# Computational Learning Theory
# Lecture 13: Online Convex Optimisation

Lecturer: James Worrell

In this lecture we briefly introduce some basic concepts behind online convex optimisation. We present the online subgradient descent procedure and show how the Perceptron algorithm can be seen as an instance of this procedure. More generally we show that both the Perceptron and Winnow algorithms are instances of follow-the-leader algorithms, but with different regularisation functions.

## 1 Subgradients

Let $S \subseteq \mathbb{R}^n$ be a convex set. One characterisation of when a function $f : S \to \mathbb{R}$ is convex is that for all $\boldsymbol{w} \in S$ there exists a vector $\boldsymbol{z} \in \mathbb{R}^n$ such that for $\boldsymbol{u} \in S$ we have

$$f(\boldsymbol{u}) - f(\boldsymbol{w}) \geq \boldsymbol{z} \cdot (\boldsymbol{u} - \boldsymbol{w}) \,. \tag{1}$$

Inequality (1) says that the linear function $g(\boldsymbol{u}) = f(\boldsymbol{w}) + \boldsymbol{z} \cdot (\boldsymbol{u} - \boldsymbol{w})$ is a lower bound of $f$ which is tangent to $f$ at $\boldsymbol{w}$. (See the book of Boyd and Vandenberghe for details.)

The set of vectors $\boldsymbol{z}$ that satisfy (1) defines the collection

$$\partial f(\boldsymbol{w}) = \{\boldsymbol{z} \in \mathbb{R}^n : f(\boldsymbol{u}) - f(\boldsymbol{w}) \geq \boldsymbol{z} \cdot (\boldsymbol{u} - \boldsymbol{w}) \text{ for all } \boldsymbol{u} \in S\}$$

of *subgradients* of $f$ at $\boldsymbol{w}$.

If $f$ is convex and differentiable at $\boldsymbol{w}$ then there is exactly one subgradient at $\boldsymbol{w}$, which is equal to the gradient: $\partial f(\boldsymbol{w}) = \{\nabla f(\boldsymbol{w})\}$.

**Example 1.** *Consider the hinge function $f : \mathbb{R} \to \mathbb{R}$, given by $f(w) = \max(0, -w)$. Then $\partial f(w) = \{0\}$ if $w > 0$, $\partial f(w) = \{-1\}$ if $w < 0$, and $\partial f(w) = [-1, 0]$ if $w = 0$. More generally, consider a labelled feature vector $(\boldsymbol{x}, y)$, with $\boldsymbol{x} \in \mathbb{R}^n$ and $y \in \{-1, +1\}$. Let $f : \mathbb{R} \to \mathbb{R}$ be the loss function*

$$f(\boldsymbol{w}) = \max\{0, -y(\boldsymbol{w} \cdot \boldsymbol{x})\}$$

*of the linear classifier $\boldsymbol{w}$ at the point $(\boldsymbol{x}, y)$. Then*

$$\partial f(\boldsymbol{w}) = \begin{cases} \{0\} & \text{if } y(\boldsymbol{w} \cdot \boldsymbol{x}) > 0 \\ \{-y\boldsymbol{x}\} & \text{if } y(\boldsymbol{w} \cdot \boldsymbol{x}) < 0 \\ \{-y\alpha\boldsymbol{x} : \alpha \in [0, 1]\} & \text{if } y(\boldsymbol{w} \cdot \boldsymbol{x}) = 0 \end{cases}$$

Let $\rho > 0$. We say that $f : S \to \mathbb{R}$ is *$\rho$-Lipschitz* if for all $\boldsymbol{u}, \boldsymbol{w} \in S$,

$$|f(\boldsymbol{u}) - f(\boldsymbol{w})| \leq \rho \|\boldsymbol{u} - \boldsymbol{w}\| \,.$$

Suppose that $f : S \to \mathbb{R}$ is $\rho$-Lipschitz and that $S$ is an open convex subset of $\mathbb{R}^n$. Then for any $\boldsymbol{w} \in S$ and subgradient $\boldsymbol{z} \in \partial f(\boldsymbol{w})$ we have $\|\boldsymbol{z}\| \leq \rho$. Indeed pick $\varepsilon > 0$ sufficiently small that $\boldsymbol{u} := \boldsymbol{w} + \varepsilon\boldsymbol{z} \in S$. Then

$$\rho\varepsilon\|\boldsymbol{z}\| = \rho\|\boldsymbol{u} - \boldsymbol{w}\| \geq |f(\boldsymbol{u}) - f(\boldsymbol{w})| \geq \boldsymbol{z} \cdot (\boldsymbol{u} - \boldsymbol{w}) = \varepsilon\|\boldsymbol{z}\|^2 \,.$$

Hence $\|\boldsymbol{z}\| \leq \rho$.

