Online learning

Online learning

Online learning algorithm

For round $t = 1, 2, \ldots, T$:

1. Observe data point $x_t \in \mathcal{X}$.
2. Choose action $a_t \in \mathcal{A}$.
3. Observe label $y_t \in \mathcal{Y}$.
4. Suffer loss $\ell(a_t, y_t)$ (e.g., $\ell(a_t, y_t) = \mathbb{1}\{a_t \neq y_t\}$ when $\mathcal{A} = \mathcal{Y}$).

Underlying (non-probabilistic) assumption

Assume there is a function class $\mathcal{F}$ such that best-in-hindsight (average) loss $\min_{f \in \mathcal{F}} \frac{1}{T} \sum_{t=1}^{T} \ell(f(x_t), y_t)$ is not too large.

All "data" (e.g., x_t, y_t) could be controlled by an adversary!

Goal: perform (almost) as well as the best-in-hindsight $f \in \mathcal{F}$.

(Average) **Regret**: $$\underbrace{\frac{1}{T} \sum_{t=1}^{T} \ell(a_t, y_t)}_{\text{your avg. loss}} \quad - \quad \underbrace{\min_{f \in \mathcal{F}} \frac{1}{T} \sum_{t=1}^{T} \ell(f(x_t), y_t)}_{\text{avg. loss of best } f \in \mathcal{F}}$$

Example

$\mathcal{X} = \mathbb{N}$, $\mathcal{A} = \mathcal{Y} = \{0, 1\}$, $\ell(a, y) := \mathbb{1}\{a \neq y\}$, $\mathcal{F}$ = threshold functions.

t	1	2	3	4	5	6	7	8	9
x_t	7	8	1	9	2	6	3	1	4
a_t	0	1	1	1	1	1	1	0	1
y_t	1	1	0	1	0	1	1	0	0

(Average) **Regret**: $$\frac{1}{9} \sum_{t=1}^{9} \ell(a_t, y_t) \quad - \quad \min_{f \in \mathcal{F}} \frac{1}{9} \sum_{t=1}^{9} \ell(f(x_t), y_t)$$

$$= \quad \frac{4}{9} \quad - \quad \frac{1}{9}$$

($f(x) = \mathbb{1}\{x > 2\}$ makes just one mistake.)

Goal: Regret $\to 0$ as $T \to \infty$ (i.e., "vanishing regret" or "no regret").

Some easy cases

Simplifying assumptions

- $N := |\mathcal{F}|$ is finite, $\mathcal{A} = \mathcal{Y} = \{0, 1\}$, $\ell(a, y) = \mathbb{1}\{a \neq y\}$ (zero-one loss).
- *Realizability*: assume there is a $f \in \mathcal{F}$ with $y_t = f(x_t)$ for all t.

  So **Regret** is simply Average Loss (i.e., average number of mistakes):

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{1}\{a_t \neq y_t\}.$$

Trivial algorithm: follow-the-leader

In round t, pick any $f_t \in \mathcal{F}$ that is *perfect-so-far*, and choose $a_t := f_t(x_t)$.

What's a bound on the average loss after T rounds?

A better algorithm

"Halving" algorithm

In round t,

$$a_t \;:=\; \operatorname*{arg\,max}_{y \in \{0,1\}} \left| \{\text{perfect-so-far } f : f(x_t) = y\} \right|$$

(i.e., go with majority's prediction).

What's a bound on the average loss after T rounds?

What if we drop realizability assumption?

Example: failure of follow-the-leader

$\mathcal{A} = \{0,1\}$, $\mathcal{Y} = [0,1]$, $\ell(a,y) := |a - y|$ (absolute loss),

t	1	2	3	4	5	6	7	8	9
$f_0(x_t)$	0	0	0	0	0	0	0	0	0
$f_1(x_t)$	1	1	1	1	1	1	1	1	1
a_t	1	1	0	1	0	1	0	1	0
y_t	$2/3$	0	1	0	1	0	1	0	1
$L_t(f_0)$	$2/3$	$2/3$	$5/3$	$5/3$	$8/3$	$8/3$	$11/3$	$11/3$	$14/3$
$L_t(f_1)$	$1/3$	$4/3$	$4/3$	$7/3$	$7/3$	$10/3$	$10/3$	$13/3$	$13/3$
leader	f_1	f_0	f_1	f_0	f_1	f_0	f_1	f_0	f_1

Follow-the-leader: go with the function $f \in \mathcal{F}$ that is best-so-far.

