Algorithms and Uncertainty, Winter 2018/19 Lecture 25 (4 pages)

Efficient Empirical Risk Minimization

Instructor: Thomas Kesselheim

Today, we will consider the question of how to *efficiently* minimize the empirical risk. In our argument about PAC-learnability, we always—somewhat naïvely—assumed to be able to find a hypothesis h such that $\mathrm{err}_S(h)$ is as small as possible. But how does this work?

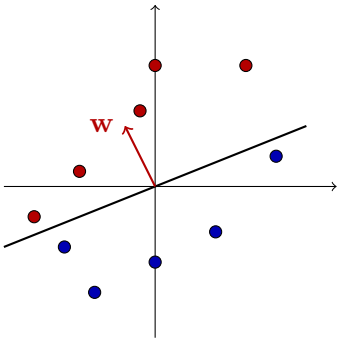
1 Linear Separators

We consider one of the easiest cases of binary classification. We would like to find a linear separator, that is, a hyperplane in $\mathbb{R}^d$ such that the points on one side of the hyperplane are positive and on the other side they are negative. We consider the homogenous case. That is, the hyperplane passes through the origin.

In this case, the hyperplane can be completely described by a normal vector $\mathbf{w} \in \mathbb{R}^d$, meaning that it contains the points $\mathbf{x} \in \mathbb{R}^d$ for which $\langle \mathbf{w}, \mathbf{x} \rangle = 0$, where $\langle \mathbf{w}, \mathbf{x} \rangle$ is the dot product of $\mathbf{w}$ and $\mathbf{x}$.

The vector $\mathbf{w}$ is orthogonal to the hyperplane. To get a hypothesis, we classify each point on the side of the hyperplane $\mathbf{w}$ is pointing to as positive.

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} +1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



By using the sign function (reinterpreting it such that it is +1 at 0), the hypothesis can be written even more succinctly

$$h_{\mathbf{w}}(\mathbf{x}) = \mathrm{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle\right) \quad .$$

2 ERM via Linear Programming

One way to find a linear separator in the realizable case is via linear programming. Recall that we would like to find a vector w such that for all $i \in \{1, \ldots, m\}$ we have $\mathrm{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle) = y_i$ or equivalently $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$.

Given any such vector $\mathbf{w}^*$, let $\gamma = \min_i y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle > 0$ and define $\mathbf{w}' = \frac{1}{\gamma}\mathbf{w}^*$. We observe that for all i the newly defined vector $\mathbf{w}'$ fulfills for all i

$$y_i \langle \mathbf{w}', \mathbf{x}_i \rangle = y_i \left\langle \frac{1}{\gamma}\mathbf{w}^*, \mathbf{x}_i \right\rangle = \frac{1}{\gamma} y_i \left\langle \mathbf{w}^*, \mathbf{x}_i \right\rangle \geq \frac{1}{\gamma}\gamma = 1 \quad .$$

So, all we have to do is to find a vector $\mathbf{w}' \in \mathbb{R}^d$ such that for all $i \in \{1, \dots, m\}$

$$\sum_{j=1}^{d} (y_i x_{i,j}) w'_j = y_i \langle \mathbf{w}', \mathbf{x}_i \rangle \geq 1 \ .$$

These are linear constraints. By finding any feasible solution to the respective linear program (with any arbitrary objective function), we find a linear classifier.

Note that this only works in the realizable case: Otherwise the vector $\mathbf{w}^*$ does not exist.

3 Perceptron Algorithm

Solving the LP is possible in polynomial time but may often take long because it does not exploit the specific structure. The Perceptron algorithm is a very old algorithm, which was already invented in 1957. It computes a sequence of vectors $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots$ and terminates if the current vector is correct on every point in the sample.

- Initialize $\mathbf{w}^{(1)} = (0, \dots, 0)$, $t = 1$
- While there is i with $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \leq 0$
  - Set $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$.
  - $t := t + 1$.

Theorem 25.1. *Let $R = \max_i \|\mathbf{x}_i\|_2$, $B = \min\{\|\mathbf{w}\|_2 \mid y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \text{ for all } i\}$. The Perceptron algorithm terminates after at most $(RB)^2$ iterations.*

Proof. Let $\mathbf{w}^*$ be a vector $\mathbf{w}$ such that $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$ for all i and $\|\mathbf{w}\|_2 = B$.

