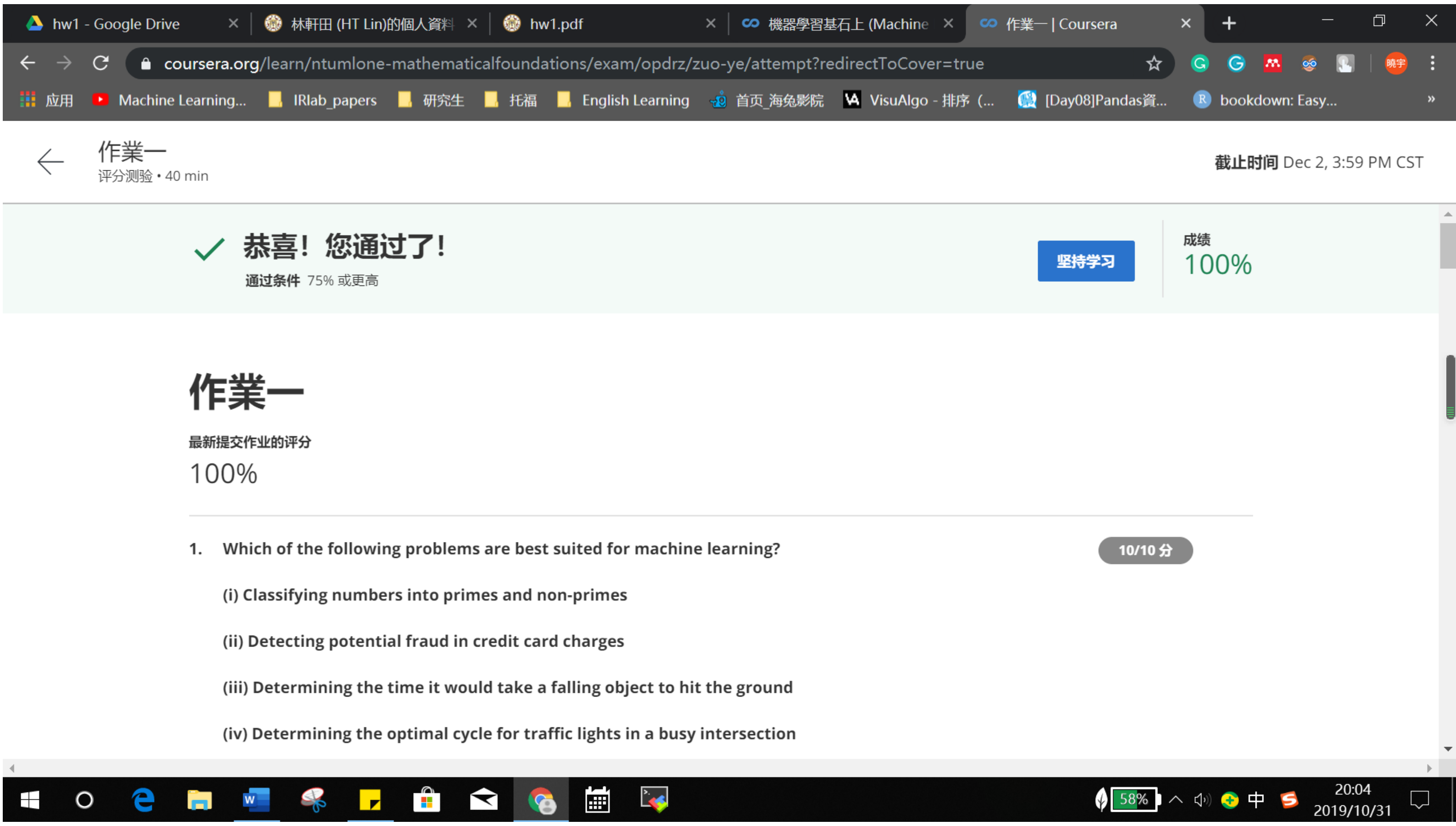
Question 1— coursera



Question 2

半監督學習（semi-supervised learning）在隱私保護上的應用。例如銀行業者希望利用機器學習的方法評估客戶的信用進行信貸風險管理。這一情境下如果利用所有資料進行建模就有可能涉及到客戶的個人隱私問題。半監督學習只需要利用一部分已標記的數據和大量未標記的數據進行建模，這樣我們可以避免使用一些客戶敏感資訊（例如身份訊息、居住地、手機號碼等），更關注某一類客戶在行為上有什麼共性。

斯晓宇 R08944052

3. ∵ f 在 $D=\{(x_n, y_n)\}_{n=1}^{N}$ 上 $f(x_n)=y_n$, where $y_n \in \mathcal{Y}=\{-1, +1\}$

f 在 test input $\{x_{N+1}, x_{N+2}, x_{N+3} \cdots x_{N+L}\}$ 上 $f(x_{N+l})=-1$ or $+1$

