1 Perceptron — full analysis

Recall the perceptron problem from last time.

**Prediction task.** We must find a vector \( w \in \mathbb{R}^d \) so that \( \hat{y} := \text{sgn}(\langle w, x \rangle) \) matches \( y \) chosen by nature, where \( \|x\| \leq 1 \) and \( y \in \{-1, +1\} \).

**Performance criterion.** \( 1[\hat{y} \neq y] \).

**Coherence.** In this problem we are considering a curious setup where past and future learning are not distinct; instead, vectors \( x \in \mathbb{R}^d \) just keep coming, along with correct labels \( y \in \{-1, +1\} \), and we can learn forever if we wish.

Coherence will be provided in the following interesting way. There will be a fixed vector \( u \in \mathbb{R}^d \) and scalar \( \gamma > 0 \) (“fixed” means: fixed across all time) so that every pair \( (x, y) \) we receive satisfies \( y = \text{sgn}(\langle w, x \rangle) \) and \( |\langle u, x \rangle| \geq \gamma \). More simply, these two cases can be combined into one: every pair \( (x, y) \) satisfies \( \langle w, xy \rangle \geq \gamma \). [ In class: a picture was drawn. ]

This problem is tractable because we don’t need to exactly learn \( u \); \( u \) has wiggle room [ added to picture ], which means that for any vector \( v \in \mathbb{R}^d \) and any \( (x, y) \) with \( \|x\| \leq 1, y \in \{-1, +1\} \),

\[
\langle v/\|v\|, xy \rangle = \langle v/\|v\| - u, xy \rangle + \langle u, xy \rangle \geq -\|v/\|v\| - u\| + \gamma,
\]

consequently

\[
\|v/\|v\| - u\| < \gamma \implies \text{sgn}(\langle v, x \rangle) = \text{sgn}(\langle u, x \rangle) = y.
\]

To learn \( w \), we will use Rosenblatt’s **Perceptron algorithm** [Rosenblatt, 1958].

1. Initialize \( w_0 := 0 \in \mathbb{R}^d \).
2. For \( i = 1, 2, \ldots \):
   a. Receive \( x_i \in \mathbb{R}^d \) with \( \|x_i\| \leq 1 \).
   b. Predict \( \hat{y}_i := \text{sgn}(\langle w_{i-1}, x_i \rangle) \).
   c. Receive \( y_i := \text{sgn}(\langle u, x_i \rangle) \).
   d. Update \( w_i := w_{i-1} + 1[y_i \neq \hat{y}_i]y_i x_i \).

The intuition behind this method is clear: with each mistake (i.e., when \( \hat{y}_i \neq y_i \)), we rotate our vector in the direction of correct predictions!

**Theorem 1.1** [Novikoff, 1962]. Let \( \hat{y}_i, y_i, \) and \( \gamma \) be defined as above in the Perceptron algorithm. Then

\[
\sum_{i \geq 1} 1[y_i \neq \hat{y}_i] \leq \frac{1}{\gamma^2}.
\]
Remark 1.2. The proof will use the “wiggle room” potential function
\[ \left\| \frac{v}{\| v \|} - u \right\|^2, \]
however it will not show that \( v/\| v \| \to u \)!
For a counterexample, suppose \( u = (1,1)^T/\sqrt{2} \), and \( x_i = (1,0)^T \) forever. (This example is affected by the behavior at 0 of sgn, but negating \( u \) and \( x_i \) avoids this and still illustrates the point.)

Remark 1.3. It may seem astonishing that Theorem 1.1 holds for all time with a constant number of mistakes. One intuition is that, due to \( \gamma \), \( u \) can hide within the unit ball in only finitely many places; the unit ball has been discretized. If this sounds confusing, we’ll make it more clear when treating margin bounds in the generalization part of the course.

