

Statistical Learning Theory: Experts and Bandits

CS4780/5780 – Machine Learning
Fall 2013

Thorsten Joachims
Cornell University

Reading: Mitchell Chapter 7.5

Generalization Error Bound: Infinite H, Non-Zero Error

- Setting
 - Sample of n labeled instances S
 - Learning Algorithm L using a hypothesis space H with $VCDim(H)=d$
 - L returns hypothesis $\hat{h}=L(S)$ with lowest training error
- Definition: The VC-Dimension of H is equal to the maximum number d of examples that can be split into two sets in all 2^d ways using functions from H (shattering).
- Given hypothesis space H with $VCDim(H)$ equal to d and an i.i.d. sample S of size n , with probability $(1-\delta)$ it holds that

$$Err_P(h_{\mathcal{L}(S)}) \leq Err_S(h_{\mathcal{L}(S)}) + \sqrt{\frac{d \left(\ln \left(\frac{2n}{d} \right) + 1 \right) - \ln \left(\frac{\delta}{4} \right)}{n}}$$

Outline

- Online learning
- Review of perceptron and mistake bound
- Expert model
 - Halving Algorithm
 - Weighted Majority Algorithm
 - Exponentiated Gradient Algorithm
- Bandit model
 - EXP3 Algorithm

Online Classification Model

– Setting

- Classification
- Hypothesis space H with $h: X \rightarrow Y$
- Measure misclassifications (i.e. zero/one loss)

– Interaction Model

- Initialize hypothesis $h \in H$
- FOR t from 1 to T
 - Receive x_t
 - Make prediction $\hat{y}_t = h(x_t)$
 - Receive true label y_t
 - Record if prediction was correct (e.g., $\hat{y}_t = y_t$)
 - Update h

(Online) Perceptron Algorithm

- Input: $S = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$, $\vec{x}_i \in \mathbb{R}^N$, $y_i \in \{-1, 1\}$
- Algorithm:
 - $\vec{w}_0 = \vec{0}$, $k = 0$
 - FOR $i=1$ TO n
 - * IF $y_i(\vec{w}_k \cdot \vec{x}_i) \leq 0$ ### makes mistake
 - $\vec{w}_{k+1} = \vec{w}_k + y_i \vec{x}_i$
 - $k = k + 1$
 - * ENDIF
 - ENDFOR
- Output: \vec{w}_k

Perceptron Mistake Bound

Theorem: For any sequence of training examples $S = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$ with

$$R = \max \|\vec{x}_i\| ,$$

if there exists a weight vector \vec{w}_{opt} with $\|\vec{w}_{opt}\| = 1$ and

$$y_i (\vec{w}_{opt} \cdot \vec{x}_i) \geq \delta$$

for all $1 \leq i \leq n$, then the Perceptron makes at most

$$\frac{R^2}{\delta^2}$$

errors.

Expert Learning Model

- Setting
 - N experts named $H = \{h_1, \dots, h_N\}$
 - Each expert h_i takes an action $y = h_i(x_t)$ in each round t and incurs loss $\Delta_{t,i}$
 - Algorithm can select which expert's action to follow in each round
- Interaction Model
 - FOR t from 1 to T
 - Algorithm selects expert h_{i_t} according to strategy A_{w_t} and follows its action y
 - Experts incur losses $\Delta_{t,1} \dots \Delta_{t,N}$
 - Algorithm incurs loss Δ_{t,i_t}
 - Algorithm updates w_t to w_{t+1} based on $\Delta_{t,1} \dots \Delta_{t,N}$

Halving Algorithm

- Setting
 - N experts named $H = \{h_1, \dots, h_N\}$
 - Binary actions $y = \{+1, -1\}$ given input x , zero/one loss
 - Perfect expert exists in H
- Algorithm
 - $VS_1 = H$
 - FOR $t = 1$ TO T
 - Predict the same y as majority of $h_i \in VS_t$
 - $VS_{t+1} = VS_t$ minus those $h_i \in VS_t$ that were wrong
- Mistake Bound
 - How many mistakes can the Halving algorithm make before predicting perfectly?