∴ 对于 test set, f 产生 $\{y_{N+1}, y_{N+2}, y_{N+3} \cdots y_{N+L}\}$ 有 2^L 种可能的情况

由 $E_{OTS}(g,f)=\frac{1}{L}\sum_{l=1}^{L}[g(x_{N+l}) \neq f(x_{N+l})]$ where $A(D)=g$

$$\begin{cases} g(x_{N+l}) \neq f(x_{N+l}), & OTS_l=1 \\ g(x_{N+l}) = f(x_{N+l}), & OTS_l=0 \end{cases}$$ OTS: off-Training-Set error.

∴ 存在 $f(x_{N+l})$ 使得 $\sum_{l=1}^{L} OTS_l = \sum_{l=1}^{L}[g(x_{N+l}) \neq f(x_{N+l})]$ 的值为 $[0, L]$ 中的任意一个整数

∴ 当 $\sum_{l=1}^{L} OTS_l = \sum_{l=1}^{L}[g(x_{N+l}) \neq f(x_{N+l})] = n$ 时, where $n \in [0, L]$

满足上述情况的 f 有 C_n^L 个.

∵ f are equally likely in probability.

∴ $P(\sum_{l=1}^{L} OTS_l = n) = \frac{C_n^L}{2^L}$

∴ $E_f\{E_{OTS}(A(D), f)\} = \sum E_{OTS}(A(D), f) \cdot P$

$= \frac{1}{L}\cdot 0 \cdot \frac{C_0^L}{2^L} + \frac{1}{L}\cdot 1 \cdot \frac{C_1^L}{2^L} + \cdots + \frac{1}{L}\cdot L \cdot \frac{C_L^L}{2^L}$

$= \frac{1}{L}\cdot\frac{1}{2^L}\cdot(0\cdot C_0^L + 1\cdot C_1^L + \cdots + L\cdot C_L^L)$

见左侧.

$= \frac{1}{L}\cdot\frac{1}{2^L}\cdot(L\cdot 2^{L-1})$

$= \frac{1}{2}$

二项式定理

$(x+y)^n = \sum_{k=0}^{n} C_k^n x^{n-k} y^k = \sum_{k=0}^{n} C_k^n x^k y^{n-k}$

$(1+x)^n = \sum_{k=0}^{n} C_k^n x^k = C_0^n + xC_1^n + x^2C_2^n + \cdots + x^nC_n^n$

$\frac{d}{dx}(1+x)^n = n(1+x)^{n-1}$
$= C_1^n + 2xC_2^n + \cdots + nx^{n-1}C_n^n$

当 $x=1$ 时 $n\cdot 2^{n-1} = 1\cdot C_1^n + 2C_2^n + \cdots nC_n^n$

∴ 对于任意 A, 上式与 A 无关 $E_f\{E_{OTS}(A(D), f)\} \equiv \frac{1}{2}$ #

R08944052 斯曉宇

4. 设 $P(A)$：选中 A 类骰子的机率

$P(D)$：选中 D 类骰子的机率

$\because$ green 1 $\in$ A or D

$P(A) \perp\!\!\!\perp P(B) \perp\!\!\!\perp P(C) \perp\!\!\!\perp P(D)$

$\therefore P(\text{green } 1) = P(A \cup D) = P(A) + P(D) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$

$\therefore P(\text{pick five green } 1\text{'s}) = (\frac{1}{2})^5 = \frac{1}{32}$ #

5.

green	Even	Odds
A		1, 3, 5
B	2, 4, 6	
C	4, 6	5
D	2	1, 3

√ green 1 $\in$ A or D

green 2 $\in$ B or D

√ green 3 $\in$ A or D

∘ green 4 $\in$ B or C

green 5 $\in$ A or C

∘ green 6 $\in$ B or C

抽5个骰子，某一种数字全为绿色的机率。

选中某一数字全为绿色会有4种可能的情况
$$\begin{cases} A \text{ or } D - 1,3 \\ B \text{ or } D - 2 \\ B \text{ or } C - 4,6 \\ A \text{ or } C - 5 \end{cases}$$

$P(\text{"Some number" that is purely green}) = [P(A \cup D)]^5 + [P(B \cup D)]^5 + [P(B \cup C)]^5 + [P(A \cup C)]^5 - [P(A)]^5 - [P(B)]^5 - [P(C)]^5 - [P(D)]^5$

$[P(A \cup D)]^5$, $[P(A \cup C)]^5$ 都包含5次全为A类的情况，故 $-[P(A)]^5$

依此类推需要减去5次全为某个类别的机率

$\therefore P(\text{"Some number" that is purely green}) = (\frac{1}{2})^5 + (\frac{1}{2})^5 + (\frac{1}{2})^5 + (\frac{1}{2})^5 - (\frac{1}{4})^5 - (\frac{1}{4})^5 - (\frac{1}{4})^5 - (\frac{1}{4})^5$

$= \frac{4}{2^5} - \frac{4}{4^5} = \frac{4}{2^5} - \frac{4}{2^{10}} = \frac{4(2^5 - 1)}{2^{\cancel{10}\, 8}} = \frac{31}{256}$ #

Findings 在下一页！

5. Findings:

可以把 problem 4–5 抽象为一个 Feasibility of Learning 的问题.

