午餐系統及分析

# 摘要

# 壹、研究動機

之前在學校訂餐的時候，都只能用紙本點餐單來點餐，代訂同學常常把紙本單弄丟，而且紙本單必須手工計算數量、金額，十分不便，紙本單是不完全記名制，只能知道是哪一班點的，常常會有人忘記自己點了什麼餐，直接隨便亂拿一份餐，我們認為應該要有一個比紙本單更優秀的解決辦法，便開始著手製作午餐系統了。

午餐系統在校內推行成功後，我們了解到廠商常常會備料過剩，於是我們打算建立一個模型，可以給廠商做為明天會有多少份餐點的依據。

# 貳、研究目的

本研究旨在於製作一套能夠取代紙本點餐單的系統，並設計一個適當的數學模型供廠商參考明天該準備多少份餐點。

製作一套能夠完全取代紙本單據的系統，這是本研究的第一目標；再根據收集到的資料給予廠商適當的預測，這是本研究的第二目標；我們意外發現對於單層的神經網路模型能夠以三分搜加速迭代，證明演算法可行且具有較高的效率，這是本研究的第三目標。

# 參、研究設備及器材

|  |  |
| --- | --- |
| 器材 | 用途 |
| *Firebase* | *App*分析用的*Third party plugin* |
| *Google analytics* | 網頁分析用的*Third party plugin* |
| *Ubuntu 18.04 LTS* | *Server OS* |
| *Nginx* | *Web Server Application* |
| *Noip Dynamic Dns* | *Server Domain name* |
| *Mac* | *iOS App*的開發環境 |
| *Iphone* | *iOS App*的測試環境 |
| *Xcode* | *iOS App* *IDE* |
| *Android*手機 | *Android App*的測試環境 |
| *Android Studio* | *Android App IDE* |
| *Chrome* | 網頁前端測試環境 |
| *Visual Studio* | 開發廠商前端的*IDE* |
| *Visaul Studio Code* | 開發後端的*IDE* |
| *Html + Css + Bootstrap* | 網頁前端開發框架 |
| *Javascript + JQuery + Ajax* | 網頁前端開發語言 |
| *Php* | 後端開發語言 |
| *Mysql* | 資料庫 |
| *Kotlin* | *Android App*開發語言 |
| *Swift* | *iOS App*開發語言 |
| *Python* | 數據分析測試語言 |
| *C#* | 廠商前端開發語言 |
| *Linux terminal bash* | *Database auto backup* |

# 肆、研究過程及方法

午餐系統後端由*Php*作為後台，*Mysql*作為資料庫，並且輸出結果到交換介面；網頁版前台、*iOS*前台、*Android*前台及其他外掛插件，從交換介面擷取資料，再將資料展現給使用者，下圖為午餐系統的架構圖。

午餐系統每天都會有大量的點餐資料，如果能對這些資料進行分析，就能夠協助廠商預測明天的餐點量，於是我們設計了一個預測模型。

## 系統開發過程

隨著業務增長，原本的架構不敷使用，我們決定重寫架構，這意味著後前台、後台以及資料庫都必須重新打掉重寫；我們也有不少過渡期功能，例如代訂繳款、合作社繳款，在還沒連結POS時，代訂必須跟全班收錢，並且將班上的錢交給合作社，合作社再將錢分配給廠商，這些功能在連結POS後就全部捨棄掉了。目前看到的成果只是冰山一角，不少程式在重寫架構時全部變成垃圾，也不少程式在過渡期中功成身退。

開發演算法時，先以*Python*開發雛形，確定演算法可行；確認演算法可行後，再將模型改寫成*C#*，並為模型製作GUI介面，引入廠商管理前端。

|  |  |
| --- | --- |
| 日期 | 功能 |
| 2017/11 | 連接學生與代訂，手工抄寫至點餐單。 |
| 2018/04 | 重寫系統架構，包含前台、後台以及資料庫。 |
| 2018/09 | 連接學生、代訂、合作社、廠商。 |
| 2019/01 | 引入*POS*繳款機制，架空代訂以及合作社。 |

