Homework #1

姓名：趙愷文 學號：R05222038, PHYS
Machine Learning Foundation (NTU, Fall 2016)

December 14, 2016

Prob. 1 Best suited for machine learning is (ii)
Reason: We can not write down all potential disadvatnages or give precise rules to prevent bad things happen to banks. However, we are able to use historical data and macine learning to figuring out the hidden policy avoiding frauds. Choices (i), (iii) and (iv), we can write down specific formula or algorithm for them. Option (v) should accout for expert's opinon.

Prob. 2 Reinforcement Learning
Machine learns policies by trying different actions and receving penalty or reward from environments. It is the framework of RL.

Prob. 3 Supervised Learning
Because in this senario, books already have their own catalog and group. We can teach machine by 'seeing' enough examples, then machine can do the job.

Prob. 4 Unsupervised Learning
Unsupervised learning algorithm can help automatically grouping the data according to some different intrinsic property. However, if we already have labels telling us face or non-face, supervised learning is also a reasonalbe choice.

Prob. 5 Active Learning
First, biological experiments are expensive, so data and labels are rare. We can do the experiments strategically and machine learns from the most 'valuable' data. So, active learning framework can help us in this case.

Prob. 6 Off-training set error
By the consideration, the hypothesis $g(x)$ can give correct answer on odd sample and fail on even one. So we only sum over odd data, which gives us the number of odd sample in the dataset.

$$E_{OTS}(f,g) = \begin{cases} \frac{1}{L}\frac{L}{2} = \frac{1}{2} & \text{if } L \text{ is even} \\ \frac{1}{L}\frac{L+1}{2} = \frac{L+1}{2L} & \text{if } L \text{ is odd} \end{cases}$$

Prob. 7 Possibles of f out of training set
The possibilities of L OTS examples: 2^L. Each configuration has two choices of y.
Possiblities of f: 2^{2^L}.

Prob. 8 Deterministic A

Prob. 9 Bin Model $\mu = 0.5$
We get 10 blank marbles, then we paint green or orange color on each one following probaility μ. So, we choose 5 out of 10 to be orange and others are green.

$$P(\nu = 0.5) = C_5^{10}(\frac{1}{2})^5(\frac{1}{2})^5 = 0.2461$$

Prob. 10 Bin Model with $\mu = 0.8$
We get 10 blank marbles, then we paint green or orange color on each one following probaility μ. So, we choose 8 out of 10 to be orange and rest 2 are green.

$$P(\nu = 0.8) = C_8^{10}(\frac{8}{10})^8(1 - \frac{8}{10})^2 = 0.3019$$

Prob. 11 Bin Model with $\mu = 0.8$ but $\nu \leq 0.1$
We get 10 blank marbles, then we paint green or orange color on each one following probaility μ. However, we paint only 1 orange or 0 orange marble.

$$P(\text{One orange marble}) + P(\text{None orange marble})$$
$$= C_1^{10}(\frac{8}{10})^1(1 - \frac{8}{10})^9 + (1 - \frac{8}{10})^{10} = 4.1984 \times 10^{-6}$$

It is a pretty small probaility.

Prob. 12 Analyze Bin Model with Hoeffding Inequality $\mu = 0.8$ but $\nu \leq 0.1$
Hoeffding Inequality is formulated as

$$P[|\nu - \mu| < \epsilon] \leq 2\exp(-2\epsilon^2 N)$$

In this problem, our $\epsilon = 0.7$,

$$P[|\nu - \mu| < \epsilon] \leq 2\exp(-2 \times 0.7 \times 10) = 0.0001109 \sim 10^{-3}$$

The inequality gives us a bound stating that the event we discussed is rare.

