Universidad Nacional de Colombia
Departarmento de Matemáticas

Maths for Machine Learning
Learning From Data exercises

Sergio Quiroga Sandoval, Manuel Fernando Valle Amortegui, Luis Ángel Ballen Avellaneda
May 30, 2023

1.2 Suppose that we use a perceptron to detect spam messages. Let's say that each email message is represented by the frequency of occurrence of keywords, and the output is if the message is considered spam.

(a) Can you think of some keywords that will end up with a large positive Weight in the perceptron?
discount, opportunity, winner, free.

(b) How about keywords that will get a negative weight?
Dear, exam, classroom, verification, students

(c) What parameter in the perceptron directly affects how many borderline messages end up being classified as spam?
The parameter that affects the mast the classification of borderline messages is the "b" in the perceptron.

1.3 The weight update rule in 1.3 has the nice interpretation that it moves in the direction of classifying $x(t)$ correctly.

(a) Show that $y(t)W^\top(t)x(t) < 0$.
We know that if $x(t)$ is misclassified by $w(t)$, then:

$$\text{sign}\left(W^\top(t)x(t)\right) \neq \text{sign}(y(t)).$$

Therefore

$$y(t)w^\top(t)x(t) > 0$$

(b) Show that $y(t)w^\top(t+1)x(t) > y(t)w^\top(t)x(t)$

$$\begin{aligned} & y(t)W^\top(t+1)x(t) \\ =& y(t)(w(t)+y(t)x(t))^\top x(t) \\ =& y(t)\left(w^\top(t)+y(t)x^\top(t)\right)x(t) \\ =& y(t)w^\top(t)x(t)+y(t)y(t)x^\top(t)x(t) \end{aligned}$$

Note that $y(t)y(t)x^\top(t)x(t) \geq 0$, then

$$y(t)w^\top(t+1)x(t) > y(t)w^\top(t)x(t).$$

(c) As far as classifying $x(t)$ is concerned, argue that the move from $w(t)$ to $w(t+1)$ is a move 'in the right direction'.

We reed to look at the following cases:

(i) $y(t) > 0$ and $W^\top(t) \times (t) < 0$: $w^\top(t)x(t) \longrightarrow$ increases. because $y(t)w^\top(t)x(t)$ increases each step.

(*ii*) $y(t) < 0$ and $w^\top(t)x(t) > 0$

The increment of $y(t)w^\top(t)x(t)$ implies that $w^\top(t)x(t) \rightarrow$ decreases. so we see that in all the cases the move $w(t)$ to $W(t+1)$ is in the right direction.

1.10 Here is an experiment that illustrates the difference between a single bin and multiple bins. Run a computer simulation for flipping 1.000 fair coins. Flip each coin independently times. Let's focus on 3 coins as follows: c1 is the first coin flipped; C_{rand} is a coin you choose at random; Cmin is the coin that had the minimum frequency of heads (pick the earlier one in case of a tie). Let v_1 , v_{rand} and v_{min} be the fraction of heads you obtain for the respective three coins.

(a) What is μ for the three coins selected?

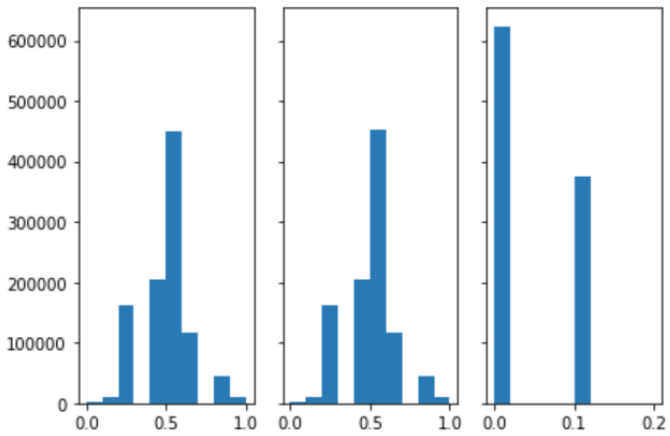
```
total = 1000
flips = 10
run_once(total, flips, True)
```

(b) Repeat this entire experiment a large n umber of times (e.g. , 100.000 runs of the entire experiment) to get several instances of v_1 , v_{rand} and v_{min} and plot the histograms of the distributions of v_1 , v_{rand} and v_{min} · Notice that which coins end up being C_{rand} and C_{min} may differ from one run to another.

```
total_coins = 1000
total_flips = 10
total_runs = 1000000
v1s, vrands, vmins = [],[],[]
for run in range(total_runs):
    v1,vrand,vmin = run_once(total_coins, total_flips)
    v1s.append(v1)
    vrands.append(vrand)
    vmins.append(vmin)

fig, axs = plt.subplots(1,3,sharey=True, tight_layout=True)
```

```
n_bins = 10
axs[0].hist(v1s,bins=n_bins)
axs[1].hist(vrands,bins=n_bins)
axs[2].hist(vmins,bins=n_bins)
```



(c) Using (b), plot estimates for $\mathbb{P}[v-\mu > \epsilon]$ as a function of ϵ, together with the Hoeffding bound $2e^{-2\epsilon^2 N}$ (on the same graph).