Regret of follow-the-leader is $1 - O(1/T)$.

Can any learner do much better?

Cover's impossibility theorem

Let $\mathcal{A} = \mathcal{Y} := \{0,1\}$, and $\mathcal{F} := \{f_0 \text{ (always } 0),\ f_1 \text{ (always } 1)\}$.
Let $\ell(a,y) := \mathbb{1}\{a \neq y\}$ (zero-one loss).

Two observations:

1. For any $(y_1, y_2, \dots, y_T) \in \{0,1\}^T$,

$$\min_{f \in \mathcal{F}} \frac{1}{T} \sum_{t=1}^{T} \ell(f(x_t), y_t) \;\leq\; \frac{1}{2}\,.$$

   (One of f_0 and f_1 has average loss $\leq 1/2$.)

2. For any learner, there exists $(y_1, y_2, \dots, y_T) \in \{0,1\}^T$ such that

$$\frac{1}{T} \sum_{t=1}^{T} \ell(a_t, y_t) \;=\; 1\,.$$

   (Think of a_t as a function of $x_1, y_1, x_2, y_2, \dots, x_{t-1}, y_{t-1}, x_t$.)

Conclusion: No learner can do better than regret $1/2$.

However:
If the learner can be randomized, then can make regret $\to 0$ as $T \to \infty$.

Using randomness

Assume $N := |\mathcal{F}|$ is finite, and $\ell(y', y) \in [0,1]$.
Let $L_t(f) := \sum_{\tau=1}^{t} \ell(f(x_\tau), y_\tau)$ be cumulative loss of f after t rounds.

Randomized Weighted Majority (Littlestone and Warmuth, 1994)

In round t, randomly pick f_t from distribution $\boldsymbol{w}_t$ on $\mathcal{F}$ given by

$$w_t(f) \;\propto\; \exp\bigl(-\eta \cdot L_{t-1}(f)\bigr)\,;$$

predict $a_t := f_t(x_t)$.

Expected regret after T rounds:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \ell(a_t, y_t) - \min_{f \in \mathcal{F}} \frac{1}{T}\sum_{t=1}^{T} \ell(f(x_t), y_t)\right] \;\leq\; \frac{1}{\eta} \cdot \frac{\ln(N)}{T} + \eta \cdot \frac{1}{2}\,.$$

Optimize bound with $\eta := \sqrt{\frac{2\ln(N)}{T}}$ to get regret bound $\sqrt{\frac{2\ln(N)}{T}}$.

Could also use $\eta_t := \sqrt{\frac{2\ln(N)}{t}}$ in round t.

Example: RWM

$\mathcal{A} = \{0,1\}$, $\mathcal{Y} = [0,1]$, $\ell(a,y) := |a-y|$ (absolute loss), $|\mathcal{F}| = 2$.

t	1	2	3	4	5	6	7	8	9
$f_0(x_t)$	0	0	0	0	0	0	0	0	0
$f_1(x_t)$	1	1	1	1	1	1	1	1	1
$w_t(f_1)$	.500	.569	.389	.549	.413	.540	.426	.535	.435
y_t	$2/3$	0	1	0	1	0	1	0	1
$L_t(f_0)$	$2/3$	$2/3$	$5/3$	$5/3$	$8/3$	$8/3$	$11/3$	$11/3$	$14/3$
$L_t(f_1)$	$1/3$	$4/3$	$4/3$	$7/3$	$7/3$	$10/3$	$10/3$	$13/3$	$13/3$
leader	f_1	f_0	f_1	f_0	f_1	f_0	f_1	f_0	f_1

Expected regret of RWM is $O(1/\sqrt{T})$.

Why RWM works

- Randomization lets us precisely "hedge" over functions in $\mathcal{F}$.

  (A generalization of RWM called "Hedge" is due to Freund and Schapire; proposed in in same paper as AdaBoost.)

- Minimizer of cumulative loss L_{t-1} can change in every round. But the distribution given by

$$w_t(f) \;\propto\; \exp\bigl(-\eta \cdot L_{t-1}(f)\bigr)$$

  (as in RWM) changes relatively slowly when η is small.