Prob. 13 Get 5 green dices

Tpye of Dice	Orange	Green
A	2, 4, 6	1, 3, 5
B	1, 3, 5	2, 4, 6
C	1, 2, 3	4, 5, 6
D	4, 5, 6	1, 2, 3

From the above table, we find each time we take a dice, the probaility of getting green 1 is

$$\begin{aligned} P(\text{Green 1}) &= P(\text{A and Green 1}) + P(\text{D and Green 1}) \\ &= (\frac{1}{4}) \times (\frac{1}{6}) + (\frac{1}{4}) \times (\frac{1}{6}) = \frac{1}{12} \end{aligned}$$

Suppose each time we pick a dice is independent. The probability of getting 5 green 1 dice is

$$P(\text{Five Green 1}) = P(\text{Green 1})^5 = (\frac{1}{12})^5 = 4.0187 \times 10^{-06}$$

Prob. 14 5 dices are purely green
No matter which kinds of dice we get, each dice has 1/2 probaility getting green color. So the probability of taking 5 green dices from bag is

$$P(\text{5 Green Dices}) = (\frac{1}{2})^5 = 0.3125$$

Prob. 15 - Naive Cycle
My code shows the result below

```
python hw_1-15.py
Prob 1-15
Initialization method: zero
Number of Updates: 45
Most frequent update example: 58
```

Prob. 16 - Random Cycle

```
python hw_1-16.py
Prob 1-16
Initialization method: zero
We run experiments 2,000 times with random cycle
Average number of updates: 40.217000
```

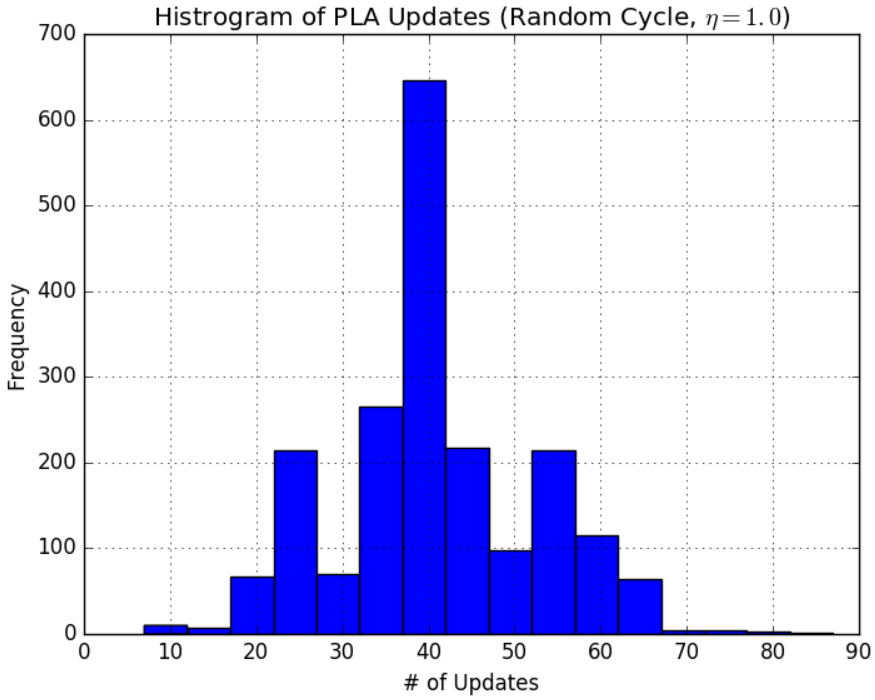

Figure 1: Histogram of Number of Updates - PLA with Random Cycle

Which gives us a little better updates performance.

Prob. 17 - Learning Rate $\eta = 0.25$

```
Prob 1-17
Initialization method: zero
We run experiments 2,000 times with random cycle, eta = 0.25
Average number of updates: 39.448500
```

