Machine Learning from Data CSCI 4100  
Assignment 2

Jae Park (661994900)

**1. Exercise 1.8**

If v ≤ 0.1, it means that either one or zero red marbles can exist. Using binomial distribution, the probability is

10C1 x 0.91 x 0.19 + 10C0 x 0.90 x 0.110 = 9 x 10-9 + 1 x 1 x 10-10 = 9.1 x 10-10

**2. Exercise 1.9**

Given: μ = 0.9, v ≤ 0.1. That means we are looking for ε that is 0.8 < |μ – v|.

P[|μ – v| > 0.8] ≤ 2e−2×(0.8)^2 ×10 = 5.522... x 10-6.

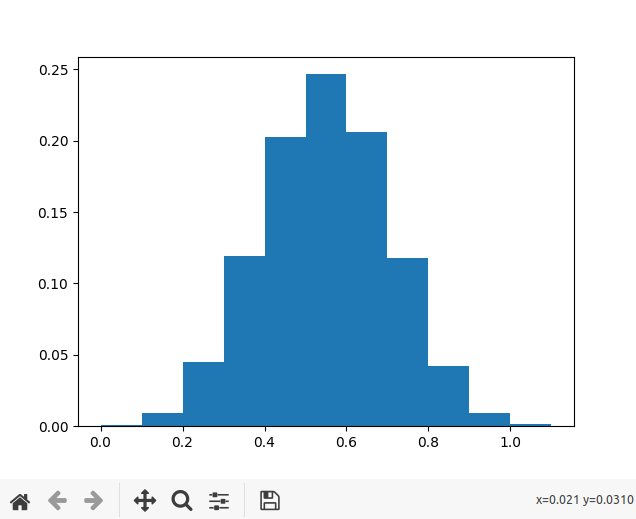
To quantify the relationship between v and μ, we use a simple **upper** bound, the *Hoeffding Inequality*, so it make sense that the result is greater than the actual probability given by 1.8

**3. Exercise 1.10**

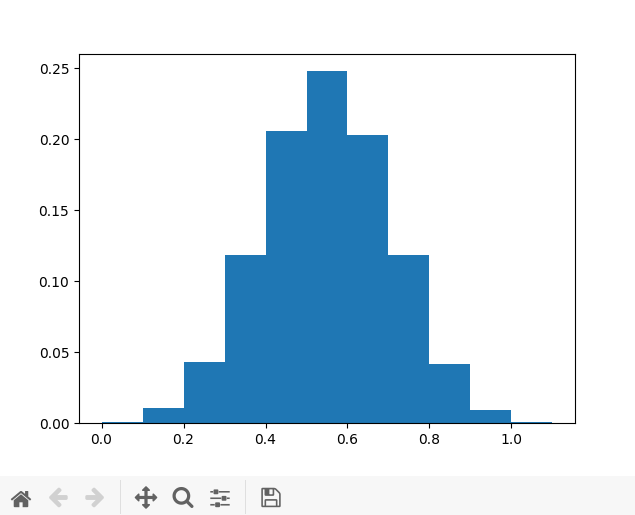
(a) μ for the three coins selected is 0.5 because the probability of getting heads for each coin is 0.5.

(b)