Bag ⟶ 巨大的data set

A, B, C, D ⟶ 四种data 类型

dice $\begin{cases} \text{绿色的数字} & h(x) \neq f(x) \\ \text{橙色的数字} & h(x) = f(x) \end{cases}$

1, 2, 3, 4, 5, 6 ⟶ hypothesis h_1; h_2, h_3, h_4, h_5, h_6

pick 5 dice from the bag ⟶ sample $N=5$.

依题意,对于任意一个 h_n $E_{out}(h_n) = \frac{1}{2}$

当$N=5$且抽到的是5个 green dice,此时 $E_{in}=1$, $|E_{out} - E_{in}| = 0.5$

所该情况可视作 Bad D

Question 4 得到: $P_D[\text{Bad } D \text{ for } h_n] = \frac{1}{32} = \frac{8}{256}$

Question 5 得到: $P_D[\text{Bad } D] = \frac{31}{256} \leq 4 \cdot P_D[\text{Bad } D \text{ for } h_n] = \frac{32}{256}$

根据 Hoeffding's Inequality:

Bound of Bad Data $P_D[\text{Bad } D] = P_D[\text{Bad } D \text{ for } h_1 \text{ or Bad } D \text{ for } h_2 \cdots \text{ or Bad } D \text{ for } h_6]$

$\leq P_D[\text{Bad } D \text{ for } h_1] + P_D[\text{Bad } D \text{ for } h_2] + \cdots + P_D[\text{Bad } D \text{ for } h_6]$

$= 6 \cdot P_D[\text{Bad } D \text{ for } h_n]$

所以理论上 $P_D[\text{Bad } D] \leq 6 \cdot P_D[\text{Bad } D \text{ for } h_n]$

但在 Question 5 中 $P_D[\text{Bad } D] \leq 4 \cdot P_D[\text{Bad } D \text{ for } h_n]$

我认为这是因为 Question 5 中只有 4种情况

所以使得有效计算 Bound of Bad Data 的

hypothesis 变为只有4种。

$\begin{cases} \text{A or D} — h_1, h_3 \\ \text{B or D} — h_2 \\ \text{B or C} — h_4, h_6 \\ \text{A or C} — h_5 \end{cases}$

说明在 hypothesis set $\mathcal{H}$ 中有一些 h 是相似的,

找到实际有效的 hypothesis 可以更精确估计 Bound.

Question6

The average number of updates is 39.22113676731794

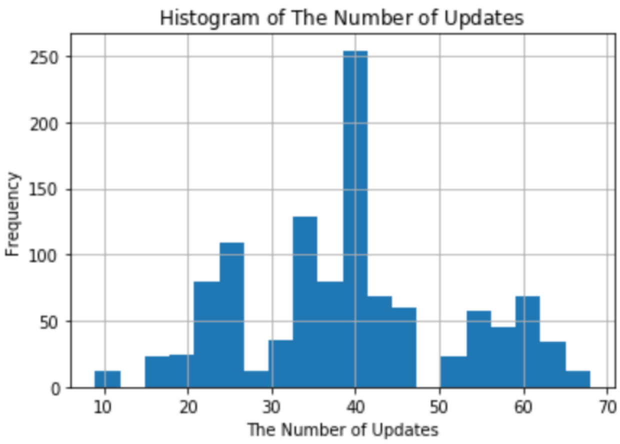
jupyter hw1_6 最后检查: 5 分钟前 (自动保存)

File Edit View Insert Cell Kernel Widgets Help Snippets

运行 代码

```
#Plot histogram to show number of updates
plt.figure()
plt.hist(save_update,bins=20)
plt.ylabel('Frequency')
plt.xlabel('The Number of Updates')
plt.title(r'$\mathrm{Histogram\ of\ The\ Number\ of\ Updates}$')
plt.grid(True)
plt.show()
```

```
The average number of updates is  39.22113676731794
```



