

Weighted Majority and the Online Learning Approach

Scribe: Mark Desnoyer built upon notes by Andres Rodriguez

1. REVIEW FROM PREVIOUS LECTURE

Note on Bayes Nets:

$$P(A, B) = P(A)P(B|A) = P(B)P(A|B) = \phi(A, B)/z$$

This means that $A \rightarrow B$ is equivalent to $B \rightarrow A$ and $A - -B$.

However, Bayes nets were originally made to encode causality. It is much easier for a doctor to tell the symptoms given a disease, than to tell the disease given the symptoms (since many diseases shared the same symptoms).

2. INDUCTION

Induction is trying to predict the future based on observations in the past. So why does induction work (and more importantly, can we know when it won't work)?

Suppose you're a chicken. For 1000 straight days, the farmer comes up and feeds you. Induction says that on the 1001st day, you're going to get fed. Unfortunately, you could be wrong and end up in the pot. Plus, if you don't go and get food, you'll die anyways in a few days.

There are three positions of if induction works:

1. It doesn't because of the above mentioned chicken case. (David Hume). This is true, but not satisfying or particularly useful
2. The "Goldilocks Position". We live in a world that is just right where induction just happens to work. (Albert Einstein). So we're lucky. Once again, not too satisfying
3. The "No Regret" view. It is true that you cannot say that induction will work for any given situations, but as far as strategies go, it's near optimal. This comes from game theory & CS.

3. LOSS AND REGRET

To formalize the No Regret view, let's define a loss function L where:

$$L(\text{wrong}) = 1$$

$$L(\text{correct}) = 0$$

Therefore, for the optimal play, we want to minimize

$$\sum_t L(p_t) \rightarrow 0$$

In this formulation, you cannot guarantee any performance because you can be arbitrarily bad if the problem is hard to predict, or is adversarial.

So, we'll assume that we have some experts that each predicts either 0 or 1. The experts are arbitrary and could be based on a number of things like

- Always constant
- Markov expert (repeats what it last saw)
- Random
- Somethign complex like it consults the color of the sky
- etc..

So, if we consult N-experts and want to do nearly as well as the best expert, we can define regret as:

$$Regret = \sum [L(p_t) - L(e^*(t))]$$

where e^* is the best expert at time t

Therefore, our goal is to:

$$\lim_{t \rightarrow \infty} \frac{Regret}{t} = 0$$

4. ON-LINE LEARNING ALGORITHMS

4.1. Best in Hindsight

This is the naive algorithm that simply picks the expert that has done the best so far. Unfortunately, this can lead to overfitting. Consider an example where you have two experts. One always predicts 1 the other always predicts 0. If the world alternates, you will get every example wrong.

4.2. Weighted Majority Algorithm

Another approach is to notice that at each timestep, each expert is either right or wrong. If it's wrong, we can eliminate the expert and predict the majority vote. In this situation, we know that for each mistake we make, we throw out at least half the experts, so

$$Regret \leq O(\log_2 N)$$

More formally, this algorithm maintains a list of weights w_1, \dots, w_n (one for each expert x_1, \dots, x_n), and predicts based on a weighted majority vote, penalizing mistakes by multiplying their weight by half.

Algorithm

1. Set all the weights to 1.
2. Predict 1 (rain) if $\sum_{x_i=1} w_i \geq \sum_{x_i=0} w_i$, and 0 otherwise.
3. Penalize experts that are wrong: for all i s.t. x_i made a mistake, $w_i^{t+1} \leftarrow \frac{1}{2}w_i^t$.
4. Goto 2

Analysis of Algorithm The sum of the weights is $w \leq \left(\frac{3}{4}\right)^m n = \left(\frac{4}{3}\right)^{-m} n$, where m is the number of mistakes. The weight of the best expert $w_i^* = \left(\frac{1}{2}\right)^{m^*} = 2^{-m^*} \leq w$. Therefore,

$$2^{-m^*} \leq w \leq \left(\frac{4}{3}\right)^{-m} n.$$

Taking the \log_2 and solving for m gives,

$$m \leq \frac{m^* + \log_2 n}{\log \frac{4}{3}} = 2.41(m^* + \log_2 n)$$

4.3. Randomized Weighted Majority Algorithm

In this algorithm, we view the weights as probabilities, and predict each outcome with probability to its weight. We also penalized each mistake by β instead of just half.

Algorithm

1. Set all the weights to 1.
2. Choose expert i in proportion to w_i .
3. Penalize experts that are wrong: for all i s.t. x_i made a mistake, $w_i^{t+1} \leftarrow \beta w_i^t$.
4. Goto 2.

Analysis The bound is

$$E[m] \leq \frac{m * \ln(1/\beta) + \ln(n)}{1 - \beta}.$$

We want β to be small if we only have a few time steps available and vice-versa.

4.4. General Weighted Majority Algorithm

Algorithm

1. Set all the weights to 1.
2. Predict expert i in proportion to w_i .
3. Receive the correct value y_t .
4. Adjust weights s.t. $w_i^{t+1} \rightarrow w_i^t \exp^{-\epsilon l(i, y_t)} \forall i$, where l is the loss function, and ϵ is the penalizer.
5. Goto 2.

Analysis The bound is

$$E[R] \leq \epsilon \sum l(i^*) + \frac{1}{\epsilon} \ln(n),$$

where R is the regret. If ϵ is large, we learn fast. If ϵ is small, we learn slow but we'll have little regret.

5. INTRO TO NEXT LECTURE

In a convex set, the line between any two points in the set is also in the set. A convex function can be defined s.t.

$$f(ax_1 + bx_2) \leq af(x_1) + bf(x_2),$$

where $a + b = 1$, and $a, b \geq 0$.