CSC 588: Machine learning theory **Spring 2021**

Lecture 17: Stability-fitting tradeoff; online learning

Lecturer: Chicheng Zhang *Scribe: Sarah Luca*

1 Stability-fitting tradeoff

In the previous lecture we discussed OARO stability:

Theorem 1. *If $\mathcal{A}$ is OARO-stable (on-average-replace-one) with rate g then*

$$\mathbb{E}_{S\sim D^m}[L_D(\mathcal{A}(S)) - L_S(\mathcal{A}(S))] \leq g(m).$$

In other words, if you have a guaranteed small expected loss difference (controlled by a rate function g where $g(m)$ is small) for a learning algorithm's ($\mathcal{A}$) output classifier when you randomly pick a sample and replace it with another sample, it is an OARO-stable algorithm.

Under this assumption we can show that the learning algorithm $\mathcal{A}$ generalizes well.

Example 1. *l_2-regularization is an example of this stability:*

Assume:

1. *$l(w, z)$ is ρ-lipschitz w.r.t w for any z*
2. *$l(w, z)$ is convex w.r.t. w for any z*
3. *$\hat{w} = \text{argmin}_{w\in\mathbb{R}^d}(\frac{\lambda}{2}\|w\|_2^2 + L_S(w))$*

1,2,3 $\Rightarrow$ $\mathcal{A}$ is $g(m) = \frac{2\rho^2}{\lambda m}$-OARO stable.

We want to apply this result to do an end-to-end analysis for regularized loss minimization (i.e. we want to find a λ such that the output classifier $\hat{w}$ has the lowest expected generalization error $\mathbb{E}_{S\sim D^m}[L_D(\mathcal{A}(S))]$).

Note, we can decompose the expected generalization error into the expected training error and the expected generalization gap:

$$\mathbb{E}_{S\sim D^m}[L_D(\mathcal{A}(S))] = \mathbb{E}_{S\sim D^m}[L_S(\mathcal{A}(S))] + \mathbb{E}_{S\sim D^m}[L_D(\mathcal{A}(S)) - L_S(\mathcal{A}(S))]$$

We can control the expected loss generalization gap with the stability rate function:

$$\mathbb{E}_{S\sim D^m}[L_D(\mathcal{A}(S)) - L_S(\mathcal{A}(S))] \leq g(m) = \frac{2\rho^2}{\lambda m}$$

If we increase the regularization strength λ too much, then the objective function could heavily skew penalizing the l_2 norm, giving a large empirical loss. Thus $\mathbb{E}_{S\sim D^m}[L_S(\mathcal{A}(S))]$ can increase when λ is large, whereas $\frac{2\rho^2}{\lambda m}$ decreases as λ increases. This gives a tradeoff between fitness and stability. This means we need to upper bound $\mathbb{E}_{S\sim D^m}[L_S(\mathcal{A}(S))]$ in terms of λ.

Define the objective function $F_s(w) = \frac{\lambda}{2}\|w\|_2^2 + L_S(w)$. Then

$$F_S(\mathcal{A}(S)) \leq F_S(w^*)$$

where $w^* \in \mathbb{R}^d$. Note that $F_S(\mathcal{A}(S))$ is an upper bound for the original empirical loss $L_S(\mathcal{A}(S))$:

$$L_S(\mathcal{A}(S)) \leq F_S(\mathcal{A}(S)) \leq F_S(w^*)$$

because l_2 regularization is always positive. Taking the expectation of both sides (ignoring the middle inequality) we can relate expected empirical error to the expected regularize

$$\begin{aligned}
\mathbb{E}_{S\sim D^m}[L_S(\mathcal{A}(S))] &\leq \mathbb{E}_{S\sim D^m}[F_S(w^*)] \\
&= \mathbb{E}_{S\sim D^m}\left[\frac{\lambda}{2}\|w^*\|^2 + L_S(w^*)\right] \\
&= \frac{\lambda}{2}\|w^*\|^2 + L_D(w^*)
\end{aligned}$$

by linearity and since $\mathbb{E}_{S\sim D^m} L_S(w^*) = L_D(w^*)$. In summary:

$$\forall\ w^*,\ \mathbb{E}_{S\sim D^m} L_D(\mathcal{A}(S)) \leq \frac{\lambda}{2}\|w^*\|^2 + L_D(w^*) + \frac{2\rho^2}{\lambda m}$$

