Aalto University
School of Science

CS-E4070 — Computational learning theory

Slide set 05 : weak and strong learning

Cigdem Aslay and Aris Gionis

Aalto University

spring 2019

reading material

- K&V, chapter 4

what we have seen so far

- **strong learning**: an algorithm A is a strong learner of a concept class $\mathcal{C}$, if for every concept $c \in \mathcal{C}$, every distribution $\mathcal{D}$, and every $\epsilon > 0$ and $\delta \in (0, 1)$, the algorithm A outputs a hypothesis $h \in \mathcal{C}$ that satisfies

$$error_{\mathcal{D}}(h) \leq \epsilon$$

with probability at least $1 - \delta$.

interesting to consider

- **weak learning**: an algorithm A is a weak learner of a concept class $\mathcal{C}$, if there exists a fixed ϵ_0 and δ_0, such that for every concept $c \in \mathcal{C}$ and every distribution $\mathcal{D}$, the algorithm A outputs a hypothesis $h \in \mathcal{C}$ that satisfies

$$error_{\mathcal{D}}(h) \leq \epsilon_0$$

  with probability at least $1 - \delta_0$.

- in other words, ϵ_0 and δ_0 are fixed, and not arbitrarily small

weak learning

- learner can be just marginally better than random
- **weak learning**: an algorithm A is a weak learner of a concept class $\mathcal{C}$, if there exists γ and τ, both greater than $1/poly(n)$, such that for every concept $c \in \mathcal{C}$ and every distribution $\mathcal{D}$, the algorithm A outputs a hypothesis $h \in \mathcal{C}$ that satisfies
$$error_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$$
with probability at least τ.
- in other words, A has a non-negligible chance of doing non-negligably better than random guessing

weak learning

- the requirement for weak learning is indeed very weak
- for instance, it is typically trivial to learn a concept class $\mathcal{C}$ with accuracy

$$error_{\mathcal{D}}(h) \leq \frac{1}{2} - \frac{1}{e(n)}$$

  where $e(n)$ is an exponentially-increasing function

- how?

  return the correct answer for instances in the training set (which has exponentially small size) and a random answer for all other instances

a surprising result

- **theorem** : if a concept class $\mathcal{C}$ is efficiently weak PAC learnable, then $\mathcal{C}$ is efficiently strong PAC learnable

turning weak learning to strong learning

proof idea

- transform a weak learner A_w to a strong learner A_s
- assume fixed parameters ϵ_0 and δ_0
- for desired accuracy and confidence parameters ϵ and δ
  show how to construct A_s from A_w
- construction should be polynomial in $1/\epsilon$ and $1/\delta$

two parts; we will show separately

- how to boost confidence δ_0 to δ (easy)
- how to boost accuracy ϵ_0 to ϵ (difficult)

warm up exercise on boosting

- consider a randomized algorithm A for a problem P
- assume that the answer to P is binary
- assume that for a given problem instance I, the algorithm A returns the correct answer for I with probabity greater than $\frac{1}{2} + \epsilon$, for some ϵ
  - i.e., A does only slightly better than random guessing

task : design an algorithm A' s.t., for any instance I, A' returns the correct answer with probability at least $1 - \delta$, for any δ

warm up exercise on boosting

answer

- repeat A on I for a total of m times
- return the majority answer

analysis

- how large should m be?
- how to analyze?

hint : apply the Chernoff bound

- extremely useful tool for tail inequalities
- many applications in analysis of randomized algorithms, machine learning, etc.
- there are many variants; useful in different scenarios

Chernoff bound (additive form known as Hoeffding bound)

- let $X_1, \ldots, X_m$ by m independent Bernoulli trials,
  with probability of success $E[X_i] = p$
  let $S = X_1 + \ldots + X_m$, then $E[S] = pm$
  then, for any $0 \leq \gamma \leq 1$ we have

$$\mathbf{Pr}[S > (p + \gamma)m] \leq e^{-2m\gamma^2}$$

and

$$\mathbf{Pr}[S < (p - \gamma)m] \leq e^{-2m\gamma^2}$$

part 1 : boosting the confidence

- suppose that a learner A_w outputs a hypothesis h, such that $error_{\mathcal{D}}(h) \leq \epsilon$ with probability at least δ_0, for any ϵ and $\delta_0 \geq 1/poly(n)$
- we want to achieve confidence $1 - \delta$, for any $\delta > 0$

constructing a strong learner

- simulate A_w a total of k times (k to be determined) each time by drawing new samples from $EX(c, \mathcal{D})$
- find k hypotheses $h_1, \ldots, h_k$
- probability all k hypotheses have error $> \epsilon$ is $\leq (1 - \delta_0)^k$
- set $(1 - \delta_0)^k \leq \delta/2$, or equivalently $k \geq (1/\delta_0) \ln(2/\delta)$
- for such k at least one hypothesis has error less than ϵ

boosting the confidence (cont'd)