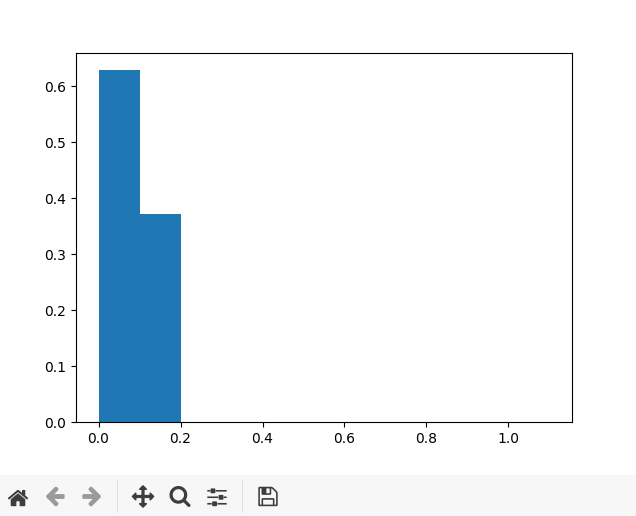
v1



vrand

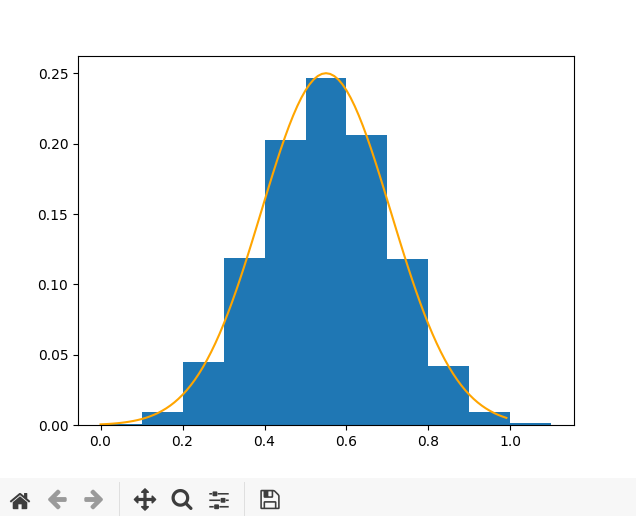


vmin

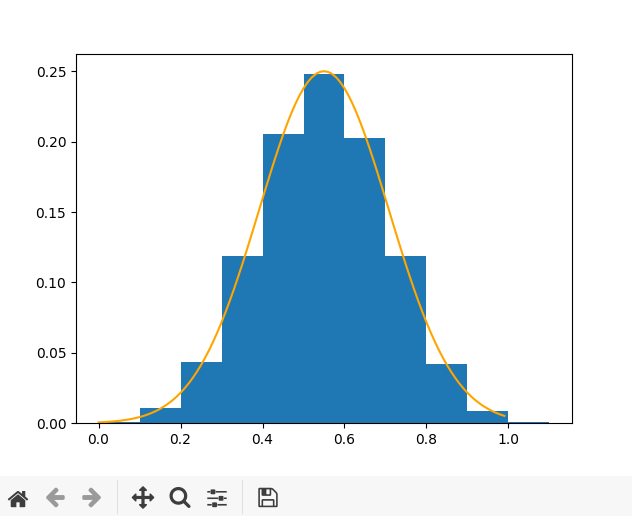


(c)

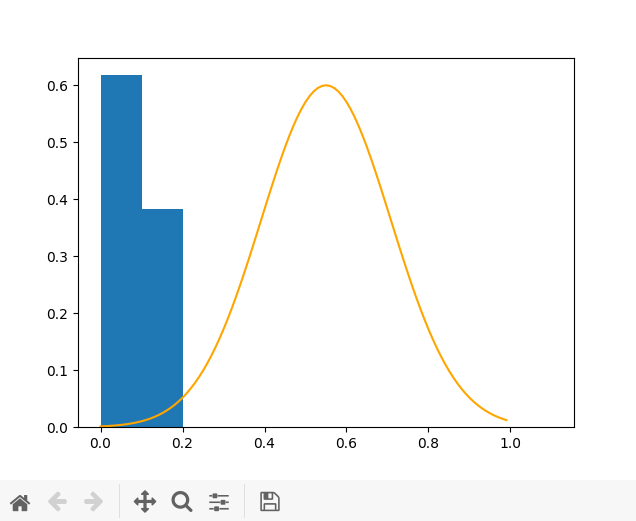
scaled, adjusted distribution of Hoeffding for v1



scaled, adjusted distribution of Hoeffding for vrand



scaled, adjusted distribution of Hoeffding for vmin



(d)

C1 and crand obey the Hoeffding bound, but cmin didn’t. The difference between c1/crand and cmin is that, for cmin, it chooses the minimum frequency of heads, i.e., the coins are not fixed during the experiment. For that matter, c1 and crand had a similar performance.

(e)

For the c1 and crand case, we may say that each coin represents a bin (model) where the hypothesis h is set before the data is generated. However, as the coins are not fixed, cmin cannot be regarded as the bin model.

**4. Exercise 1.11**

(a)

No. As discussed in the lecture, there is no *guarantee* that S will choose a hypothesis that always performs better than the counterpart outside of the training data. It could have been a sheer luck that a hypothesis was chosen such that it performs well within the in-sample data set D, but that was done through the process of verification, not real learning.

(b)

Yes. Suppose an extreme case where p is a very small probability, meaning there is very little +1’s compared to -1’s in Y. And somehow, through random selection, D only contained +1, so our hypothesis S is chosen which says y is always +1. However, we have more -1’s in Y, so from the probabilistic point of view, C will perform much better outside of the training data, which tells us y always -1 (opposite of S).

