Introduction to Machine Learning

Online Learning and Mistake Bounds

Varun Chandola

February 14, 2017

Outline

Contents

1 Analysis of Machine Learning Problems

- Why?
  - To understand the complexity of problem classes. For instance, one might be interested in finding out if conjunctive concept class is tougher than the disjunctive concept class.
  - To get bounds on the best or worst performance one could expect from any algorithm for a given concept class.
- How to measure complexity?
  - Size of training data needed to learn
  - Resources taken to learn (CPU cycles or memory)
  - **Number of mistakes made (for online learning only)**

2 Online Learning

- $X = \{true, false\}^d$
- $D = X^{(1)}, X^{(2)}, \ldots$
- $D \subseteq X$
- $c \in \mathcal{C}, c : X \rightarrow \{0, 1\}$

1: **for** $i = 1, 2, \ldots$ **do**
2: Learner given $x^{(i)} \in X$
3: Learner predicts $c_*(x^{(i)})$
4: Learner is told $c(x^{(i)})$
5: **end for**

Learning Objective
"Discover" c with minimum number of prediction mistakes

While the objective of the learner is to learn the target concept while making as few mistakes as possible, we are also interested in estimating the bounds on the number of mistakes for learning the concept.

This can also be treated as the quality of the learning algorithm. The algorithm's learning behavior is measured by counting the number of mistakes it makes while learning a function from a specified class of functions. Obviously, the computational complexity (space and time) is also considered. The idea is to put a lower and upper bound on the number of mistakes.

3 Optimal Mistake Bounds for a Concept Class

- $\mathcal{L}$ - Learning algorithm
- c - Target concept ($c \in \mathcal{C}$)
- D - One possible sequence of training examples
- $M_{\mathcal{L}}(c, D)$ - Number of mistakes made by $\mathcal{L}$ to learn c with D examples
- $M_{\mathcal{L}}(c) = \max_{\forall D} M_{\mathcal{L}}(c, D)$
- Worst case scenario for $\mathcal{L}$ in learning c
- $M_{\mathcal{L}}(\mathcal{C}) = \max_{\forall c \in \mathcal{C}} M_{\mathcal{L}}(c)$
- Worst case scenario for $\mathcal{L}$ in learning any concept in $\mathcal{C}$

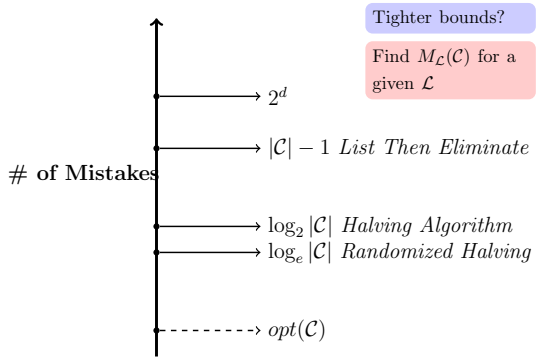
Optimal Mistake Bound
$opt(\mathcal{C}) = \min_{\forall \mathcal{L}} M_{\mathcal{L}}(\mathcal{C})$

Note that the optimal mistake bound is independent of the training examples and the algorithm and even the actual concept to be learnt. It essentially puts a upper bound on the performance of any algorithm that is designed to learn a concept belonging to $\mathcal{C}$.

In other words, $opt(\mathcal{C})$ is the number of mistakes made by the best algorithm to learn the hardest concept in $\mathcal{C}$ using the hardest sequence of training data. It defines the complexity of $\mathcal{C}$.

3.1 Bounds on the Optimal Mistake Bound

It is clear that exact estimation of $opt(\mathcal{C})$ is not possible as it requires knowledge of all possible learners. But one can definitely impose bounds on this bound. Obvious bounds are 2^d, the maximum possible number of examples and $\mathcal{C} - 1$ which is obtained by a learner which "tries" every possible concept and eliminates them one by one.



For instance, for the space of conjunctive concepts, we have already seen that there exists an algorithm that can learn any concept with $d + 1$ mistakes. In general, one can prove that $opt(\mathcal{C}) \leq \log(|\mathcal{C}|)$ using the *Halving Algorithm*.

