* Hard-Margin Support Vector Machine
  1. “hard-margin” decision stump

首先可以知道一定是切在跟之間，接著就是要讓邊界最大化，那麼分界點就是取中間值，所以最大邊界長度就是：

* 1. dual problem for the uneven margin SVM

Primal問題：

首先列出Lagrange function：

其中所有的都大於等於0。定義：

如果資料是線性可分的，那麼：

因為線性可分確保了下面兩個式子：

加上我們限制所有的都大於等於0，取max後會讓裡面的東西變成0。

然後可以跟Dual SVM講義第9頁的說明一樣，得到：

接著因為我們確保了三個constraint qualification：線性的限制(linear constraints)、convex primal跟資料是線性可分(feasible primal)，所以上面的大於小於可以是等於：

接著來求取min部分，inner problem的optimal；先對b偏微分求極值：

然後將這個inner optimal代入Lagrange function不會影響最佳性質：

接著對w偏微分求極值：

一樣可以將這個inner optimal代入Lagrange function不會影響最佳性質：

列出primal-dual optimal的四個KKT條件：

* primal feasible
* dual feasible
* dual-inner optimal
* primal-inner optimal

因此最後我們得到我們SVM的dual problem：

或著改成取min：

* 1. Converting without solving the QP problem

在此我先證明，even margin SVM得到的超平面，其兩側必定都存在至少一個支持向量；這裡我使用反證法：

假設SVM最後求得的跟b，有下列的情況：

也就是的資料中最小值並不在邊界上而在邊界外。

為了方便說明，我將跟的最小值的資料分別叫做跟，並且令得到的值叫做 ：

這兩個式子也代表了夾住原本超平面的兩個邊界超平面：

這時我們來去找，在相同的法向量還有的情況下，哪個超平面可以和那兩個邊界超平面之間的距離是1比1：

我們想要找的超平面：

算出離兩個邊界超平面的垂直距離：

這個超平面就是我們新的最佳超平面，但是要做一些調整：

我令這個新的截距部分叫做：

雖然此時這個新的超平面跟那兩個邊界超平面距離是1比1，但的部分不是真的等於1：

這是資料方向的邊界超平面：

這是資料方向的邊界超平面：

可以觀察到他們確實是1比1。所以接著要對進行調整，同除以：

這是資料方向的邊界超平面

這是資料方向的邊界超平面

這是新的最佳超平面

而這個新得到的最佳法向量(權重)：

因為是小於的負數，因此可以知道，所以可以知道這個新的最佳法向量比原本的更好，造成矛盾。

因此可以知道SVM求得的最佳超平面，其兩側必定有支持向量存在。

上面的推論過程，剛好可以拿來計算此題提到的轉換。

原本是從even margin得到的最佳超平面，如果想要得到，方法就跟上面一樣，找到一個新的超平面，他跟原本的兩個邊界超平面之間的距離變成1比1126，所以上面推論過程中，比例的部分就會變成(為了更general我將1126換成了)：

；接著我們可以確認和兩個邊界超平面的距離的確是1比 ：

並且可知需要同除以的係數就是，恰好對應上面推導時的：

所以可以知道，。

將1126代入即可：，。

* 1. optimal solution of the uneven margin

我認為並不會是任何的最佳解，原因在於dual-inner optimal：

從第三題的推論可以知道，只差了常數倍，所以可以知道跟應該會差常數倍，而不是等於的關係，換句話說就是跟應該是差常數倍，而不是等於。(這裡我的就是)

