size.

Details can be found in [3].

3 Conclusion

To conclude today’s class, in the first part of our lecture we proved a Statistical Theorem which gives a strong characterization of what is learnable in terms of VC dimension. In the second half, we scratched the surface of computationally efficient algorithms for learning, and saw that computational barrier are often much more restrictive than statistical ones. However, we saw that under helpful assumptions like the large margin hypothesis, we can develop computationally efficient algorithms to compute provably good hypotheses in polynomial, linear, and even possibly sublinear time.
References