Interpretations:

1. Suppose $\lambda = \mathcal{O}(\frac{1}{\sqrt{m}})$ (or any other function of m where $\frac{1}{m} \ll \lambda \ll 1$), with this setting of λ, the complexity penalty $\frac{\lambda}{2}\|w^*\|^2$ decreases with increasing sample size. In other words:

   $$\frac{1}{2\sqrt{m}}\|w^*\|^2 + \frac{2\sqrt{m}\rho^2}{m}$$

   decreases as m increases for $m > 1$. So

   $$\mathbb{E}_{S\sim D^m} L_D(\mathcal{A}(S)) \leq \min_{w^*\in\mathbb{R}^d}\left(L_D(w^*) + \frac{1}{2\sqrt{m}}\|w^*\|^2 + \frac{2\sqrt{m}\rho^2}{m}\right)$$

   For example, you can think of the set of linear predictors as belonging to the nested set of classifiers (see Figure 1) where $\mathcal{H}_i = \{w : \|w\|_2 \leq f(i)\}$ where $f(i) = i$, or $f(i) = 2^i$ or $f(i)$ is another other monotonically increasing function. In other words, it can compete with any of the hypothesis classes by paying the appropriate penalties. This echos the model selection result we have obtained by using cross validation and structural risk minimization.

   (Chicheng notes: another good choice of λ is $\lambda = \frac{\rho}{\sqrt{m}}$; this requires the knowledge of ρ. For this choice of λ, we have

   $$\mathbb{E}_{S\sim D^m} L_D(\mathcal{A}(S)) \leq \min_{w^*\in\mathbb{R}^d}\left(L_D(w^*) + \frac{\rho(1+\|w^*\|^2)}{2\sqrt{m}}\right)$$

   which can be better than the above bound when ρ is far away from 1.)

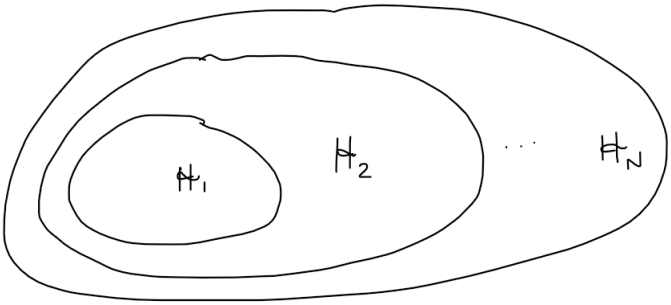

Figure 1:

2. Fix hypothesis class:

   Say we want to focus on linear predictors with bounded norms:

   $$\mathcal{H} = \{w \in \mathbb{R}^d : \|w\|_2 \leq B\}$$

   The result tells us that:

   $$\mathbb{E}_S[L_D(\mathcal{A}(s))] \leq \min_{w \in \mathcal{H}} L_D(w) + \frac{\lambda}{2}B^2 + \frac{2\rho^2}{\lambda m}$$

   If we pick $\lambda^* = 2\frac{\rho}{B}\sqrt{\frac{1}{m}}$, we get

   $$\frac{\lambda}{2}B^2 + \frac{2\rho^2}{\lambda m} = 2\rho B\sqrt{\frac{1}{m}}.$$

   This says that this algorithm has expected excess loss of this quantity which decreases as the number of training samples increases. This is a PAC-like guarantee, except that the guarantee is measured with expected loss, as opposed to bounding the loss of the output predictor with high probability.

   We can set the number of training samples m such that the excess loss upper bound is at most ϵ :

   $$m \geq \frac{4\rho^2 B^2}{\epsilon^2} \Rightarrow \mathbb{E}_S[L_D(\mathcal{A}(S))] - \min_{w \in \mathcal{H}} L_D(w)] \leq \epsilon.$$

2 Online Learning

Online learning is a sequential decision making problem where the learner tries to collect information online and act on this information.