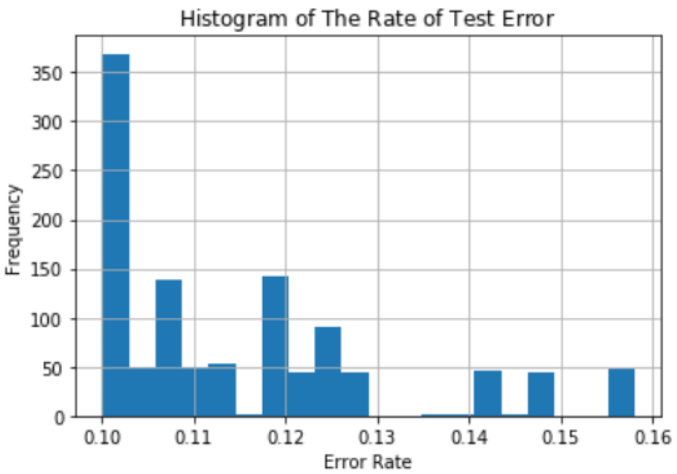
Question7

The average error rate(pocket) on the test set is 0.1149

In [49]:
```python
avg_error = mean(eval)
print("The average error rate on the test set is ",avg_error)

#Plot histogram to show number of test error
plt.figure()
plt.hist(eval,bins=20)
plt.ylabel('Frequency')
plt.xlabel('Error Rate')
plt.title(r'$\mathrm{Histogram\ of\ The\ Rate\ of\ Test\ Error}$')
plt.grid(True)
plt.show()
```

```
The average error rate on the test set is  0.11490230905861457
```



Question8

The average error rate(w100) on the test set is 0.35128

Findings:

實驗 Pocket PLA 演算法，第 7 題 return w_hat(pocket vector)，第 8 題 return w100。 w_hat(pocket vector)保存每次更新使得錯分率更小的權重， 相較於直接使用更新到 100 次的權重 w100。 在有限的 iteration 和 update 次數中，使用 w_hat(pocket)的平均錯誤率=0.115 顯著小於 w100 的平均錯誤率=0.351 可以驗證使用 pocket 演算法能得到最佳的分類線，雖然仍有分錯的點，但已經是最少的了。

jupyter hw1_8 最后检查: 14 分钟前 (自动保存)

Logout

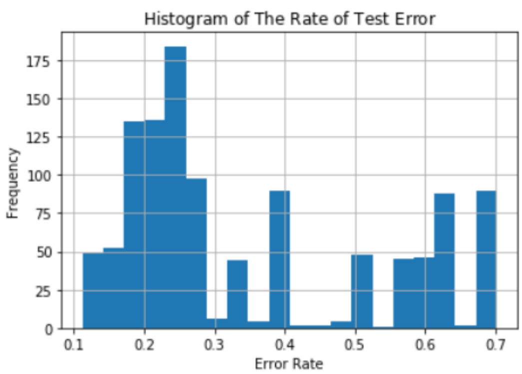
File Edit View Insert Cell Kernel Widgets Help Snippets 不可信 Python 3

运行 代码

In [11]:
```
avg_error = mean(eval)
print("The average error rate on the test set is ",avg_error)

#Plot histogram to show number of test error based on W100
plt.figure()
plt.hist(eval,bins=20)
plt.ylabel('Frequency')
plt.xlabel('Error Rate')
plt.title(r'$\mathrm{Histogram\ of\ The\ Rate\ of\ Test\ Error}$')
plt.grid(True)
plt.show()
```

```
The average error rate on the test set is  0.3512877442273535
```



實驗Pocket PLA演算法，第7題return w_hat(pocket vector)，第8題return w100。 w_hat(pocket vector)保存每次更新使得錯分率更小的權重， 相較於直接使用更新到100次的權重w100。 在有限的iteration和update次數中, 使用w_hat(pocket)的平均錯誤率=0.115 顯著小於 w100的平均錯誤率=0.351 可以驗證使用pocket演算法能得到最佳的分類線，雖然仍有分錯的點，但已經是最少的了。

R08944052 斯曉宇

9. It doesn't work

Refer to page16 in Lecture2

$R^2 = \max_n \|x_n\|^2$ $\rho = \min_n y_n \frac{w_f^T}{\|w_f\|} x_n$

The number of mistake corrections $T \le \frac{R^2}{\rho^2}$

如果 x_n 线性缩小10倍，x_n 的模长 $\|x_n\|$ 也会缩小10倍.

$R' = \max_n \|\frac{x_n}{10}\|$ $\rho' = \min_n y_n \frac{w_f^T}{\|w_f\|} \frac{x_n}{10}$

$$\frac{R'^2}{\rho'^2} = \frac{(R/10)^2}{(\rho/10)^2} = \frac{R^2}{\rho^2} \ge T$$

∴ 线性缩放 x_n 不会对收敛速度有所影响.