## 系統使用簡介

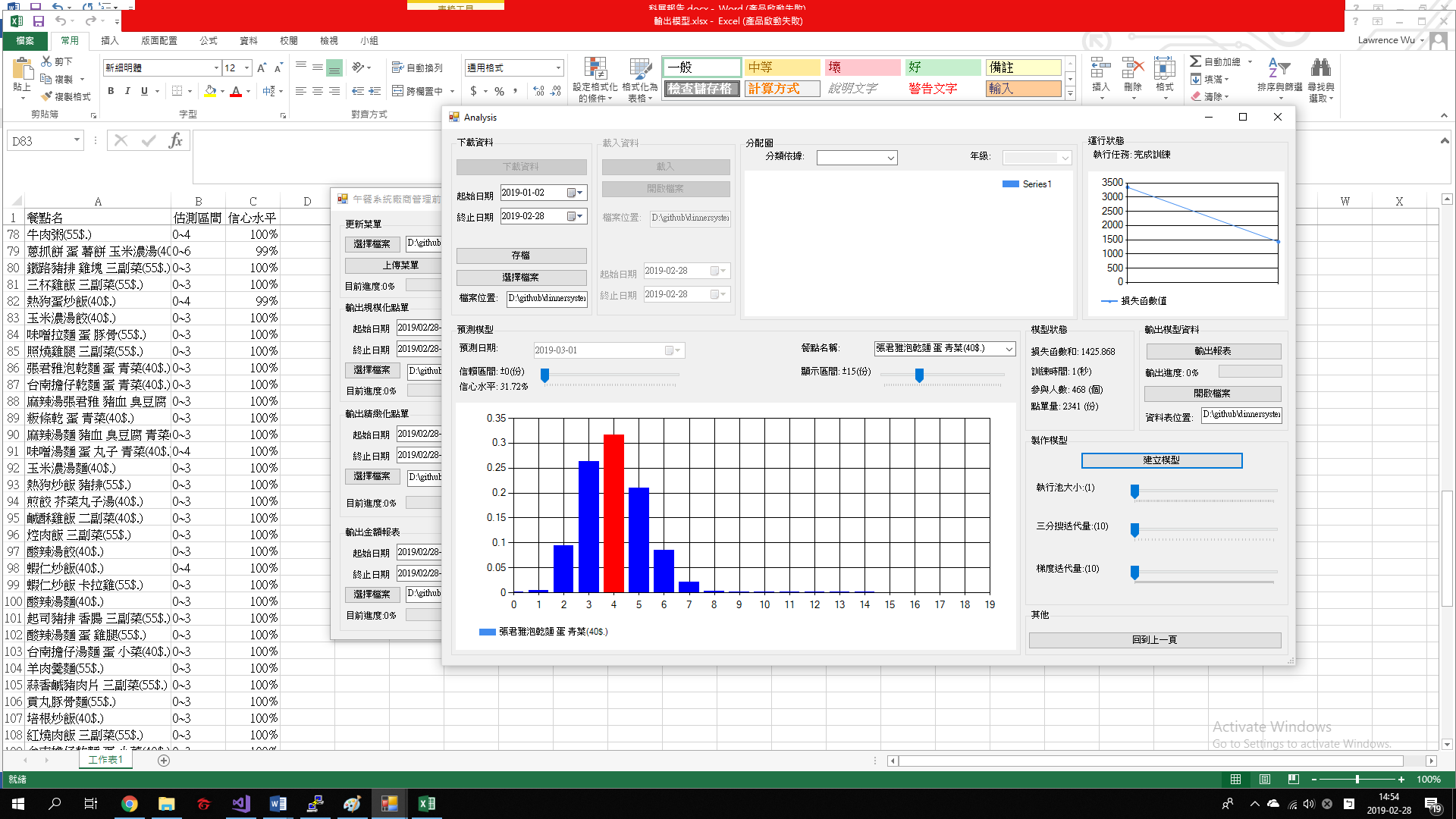
下表為實際使用系統點餐的擷圖，以下簡介以下四種最常被用到的功能。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 登入 | 點餐 | 查看點單 | 繳款 |
| 安卓前端 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52995536_341576016452924_4492182019454795776_n.jpg | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\53020164_327182224573781_1994272908897157120_n.jpg | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52859457_560714564433881_1700737264587374592_n.jpg |  |
| 蘋果前端 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52785773_2010996985860581_7006054668753174528_n.jpg |  |  |  |
| 網頁前端 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52877884_1913323308776129_3492376395303092224_n.jpg | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\53509579_540146379724500_6811099317139406848_n.jpg |  |  |