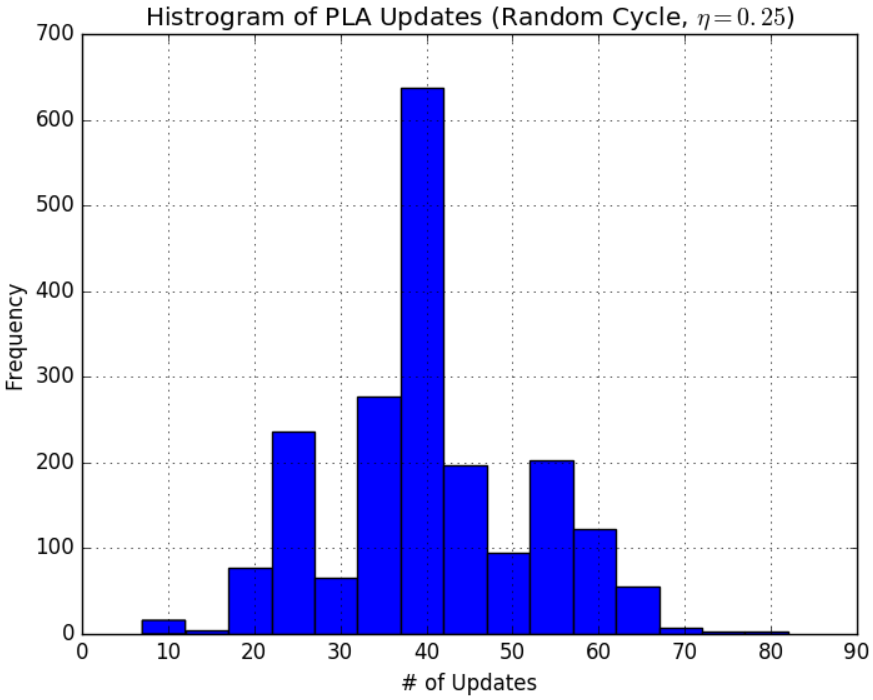

Figure 2: Histogram of Updates Numbers - Random Cycle PLA with learning rate = 0.25

Discussion:
From results above, we can not see any differences from histogram which bothers us. Because we expect smaller learning rate indicates algorithm updates more slowlly, we should see higher number of updates as lowering learning rate.
So, we can do experiments more carefully, and examine it with different weights initialization methods. Here, we use zero init, uniform init and normal distribution init.

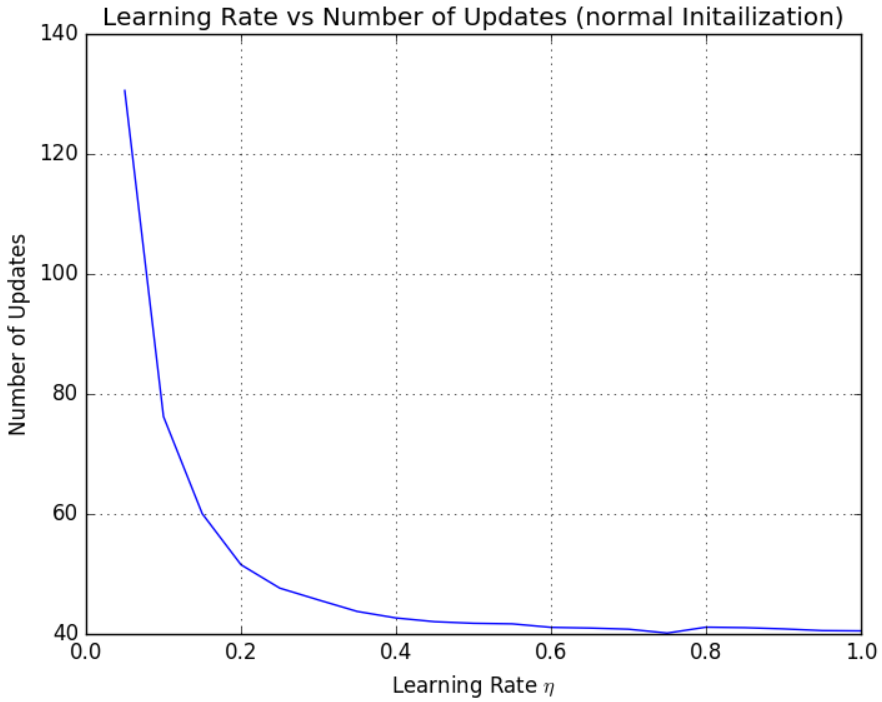

Figure 3: Normal Init: Learning Rate vs Updates

Here we go! This is our expected result. Higher learning rates tunes the 'line' more quickly, that we need less updates to acheive the 'perfect line'. Besides, we find out weights initialization could influence the curve significantly. The normal distribution initialization fits our imagination best. And one interesting observation is that there is a minimum updates number as we use uniform init.