* Operation of Kernels

1. multiplication of valid kernels is still a valid kernel

這裡我令：

所以可以知道：

所以可知新的轉換就是：

1. distances in the Z space

距離公式(平方)是：

使用我們的傳統藝能 kernel trick：

對微分求極值：

所以極值發生在跟的時候，代回去原本的函數可得：

別忘記最後還要開個根號：

最大距離為

最小距離為0

1. Gaussian kernel
2. cosine

所以我們可以構造出一個轉換：

這樣一來：

* Experiments with Soft-Margin Support Vector Machine

1. smallest number of support vectors

根據libsvm的執行結果：

# (0.1,2):860 # (0.1,3):789 # (0.1,4):740

# (1,2):783 # (1,3):721 # (1,4):666

# (10,2):712 # (10,3):659 # (10,4):629

可以發現(10,4)具有最少的支持向量。

1. lowes

我依序傳入[0.01,0.1,1,10,100]，根據libsvm的執行結果：

# Accuracy = 95.4% (1908/2000) (classification)

# 0.01:95.39999999999999

# Accuracy = 98.8% (1976/2000) (classification)

# 0.1:98.8

# Accuracy = 99.5% (1990/2000) (classification)

# 1:99.5

# Accuracy = 99.4% (1988/2000) (classification)

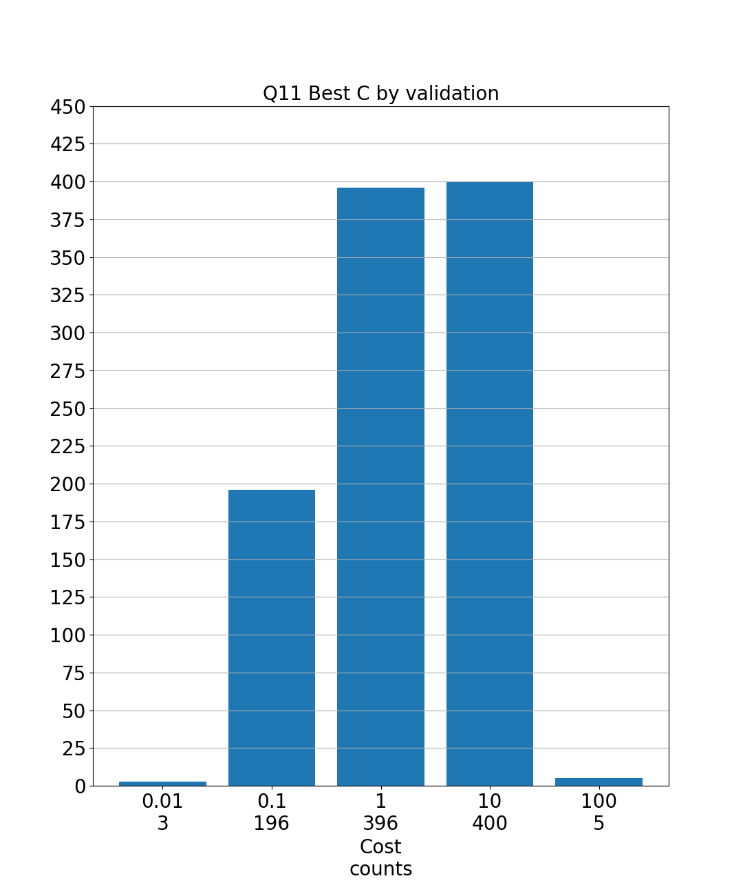
# 10:99.4

# Accuracy = 99.45% (1989/2000) (classification)

# 100:99.45

最低的是。

1. times of each C is selected



可以看到在1000次實驗中，最低大多落在跟。

會這麼低的原因是對錯誤的容忍太低了，導致超平面變得太複雜，發生overfitting。會這麼低則是因為對錯誤的容忍度太高了，大多都是錯的，而則是逐漸提高準確率，並在的時候達到最高……。