下表為實際使用廠商管理插件的截圖，以下簡介三種最常被用到的功能。

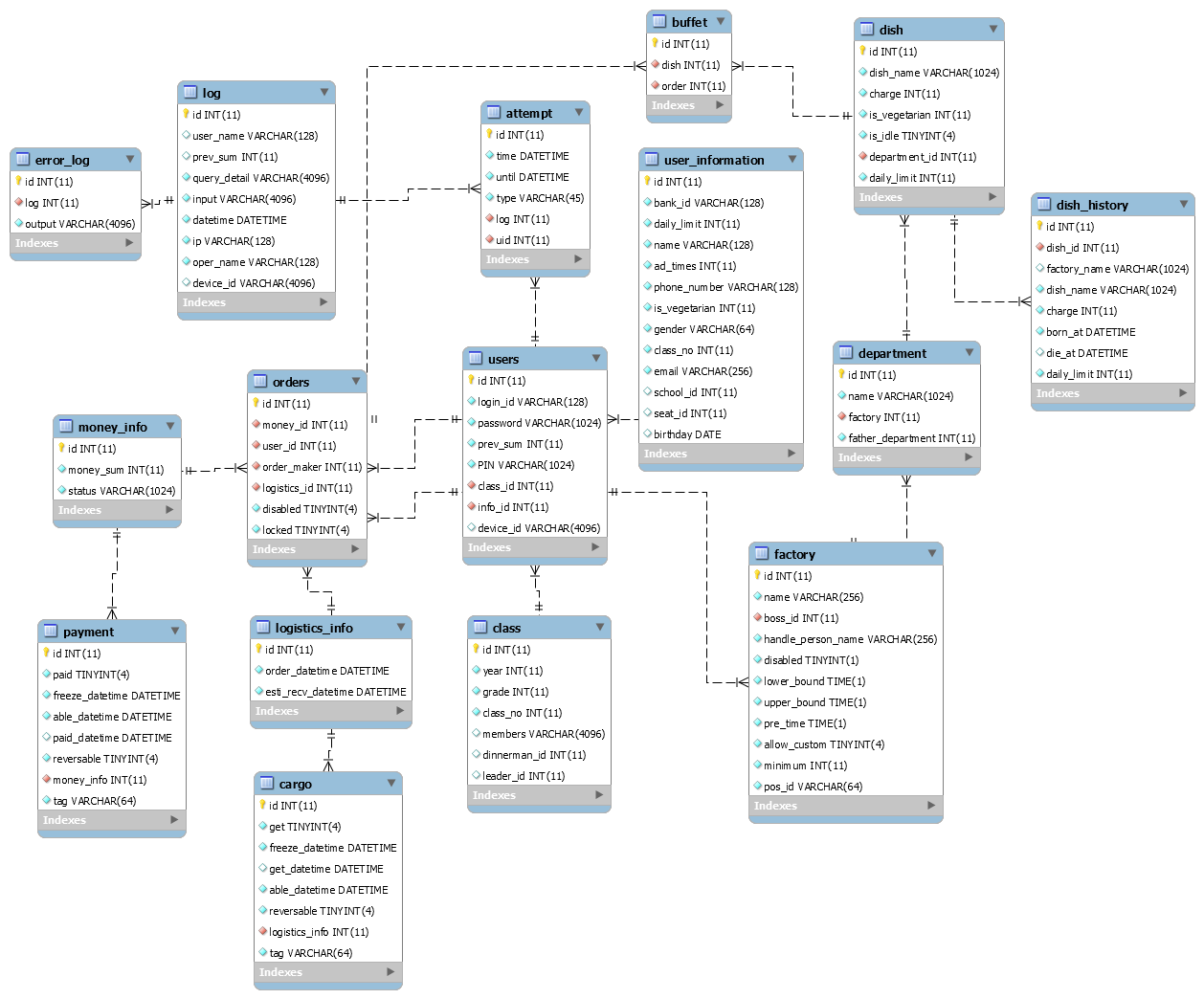
|  |  |  |
| --- | --- | --- |
|  | 廠商管理插件 | 匯出表格 |
| 更新菜單 |  |  |
| 查看點單 |  |  |
| 查看預測模型 |  |  |