(c)

In order for S to be considered a better hypothesis than C, it has to be the case that more +1’s are present out of the 25 training examples than the -1’s. So the probability that 13 to 25 +1’s will be present out of 25 sample with p 0.9 is

25C13 x 0.913 x 0.112 + 25C14 x 0.914 x 0.111 + … + 25C25 x 0.925 x 0.10 ≈ 0.9999998379

(d)

No. Assuming that the data set D is “well-chosen” (i.e., IID), it is fair to say that the smart algorithm S will choose a hypothesis that agrees the most with D and will approximate v to μ. Therefore, regardless of what the p value is, S will always choose a better hypothesis between h1 and h2, and C will be left to choose whichever the worse hypothesis is.

**5. Exercise 1.12**

(c) is what we can promise. We cannot guarantee that we will find a hypothesis that achieves Eout(g)≈Ein(g), but at least we will know if we find it (pg.25). If we did find it, we can say that with some probabilistic certainty, g will approximate f outside of the sample.

**6. Problem 1.3**

(a)

Showing min1≤n≤N yn(w\*Txn) > 0 is basically same as saying that all of yn(w\*Txn) is positive for every n, as the minimum of them has to be greater than 0. So the question is asking is yn(w\*Txn) always positive for every iteration n? And the answer is yes because we said w\* is an optimal set of weight that separates the data. This means that yn and w\*Txn will always have the same sign. This makes yn(w\*Txn) to be always positive.

(b)

wT(t)w\* ≥ wT(t-1)w\* + p … (1)

Using the known update rule w(t+1) = w(t) + y(t)x(t),

wT(t-1)w\* + y(t-1)x(t-1)w\* ≥ wT(t-1)w\* + p

and replacing p with the proven above in part (a),

wT(t-1)w\* + y(t-1)x(t-1)w\* ≥ wT(t-1)w\* + min1≤n≤N[yn(t-1)(w\*Txn(t-1)]

Cancel out the equivalent left terms and we get a base case that is,

y(t-1)x(t-1)w\* ≥ min1≤n≤N[yn(t-1)(w\*Txn(t-1)] … (2)

This is certainly true because at iteration *t-1*, or at any iteration as a matter of fact, the min out of 1 ≤ n ≤ N will be smaller than what is on the left hand side of the inequality, in this case y \* x \* w\* at iteration t-1. Given this holds true, we go back to inequality (1) where the form is simpler.

Base case:

wT(t)w\* ≥ wT(t-1)w\* + p

Next step: we look at an iteration before the update, at iteration t-2

wT(t)w\* ≥ wT(t-1)w\* + pt-1 ≥ wT(t-2)w\* + pt-1 + pt-2

where the subscript of p denotes the iteration of update in which p takes place. As demonstrated by inequality (2), we can see that this relationship also hold true because (2) shows that p is of something smaller or equal to than what is actually being updated.

Iterative step:

wT(t)w\* ≥ wT(t-1)w\* + pt-1 ≥ … ≥ wT(0)w\* + t \* p

We know that w(0) = 0, so wT(t)w\* ≥ tp

(c)

Using a simple property (a+b)2 = a2 + 2ab + b2,

||w(t)||2 = ||w(t-1) + y(t-1)x(t-1)||2 = ||w(t-1)||2 + 2 \* (y(t-1)wT(t-1)x(t-1)) + ||y(t-1)x(t-1)||2

We know y(t-1)wT(t-1)x(t-1) ≤ 0 because x(t-1) was misclassified by w(t-1)

Therefore, ||w(t)||2 = ||w(t-1)||2 + 2 \* (negative vector) + ||y(t-1)x(t-1)||≤ ||w(t-1)||2 + ||x(t-1)||2

(d)

Base case: ||w(0)||2 ≤ 0 \* R2 holds true because w(0) = 0.

Inductive step:

Assume step t-1 holds true, that is

||w(t-1)||2 ≤ (t-1)R2 ….(1)

Now, so that it is still true for step t:

From the proof of (c) above, we know that

||w(t)||2 ≤ ||w(t-1)||2 + ||x(t-1)||2 ...(2)

We can combine (1) and (2) by substituting ||w(t-1)||2 appropriately, and we get

||w(t)||2 ≤ (t-1)R2 + ||x(t-1)||2

R = max1≤n≤N||xn||, so ||x(t-1)||2 can be written as R2.

||w(t)||2 ≤ (t-1)R2 + ||x(t-1)||2 = (t-1)R2 + R2 = (t)R2

∴ ||w(t)||2 ≤ tR2

This shows that induction holds true for the tth step.