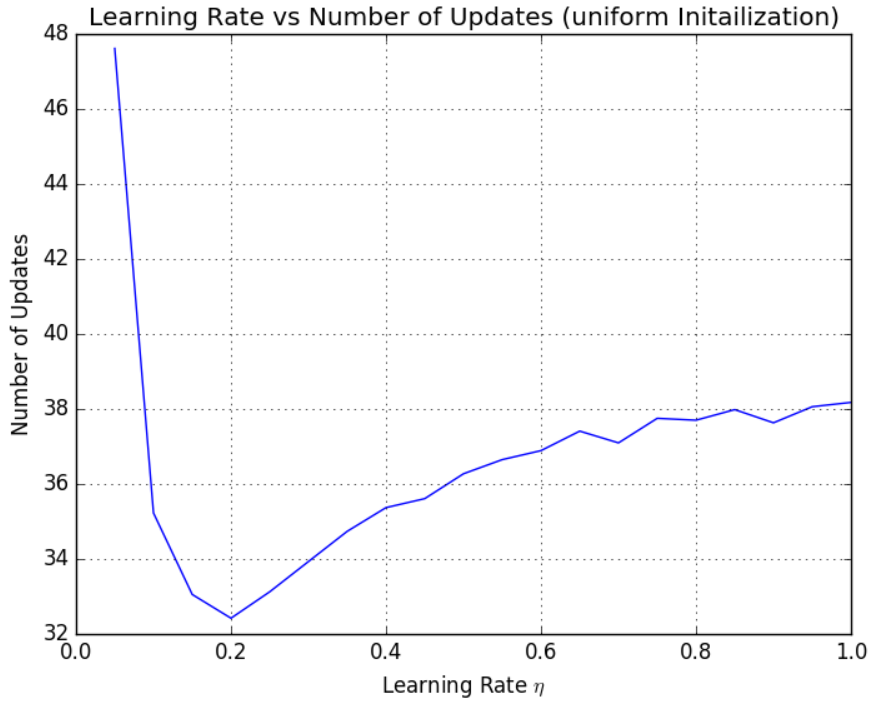

Figure 4: Uniform Init: Learning Rate vs Updates

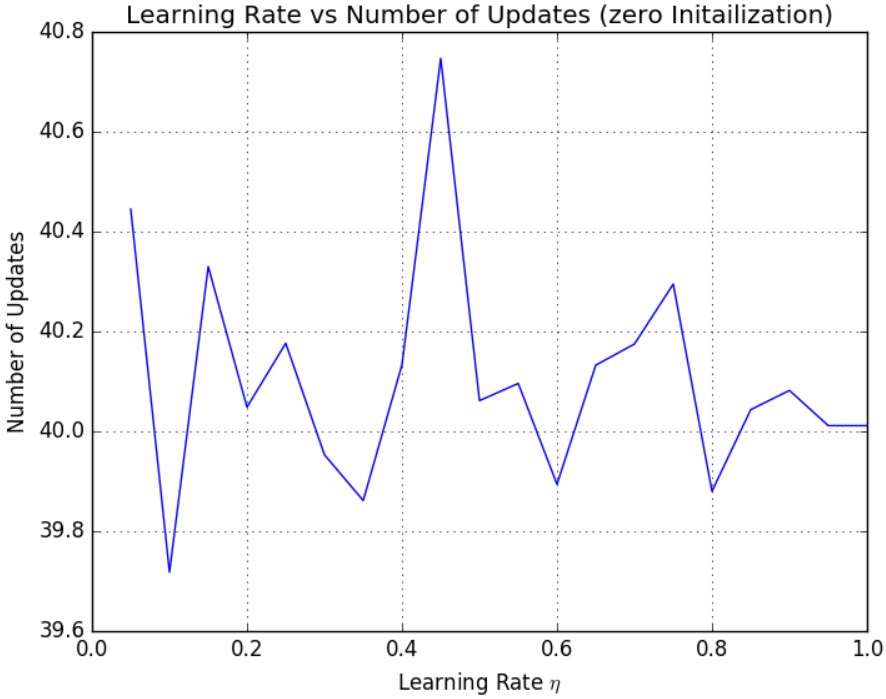

Figure 5: Zero Init: Learning Rate vs Updates

Prob. 18 - Pocket PLA

```
Initialization method: zero
Pocket Updates 3 times within 50 iterations.
Get 88 errors on 500 testing dataset, accuracy: 82.4000 percents
Average Error Rate after 2000 experiments: 18 percents
```