下圖為實際使用預測模型的截圖，我們特別製作可以查看模型的GUI介面，方便廠商估計明天需要準備多少餐點。



## 三、資料庫

下列為午餐系統的資料結構模型，每一條線代表一個關聯性連接。



午餐系統的資料庫為關聯性資料庫，並使用*innoDB*做為引擎，*innoDB*支援交易機制，比起*myisam*，使用*innoDB*更方便處理死結回溯的問題。

### (一)、*Deadlock*

在單線程的測試環境下，很少會遇到資料庫死結*(Deadlock)*，而在系統真正運行的時候，常會遇到不可預知的死結。一個*Procedure*包裝了多個語句*(Syntax)*，若是在尚未執行完*Procedure*前，發生了死結，則會發生不可預期的後果。

我們針對容易發生死結的*Procedure*加上*start transcation*、*rollback*、*commit*，若是在*Procedure*尚未結束前發生死結，則回溯*(Rollback)*整個*Procedure*的操作。

### (一)、*InnoDB Locks*

下表為*InnoDB*內建的鎖，例如資料庫同時接到「繳款」以及「刪除」的指令，資料庫會先在該筆訂單上*IX*鎖，確保另外一個指令沒辦法更改資料，再拿到*X*鎖修改資料，最後釋放*X*鎖以及*IX*鎖，使得另外一個指令開始運行。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | X | IX | S | IS |
| X | 衝突 | 衝突 | 衝突 | 衝突 |
| IX |  | 相容 | 衝突 | 相容 |
| S |  |  | 相容 | 相容 |
| IS |  |  |  | 相容 |

## 四、後端

下圖為後端的架構，我們將後端分成多個模組，方便維護，也方便開發新功能。

#### (一)、*Session* 阻塞

*php*為了保證執行緒安全，同一個*session*同時只能給一個請求使用。在每個請求都不會提前釋放*session*的狀況下，同時送出大量請求，會使得效能非常低落，如下圖所示。

上圖中，請求(3)等候前面兩個請求處理完才開始執行，成為緩慢的串行命令。如果程式會先複製好*session*再執行，則每個請求只需要等候其他請求複製完資料，就能夠先開始執行了，如下圖所示。

上圖中，請求(3)僅等候前面兩個請求複製資料，因為不必等候其他請求，因此能受益於*CPU*的平行處理，使得效能提升。

#### (二)、資料庫存取阻塞

資料庫的存取速度遠遠低於記憶體的存取速度，如果每次使用常駐資料時，都向資料庫要求一次資料，則這些常駐資料請求會拖累系統效能。後端會將常駐資料先快取於*session*，需要使用資料時直接從*session*調用資料，就不必再向資料庫請求資料了。

## 五、前端

大多數使用者對於系統的印象就只有前端，所以前端被稱為為整套系統的門面。前端的任務在於將資料轉換成使用者能看懂的格式，並且協助使用者對伺服器下達命令，我們共有三種給學生的前端，分別是*iOS*、*Android*以及網頁前端。

#### (一)、*Firebase、Google Analytics*

#### (二)、比較、使用分布

下圖為各種前端的使用率分配圖，網頁佔了接近一半的使用率；安卓少了一些，不過使用率仍比蘋果高；蘋果的使用率最低，占了五分之一再少一點。

下表比較了三種前端之間的優劣利弊，我們首先發布了網頁前端，再來是*iOS App*，最後發布*Android App*。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 特色 | 優勢 | 劣勢 |
| 網頁版 | 最早推行的前端 | 不論系統，皆可使用 | 僅支援主流瀏覽器  不支援自助點餐 |
| *Android*前端 | 具有良好的防呆機制 | 支援自助點餐 | 開發耗時較久 |
| *iOS*前端 | 具有良好的防呆機制 | 支援自助點餐 | 開發耗時較久，上架審查機制嚴格 |