4 Analyzing Online Learning Algorithms

4.1 Halving Algorithm

- $\xi_0(V, x) = \{c \in V : c(x) = 0\}$
- $\xi_1(V, x) = \{c \in V : c(x) = 1\}$

$V_0 \leftarrow \mathcal{C}$
for $i = 1, 2, \ldots$ **do**
  if $|\xi_1(V_{i-1}, x^{(i)})| \geq |\xi_0(V_{i-1}, x^{(i)})|$ **then**
    predict $c(x^{(i)}) = 1$
  else
    predict $c(x^{(i)}) = 0$
  end if
  if $c(x^{(i)}) \neq c_*(x^{(i)})$ **then**
    $V_i = V_{i-1} - \xi_{c(x^{(i)})}(\mathcal{C}, x^{(i)})$
  end if
end for

- Every mistake results in halving of the version space

- Not computationally feasible
  - Need to store and access the version space
- Are there any *efficient implementable* learning algorithms
  - With comparable mistake bounds

5 Learning Monotone Disjunctions

- A restricted concept class: **monotone disjunctions** of at most k variables
  - $\mathcal{C} = \{x_{i_1} \vee x_{i_2} \vee \ldots x_{i_k}\}$
  - $|\mathcal{C}|$?
  - Mistake bound $= \log_2 |\mathcal{C}|$
- An efficient algorithm - *Winnow*

What are monotone disjunctions? These are disjunctive expressions in which no entry appears negated.

The size of the concept space can be determined by counting the number possible concepts with $0, 1, \ldots, k$ variables:

$$|\mathcal{C}| = \binom{d}{k} + \binom{d}{k-1} + \ldots + \binom{d}{0}$$

Note that $\log_2 |\mathcal{C}| = \Theta(k \log_2 d)$

5.1 Linearly Separable Concepts

- Concept c is linearly separable if $\exists w \in \mathbb{R}^d, \Theta \in \mathbb{R}$ such that:

$$\forall x, c(x) = 1 \Leftrightarrow w^\top x \geq \Theta$$

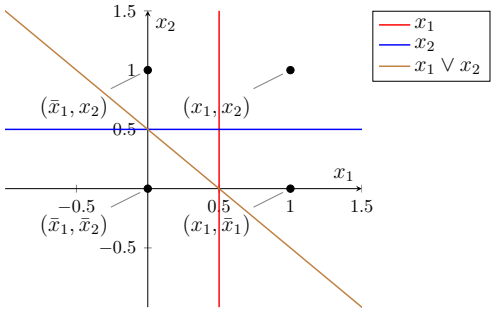
$w^\top x$ denotes the inner product between two vector. Since $x \in \{0,1\}^d$, the inner product is essentially the sum of the weights corresponding to attributes which are 1 for x.

- Monotone disjunctions **are linearly separable**

- For a disjunction $x_{i_1} \vee x_{i_2} \vee \ldots x_{i_k}$

$$x_{i_1} + x_{i_2} + \ldots + x_{i_k} = \frac{1}{2}$$

separates the points labeled 1 and 0 by the disjunctive concept.



5.2 Winnow Algorithm

The name *winnow* comes from the fact that the algorithm finds the k attributes out of a large number of attributes, most of which ($d - k$ to be exact) are not useful.

$\Theta \leftarrow \frac{d}{2}$
$w \leftarrow (1, 1, \ldots, 1)$
for $i = 1, 2, \ldots$ **do**
  if $w^\top x^{(i)} > \Theta$ **then**
    $c_*(x^{(i)}) = 1$
  else
    $c_*(x^{(i)}) = 0$
  end if
  if $c_*(x^{(i)}) \neq c(x^{(i)})$ **then**
    if $c(x^{(i)}) = 1$ **then**
      $\forall j : x_j^{(i)} = 1, w_j \leftarrow 2w_j$
    else
      $\forall j : x_j^{(i)} = 1, w_j \leftarrow 0$
    end if

15: **end if**
16: **end for**

- *Move* the hyperplane when a mistake is made
- Θ is often set to $\frac{d}{2}$
- *Promotions* and *eliminations*

5.3 Analyzing Winnow

- Winnow makes $O(k \log_2 d)$ mistakes
- Optimal mistake bound
- One can use different values for Θ
- Other variants exist
  - *Arbitrary* disjunctions
  - k-DNF (disjunctive normal forms)
    - $(x_1 \wedge x_2) \vee (x_4) \vee (x_7 \wedge \neg x_3)$

The Winnow algorithm is a type of a linear threshold classifier which divides the input space into two regions using a hyperplane.

References