Examples:

1. Spam detection:

   At each time step, $t = 1, 2, ..., T$:

   – Spam filter receives an email: $x_t \in \mathcal{X}$

- Based on new information from email, predict $\hat{y}_t \in \{-1, +1\}$ where -1 denotes no spam, and $+1$ denotes spam.
- User continuously checks email and spam folders so every time spam filter makes a prediction, it gets immediate feedback. It sees the ground truth label $y_t \in \{1, +1\}$.

We want to make the filter as accurate as possible (i.e. minimize $\sum_{t=1}^{T} I(\hat{y}_t \neq y_t)$)

This setup is called online classification and can be extended to real valued outputs which corresponds to online regression. In statistical learning, rather than receive input and feedback continuously in time (as in this online example), you receive in put in batches and use this to generate a prediction rule that can be used for future predictions.

2. Sequential investment (portfolio selection):

   W_1 = initial capital.
   At each time step $t = 1, 2, ..., T$:

   - Decide how to divide the capital $P_t \in \Delta^{N-1} = \{P \in \mathbb{R}_t^N : \sum_i P(i) = 1\}$ and for each asset $i \in \{1, ..., N\}$ how much money to invest $W_t P_t(i)$.
   - Observe relative price change of the asset $r_t \in \mathbb{R}_t^N$ (the price of the asset after time t divided by the price at the beginning.) In other words, $r_t > 1 \Rightarrow$ the price increases, $r_t < 1 \Rightarrow$ the price decreases, $r_t = 1 \Rightarrow$ the price stays the same.

   At the end of day t, we have the price of the asset i : $W_t P_t(i) r_t(i)$ Then

   $$W_{t+1} = \sum_i W_t P_t(i) r_t(i) = W_t \langle P_t, r_t \rangle$$

   Goal: maximize W_{T+1} the final capital.

3. Aggregating weather prediction:
   For each day $t = 1, 2, ..., T$

   - Obtain weather temperature predictions from N experts/models
   - Based on predictions of all models, make a final prediction by following a randomly chosen expert drawn from $P_t \in \Delta^{N-1}$
   - Observe the losses of each model $l_t \in [0, 1]^N$. When expert i predicts today's weather well, that entry of $l_t(i)$ will be small.

   Goal: make accurate predictions by minimizing $\sum_t \langle P_t, l_t \rangle$

Examples 1-3 have been in a fully informational setting (i.e. spam labels, relative changes of the price of assets, and loss of each model). The next example uses learning with limited feedback (bandit feedback).

4. Production recommendation (multi-armed bandits):
   For $t = 1, 2, ..., T$

   - Recommend a product $a_t \in \{1, ..., k\}$ to a new customer
   - The customer will see recommendation and react to it (i.e. if the ad is a link, the customer will either click the link or not)
   - Observe loss of particular ad a_t : $l_t(a_t)$, where $l_t \in [0, 1]^k$ (i.e. clicked$\Rightarrow$ $l_t(a_t) = 0$, not clicked$\Rightarrow$ $l_t(a_t) = 1$)

The algorithm is only observing the loss of the recommend product, but we are assuming there is a ground truth loss vector l_t that is k-dimensional that encodes the users preferences of the product (i.e. $l_t(i)$ is the counterfactual loss of users reaction to the ith product.

Goal: minimize cummulative loss $\sum_t l_t(a_t)$

Note: the specific feedback $l_t(a_t)$ for this example is called bandit feedback.

5. Personalized product recommendation:
   Given N policies: $\Pi^1 ... \Pi^N$, each Π^i is a mapping from the customer space to the product space $(\Pi^i : \mathcal{X} \to \{1, ..., k\})$.
   For $t = 1, 2, ..., T$
   - Observe contextual information of customer x_t
   - Randomly select one of the N policies $\Pi_t \sim W_t \in \Delta^{N-1}$
   - Recommend the product with customer information as input: $\Pi_t(x_t)$
   - Observe users reaction (loss) on the recommended product: $l_t(\Pi_t(x_t))$. Similarly, here we don't observe the other entries of the k-dimensional loss vector.