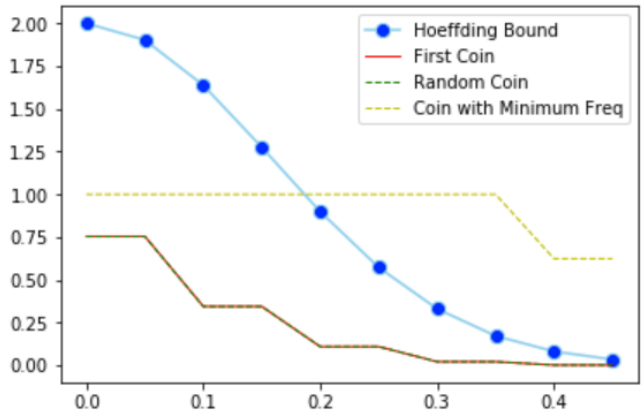
```
    eps = np.arange(0.0,0.5,0.05)
    bounds = hoeffding_bound(eps, total_flips)
    v1s, vrands, vmins = np.array(v1s), np.array(vrands), np.
        array(vmins)
    v1d = np.abs(v1s-0.5)
    vrandd = np.abs(vrands-0.5)
    vmind = np.abs(vmins-0.5)

    p1, prand, pmin = np.zeros(len(eps)),np.zeros(len(eps)),np.
        zeros(len(eps))

    for idx in range(eps.shape[0]):
        ep = eps[idx]
        p1[idx] = np.sum(v1d > ep)/total_runs
        prand[idx] = np.sum(vrandd > ep)/total_runs
        pmin[idx] = np.sum(vmind > ep)/total_runs

    #plt.ylim((0,0.01))
    plt.plot(eps, bounds, marker='o', markerfacecolor='blue',
        markersize=8, color='skyblue', label='Hoeffding Bound')
    plt.plot(eps, p1, marker='', color='r', linewidth=1, label='
        First Coin')
```

```
    plt.plot(eps, prand, marker='', color='g', linewidth=1,
        linestyle='dashed', label='Random Coin')
    plt.plot(eps, pmin, marker='', color='y', linewidth=1,
        linestyle='dashed', label='Coin with Minimum Freq')
    plt.legend()
```



(d) Which coins obey the Hoeffding bound, and which ones do not? Explain why.

The first and random coins follow the Hoeffding bound. The coin with minimum frequency doesn't obey Hoeffding bound. This is because that for the first two coins, the coins were chosen before the experiment. While for the last one, we have to flip all the coins first, and use the data to compute out which is the coin with minimum frequency of heads. This violates the Hoeffding inequality condition which says the hypothesis has been fixed before samples were drawn.

(e) Relate part (d) to the multiple bins in Figure 1.10.

When we choose the coin having the minimum frequency of heads. We are like choosing the bin from 1000 bins (our hypothesis space). But we choose bin after we finish sampling the data. This is akin to learning algorithm for the final hypothesis. The other two coins were chosen before the sampling, which is choosing bin beforehand.

1.11 we are given a dataset D of 25 training examples from an unknown target function $f : \chi \to Y$, where $\chi = \mathbb{R}$ and $y = \{-1, +1\}$. To learn f, we use a simple hypothesis set $H = \{h_1, h_2\}$ where h_1 is the constant $+1$ function and h_2 is the constant -1 . We consider two learning algorithms, S(smart) and C(crazy). S chooses the hypothesis that agrees the most with D and C chooses the other hypothesis is deliberately. Let us see how these algorithms perform out of sample from the deterministic and probabilistic points of view. Assume in the probabilistic view that there is a probability distribution on χ, and let

$$P[f(x) = +1] = P.$$

(a) S produce a hypothesis that is guaranteed to perform better than random on any point outside D? It is not guaranteed that S will perform better than random on any point outside D. Let's see an example:

Define $f : \mathbb{R} \to \{+1, -1\}$

$$f(x) = \begin{cases} +1 & \text{if } x \in (0,1) \\ -1 & \text{else.} \end{cases}$$

In the possible context of having Dc $(0, 1)$ the model S will fit the data by selecting the hypothesis

$$h_1 = +1$$

But when we test it in Random data outside D, for example in random points from the real interval $[1, 2]$ it won't predict well any point Not as the model C that will have a 50/50 chance of good prediction.

(b) Assume for the rest of the exercise that all the examples in D have $Y_n = +1$. Is it possible that the hypothesis that C produces turns out to be better than the hypothesis that S produces?

Yes, the above example is in fact this scenario.

(c) c) if $p = 0.9$ we have the following:

S will choose the hypothesis h_1, so then the chance of producing good predictions or random data will be 90%

C will the model h_2 so h_2 will have only 10% of chance of predicting well a point outside D.

Then S produces a better hypothesis than C.

(d) If $p < 0.5$, then (will predict better than S because is a higher probability of getting a point with -1 label than +1. (knowing that that S chose h_1).

1.12 A friend comes to you with a learning problem. She says the target function is completely unknown, but she has 4, 000 data points. She is willing to pay you to solve her problem and produce for her a g which approximates f. What is the best that you can promise her among the following:

(a) After learning you will provide her with a g that you will guarantee approximates well out of sample.

(b) After learning you will provide her with a g, and with high probability the g which you produce will approximate well out of sample.

(c) One of two things will happen: (i) You will produce a hypothesis g; (ii) You will declare that you failed. If you do return a hypothesis g, then with high probability the g which you produce will approximate well out of sample.

The answer is c) because:

We don't know anything a priori-information about the function f so there is a problem with it's complexity.

If can be so complex that we can't learn using the provided data only, but there also is a possibility the f is not very complex so we can reach a good approximation with g.