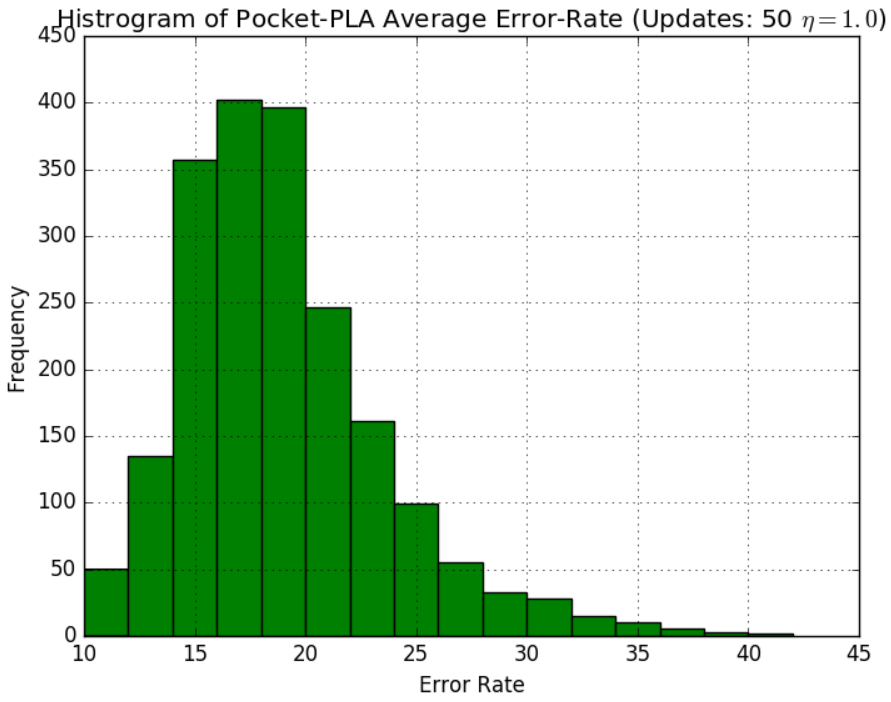

Figure 6: Histogram of Error Rate - Pocket PLA 50 updates

Prob. 19 - Pocket PLA with 100 updates

```
python hw_1-19.py
Prob 1-19
Initialization method: zero
Pocket Updates 4 times within 100 iterations.
Get 72 errors on 500 testing dataset, accuracy: 85.6000 percents
Average Error Rate after 2000 experiments: 14 percents
```

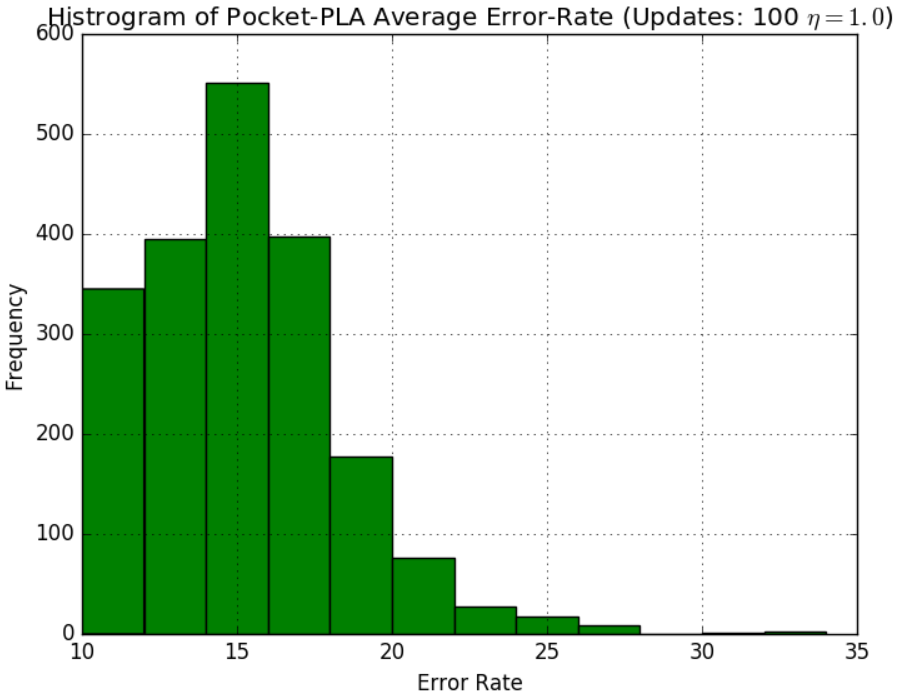

Figure 7: Histogram of Error Rate - Pocket PLA 100 updates

Discussion:
Compare two histograms we can conclude that more updates give us lower error rate. Not only the average rate decreases from 18 to 14 but the distribution is aligend to left-hand side. It states that the machine makes less mistakes.

