CSC 665: Online classification

Chicheng Zhang

November 26, 2019

1 Online learning

1. Sequential decision making problem
2. Different from statistical learning, here training and test are interleaved
3. Has many applications, e.g. spam filtering, (personalized) product recommendation, experimental design, sequential investment, etc

General setup:

Algorithm 1 Online learning: general setup

Require: Context space $\mathcal{X}$, action space $\mathcal{A}$.
  for timesteps $t = 1, 2, \ldots, T$: **do**
    (Optional) Observe context $x_t \in \mathcal{X}$
    Take action $a_t \in \mathcal{A}$
    Receive feedback b_t (that reveals information about loss ℓ_t)
  end for
  Goal: minimize cumulative loss $\sum_{t=1}^{T} \ell_t(a_t)$

Examples:

1. Spam filtering (online classification).
   (a) Each x_t (in $\mathcal{X} = \mathbb{R}^d$) denotes the feature representation of an email.
   (b) Use $\mathcal{A} = \{\pm 1\}$: $+1$ denotes non-spam, -1 denotes spam.
   (c) Feedback $b_t = y_t$: true label of email
   (d) Loss: $\ell_t(a) = \mathbf{1}(a \neq y_t)$ - classification error
2. Spam filtering, modified (partial information online classification). Same as the setup before, except that the feedback model is different:
$$b_t = \begin{cases} y_t & a_t = -1 \\ \perp & a_t = +1 \end{cases}$$
   In words, if an email is classified as non-spam, then it goes to the inbox and user marks spam if necessary; however if an email is classified as spam, then the user does not check the spam folder and never provided feedback on it.
3. Product recommendation (multi-armed bandits).
   (a) No context
   (b) $\mathcal{A} = \{1, \ldots, K\}$: K products

(c) Loss: $\ell_t(a)$ - the cost of recommending product a to user t (characterizing user's preferences on all products)

(d) Feedback: $b_t = \ell_t(a_t)$ - user's preferences on product recommended (but not other $K-1$ products)

4. Personalized product recommendation (contextual bandits). Same as the setup before, except that a context x_t is given at each timestep t, that reveals "charateristics" about user t. The goal is to utilize the contexts to make better product recommendations.

Some terminlogies:

1. Full information vs. Partial information: if b_t reveals the true loss function ℓ_t, then it is called full-information setting; otherwise it is called partial-information setting. Both settings have many applications in practice.

2. Stochastic vs. Adversarial: if (x_t, ℓ_t)'s are iid, then it is called the stochastic setting (where techniques in statistical learning can potentially carry over); Adversarial setting refers to the setup where we don't have assumptions on the data generation process.

2 Online classification

Convention: as $\mathcal{A} = \mathcal{Y} = \{\pm 1\}$, we often write a_t as $\hat{y}_t$. The goal is to minimize the cumulative number of classification errors, $\sum_{t=1}^{T} \mathbf{1}(\hat{y}_t \neq y_t)$.

As a starting point, let's consider a simple setting when $\mathcal{H}$ is finite, and we are in the realizable setting: there exists a classifier $h^\star$ in $\mathcal{H}$ that agrees with all the examples. How can we design a learning algorithm that makes a small number of mistakes?

The consistency algorithm: a first trial. One plausible idea is to utilize the consitency algorithm (or ERM algorithm) we studied in statistical learning: at time t, define version space $V_t = \big\{h \in \mathcal{H} : h(x_s) = y_s \forall s \leq t-1\big\}$. The name *version space* comes from the fact that this set of classifiers gets constantly updated, therefore having "versions" indexed by timesteps. The algorithm then selects a classifier $h_t \in V_t$ and uses that to make prediction: $\hat{y}_t = h_t(x_t)$.

Theorem 1. *The consistency algorithm makes $|\mathcal{H}| - 1$ mistakes (regardless of the length of time horizon T).*

Proof. First, by the definition of version space, and the realizability assumption, the algorithm maintains the invariant that $h^\star \in V_t$.

Second, let $M_t = \mathbf{1}(\hat{y}_t \neq y_t)$ be the mistake indicator at time t. We claim that $M_t \leq |V_t| - |V_{t+1}|$. We look at two cases:

1. if $M_t = 0$, as $V_{t+1} \subset V_t$, the right hand side is positive, thus the inequality is true.

