Ch 21 Online Learning, part 2

Friday, March 27, 2020 3:05 PM

These notes: 1) doubling trick
2) online-to-batch conversion
3) online convex opt.
4) perceptron

Aside: the "Doubling Trick"

Given an algorithm that needs to know T in advance, and has an error (or regret) $\sum_{i=1}^{T} \text{loss}_i \le \alpha \cdot \sqrt{T}$

we can do the following: split T into $\log_2(T)$ groups of the following sizes:

1 2 4 8 16 ...

1 3 7 15 31 ... (running sum)

$T \le \sum_{m=0}^{L} 2^m$ $L = \lfloor \log_2(T) \rfloor$ ← floor $\le \log_2(T)$

i.e. $T=7 = \overset{0}{1}+\overset{1}{2}+\overset{2}{4}$ $L=2$

$T=8 = 1+2+4+\text{part-of-8}$, $T=9 = 1+2+4+\text{part-of-8}$ } $L=3$

on each group of size 2^m, run algorithm, so error on this chunk is $\le \alpha \cdot \sqrt{2^m} = \alpha \cdot \sqrt{2}^m$

Recall if $S = \sum_{m=0}^{L} \rho^m = 1+\rho+\dots+\rho^L$

$\rho \cdot S = \rho + \dots + \rho^L + \rho^{L+1}$

so $S - \rho S = 1-\rho^{L+1}$

so $\rho \ne 1 \Rightarrow S = \frac{1-\rho^{L+1}}{1-\rho}$

So, total error is $\le \sum_{m=0}^{L} \alpha \sqrt{2}^m$

$= \alpha \frac{1-\sqrt{2}^{L+1}}{1-\sqrt{2}}$ $\quad \sqrt{2}^L = \sqrt{2^L} \le \sqrt{T}$

$\le \alpha \frac{\sqrt{2}\sqrt{T}-1}{\sqrt{2}-1} \le \alpha \cdot \frac{\sqrt{2}}{\sqrt{2}-1} \cdot \sqrt{T}$ $\quad \frac{\sqrt{2}}{\sqrt{2}-1} \approx 3.4$

so if we knew T in advance, error bound is $\alpha\sqrt{T}$

if we don't know in advance, $3.4 \cdot \alpha\sqrt{T}$ not much worse

Used elsewhere in applied math + computer science

ex linesearches in optimization methods ← mainly theoretical (not good practical performance)

ex growing memory $x = [3,1,4]$ // 3 units of memory allocated

x.append(5) // now we need 4 units of memory, so find 4 free units, copy old memory to new location (slow!) since often want contiguous

better: allocate $2^{\lceil \log_2(d) \rceil}$ at first, then when full, double memory. Fewer copies, only at most 2x sub-optimal

Aside: Online-to-batch Conversion, converting an online learner to a PAC learner

Thm: Usual PAC learning problem (realizable), binary classification

and suppose we have an online algorithm A w/ a mistake bound $M_A(\mathcal{H}) < \infty$

Run $m=T$ iid examples through algo A (labels via $h^* \in \mathcal{H}$)

Let h_t be the hypothesis the online algo. used at round t

Then $\mathop{\mathbb{E}}_{\substack{(x_t)\sim S^m \\ r\sim \text{Unif}([T])}} L_D(h_r) \le \frac{M_A(\mathcal{H})}{T}$ $(T=m)$

not of PAC form, but as we did in ch13 Reg.+Stability, convert to PAC via Markov's ineq (exer. 13.1) proof sketch $\mathbb{E}_{r\sim \mathrm{Uni}([T])} L_D(h_r) = \frac{1}{T}\sum_{t=1}^{T} L_D(h_t)$

so $\mathbb{E}_{\substack{(x_t)\sim S^m \\ r\sim \mathrm{Uni}([T])}} L_D(h_r) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x_t)\sim S^m} L_D(h_t) = \frac{1}{T}\sum_{t=1}^{T} \underbrace{\mathbb{E}_{(x_t)\sim S^m} \mathbb{E}_{x\sim S} \mathbb{1}_{h_t(x)\neq h^*(x)}}$

h_t depends only on $(x_1,\dots,x_{t-1})$

$x_t \sim S$

So $\mathbb{E}_{x\sim S} \mathbb{1}_{h_t(x)\neq h^*(x)} = \mathbb{E}_{x_t\sim S} \mathbb{1}_{h_t(x_t)\neq y_t}$

$\dots = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x_{t'})\sim S^m} \mathbb{1}_{h_t(x_t)\neq y_t}$

i.e. $\mathbb{E}_{(x_{t'})\sim S^m} (\dots)$ ↖ only 1^{st} t matters

$= \mathbb{E}_{(x_{t'})\sim S^{t-1}} \mathbb{E}(\dots \mid (x_{t'})\sim S^{t-1})$ law of total expectation

$\leq \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}_{h_t(x_t)\neq y_t}$ x_t adversarial

$\leq \frac{1}{T} M_A(\mathcal{H})$ □

Corollary

If the loss function is convex (eg, parameterize $h\in\mathcal{H}$ by $w\in\mathcal{H}$, and $\forall z\in X\times Y$, $w\mapsto \ell(w,z)$ is convex)

then can choose $h = \frac{1}{T}\sum h_t$ (instead of $h=h_r$, $r\sim \mathrm{Uni}([T])$) and have the same bound

proof: Jensen's Ineq.