**正當我想這樣敘述的時候，仔細回想第十題的正確率：**

# Accuracy = 95.4% (1908/2000) (classification)

# 0.01:95.39999999999999

# Accuracy = 98.8% (1976/2000) (classification)

# 0.1:98.8

# Accuracy = 99.5% (1990/2000) (classification)

# 1:99.5

# Accuracy = 99.4% (1988/2000) (classification)

# 10:99.4

# Accuracy = 99.45% (1989/2000) (classification)

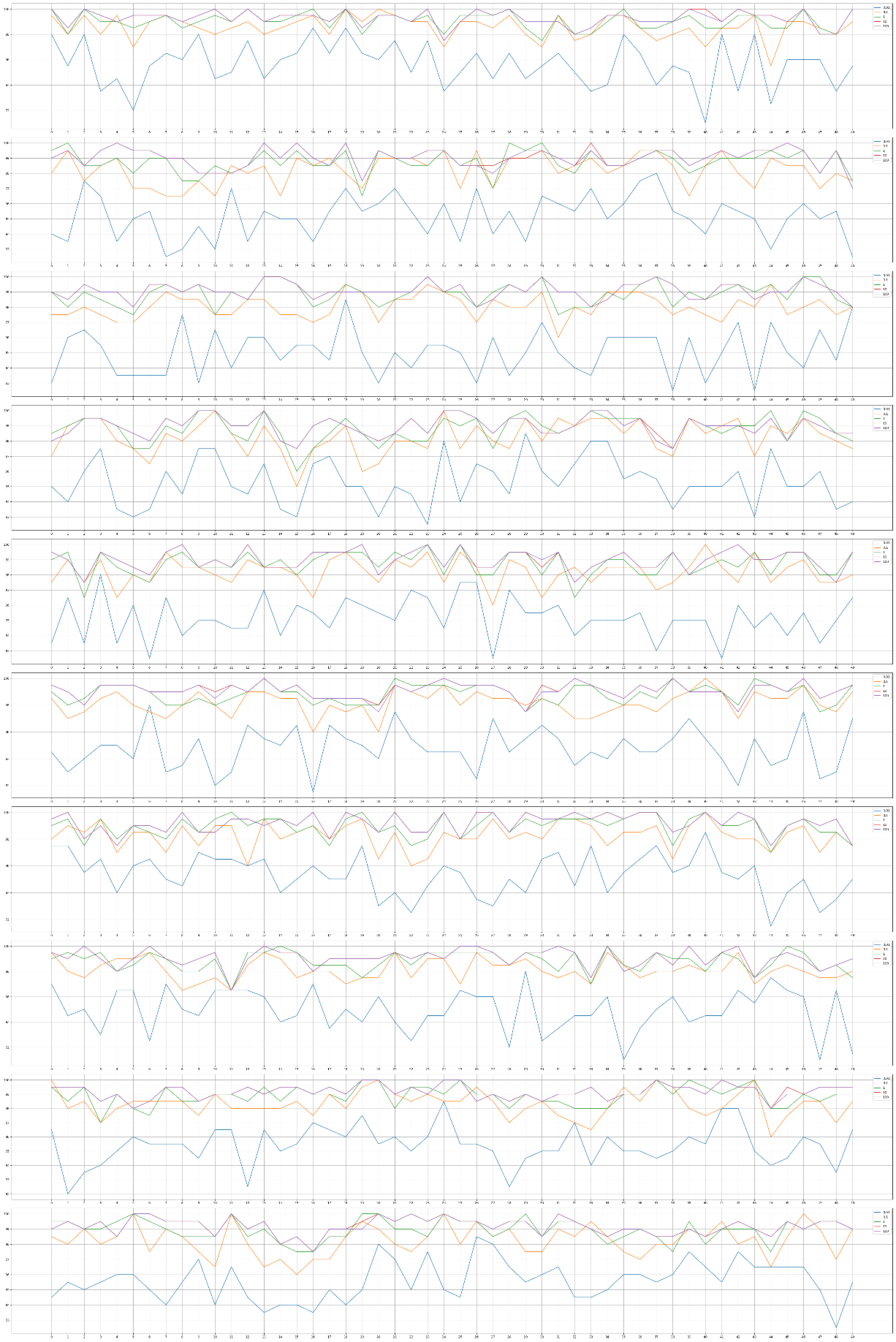
# 100:99.45

發現其實他們準確率都很高，尤其這三個非常接近，於是我就想有沒有可能是因為每次的模擬， 跟 的正確率一樣，但是因為我們都是挑數值小的，所以導致被選的次數才這麼低，因此我將每次實驗得到的數據做出下面的圖表。(在下一頁)