- one hypothesis of $h_1, \ldots, h_k$ has error less than ϵ with probability at least $1 - \delta/2$
  - we want to find which one
- draw a *"large enough"* sample S using $EX(c, \mathcal{D})$
- output the hypothesis h_i that makes less mistakes on S
- let $m = |S|$
  - how large should m be?
- consider any h_j with error $error(h_j)$
- we want to bound by $\delta/2k$ the probability that h_j's error on S is greater than $error(h_j) + \gamma$
- by Chernoff bound it suffices to take $m \geq (c_0/\gamma^2)\ln(2k/\delta)$

boosting the confidence (cont'd)

- for each h_j, the probability that h_j's error on S is greater than $error(h_j) + \gamma$ is bounded by $\delta/2k$

(A) by the union bound, the probability that any of the k hypotheses deviates its error by more than γ is bounded by $k(\delta/2k) = \delta/2$

(B) recall that with probability at least $1 - \delta/2$ there is a hypothesis having error less than ϵ

- putting (A) and (B) together, we can find a hypothesis h_i having error at most $\epsilon + \gamma$
- and the failure probability (applying union bound again) is bounded by $\delta/2 + \delta/2 = \delta$
- to achieve error ϵ', set $\epsilon = \epsilon'/2$ and $\gamma = \epsilon'/2$

boosting the confidence (algorithm recap)

constructing a strong learner A_s from a weak learner A_w

1. simulate A_w a total of $k \geq (1/\delta_0)\ln(2/\delta)$ times
   - find k hypotheses $h_1, \ldots, h_k$
2. draw a sample S of size $|S| = m \geq (c_0/\epsilon^2)\ln(2k/\delta)$
3. output the hypothesis h_i that makes less mistakes on S

- note that the strong learner A_s makes a polynomial (in $1/\epsilon$ and $1/\delta$) number of calls to the weak learner A_w (which is assumed polynomial)

part 2 : boosting the accuracy

- suppose that a learner A_w outputs a hypothesis h, such that $error_{\mathcal{D}}(h) \leq \beta$ with probability at least $1 - \delta$, for a fixed $\beta < 1/2$ and any $\delta > 0$
- we want to achieve accuracy ϵ, for any $\epsilon > 0$

- it seems to be an almost impossible task
- learner may always return a hypothesis with large error
- not clear how repeated runs can help to boost accuracy

boosting the accuracy

high-level idea

- take advantage of the fact that the learner A_w can find a hypothesis with large error β, but can do so for any input distribution
- run A_w not only on the target distribution $\mathcal{D}$, but also on *"regions"* of $\mathcal{D}$ in which the previously-learned hypothesis performs poorly

for instance

- first run A_w on $\mathcal{D}$ and obtain h having error β
- then run A_w on inputs from $\mathcal{D}$ in which h errs
- we hope to learn *"something new"*

boosting the accuracy — a two-step process

step 1

- assume weak learner A_w with guaranteed error β
- build a new learner A that uses A_w as a subroutine and has error $g(\beta)$
- A invokes A_w on three different distributions and learns three hypotheses h_1, h_2, h_3
- learner A forms $h = \text{majority}\{h_1, h_2, h_3\}$
- hypothesis h is guaranteed to have error $g(\beta)$

step 2

- step 1 is repeated in a recursive manner
- overall accuracy is boosted to a desirable level ϵ

boosting the accuracy — step 1

- as usual, c is target concept, and $\mathcal{D}$ target distribution
- weak learner A_w achieves error β on any distribution

1. we invoke learner A_w on instances sampled from $EX(c, \mathcal{D})$ and find hypothesis h_1

– we know that $error_{\mathcal{D}}(h_1) \leq \beta$

2. we create a new distribution $\mathcal{D}_2$ by filtering $\mathcal{D}$ using h_1