(e)

wT(t)w\* ≥ wT(t-1)w\* + p

∴ wT(t)w\* - wT(t-1)w\* ≥ p …(1)

||w(t)||2 ≤ tR2 (square root on both sides)

∴ √tR ≤ 0 ≤ ||w(t)|| ≤ √tR, and the zero bound is given because ||w(t)|| is always positive, but we will write it as ||w(t)|| ≤ √tR for the sake of the proof, and this partial inequality is also certainly true.

||w(t)|| ≤ √tR

≥

≥

≥ …(2)

multiply (1) and (2) and we get

(wT(t)w\* - wT(t-1)w\*) x ≥ p \*

rewritten,

≥

We can see that ≥becauseis not negative.

≥≥

∴ ≥

≥ (is not negative since both positive)

≥

use transpose matrix property ≤ ll a ll x ll b ll

≥

≥

≥

**7. Problem 1.7**

(a)

P[one coin of the sample (10) has v = 0] = p = (1 - μ)10

When μ = 0.05, p = 0.59873…

P[at least one coin of 1 coin(s) has v = 0] = 1 – (1 – p) = 0.598…

P[at least one coin of 1,000 coin(s) has v = 0] = 1 – (1 – p)1000 = 1.000…

P[at least one coin of 1,000,000 coin(s) has v = 0] = 1 – (1 – p)1000000 = 1.000…

When μ = 0.8, p = 1.024 x 10-7

P[at least one coin of 1 coin(s) has v = 0] = 1 – (1 – p) = 1.024 x 10-7

P[at least one coin of 1,000 coin(s) has v = 0] = 1 – (1 – p)1000 = 1.024 x 10-4

P[at least one coin of 1,000,000 coin(s) has v = 0] = 1 – (1 – p)1000000 = 0.09733…

(b)

|  |  |  |
| --- | --- | --- |
| i | | v - μ | | P[ |v - μ |] |
| 0 | 1/2 | 0.0156 |
| 1 | 1/3 | 0.0938 |
| 2 | 1/6 | 0.2344 |
| 3 | 0 | 0.3125 |
| 4 | 1/6 | 0.2344 |
| 5 | 1/3 | 0.0938 |
| 6 | 1/2 | 0.0156 |

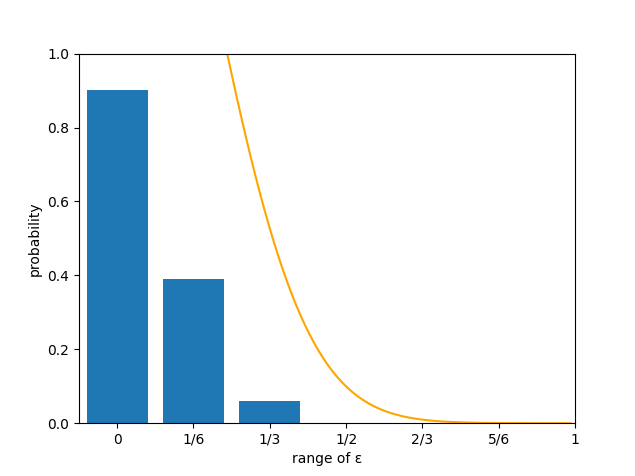
For 0 ≤ ε < 1/6, i = 3, P = 1 – (0.3125)2 = 0.9023….

For 1/6 ≤ ε < 1/3, i = 2 or 3 or 4 , P = 1 – (0.2344+0.3125+0.2344)2 = 0.3896...

For 1/3 ≤ ε < 1/2, i = 1 or 2 or 3 or 4 or 5,

P = 1 – (0.0938+0.2344+0.3125+0.2344+0.0938)2 = 0.0612...

For 1/2 ≤ ε ≤ 1, i = 0 or1 or 2 or 3 or 4 or 5 or 6, P = 1 – (1)2 = 0



**8. Problem 1.8**

(a)

E(t) =

for some number a,

E(t) =

because a =

a \* = a \* P( T a)

P(T a)

(b)

Let t = (u-)2. Note that E(t) = E((u-)2) = Var(u).

|u-| is exactly same as t = (u-)2 2

Therefore, P[||u-| ] = P[t 2].

Y is always non-negative, so applying (a), we get

P[(u-)2 2] = P[t 2] ≤ = =

Reducing 2 to , we get

P[(u-)2 ] ≤

(c)

P[(u-)2 ] = P[t ] ≤ =

P[(u-)2 ] ≤ …(1)

= = = … (2)

plug in …(2) into … (1)

P[(u-)2 ] ≤ = \*

P[(u-)2 ] ≤