可以看到確實如此，跟甚至跟，三者在模擬中的表現有很多時候都是一樣高的，尤其，他就只有5筆數據是準確率大於其他人，其他都是等於跟一點點的小於，導致他不容易被選到。

這時結合11題，最低的是，我認為是因為模擬的過程中跟出現了「一點點」的overfitting：由於我們都是從同個資料集合中隨機抽200個出來作為驗證資料，經過這1000次模擬下來，就好像是做了很多遍的cross validation，而每次準確率都可以很高，代表模型在這樣手上有的資料的分布可以有良好的汎化能力，不論是怎樣的相同取樣方法，都不會發生overfitting；不過如果測試資料有稍微不一樣的分佈，那麼預測可能就會稍微不準，也就是我認為11題發生的狀況。

但是overfitting的狀況其實真的不嚴重，從那非常高的準確率就可以知道資料是有一定的分佈存在的。



這是前500筆數據

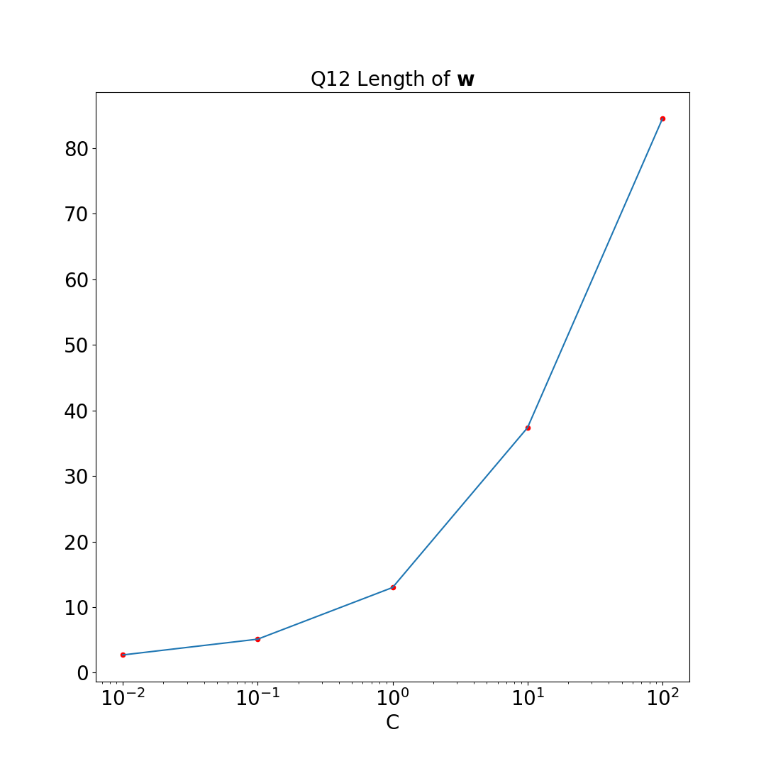
藍色是0.01，橘色是0.1，綠色是1，紅色是10，紫色是100。



這是後500筆數據

藍色是0.01，橘色是0.1，綠色是1，紅色是10，紫色是100。

可以明顯看到綠色紅色跟紫色非常頻繁的重疊在一起，那麼此時就會選小的。



可以發現當C越大，就越大。C越大代表我們對錯誤量的容忍程度越低，而越大代表SVM的margin越小。

因為我們對錯誤量的容忍程度越低，會需要讓超平面更靠近犯錯的資料，所以才會導致SVM的margin變小，導致變大。

* Bonus

1. Dual of Dual

先列出原本的hard margin SVM dual：

首先建構出Lagrange function：

然後經過跟講義一樣的步驟，可以得到Lagrange dual：

接著求inner optimal，對偏微分求極值：

將inner optimal代回原lagrange function當中的，可得：

然後再代回原本lagrange function當中，會發現很多東西都削掉了：

所以我們就得到了dual的dual problem：

如果用 進行替換可以得到

都不見了，所以可以不用寫上了。如果換成取min，我們就會得到：

列出KKT條件：

* primal feasible 跟
* dual feasible
* dual-inner optimal
* primal-inner optimal

可以發現形式就跟以前的primal一樣，所以只要補回primal的限制就會得到SVM的primal problem：

但是還有其背後的一堆KKT條件。