## 2 Online Convex Optimisation

The setup of online convex optimisation is as follows. Let $S \subseteq \mathbb{R}^n$ be a convex set and $G$ a set of convex loss functions $S \to \mathbb{R}$.

for $t = 1, 2, \ldots$
  the learner chooses prediction $\boldsymbol{w}_t \in S$
  the adversary chooses convex loss function $f_t \in G$
  the learner loses $f_t(\boldsymbol{w}_t)$

The regret of the learner over the first $T$ rounds is then the difference between their total loss and the total loss of the best comparator:

$$\text{Regret}(T) = \sup_{\boldsymbol{u} \in S} \sum_{t=1}^{T} (f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u})) \,.$$

Our goal is to minimize regret. In particular, we would like the regret to be sublinear in the number of rounds $T$ so that our *average* regret goes to 0 as $T$ goes to infinity.

### 2.1 Follow the (Regularised) Leader

A natural policy for the learner in the situation of online convex optimisation is called *follow the leader*. The idea is to select in round $t$ a vector $\boldsymbol{w}_t \in S$ that minimises the total loss on the previous $t-1$ rounds:

$$\boldsymbol{w}_t \in \arg\min_{\boldsymbol{w} \in S} \sum_{i=1}^{t-1} f_i(\boldsymbol{w}) \,. \tag{2}$$

Follow the leader is similar to the idea of minimising empirical error in the setting of batch learning.

The following is an example of a series of loss functions for which follow the leader generates regret linear in the number of rounds.

**Example 2.** *Let $S = [-1, 1]$ be the set of possible predictions of the learner. Assume that the adversary provides the following alternating sequence of (linear) loss functions:*

$$f_1(w) = -0.5w, \; f_2(w) = w, \; f_3(w) = -w, \; f_4(w) = w, \ldots$$

*In response the learner will generate the following sequence of predictions:*

$$w_1 = 0, \; w_2 = 1 \; w_3 = -1, \; w_4 = 1, \ldots$$

*The total loss over the first $T$ rounds is $T - 1$, but the comparator $w = 0$ obtains a loss of $0$ over the first $T$ rounds, so the regret is at least $T - 1$. This falls short of our goal sublinear regret.*

Intuitively, the bad behaviour of follow-the-leader algorithm in Example 2 stems from from the instability of the predictions, which oscillated between the two vertices of the set $S$. We can improve the stability of the follow-the-leader algorithm by adding a *regulariser term* $R(\boldsymbol{w})$ to the expression that the learner minimises in each step, for some convex function $R : S \to \mathbb{R}$. This leads to the *follow-the-regularised-leader* (FTRL) rule:

$$\boldsymbol{w}_t \in \arg\min_{\boldsymbol{w} \in S} \left( \sum_{i=1}^{t-1} f_i(\boldsymbol{w}) + R(\boldsymbol{w}) \right) \,. \tag{3}$$

The following theorem gives an upper bound on the regret of follow the regularised leader. Notice that the first term in the upper bound corresponds to the stability of the algorithm.

**Theorem 1.** *For any number of rounds $T$ and any $\boldsymbol{u} \in S$:*

$$\sum_{t=1}^{T}(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u})) \leq \sum_{t=1}^{T}(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{w}_{t+1})) + R(\boldsymbol{u}) - R(\boldsymbol{w}_1)\,.$$

*Proof.* Subtracting $\sum_{t=1}^{T} f_t(\boldsymbol{w}_t)$ from both sides, it is sufficient to show that

$$\sum_{t=1}^{T} f_t(\boldsymbol{w}_{t+1}) + R(\boldsymbol{w}_1) \leq \sum_{t=1}^{T} f(\boldsymbol{u}) + R(\boldsymbol{u}) \quad \text{for all } \boldsymbol{u} \in S \tag{4}$$

We prove (4) by induction on $T$. Assume by induction that

$$\sum_{t=1}^{T-1} f_t(\boldsymbol{w}_{t+1}) + R(\boldsymbol{w}_1) \leq \sum_{t=1}^{T-1} f(\boldsymbol{u}) + R(\boldsymbol{u}) \quad \text{for all } \boldsymbol{u} \in S$$

Adding $f_T(\boldsymbol{w}_{T+1})$ to both sides we have

$$\sum_{t=1}^{T} f_t(\boldsymbol{w}_{t+1}) + R(\boldsymbol{w}_1) \leq \sum_{t=1}^{T-1} f(\boldsymbol{u}) + f_T(\boldsymbol{w}_{T+1}) + R(\boldsymbol{u}) \quad \text{for all } \boldsymbol{u} \in S$$