## 六、預測模型

我們先向廠商、合作社打聽了一下平常點餐的趨勢，廠商表示訂購便當的意願隨著在學校的時間漸漸降低。我們經過初步分析，得知高一最多人點餐，高二的點餐人數較少，高三幾乎沒有人點餐。

我們將預測模型拆成兩個子模型，一為比例模型，二為數量模型。顧名思義，比例模型能夠給出明天的點餐比例，數量模型能夠給出明天的點餐總數，兩個模型一起使用就能得到明天各種餐點的數量。

### (一)、資料結構

### (二)、比例模型

設計比例模型旨在於使用數學模型來預測各種類別的比例，我們使用機率矩陣的穩定狀態來預測比例，無法求得穩定狀態時，以矩陣快速冪代替。

我們可以將現實世界中的操作抽象化，成為圖論中的操作，方便程式預測，下表為現實世界中的操作與圖論中的操作對照表。

|  |  |
| --- | --- |
| 圖論上的操作 | 現實世界的意義 |
| 從圖上的任意一個點出發 | 第一天先隨便點一道菜 |
| 經由任意一條邊 | 經過了一天 |
| 到達圖上的另外一個點 | 第二天點了另外一道餐 |
| 經過一個自環 | 經過了一天，決定要吃同一道菜 |
| 到了同一個點 | 第三天點了同一道菜 |

#### 1. 機率矩陣

對於點餐向量，代表原先餐佔所有餐的比例，若，則代表經過天後餐的比例。我們定義，而且經過多次轉移之後，會漸漸趨近於一個穩定狀態，則我們可以得到方程式。

此時，我們稱呼為矩陣的穩定狀態。

#### 求穩定狀態

我們有兩種方法可以求穩定矩陣，第一種叫做「反矩陣解聯立」，第二種叫做「矩陣快速冪」，下表比較了兩種方法的優劣。

|  |  |  |
| --- | --- | --- |
|  | 反矩陣解聯立 | 矩陣快速冪 |
| 優點 | 能夠求出真正的穩定狀態 | 保證有輸出值 |
| 缺點 | 不一定能夠求出穩定狀態 | 只能求出近似穩定狀態 |

在反矩陣解聯立無法求出穩定狀態時，我們使用矩陣快速冪作為替代方案，由於矩陣快速冪有一定的機率會失準，我們選擇多求出幾個近似的穩定狀態，再將各個近似狀態平均，藉此取得較為精準的穩定狀態。

##### (1). 反矩陣解穩定狀態

定義穩定狀態，我們可以得到下列聯立方程式。

化簡後得

對於聯立方程組有以下性質，為給定的係數矩陣，為給定的值矩陣，為未知數矩陣。

將視為方程式中的係數，且，我們得知有一行方程式無用，於是可以得到下式。

於是我們可以求得穩定狀態，不過並不是每一個都有反矩陣，於是我們使用矩陣快速冪作為替代方案。

#### 3. 矩陣快速冪

矩陣乘法具有結合律，並根據下列二式，我們可以得知只需要在的時間內就能得知的值。

為任意機率向量，給定越大的數字，求出來的向量越接近穩定狀態，給定數列，共有個元素，其中為一個隨機整數，且界於與之間。

該演算法的時間複雜度為，使用越大的與越大的以及，預測出來的越接近穩定狀態，模型預設。

### (三)、數量模型

設計數量模型旨在於使用數學模型來預測總餐數會有多少份，數量模型的核心為*logistic*模型，我們對每個人建立一個*logistic*模型，再使用統計演算法求出總餐數約有多少份。