We will bound the angle between $\mathbf{w}^*$ and $\mathbf{w}^{(t)}$ for every t. More precisely, we will show that the vector $\mathbf{w}^{(t+1)}$, which is computed in the t^{th} iteration, fulfills

$$\frac{\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \rangle}{\|\mathbf{w}^*\|_2 \cdot \|\mathbf{w}^{(t+1)}\|_2} \geq \frac{\sqrt{t}}{RB} \ .$$

Note that the left-hand side is exactly the cosine of the angle between $\mathbf{w}^*$ and $\mathbf{w}^{(t)}$. After $T = (RB)^2$ iterations, the right-hand side is 1, so the angle is 0, and the algorithm must have terminated.

Consider one iteration t, in which the update $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$ is performed. For the numerator, we then have

$$\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \rangle = \langle \mathbf{w}^*, \mathbf{w}^{(t)} + y_i \mathbf{x}_i \rangle = \langle \mathbf{w}^*, \mathbf{w}^{(t)} \rangle + \langle \mathbf{w}^*, y_i \mathbf{x}_i \rangle = \langle \mathbf{w}^*, \mathbf{w}^{(t)} \rangle + y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq \langle \mathbf{w}^*, \mathbf{w}^{(t)} \rangle + 1 \ .$$

By applying the argument repeatedly and $\langle \mathbf{w}^*, \mathbf{w}^{(1)} \rangle = 0$, we get $\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \rangle \geq t$.

For the denominator, we have to upper-bound $\|\mathbf{w}^{(t+1)}\|_2$. To this end, we write

$$\|\mathbf{w}^{(t+1)}\|_2^2 = \|\mathbf{w}^{(t)} + y_i \mathbf{x}_i\|_2^2 = \langle \mathbf{w}^{(t)} + y_i \mathbf{x}_i, \mathbf{w}^{(t)} + y_i \mathbf{x}_i \rangle = \langle \mathbf{w}^{(t)}, \mathbf{w}^{(t)} \rangle + 2 \langle \mathbf{w}^{(t)}, y_i \mathbf{x}_i \rangle + \langle y_i \mathbf{x}_i, y_i \mathbf{x}_i \rangle$$

Note that $\langle \mathbf{w}^{(t)}, y_i \mathbf{x}_i \rangle \leq 0$ because $\mathbf{x}_i$ was classified incorrectly by $\mathbf{w}^{(t)}$. Therefore

$$\|\mathbf{w}^{(t+1)}\|_2^2 \leq \|\mathbf{w}^{(t)}\|_2^2 + \|y_i \mathbf{x}_i\|_2^2 = \|\mathbf{w}^{(t)}\|_2^2 + \|\mathbf{x}_i\|_2^2 \leq \|\mathbf{w}^{(t)}\|_2^2 + R^2 \ .$$

Applying the argument repeatedly and $\|\mathbf{w}^{(1)}\|_2^2 = 0$, we have $\|\mathbf{w}^{(t+1)}\|_2^2 \leq tR^2$ and therefore $\|\mathbf{w}^{(t+1)}\|_2 \leq \sqrt{t} R$. □

4 Stochastic Gradient Descent

One can also interpret the Perceptron algorithm as an application of an algorithm called Stochastic Gradient Descent. To see this, let us first rephrase the problem a bit. Given any normal vector $\mathbf{w}$ defining a hypothesis $h_{\mathbf{w}}$ and a labeled point $\mathbf{z} = (\mathbf{x}, y)$, we let

$$\ell^{0-1}(h_{\mathbf{w}}, \mathbf{z}) = \begin{cases} 0 & \text{if } y\langle \mathbf{w}, \mathbf{x}\rangle > 0 \\ 1 & \text{otherwise} \end{cases}$$

be the *0/1 loss function*. The training error is simply

$$\mathrm{err}_S(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} \ell^{0-1}(h_{\mathbf{w}}, \mathbf{z}_i) \ .$$

This function is not continuous. Therefore it is difficult to optimize it. Indeed, in the unrealizable case, the problem is NP-hard. One therefore often uses surrogate functions, which can be optimized more easily. A suitable choice in this case is the following surrogate loss function

$$\ell^{\text{surrogate}}(h_{\mathbf{w}}, \mathbf{z}) = \max\{0, -y\langle \mathbf{w}, \mathbf{x}\rangle\} \ .$$

In the realizable case, a perfect classifier $\mathbf{w}^*$ fulfills

$$\ell^{\text{surrogate}}(h_{\mathbf{w}}, \mathbf{z}) = 0$$

for all $\mathbf{z}$ in the training set. At the same time, if some $\mathbf{z}$ is labeled incorrectly, the surrogate loss will also be positive on this point.