Proof. Let positive integer \( t \) be given, and define mistake set \( M_t := \{ i \leq t : y_i \neq \hat{y}_i \} \). Momentarily we’ll show
\[ |M_t| \leq \frac{1}{\gamma^2}. \]
This in turn proves the claim since \( |M_t| \) is nondecreasing, whereby \( \lim_{t \to \infty} |M_t| \) exists, and thus
\[ \sum_{i \geq 1} 1[\hat{y}_i \neq y_i] = \lim_{t \to \infty} |M_t| \leq \frac{1}{\gamma^2}. \]

To prove \( |M_t| \leq 1/\gamma^2 \), let’s go back to our wiggle room potential function:
\[ 0 \leq \left\| \frac{w_i}{\| w_i \|} - u \right\|^2 = 2 \left( 1 - \frac{\langle w_i, u \rangle}{\| w_i \|} \right). \]
Let us consider this final numerator and denominator.
For the numerator,
\[ \langle w_i, u \rangle = \left( \sum_{i \in M_t} x_i y_i, u \right) = \sum_{i \in M_t} \langle x_i y_i, u \rangle \geq |M_t| \gamma. \]
For the denominator, for any \( i \leq t \),
\[ \| w_i \|^2 = \| w_{i-1} + 1[i \in M_t] x_i y_i \|^2 \]
\[ = \| w_{i-1} \|^2 + 2 \| w_{i-1} \| \langle w_{i-1}, x_i y_i \rangle + 1[i \in M_t]^2 \| x_i y_i \|^2 \]
\[ \leq \| w_{i-1} \|^2 + 0 + 1[i \in M_t], \]
thus, by induction,
\[ \| w_t \|^2 \leq \| w_0 \|^2 + \sum_{i \leq t} 1[i \in M_t] = |M_t|. \]
Putting these pieces together with the wiggle room potential,
\[ 0 \leq 2(1 - \frac{\langle w_i, u \rangle}{\| w_i \|}) \leq 2 \left( 1 - \frac{\gamma |M_t|}{\sqrt{|M_t|}} \right) = 2 \left( 1 - \gamma \sqrt{|M_t|} \right), \]
which implies \( |M_t| \leq 1/\gamma^2 \)

Before closing, note that the funny merged training/testing setting of this problem is standard in the learning theory community.
Remark 1.4. The perceptron algorithm here is in the **online learning setting**: every round, nature reveals an $x$, the algorithm makes a prediction $\hat{y}$, then the algorithm suffers a loss (in the perceptron case $1[\hat{y} \neq y]$) and nature reveals something the algorithm can use to improve (in this case $y$). The online learning setting allows the sequence of $(x, y)$ choices to depend on the actions of the algorithm, and thus the setting is also sometimes called **adversarial**.

Note that in the perceptron case, we can not convert the bound into a statement about the final predictor’s performance on the future relating to its performance on the past. That is to say, the number of mistakes on $(x_i, y_i)$ committed by $w_t$ may differ from $|M_t|$. To see this, suppose $u = (1, 0)$, $x_1 := x_2 := (1/4, -1/2)$, and $x_i = (1/4, 1/2)$ for $i \geq 3$; then $w_1 = x_2$ and $\langle w_1, x_1 \rangle < 0$, thus $w_1$ makes 2 mistakes on the past, whereas $|M_1| = 1$.

Note that if the loss were convex, than the averaged iterate would be guaranteed to perform at least as well as the average over the past (if this comment makes no sense now, it’ll make sense after we cover convexity!).

2 Decomposing learning problems into representation, optimization, generalization

In order to split learning problems into **representation**, **optimization**, and **generalization**, we’ll use the following notation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{R}$</td>
<td>Performance measure on the future; the <em>risk</em>.</td>
</tr>
<tr>
<td>$\hat{\mathcal{R}}$</td>
<td>Performance measure on the past; the <em>empirical risk</em>.</td>
</tr>
<tr>
<td>$F$</td>
<td>The function class we will search over.</td>
</tr>
<tr>
<td>$f \in F$</td>
<td>Predictor output by our algorithm.</td>
</tr>
<tr>
<td>$f \in F$</td>
<td>Benchmark predictor within $F$ we compete with.</td>
</tr>
<tr>
<td>$\bar{f}$</td>
<td>Benchmark predictor, <em>not necessarily</em> in $F$.</td>
</tr>
</tbody>
</table>

Remark 2.1. The appearance of the function class $F$ may come as a surprise; after all, it did not enter into our discussion of the two aspects of formalizing a machine learning problem, namely coherence and performance criteria. This will be explained momentarily in Remark 2.2. Note that in the perceptron example, we did in fact have a constrained set of functions, namely the linear predictors $\{x \mapsto \text{sgn}(\langle w, x \rangle) : w \in \mathbb{R}^d\}$. 