In particular this holds for $\boldsymbol{u} = \boldsymbol{w}_{T+1}$, that is,

$$\sum_{t=1}^{T} f_t(\boldsymbol{w}_{t+1}) + R(\boldsymbol{w}_1) \leq \sum_{t=1}^{T} f_t(\boldsymbol{w}_{T+1}) + R(\boldsymbol{w}_{T+1}) \tag{5}$$

But since $\boldsymbol{w}_{T+1} \in \arg\min_{\boldsymbol{u} \in S} \left( \sum_{t=1}^{T} f_t(\boldsymbol{u}) + R(\boldsymbol{u}) \right)$ the inequality (5) directly implies (4). □

# 3 Online Subgradient Descent

In this section we present an algorithm for online convex optimisation that can be seen as an instance of FTRL with regulariser $R(\boldsymbol{w}) = \frac{1}{2\eta}\|\boldsymbol{w}\|^2$, where $\eta > 0$ is a parameter of the algorithm.

**Online Subgradient Descent** (with parameter $\eta > 0$)
**for** $t = 1, \ldots, T$
  Predict $\boldsymbol{w}_t := \arg\min_{\boldsymbol{w} \in S} \left\| \boldsymbol{w} - \eta \sum_{i=1}^{t-1} \boldsymbol{z}_i \right\|$
  Receive $g_t : S \to \mathbb{R}$
  Choose a subgradient $\boldsymbol{z}_t \in \partial g_t(\boldsymbol{w}_t)$
**end**

The prediction $\boldsymbol{w}_t$ is chosen to be the closest point in $S$ to the sum of the subgradients $-\eta \sum_{i=1}^{t-1} \boldsymbol{z}_i$. In case $S = \mathbb{R}^n$ we simply have $\boldsymbol{w}_t = -\eta \sum_{i=1}^{t-1} \boldsymbol{z}_i$.

**Remark 1.** *Suppose that each loss function $g_t$ is a linear function $g_t(\boldsymbol{w}) = \boldsymbol{v}_t \cdot \boldsymbol{w}$. Then the above algorithm makes exactly the same predictions as FTRL with regulariser $R(\boldsymbol{w}) = \frac{1}{2\eta}\|\boldsymbol{w}\|^2$. Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$ be the predictions on the online subgradient descent algorithm. Observe that the only choice for $\boldsymbol{z}_i \in \partial g_i(\boldsymbol{w}_i)$ is $\boldsymbol{z}_i = \boldsymbol{v}_i$, and so we have*

$$\boldsymbol{w}_t = \arg\min_{\boldsymbol{w} \in S} \frac{1}{2\eta} \left\| \boldsymbol{w} - \eta \sum_{i=1}^{t-1} \boldsymbol{v}_i \right\|^2 . \tag{6}$$

*On the other hand, the FTRL update rule (3) with the given sequence of loss functions is*

$$\boldsymbol{w}_t = \arg\min_{\boldsymbol{w} \in S} \left( \frac{1}{2\eta} \|\boldsymbol{w}\|^2 - \sum_{i=1}^{t-1} \boldsymbol{v}_i \cdot \boldsymbol{w} \right) . \tag{7}$$

*But, completing the square, we see that the objectives in the two optimisation problems (6) and (7) are equal up to a constant.*

The main result of this section shows that the online subgradient descent procedure has total regret $O(\sqrt{T})$ over $T$ rounds, assuming that the loss functions $f_t$ are convex, $\rho$-Lipschitz for some fixed $\rho$, and defined on a region $S$ of bounded diameter. This achieves our goal of sublinear regret.

**Theorem 2.** *Consider a run of the online subgradient descent algorithm in which the loss functions $g_1, \ldots, g_T : S \to \mathbb{R}$ are $\rho$-Lipschitz and such that $\sup\{\|\boldsymbol{u} - \boldsymbol{v}\| : \boldsymbol{u}, \boldsymbol{v} \in S\} \leq D$ for some $\rho, D > 0$. If $\eta = \frac{D}{\rho\sqrt{2T}}$ then the regret after $T$ rounds is at most $D\rho\sqrt{2T}$.*

*Proof.* For all $\boldsymbol{u} \in S$ we have

$$\begin{aligned}
\sum_{t=1}^{T} (g_t(\boldsymbol{w}_t) - g_t(\boldsymbol{u})) &\leq \sum_{t=1}^{T} \boldsymbol{z}_t \cdot (\boldsymbol{w}_t - \boldsymbol{u}) && (8)\\
&\leq \sum_{t=1}^{T} \boldsymbol{z}_t \cdot (\boldsymbol{w}_t - \boldsymbol{w}_{t+1}) + \frac{\|\boldsymbol{u}\|^2 - \|\boldsymbol{w}_1\|^2}{2\eta} && (9)\\
&\leq \sum_{t=1}^{T} \|\boldsymbol{z}_t\| \cdot \eta \|\boldsymbol{z}_t\| + \frac{\|\boldsymbol{u}\|^2 - \|\boldsymbol{w}_1\|^2}{2\eta} && (10)\\
&\leq T\rho^2\eta D + \frac{D^2}{2\eta} .
\end{aligned}$$