A closer look at RWM

Let $\lambda_t(f) := \ell(f(x_t), y_t)$, so expected loss of RWM in round t is

$$\sum_{f \in \mathcal{F}} \lambda_t(f) \cdot w_t(f) \;=\; \langle \boldsymbol{\lambda}_t, \boldsymbol{w}_t \rangle .$$

Interpretation: $\boldsymbol{w}_t$ is solution to a convex optimization problem

$$\boldsymbol{w}_t \;:=\; \operatorname*{arg\,min}_{\boldsymbol{w} \in \Delta_{\mathcal{F}}} \sum_{i=1}^{t-1} \langle \boldsymbol{\lambda}_i, \boldsymbol{w} \rangle - \frac{1}{\eta} H(\boldsymbol{w}) ,$$

where $\Delta_{\mathcal{F}}$ is the set of probability distributions over $\mathcal{F}$, and H is Shannon's entropy function.

Without the "$-(1/\eta)H(\boldsymbol{w})$", optimization problem is

$$\operatorname*{arg\,min}_{\boldsymbol{w} \in \Delta_{\mathcal{F}}} \sum_{i=1}^{t-1} \langle \boldsymbol{\lambda}_i, \boldsymbol{w} \rangle \;=\; \operatorname*{arg\,min}_{f \in \mathcal{F}} \sum_{\tau=1}^{t-1} \lambda_\tau(f) \;=\; \operatorname*{arg\,min}_{f \in \mathcal{F}} L_{t-1}(f) ,$$

which is "follow-the-leader"—i.e., go with $f \in \mathcal{F}$ that's doing best-so-far.
Doesn't guarantee vanishing regret!

RWM is **regularized follow-the-leader** (with *negative entropy regularization*).

Some form of "regularization" is crucial for vanishing regret.

Application to learning decision trees

Usual approach

Split training data S into two parts, S' and S'':

- Use first part S' to **grow tree $\mathcal{T}$ until all leaves are pure**.
- Use second part S'' to **choose a good pruning of tree $\mathcal{T}$**.

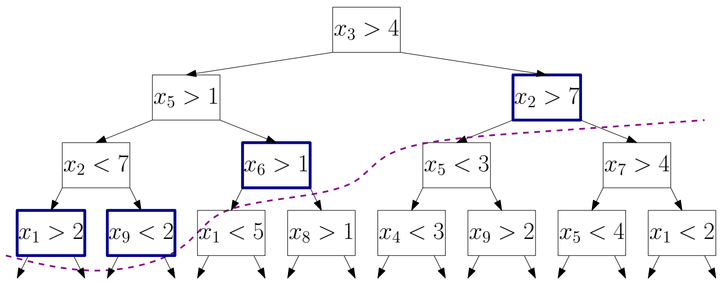
Alternative approach (Helmbold and Schapire, 1995)

Split training data S into two parts, S' and S'':

- Use first part S' to **grow tree $\mathcal{T}$ until all leaves are pure**.
- Use second part S'' to **learn weighting over all prunings of tree $\mathcal{T}$**: Let $\mathcal{F}$ = all prunings of $\mathcal{T}$, and run RWM using S''.

  Problem: $|\mathcal{F}|$ is exponentially large!

Dynamic programming



- Each $f \in \mathcal{F}$ corresponds to a pruning of the tree $\mathcal{T}$.
- $L_t(f)$ depends on prediction error statistics at leaves of f; weight of f is just $\exp(-\eta L_t(f))$.
- Maintain error statistics for all nodes in $\mathcal{T}$.
- Can use dynamic programming to combine weights of all $f \in \mathcal{F}$ to make weighted prediction.
- See (Helmbold & Schapire, 1995) for details.

Other examples

- **Perceptron algorithm**: online learning algorithm for $\mathcal{F}$ = linear classifiers.
- **Winnow algorithm**: much like Perceptron, but works better when there is a linear separator that uses only a few features.
- **Online gradient descent**: cosmetically similar to *stochastic gradient descent*, for online convex optimization problem

  (i.e., in each round, there is a different convex cost function).

More on online learning

Many problems and algorithms in online learning:

- Partial feedback ("multi-armed bandit")
- Bayesian inference
- Online convex optimization
- Tracking / filtering
- ...

Many connections between online learning and other subjects:

- Game theory, calibration, boosting
- Stochastic optimization
- Compression, coding theory
- ...