With this notation, we would like to ensure that

$$\mathcal{R}(\hat{f}) - \mathcal{R}(\bar{g})$$

is small. But we can’t measure $\mathcal{R}(\hat{f})$, let alone $\mathcal{R}(\bar{g})$! However, we can measure $\hat{\mathcal{R}}$, thus note the decomposition

$$\mathcal{R}(\hat{f}) - \mathcal{R}(\bar{g}) = \hat{\mathcal{R}}(\hat{f}) - \hat{\mathcal{R}}(\bar{g})$$

These terms are controlled as follows.

**Representation.** eq. $[\star]$ is *representation*, part 1 of the course. The current plan includes: linear predictors, linear predictors over rich bases, RKHSes, decision trees, branching programs, 3-layer and 2-layer networks, deep networks, representing polynomials and smooth functions with deep networks, probability modeling and GANs.

**Optimization.** eq. $[\bullet]$ denotes *optimization*, and is part 2 of the course. (I’m still planning it.)
Deviations. To analyze eq. (□), we will use the formalism of statistical learning theory. We will assume that \( \hat{R} \) is measured on a training set, where \( R \) is the expectation of the error over a new random example. In this setting, eq. (□) denotes uniform deviations over \( \mathcal{F} \), whereas eq. (♦) denotes deviations of \( \bar{f} \). Controlling this term is part 3 of the course. (I’m still planning it.)

Remark 2.2. Let’s come back to the comment: why not make \( \mathcal{F} \) huge? Roughly: as \( \mathcal{F} \) becomes larger, while eq. (⋆) shrinks, eq. (□) grows, and eq. (△) becomes more difficult. In reality, the interplay between these three is much more intricate. For instance, if \( R(\bar{f}) \approx 0 \), then eq. (△) becomes easier and eq. (□) becomes more efficient!

3 Representation overview

The goal of representation is to choose \( \mathcal{F} \) so that \( \inf_{f \in \mathcal{F}} R(f) \) is as small as possible; in other words, something of the form

\[
\inf_{f \in \mathcal{F}} R(f) \approx \inf \{ R(g) : g \in \text{“all possible functions”} \}.
\]

Warning: often it is not possible to minimize over all functions! (E.g., in measure-theoretic terms it is often not well-defined.)

How can we possibly guarantee \( R(f) \) is small?

- Ideally, we’d have prior information which informs our choices: we can actually search over a small set. The holy grail small set is “problems arising in nature” or even “problems we know we will run into”. Magically neural nets seem to hit a pretty good sweet spot here.

- As a fallback, we can make sure we work with function classes that can approximate continuous functions arbitrarily well. As a concrete example, in binary classification in the statistical setting, we often work with

\[
R(f) = \int \ell_{\log}(y f(x)) \, d \mu(x)
\]

where \( \ell_{\log}(z) = \ln(1 + e^z) \), the logistic loss.

Since \( \ell_{\log} \) is 1-Lipschitz, for any other \( g \),

\[
R(f) = \int \ell_{\log}(-y f(x)) \, d \mu(x)
= \int \left( \ell_{\log}(-y g(x)) + \ell_{\log}(-y f(x)) - \ell_{\log}(-y g(x)) \right) \, d \mu(x)
\]

\[
\leq R(g) + \int |f(x) - g(x)| \, d \mu(x);
\]

in other words, it suffices to ensure the last term is small. Bounds of this form will be common here.

- Approximating arbitrary continuous functions will require massive function classes, and will be pessimistic in general (we don’t actually need to compete against all continuous functions). So we will also provide a few very specialized cases where functions we work with model something very nice.

[ At this point I gave an overview of the representation lectures. See the schedule and also the next lecture notes for a more detailed version. ]

[ To close the lecture I gave a construction, due to Minsky-Papert, establishing that the representation question matters: specifically, I gave a function fitting example which has an exact solution, but searching over linear predictors incurs error at least 1/4. I’ll recap it next time and include it in those notes. ]
References