   This is a contextual bandit model. The difference between this model and the multi-armed bandit model is each customer is equipped with a context x_t and we can use contextual information to refine the recommendations. In other words, the policies Π^i can help make the loss of a particular product recommendation l_t smaller than in the multi-armed bandit case.

3 Online (Convex) Optimization

Online learning can be captured in a simplified model with online convex optimization. This model can be thought of as a game between a learner and an adversary. We need a few definitions:

Definition 2. *The decision set called Ω is given to the learner as the action space. Often this set is convex.*

Definition 3 (Convex Set)**.** *The set Ω is convex if $\forall u, v \in \Omega \ \forall \alpha \in (0, 1)$, then*

$$\alpha u + (1 - \alpha)v \in \Omega$$

Given a decision set Ω, the game proceeds as follows:
For $t = 1, 2, ..., T$

- The learner commits to an action (i.e. picks $W_t \in \Omega$)
- Environment picks loss function $f_t : \Omega \to \mathbb{R}$
- Learner suffers loss $f_t(W_t)$, and observes information on f_t.

Goal: minimize cumulative loss: $\sum_{t=1}^{T} f_t(W_t)$

We also need to distinguish between a few settings of online learning:

- Stochastic: loss functions are chosen randomly from a distribution $(\{f_t\}_{t=1}^{T} \overset{i.i.d}{\sim} D)$
- Oblivious adversary: loss functions $\{f_t\}_{t=1}^{T}$ chosen ahead of time.

– Adaptive adversary: (in contrast with oblivious) $\forall t,\ f_t$ can depend on the previous decision made by the learning algorithm $\{W_s\}_{s=1}^t$.

This is categorization based on the choices of f_t, but we also have categorization based on the feedback model:

– The learner sees full information and sees f_t (or $\nabla f_t(W_t)$).

– Bandit setting: learner only sees loss of the specific action taken $f_t(W_t) \in \mathbb{R}$

– Other feedback settings (partial information)

If we go back to our examples from before, we can consider them in the context of online convex optimization:

1. Spam detection:
   – The action set can be the set of all linear predictors ($\Omega = \{w : \|w\|_2 \leq B\}$)
   – The loss function is the classification loss defined in terms of the linear predictors: $f_t(w) = I(y_t\langle w, x_t\rangle \leq 0) = \ln(1 + e^{-y_t\langle w, x_t\rangle})$
2. Sequential investment (portfolio selection):
   – The action space is the decision made by learners $\Omega = \Delta^{N-1}$.
   – The loss function is $f_t(P) = -\ln(\langle P, r_t\rangle)$
3. Aggregating weather prediction:
   – The action space is $\Omega = \Delta^{N-1}$
   – The loss function is $f_t(P) = \langle P, l_t\rangle$.
4. Production recommendation (multi-armed bandits):
   – The decision space is the discrete set $\Omega = \{1, ..., k\}$
   – The loss function is $f_t(a) = l_t(a)$
5. Personalized product recommendation:
   – The action space is $\Omega = \Delta^{N-1}$.
   – The loss function is $f_t(w) = \mathbb{E}_{i\in w}[l_t(\Pi^i(x_t))] = \sum_{i=1}^N w_i l_t(\Pi^i(x_t))$

For all of these online optimization problems, the performance metric is the cumulative loss. An analog of this to statistical learning is the notion of regret, a comparison of the performance of the algorithm's decision over time with the algorithm's decision in hindsight. Regret is defined as follows:

$$R_T = \sum_{t=1}^{T} f_t(W_t) - \min_{W\in\Omega} \sum_{t=1}^{T} f_t(W)$$

The goal is to achieve a sublinear regret rate ($R_T = o(T)$).

In the next lecture, we will show that if an online learning algorithm can achieve a small R_T guarantee, then the algorithm can be converted to a statistical learning algorithm.