2. if $M_t = 1$, note that h_t will be excluded from V_t. This implies that $|V_{t+1}| - |V_t| \geq 1$.

Define potential $\Phi_t = |V_t|$; in this notation, $M_t \leq \Phi_t - \Phi_{t+1}$. Summing over all round t from 1 to T, we have

$$\sum_{t=1}^{T} \leq \Phi_1 - \Phi_{T+1} \leq \Phi_1 = |\mathcal{H}|.$$

□

Can we design a better algorithm for online binary classification under realizability? It turns out that using a more carefully designed prediction strategy we can do much better, reducing the mistake bound from linear in $|\mathcal{H}|$ to only logarithmic.

The halving algorithm. Same as the consistency algorithm, Halving will also keep track of version space, defined in the same way. At timestep t, observe that example x_t divides the version space V_t into two parts: $V_t^+ = \{h \in V_t : h(x_t) = +1\}$, and $V_t^- = \{h \in V_t : h(x_t) = -1\}$. We know that at the end of round t, one of them would become the update version space V_{t+1}.

The halving algorithm makes prediction as follows:

$$\hat{y}_t = \begin{cases} +1 & |V_t^+| \geq |V_t|/2, \\ -1 & \text{otherwise}, \end{cases}$$

which is equivalent to a majority vote over classifiers in V_t (tie broken in favor of $+1$).

What is the advantage of this new prediction strategy? We claim that $M_t \leq \log |V_t| - \log |V_{t+1}|$. Why? We also conduct a case analysis on M_t:

1. If $M_t = 0$, the right hand side is $\log |V_t| - \log |V_{t+1}|$, which is at least 0; therefore the inequality is true.
2. If $M_t = 1$, then $y_t = -\hat{y}_t$. It can be seen that from the definition of $\hat{y}_t$, $V_{t+1} = V_t^{y_t} = V_t^{-\hat{y}_t}$ is the minority class, implying that $|V_{t+1}| \leq |V_t|/2$. The claimed inequality is shown by taking logarithms on both sides.

Define potential $\Phi_t = \log |V_t|$; in this notation, $M_t \leq \Phi_t - \Phi_{t+1}$. Summing over all round t from 1 to T, we have

$$\sum_{t=1}^{T} \leq \Phi_1 - \Phi_{T+1} \leq \Phi_1 = \log |\mathcal{H}|.$$

To summarize, we have the following theorem.

Theorem 2. *The halving algorithm makes* $\log |\mathcal{H}|$ *mistakes (regardless of the length of time horizon* T*).*

3 Mistake bound model and the minimax analysis of realizable online learning

Definition 1. *1. Algorithm* $\mathcal{A}$ *is said to acheive a mistake bound* B *with hypothesis class* $\mathcal{H}$*, if for any sequence of examples* S *realizable with respect to* $\mathcal{H}$*,* $\mathcal{M}_{\mathcal{A}}(S)$*, the cumulative number of mistakes made by* $\mathcal{A}$ *on* S *is at most* B*.*

2. $\mathcal{H}$ *is online learnable if there exists an algorithm* $\mathcal{A}$ *such that* $\mathcal{A}$ *achieves a finite mistake bound on* $\mathcal{H}$*.*

In light of Theorem 1 or Theorem 2, we have the following simple fact.

Corollary 1. *Any finite hypothesis class* $\mathcal{H}$ *is online learnable.*

What happens for infinte hypothesis classes? Can we develop similar capacity measure like VC dimension that measures the fundamental complexity of online learning? Imaging online classification as a game between the learner and the environment, where the learning algorithm makes online predictions at every timestep that tries to minimize the number of mistakes, while the environment shows examples sequentially to "trick" the learner to make as many mistakes, without violating the realizability assumption. We can formulate the design of online classification algorithms as the following optimization problem:

$$\min_{\mathcal{A}} \max_{S: S \text{ realizable by } \mathcal{H}} \mathcal{M}_{\mathcal{A}}(S).$$

In other words, we would like to design an algorithm, $\mathcal{A}$, that has the smallest number of worst case mistakes (where every sequence of example realizable by $\mathcal{H}$ is a "case").

To analyze this, let us first look at the flip side: can we design a strategy of the environment to enforce any learner to make a lot of mistakes? To study this, we need the concept called mistake trees. (See handwritten notes for details.)