6.883: Online Methods in Machine Learning

Alexander Rakhlin

LECTURE 2

1. LINEAR CLASSIFICATION

1.1 Perceptron

Some of the oldest algorithms in machine learning are, in fact, online. In addition to the bit prediction method of the last lecture, let us discuss Perceptron (1957). Kernel methods (for both classification and regression) were introduced by Aizerman, Braverman, and Rozenoer [ABR64, ABR70] specifically for online updates, right after Novikoff's proof (1962) of Perceptron's finite-step convergence.

Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$ be labeled training data. We say data are linearly *separable* (or, *realizable*) if there exists a hyperplane that correctly classifies all the examples:

$$\forall t \in [n], \quad y_t \langle w^*, x_t \rangle > 0. \tag{1}$$

While we might not be able to identify w^* in general, we search for some $\widehat{w}$ that also separates the two classes. This is a linear program (why?) and can be solved by a variety of known techniques.

The objective we wish to optimize can be written as

$$f(w) = \sum_{t=1}^{n} \mathbf{1}\{y_t \langle w, x_t \rangle \geq 0\} \tag{2}$$

which is the cumulative fit to data, or the number of mistakes on the data made by the half-space. We will call this a **batch objective**, to distinguish from an online objective, to be covered later. By our assumption, $f(w^*) = 0$.

By appending an extra coordinate to all the vectors, we may also cover the non-homogenous hyperplanes. This is immediate.

By scaling w^* we obtain many solutions to (1). To fix this problem, we have two options. One is to fix w^* to be *unit norm* and define γ to be the largest amount (called *margin*) by which (1) are above zero; or, conversely, posit that

$$\forall t \in [n], \quad y_t \langle w^*, x_t \rangle \geq 1 \tag{3}$$

where w^* is the smallest-norm vector that guarantees (3). We will opt for the second version. One may always move between the two assumptions on the objective by rescaling the problem by $\gamma = 1/\|w^*\|$.

The linear program can be solved in various ways, but one that is of particular interest is the Perceptron recursive update, given below.

Perceptron converges to a solution in a finite number of steps (which depends on $\|w^*\|$ and the largest $\|x_t\|$). This is not really all that surprising if you know convex optimization: since data are separated by a margin, we only need to get a solution of constant accuracy (or, $O(\gamma)$ if we rescale everything; $O(1/\gamma^2)$ is what you would expect from gradient descent style methods, more on this later).

Algorithm 1 Perceptron

```
Init: $t = 1, w_1 = 0$
while there exists misclassified $i \in [n]$ (i.e. $y_t \langle w_t, x_i \rangle < 0$) do
    $w_{t+1} = w_t + y_i x_i$
    $t = t + 1$
end while
```

Let m be the number of corrections made by the Perceptron. Let's prove that m cannot be too large. The cosine of the angle between w_m and w^* is

$$\left\langle \frac{w_m}{\|w_m\|}, \frac{w^*}{\|w^*\|} \right\rangle \leq 1, \tag{4}$$

where $\|\cdot\|$ is the Euclidean norm for the rest of this lecture.

Let x_t denote the example used to update from step t to $t+1$. On the one hand,

$$\langle w_{t+1}, w^* \rangle = \langle w_t + y_t x_t, w^* \rangle = \langle w_t, w^* \rangle + y_t \langle x_t, w^* \rangle \geq \langle w_t, w^* \rangle + 1 \tag{5}$$

and thus $\langle w_m, w^* \rangle \geq m$. On the other hand,

$$\|w_{t+1}\|^2 \leq \|w_t\|^2 + 2y_t \langle w_t, x_t \rangle + \|x_t\|^2 \leq \|w_t\|^2 + R^2, \tag{6}$$

where $R = \max_i \|x_i\|^2$. Hence,

$$\|w_m\|^2 \leq mR^2.$$

Putting everything together,

$$\frac{m}{\sqrt{m}R\|w^*\|} \leq \left\langle \frac{w_m}{\|w_m\|}, \frac{w^*}{\|w^*\|} \right\rangle \leq 1, \tag{7}$$

or,

$$m \leq R^2 \|w^*\|^2.$$

We summarize what we have proved:

Lemma 1. *Let* $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$ *and* $R = \max_i \|x_i\|$. *Denote by* w^* *the vector of smallest norm such that* (3) *holds. Then Perceptron will terminate in at most* $R^2 \|w^*\|^2$ *rounds.*