Prob. 20 - w_{100} PLA

```
Prob 1-20
Initialization method: zero
Pocket Updates 7 times within 100 iterations.
Get 54 errors on 500 testing dataset, accuracy: 89.2000 percents
Average Error Rate after 2000 experiments: 25 percents
```

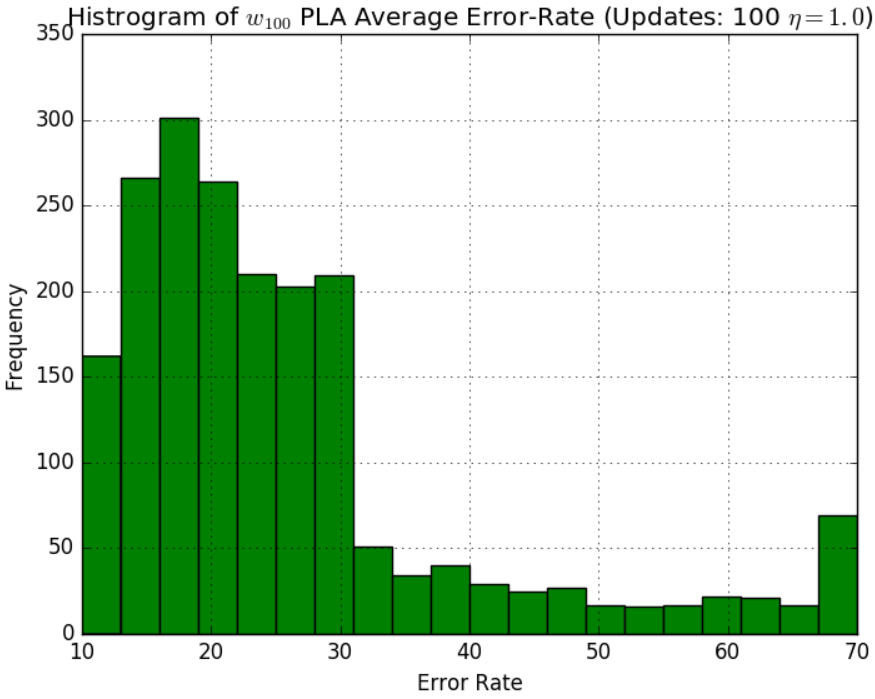

Figure 8: Histogram of Error Rate - w_{100} PLA

Discussion:
The performance becomes worse than before as we expected. Because we ignore the best card in our hand but choosing the alternative, it pay us some price. The average error rate grows and its distribution becomes more unifrom. It states that algorithm sometime can make a disaster.

Prob. 21 New update always classify correctly
If algorithm makes mistake at (x_n, y_n),

$$y_n \neq \text{sign}(w_t^T x_n) \text{ but } y_n = \text{sign}(w_{t+1}^T x_n)$$

Indicating $y_n w_t^T x_n < 0$ by follwing updates, we are going to show $y_n w_{t+1}^T x_n > 0$

$$w_{t+1} \leftarrow w_t + y_n x_n [\frac{-y_n w_t^T x_n}{|x_n|^2} + 1]$$

We multiply x_n and y_n from both sides can get

$$\begin{aligned} y_n w_{t+1} x_n &= y_n w_t x_n + y_n y_n x_n [\frac{-y_n w_t^T x_n}{|x_n|^2} + 1] x_n \\ &= y_n w_t x_n + |y_n|^2 [\frac{-y_n w_t^T}{|x_n|^2}] |x_n|^2 + |y_n|^2 |x_n|^2 \\ &= (1 - |y_n|^2) y_n w_t x_n + |y_n x_n|^2 \end{aligned}$$

Due to binary classification, our labels are chooen $y = \pm 1,\ |y_n|^2 = 1$, so the first term is cancelled. Such that

$$y_n w_{t+1} x_n \geq 0$$

It shows that updated weights could be correct on sample (x_n, y_n)