Weighted Majority Algorithm

- Setting
 - N experts named $H = \{h_1, \dots, h_N\}$
 - Binary actions $y = \{+1, -1\}$ given input x , zero/one loss
 - There may be no expert in H that acts perfectly
- Algorithm
 - Initialize $w_1 = (1, 1, \dots, 1)$
 - FOR $t = 1$ TO T
 - Predict the same y as majority of $h_i \in H$, each weighted by $w_{t,i}$
 - FOREACH $h_i \in H$
 - IF h_i incorrect THEN $w_{t+1,i} = w_{t,i} * \beta$
 - ELSE $w_{t+1,i} = w_{t,i}$
- Mistake Bound
 - How close is the number of mistakes the Weighted Majority Algorithm makes to the number of mistakes of the best expert in hindsight?

Regret

- Idea
 - Compare performance to best expert in hindsight
- Regret
 - Expected loss of algorithm A_w at time t is

$$E_{A_w}[\Delta_{t,i}] = w_t \Delta_t$$

for randomized algorithm that picks recommendation of expert i at time t with probability $w_{t,i}$

- Overall loss of best expert i^* in hindsight is

$$\sum_{t=1}^T \Delta_{t,i^*}$$

- Regret is difference between expected loss of algorithm and best fixed expert in hindsight

$$\text{Regret}(T) = \sum_{t=1}^T w_t \Delta_t - \min_{i^* \in [1..N]} \sum_{t=1}^T \Delta_{t,i^*}$$

Exponentiated Gradient Algorithm for Expert Setting (EG)

- Setting
 - N experts named $H = \{h_1, \dots, h_N\}$
 - Any actions, any loss function
 - There may be no expert in H that acts perfectly
- Algorithm
 - Initialize $w_1 = \left(\frac{1}{N}, \dots, \frac{1}{N}\right)$
 - FOR t from 1 to T
 - Algorithm randomly picks i_t from $P(I_t = i_t) = w_{t,i}$
 - Experts incur losses $\Delta_{t,1} \dots \Delta_{t,N}$
 - Algorithm incurs loss Δ_{t,i_t}
 - Algorithm updates w for all experts i as
$$\forall i, w_{t+1,i} = w_{t,i} \exp(-\eta \Delta_{t,i})$$
Then normalize w_{t+1} so that $\sum_j w_{t+1,j} = 1$.

Regret Bound for Exponentiated Gradient Algorithm

- Theorem

The regret of the exponentiated gradient algorithm in the expert setting is bounded by

$$\text{Regret}(T) \leq \Delta \sqrt{2T \log(N)}$$

where $\Delta = \max\{\Delta_{t,i}\}$ and $\eta = \frac{\sqrt{\log(N)}}{\Delta \sqrt{2T}}$.

Bandit Learning Model

- Setting
 - N bandits named $H = \{h_1, \dots, h_N\}$
 - Each bandit h_i takes an action in each round t and incurs loss $\Delta_{t,i}$
 - Algorithm can select which bandit's action to follow in each round
- Interaction Model
 - FOR t from 1 to T
 - Algorithm selects expert h_{i_t} according to strategy A_{w_t} and follows its action y
 - Bandits incur losses $\Delta_{t,1} \dots \Delta_{t,N}$
 - Algorithm incurs loss Δ_{t,i_t}
 - Algorithm updates w_t to w_{t+1} based on Δ_{t,i_t}

Key difference compared to Expert Model

Exponentiated Gradient Algorithm for Bandit Setting (EXP3)

- Initialize $w_1 = \left(\frac{1}{N}, \dots, \frac{1}{N}\right)$, $\gamma = \min \left\{ 1, \sqrt{\frac{N \log N}{(e-1)\Delta T}} \right\}$
 - FOR t from 1 to T
 - Algorithm randomly picks i_t with probability
$$P(i_t) = (1 - \gamma)w_{t,i} + \gamma/N$$
 - Experts incur losses $\Delta_{t,1} \dots \Delta_{t,N}$
 - Algorithm incurs loss Δ_{t,i_t}
 - Algorithm updates w for bandit i_t as
$$w_{t+1,i_t} = w_{t,i_t} \exp\left(-\eta\Delta_{t,i_t}/P(i_t)\right)$$
- Then normalize w_{t+1} so that $\sum_j w_{t+1,j} = 1$.

Other Online Learning Problems

- Stochastic Experts
- Stochastic Bandits
- Online Convex Optimization
- Partial Monitoring