See "Online Learning and Online Convex Optimization", Shalev-Shwartz, Foundations and Trends in Machine Learning, 2011

(2) Online Convex Optimization

Setup is nearly the same, but instead of 0-1 loss $|h(x)-y|$, now allow a general loss $\ell: \mathcal{H}\times Z \to \mathbb{R}$, $Z = X\times Y$, parameterize h as $w\in\mathcal{H}$,

assume: 1) $\mathcal{H}$ is a convex set (ex: $\mathcal{H} = \{w : \|w\|\leq B\}$)

2) $\forall z$, $w\mapsto \ell(w,z)$ is a convex function

i.e., regression or binary classif. w/ a convex surrogate loss

Analyze the regret:

Def: $\mathrm{Regret}_A(w^*, T) := \sum_{t=1}^{T} \ell(w^{(t)}, z_t) - \sum_{t=1}^{T} \ell(w^*, z_t)$

Def: $\mathrm{Regret}_A(\mathcal{H}, T) := \sup_{w^*\in\mathcal{H}} \mathrm{Regret}_A(w^*, T)$

$$= \sum_{t=1}^{T} \ell(w^{(t)}, z_t) - \min_{w^*\in\mathcal{H}} \sum_{t=1}^{T} \ell(w^*, z_t)$$

So, minimize $\sum_{t=1}^{T} \ell(w^{(t)}, z_t)$] online cvx opt (ch 21) ← training = testing

w can change per iteration

z_t adversarial, choose w_t before we see y_t

vs $\min \sum_{t=1}^{T} \ell(w, z_t)$] "batch" cvx opt (ch 14) ← this is training. Then later test

Fix w, z_t iid, all labels known,

focus on generalization

(Different rules than ch. 14)

Algo: "Online sub Gradient Descent"

Stepsize $\eta > 0$

Init. $w^{(1)} = 0$ (if $0 \in \mathcal{H}$)

for $t = 1, 2, \dots, T$

Our "prediction" is $w^{(t)}$

Observe $z_t = (x_t, y_t)$

Loss function is $f_t(\cdot) = \ell(\cdot, z_t)$

So suffer loss $f_t(w^{(t)})$

Choose $v_t \in \partial f_t(w^{(t)})$ // subgradient

Update

$$w^{(t+1)} = \mathrm{Proj}_{\mathcal{H}}(w^{(t)} - \eta v_t)$$

$\mathrm{Proj}_C(v) = \underset{w \in C}{\mathrm{argmin}} \|w - v\| = \underset{w \in C}{\mathrm{argmin}} \frac{1}{2}\|w - v\|^2$

Thm (online [projected sub-]gradient descent (OGD)) and assuming convexity

Choosing $w^{(t)}$ as in OGD, (i.e. $A = $ OGD)

① $\forall w^* \in \mathcal{H}$, $\mathrm{Regret}_{OGD}(w^*, T) \le \dfrac{\|w^*\|^2}{2\eta} + \dfrac{\eta}{2}\sum_{t=1}^{T}\|v_t\|^2$

So ...

② if f_t is ρ-Lipschitz $\forall t \in [T]$ then w/ $\eta = 1/\sqrt{T}$ i.e. $\|v_t\| \le \rho$

$\mathrm{Regret}_{OGD}(w^*, T) \le \frac{1}{2}\sqrt{T}\cdot(\|w^*\|^2 + \rho^2)$

sublinear regret!

③ and if we also assume $\mathcal{H}$ is B-bounded, then w/ $\eta = \dfrac{B}{\rho\sqrt{T}}$

$\mathrm{Regret}_{OGD}(w^*, T) \le B\rho\sqrt{T}$

compare to Cor 14.12 in batch PAC learning case. Same η, same bound, different interpretation

(if T unknown, then η unknown, so do doubling trick)

Proof sketch: see book, not very enlightening

③ Online Perceptron

For homogeneous half-spaces $\mathcal{H} = \{x \mapsto \mathrm{sign}(\langle w, x\rangle)\}$

+ binary classification

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$

Ch 9: batch version

Ch 21: online version

... but we saw $\mathrm{Ldim}(\mathcal{H}) = \infty$ if $d \ge 2$ (ie. 1D + offset)

So a mistake bound not possible

... instead, use a surrogate convex loss

Because we're not trying to generalize, we can change the surrogate loss to depend on the round t (including on z_t and $w^{(t)}$), as long as it's a surrogate, ie.,

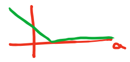
$$\underbrace{f_t}_{\text{surrogate loss}}(w^{(t)}) \ge \underbrace{\ell}_{\text{true loss, eg, 0-1 loss}}(w^{(t)}, z_t)$$

So that we bound the risk.

Use true loss is 0-1 loss, $\ell(w, (x, y)) = \mathbb{1}_{[y\langle w, x\rangle \le 0]} = \begin{cases} 1 & y \ne \mathrm{sign}(\langle w, x\rangle) \\ 0 & \text{else} \end{cases}$

Surrogate is

$$f_t(w) = \begin{cases} 0 & \text{when we had correct prediction} \\ \underbrace{\max(0, 1 - \underbrace{y_t \langle w, x_t \rangle}_{a})}_{\text{hinge-loss}(a)} & \text{if we got it wrong} \end{cases}$$



Algo: "Perceptron" Rosenblatt '58

$w_1 = 0$ no stepsize!

for $t = 1, 2, \dots, T$

    receive x_t

    predict $p_t = \text{sign}(\langle w^{(t)}, x_t \rangle)$

    if $y_t \langle w^{(t)}, x_t \rangle \le 0$ // we got it wrong

        $w^{(t+1)} = w^{(t)} + y_t x_t$

    else   // we got it right

        $w^{(t+1)} = w^{(t)}$ // no change

See Thm 21.16 for analysis