#### 1. *logistic*模型

*logistic*模型最主要的函數為*sigmoid*函數，如下。

將輸入先經過線性變換，再交給*sigmoid*進行輸出，定義為線性變換用的向量，為輸入的向量。

一組訓練用的資料為輸入值以及輸出值，為一個布林值，只會是1或是0，我們可以寫出損失函數如下。

可以得知損失函數越大，該模型精確度就愈高。

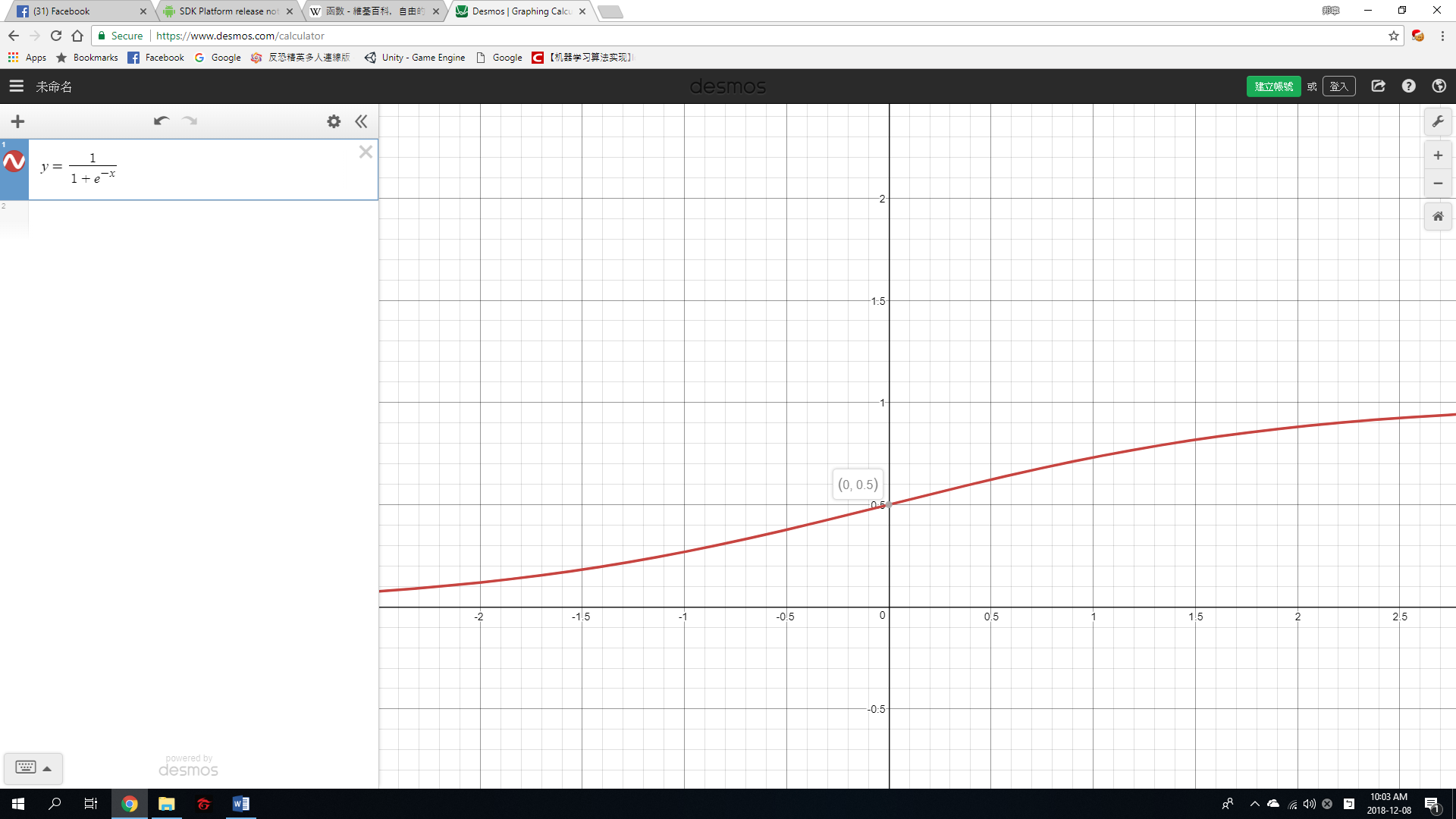
我們無法找出一個使得損失函數最大化，但是可以使用最大似然估計來估測，為偏差倍率，偏差倍率越大，模型訓練越快，越容易錯過最佳解；偏差倍率越小，模型訓練越慢，越容易找到最佳解。