This is why we can also find a vector $\mathbf{w}$ to minimize $\frac{1}{m}\sum_{i=1}^{m} \ell^{\text{surrogate}}(h_{\mathbf{w}}, \mathbf{z}_i)$ rather than $\frac{1}{m}\sum_{i=1}^{m} \ell^{0/1}(h_{\mathbf{w}}, \mathbf{z}_i)$.

4.1 Algorithm

The advantage of the surrogate loss function is that it is convex. To minimize $\frac{1}{m}\sum_{i=1}^{m} \ell^{\text{surrogate}}(h_{\mathbf{w}}, \mathbf{z}_i)$, let us consider the gradient of one term $\ell^{\text{surrogate}}(h_{\mathbf{w}}, \mathbf{z})$. The partial derivative by w_j is

$$\frac{\partial \ell^{\text{surrogate}}}{\partial w_j} = \begin{cases} 0 & \text{if } y\langle \mathbf{w}, \mathbf{x}\rangle > 0 \\ -yx_j & \text{if } y\langle \mathbf{w}, \mathbf{x}\rangle < 0 \end{cases} \ .$$

So the gradient is given by

$$\nabla_{\mathbf{w}} \ell^{\text{surrogate}} = \begin{cases} 0 & \text{if } y\langle \mathbf{w}, \mathbf{x}\rangle > 0 \\ -y\mathbf{x} & \text{if } y\langle \mathbf{w}, \mathbf{x}\rangle < 0 \end{cases} \ .$$

Indeed, in a step of the Perceptron algorithm, we subtract this gradient from the current vector $\mathbf{w}^{(t)}$. This is an algorithm template we can use for any loss function ℓ as long as it is convex. This algorithm is called stochastic gradient descent.

- Initialize $\mathbf{w}^{(1)} = (0, \ldots, 0)$, $t = 1$
- For all i (maybe multiple times)
  - Set $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} \ell(\mathbf{w}^{(t)}, \mathbf{z}_i)$.
  - $t := t + 1$.

As a matter of fact, this algorithm is familiar: We have already seen that Follow-the-Regularized-Leader with Euclidean Regularization is equivalent to it.

4.2 A Side Remark: Subgradients

The surrogate loss function is not differentiable if $y\langle \mathbf{w}, \mathbf{x}\rangle = 0$. However, this is not necessary for (stochastic) gradient descent.

It suffices if a function is convex. Recall that a differentiable function $f\colon S \to \mathbb{R}$, $S \subseteq \mathbb{R}^d$ is convex if and only if for all $\mathbf{u}, \mathbf{v} \in S$

$$f(\mathbf{u}) \geq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), (\mathbf{u} - \mathbf{v})\rangle \ .$$

This means that the function f has to stay above its tangent. More generally, if f is not differentiable, the tangent might not be unique. The characterization stays the same: It is convex if and only if for all $\mathbf{v} \in S$ there exists a $\mathbf{g} \in \mathbb{R}^d$ such that for all $\mathbf{u} \in S$

$$f(\mathbf{u}) \geq f(\mathbf{v}) + \langle \mathbf{g}, (\mathbf{u} - \mathbf{v})\rangle \ .$$

In case of a differentiable function, the only possible choice for $\mathbf{g}$ is the gradient at $\mathbf{v}$. If f is not differentiable, then there are multiple such vectors, which are then called *subgradients*.

4.3 (Surrogate) Loss Functions

As we see, it is sometimes helpful to replace the actual optimization objective with a surrogate one, which has nicer properties such as convexity. Indeed, by introducing such loss functions, in terms of optimization, there is no fundamental difference anymore between (binary) classification and predicting labels via regression.