– w.p. $1/2$ we draw $(\mathbf{x}, c(\mathbf{x}))$ from $\mathcal{D}$ such that $c(\mathbf{x}) = h_1(\mathbf{x})$

– w.p. $1/2$ we draw $(\mathbf{x}, c(\mathbf{x}))$ from $\mathcal{D}$ such that $c(\mathbf{x}) \neq h_1(\mathbf{x})$

– notice $error_{\mathcal{D}_2}(h_1) = 1/2$, i.e., h_1 on $\mathcal{D}_2$ is random guessing

– invoking A_w on $\mathcal{D}_2$ we get h_2 with error $\beta < 1/2$ (on $\mathcal{D}_2$) i.e., $h_2 \neq h_1$, i.e., h_2 learns *“something new”*

boosting the accuracy — step 1 (cont'd)

3. we create a third distribution $\mathcal{D}_3$ by filtering $\mathcal{D}$ using both h_1 and h_2
   - we sample from $EX(c, \mathcal{D})$ until we find instance $(\mathbf{x}, c(\mathbf{x}))$ for which $h_1(\mathbf{x}) \neq h_1(\mathbf{x})$
   - i.e., $\mathcal{D}_3$ focuses on the region of $\mathcal{D}$ that h_1 and h_2 disagree
   - invoking A_w on $\mathcal{D}_3$ returns h_3
   - h_3 learns *"something new"* for the input instances in which h_1 and h_2 disagree
4. the learning algorithm returns $h = \text{majority}\{h_1, h_2, h_3\}$

boosting the accuracy — step 1 (analysis, sketch)

- define
  $error_{\mathcal{D}_1}(h_1) = \beta_1, \quad error_{\mathcal{D}_2}(h_2) = \beta_2, \quad error_{\mathcal{D}_3}(h_3) = \beta_3$
- we want to show that although $\beta_1, \beta_2, \beta_3$ can be as large as β, the $error_{\mathcal{D}}(h)$ will be significantly smaller than β
- it can be shown that $error_{\mathcal{D}}(h)$ is maximized if $\beta_i = \beta$ (for details see K&V)
- hypothesis h makes two types of errors
  - 1st type error : both h_1 and h_2 make an error
  - 2nd type error : h_1 and h_2 disagree and h_3 makes error

boosting the accuracy — step 1 (analysis, sketch)

- thus,

$$\begin{aligned} error_{\mathcal{D}}(h) &= \mathbf{Pr}_{\mathbf{x}\in\mathcal{D}}[h_1(\mathbf{x}) \neq c(\mathbf{x}) \wedge h_2(\mathbf{x}) \neq c(\mathbf{x})] \\ &\quad + \mathbf{Pr}_{\mathbf{x}\in\mathcal{D}}[h_3(\mathbf{x}) \neq c(\mathbf{x}) \mid h_1(\mathbf{x}) \neq h_2(\mathbf{x})] \\ &\qquad \mathbf{Pr}_{\mathbf{x}\in\mathcal{D}}[h_1(\mathbf{x}) \neq h_2(\mathbf{x})] \\ &= \mathbf{Pr}_{\mathbf{x}\in\mathcal{D}}[h_1(\mathbf{x}) \neq c(\mathbf{x}) \wedge h_2(\mathbf{x}) \neq c(\mathbf{x})] \\ &\quad + \beta_3 \, \mathbf{Pr}_{\mathbf{x}\in\mathcal{D}}[h_1(\mathbf{x}) \neq h_2(\mathbf{x})] \end{aligned}$$

  l.h.s. is maximized when $\beta_3 = \beta$

- with further algebraic derivations (see K&V) we can show

$$error_{\mathcal{D}}(h) \leq 3\beta^2 - 2\beta^3$$

  and thus, $g(\beta) = 3\beta^2 - 2\beta^3$, as desired

summary

- a weak learner can be transformed to a strong learner
- confidence can be boosted by iterative runs of the weak learner
- accuracy can be boosted by focusing on regions of the target distribution that are more difficult to learn
  - two step process:
    1. reduce error quadratically
    2. recursive application of 1. to reduce error to ϵ

- analysis is quite involved (in particular the recursive part)
- algorithm is not practical

- can we design a practical boosting algorithm?
  yes! AdaBoost