1.2 Online interpretation

There are two interpretations of Lemma 1. One is online, and one is offline ("batch"). The online interpretation is as follows. Suppose n is large enough and we do not get to look at the same data point twice. That is, the data are streaming. Looking at the proof, we see that nothing changes, and we have a *mistake bound*: the number of mistakes made on *any* sequence (with the margin assumption) is at most $R^2\|w^*\|^2$. We will be able to make some interesting conclusions from this mistake bound, and relate it to mistakes made by our Bit Prediction method from last lecture. But for now, we leave it, and turn to the "batch" interpretation.

1.3 Batch interpretation

If you know about gradient descent, you may wonder if Perceptron is an instance of this concept. The step "find a misclassified example" may be replaced by "cycle through the data until you find a misclassified example". That is, no update is made for a correctly classified example. This can be realized as a zero gradient, and naturally leads to the function

$$\boldsymbol{\ell}(w,(x_t,y_t)) = \max\{0, -y_t \langle w, x_t\rangle\}$$

which we will call "hinge at zero" for reasons that will be clear soon. We will call a function that evaluates a fit of w and (x_t,y_t) a *loss function.* As such, $\mathbf{1}\{y_t \langle w, x_t\rangle < 0\}$ is a zero-one, or indicator, loss function.

Of course, we still have that

$$\sum_{t=1}^{n} \max\{0, -y_t \langle w^*, x_t\rangle\} = 0$$

and

$$\nabla_w \boldsymbol{\ell}(w,(x_t,y_t)) = -y_t x_t \cdot \mathbf{1}\{y_t \langle w, x_t\rangle < 0\}\,.$$

To be precise, it's a subgradient (see next lecture). The Perceptron update can now be written as

$$w_{t+1} = w_t - \nabla_w \boldsymbol{\ell}(w,(x_t,y_t)).$$

This type of update is called *gradient descent.* As we will see in the next lecture, one would typically have a "step-size" η (also known as "learning rate"), but here it is not necessary and can be set to $\eta = 1$. Let's see why. An update $\eta y_t x_t$ is equivalent to rescaling x_t's by a factor η, which changes the norm $\|w^*\|$ (since it's defined as the minimum norm hyperplane) by exactly the opposite factor, leaving the bound in Lemma 1 unchanged.

1.4 Non-separable case

When there is no perfect separator w^* for the data at hand, we may still ask whether one can find a $\widehat{w}$ that has the lowest number of errors:

$$\widehat{w} = \operatorname*{argmin}_{\|w\|=1} \sum_{t=1}^{n} \mathbf{1}\{y_t \langle w, x_t\rangle < 0\}\,. \tag{8}$$

This problem is known to be NP-hard in general [BBD99]. We could replace this objective with

$$\sum_{t=1}^{n} \max\{0, -y_t \langle w, x_t\rangle\},$$

as we did in the previous section. However, the meaning of the above sum is no longer related to the number of mistakes in (8). Instead, one opts for *surrogate losses* – those that upper bound the indicator loss. One classical example is the *hinge loss*

$$\boldsymbol{\ell}^h(w,(x_t,y_t)) = \max\{0, 1 - y_t \langle w, x_t\rangle\}$$

Check that

$$\mathbf{1}\{y_t \langle w, x_t\rangle < 0\} \le \max\{0, 1 - y_t \langle w, x_t\rangle\}$$

pointwise. The advantage of the hinge loss over the indicator loss is its convexity. Plus, the objective forces the hyperplane to make the margin to the points larger. Any point that is correctly classified, but close to the hyperplane will still incur loss and thus repel the hyperplane. That is, of course, if we make sure that the norm of w is minimized.

References

[ABR64] M.A. Aizerman, E.M. Braverman, and L.I. Roeznoer. The problem of patter recognition learning and the method of potential functions. *Avtomatika i Telemekhanika*, (25):1175–1193, 1964.

[ABR70] MA Aizerman, EM Braverman, and LI Rozonoer. *The Method of Potential Functions in the Theory of Machine Learning*. Nauka, Moscow, 1970.

[BBD99] Peter Bartlett and Shai Ben-David. Hardness results for neural network approximation problems. In *Computational Learning Theory*, pages 50–62. Springer, 1999.