下式為迭代方程式，為對的梯度。

當趨近零時

經過迭代之後，我們可以得到向量。

對於輸入值，線性變換後的結果必為零，sigmoid的輸出值必為0.5，如下圖。



這樣的模型很明顯不是我們想要的，所以我們需要一個常數來修正模型，其中為一個常數，將模型修正如下。

我們使用下列方法來求出，其中為線性變換參數。

使用最大似然估計來估測，再將還原成與。

我們可以獲得較為精確的logistic模型，模型的訓練時間複雜度為，模型預測一組資料的時間為，其中為迭代次數，為輸入的向量大小，為訓練資料數量。

#### 2. 決策模型

決策模型只能預測特定一個人，模型會根據他之前的點餐行為，預測他今天是否會選擇點餐，以下是模型的輸入輸出圖，我們可以用點餐序列作為模型輸入。

點餐機率有多少

……..

本模型具有一定的規律鑑別能力，像是很多人是禮拜二家裡會準備便當，所以就不會訂購學校的便當，模型能夠偵測出這個人禮拜二通常都不會點餐，宏觀模型沒辦法達到這一點。

以下是一個三循環的測試資料，我們將這組資料交給決策模型進行訓練。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 日期 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 是否點餐 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

以下是決策模型的訓練成果，模型最後一項的線性變換參數為。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | Loss |
| N=1 | 正相關 | 無相關 | - | - | - | - | -3.37602795 |
| N=2 | 正相關 | 正相關 | 負相關 | - | - | - | -0.03053806 |
| N=3 | 正相關 | 正相關 | 負相關 | 負相關 | - | - | -0.00921238 |
| N=4 | 正相關 | 正相關 | 負相關 | 正相關 | 無相關 | - | -0.00816815 |
| N=5 | 正相關 | 正相關 | 負相關 | 正相關 | 正相關 | 無相關 | -0.03469504 |

負相關代表「該值愈大，輸出值越接近零」，正相關代表「該值愈大，輸出值越接近一」，不相關代表該輸入與輸出幾乎無關聯性，*Loss*為損失函數值。

由上表我們可以得知，取太大或是太小的*N*容易無法判斷模式，模型預設，因為一個禮拜有七天，大多數的規律都是七天一個循環。

#### 4. 統計演算法

我們想要知道有幾個人點餐的機率最高，我們將甲生點餐的事件寫成，甲生不點餐的事件寫成，總共有個人點餐的事件寫成。

求出每一項*N*需要，我們可以想成每個人只有點餐跟不點餐這兩個選項，枚舉每一種狀態，再將所有機率加總。演算法的時間複雜度不甚理想，我們需要對演算法優化。

我們使用*DP*來進行優化，優化之後只需要的時間複雜度，效率大幅提升，以下是*DP*的遞迴關係式。

每次執行完迴圈後的值會賦予，最後得到的就是結果。

### (四)、預測模型

比例模型可以得出比例，數量模型可以得出總數，再經由下面公式即可獲得每個分類的數量預估值。

下圖為預測模型的輸出，與前三天的資料進行對比，每一種顏色代表一天的資料，每一個長條代表一種類別的數量，虛線代表預測值。

# 伍、研究結果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 登入 | 點餐 | 查看點單 | 繳款 |
| 安卓前端 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52995536_341576016452924_4492182019454795776_n.jpg | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\53020164_327182224573781_1994272908897157120_n.jpg | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52859457_560714564433881_1700737264587374592_n.jpg |  |
| 蘋果前端 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52785773_2010996985860581_7006054668753174528_n.jpg |  |  |  |
| 網頁前端 | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52877884_1913323308776129_3492376395303092224_n.jpg | C:\Users\lawre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\53509579_540146379724500_6811099317139406848_n.jpg |  |  |

# 陸、討論

比例模型的機率矩陣過為稀疏

學生的點餐習慣

# 柒、結論

# 捌、參考資料及其他

板橋高中資訊培訓講義

StackOverflow

Logistic regression <https://blog.csdn.net/SzM21C11U68n04vdcLmJ/article/details/78221784>

午餐系統<http://dinnersystem.ddns.net>