Line (8) holds because $\boldsymbol{z}_t$ is a subgradient of $g_t$ at $\boldsymbol{w}_t$. Line (9) follows from Theorem 9 applied to FTRL with linear loss functions $f_1, \ldots, f_T : S \to \mathbb{R}$, with $f_t(\boldsymbol{w}) = \boldsymbol{z}_t \cdot \boldsymbol{w}$ for $t = 1, \ldots, T$.[1] Line (10) follows from the Cauchy-Schwartz inequality, using the fact that

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \leq \left\| \eta \sum_{i=1}^{t} \boldsymbol{z}_i - \eta \sum_{i=1}^{t-1} \boldsymbol{z}_i \right\| = \|\eta \boldsymbol{z}_t\| .$$

Putting $\eta = \frac{D}{\rho\sqrt{2T}}$, the right-hand side above evaluates to $D\rho\sqrt{2T}$. □

## 3.1 Perceptron

The Perceptron algorithm can be seen as an instance of the online subgradient descent algorithm with parameter $\eta = 1$. Specifically let the loss function $g_t : \mathbb{R}^n \to \mathbb{R}$ in round $t$ of the online subgradient descent algorithm be

$$g_t(\boldsymbol{w}) = \max(0, -y_t(\boldsymbol{w} \cdot \boldsymbol{x}_t)) ,$$

where $(\boldsymbol{x}_t, y_t)$ is the labelled example in round $t$ of the Perceptron algorithm. Notice that the subgradient $\boldsymbol{z}_t$ at round $t$ satisfies $\boldsymbol{z}_t = \boldsymbol{0}$ if the prediction $\boldsymbol{w}_t$ has loss 0 and $\boldsymbol{z}_t = -y_t\boldsymbol{x}_t$ otherwise. Thus the prediction $\boldsymbol{w}_t = -\sum_{i=1}^{t-1} \boldsymbol{z}_i$ is exactly matches the update rule of the Perceptron algorithm as presented in Lecture 11.

[1] Note that by Remark 1 FTRL makes predictions $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$ when given this input by the adversary.

# 4 Winnow

Let us briefly sketch how the Winnow algorithm is based on follow-the-regularised-leader using the entropy function as a regulariser.

Let $\boldsymbol{p}$ be a probability distribution of $\{1, \ldots, n\}$. The entropy function is defined by

$$H(\boldsymbol{p}) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}.$$

Here, by convention, $0 \log(1/0)$ is defined to be $0$. Note that when $\boldsymbol{p}$ puts all of its weight on a single feature, we have $H(\boldsymbol{p}) = 0$. When $\boldsymbol{p}$ is uniform, we have $H(\boldsymbol{p}) = \log n$. In general, higher entropy implies "more uniform" weights. Because of this, using the negative entropy as the regularization term makes sense. Encouraging high entropy leads to more uniform weights, which leads to more stability in the algorithm.

Consider the FTRL algorithm with regulariser $-\frac{1}{\eta}H$ for some parameter $\eta > 0$. Assume that in round $t$ the adversary supplies the hinge loss function $f_t : \mathbb{R}^n \to \mathbb{R}$ defined by

$$f_t(\boldsymbol{w}) = \max(0, -y_t(\boldsymbol{w} \cdot \boldsymbol{x}_t))$$

where $(\boldsymbol{x}_t, y_t)$ is the $t$-th input to the Winnow algorithm. As in the case of online subgradient descent we can soundly apply the FTRL rule instead to the "surrogate" linear loss functions

$$g_t(\boldsymbol{w}) = \boldsymbol{z}_t \cdot \boldsymbol{w}$$

where $\boldsymbol{z}_t$ is a subgradient of $f_t$ at $\boldsymbol{w}_t$. Again we have that $\boldsymbol{z}_t = \boldsymbol{0}$ if the prediction $\boldsymbol{w}_t$ was correct on $(\boldsymbol{x}_t, y_t)$ and $\boldsymbol{z}_i = -y_i \boldsymbol{x}_i$ otherwise.

The prediction $\boldsymbol{w}_{t+1}$ of FTRL algorithm in round $t+1$ is obtained by the following rule:

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w} \in S} \sum_{i=1}^{t} \boldsymbol{z}_i \cdot \boldsymbol{w} - \frac{1}{\eta} H(\boldsymbol{w})\,. \tag{11}$$

Solving for $\boldsymbol{w}$ using calculus one obtains an update rule equivalent to that given in the Winnow algorithm in Lecture 12.