圖形識別與機器學習 –線性交點

系級: 電機碩一 學號: 11278041 姓名: 陳大荃

學校系所: 中原大學 電機工程學系

1. 摘要

仿效 Layer Machines 的運作原理以更簡單的方式呈現其在持續不斷更動 Weight 後期望能夠更貼近最佳解。

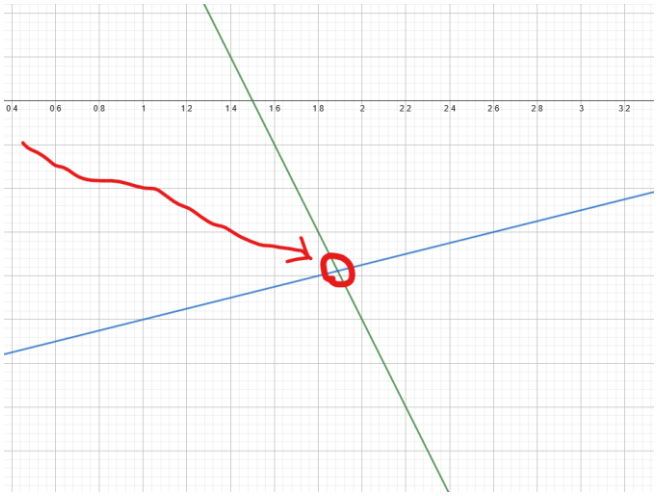

圖 一、以類似機器學習的方式接近兩線交錯點

2. 引言

在機器學習中所使用的數據集可以視為如圖一中的兩線為中心分部，向線的兩側散開。若是演算法能夠找出這兩線的交點，就能成為區分資料群體的最佳解。

3. 方法

實驗器材與對象: 以電腦根據使用者設定的線條數、各線的參數、以及起始點等設定繪製線條及自動以演算法一步步找出最佳解。以 C 及 GNUPlot 實作的程式開源在 GitHub 中，產生的圖檔也公開在 GitHub 中，請點連結前往。

實驗一: 測試多條線的最佳解

預設環境: 請至程式開發文件確認所有軟體的需求

測試方法: 將設定檔中的”line_cnt”設為目前要測試的線條數，接著完成各線的”line_title”, “line_symbol”, “line_param1”, “line_param2”, 及”line_param3”即可設定各線為$line_{param1}x + line_{param2}y = line_{param3}$。

結果 1: 兩線最佳解

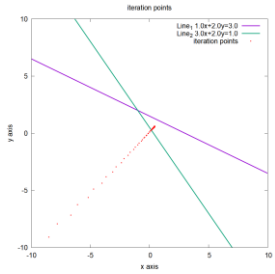

圖 二、兩線最佳解路徑

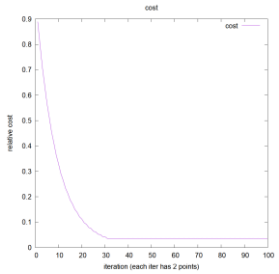

圖 三、最佳解與兩線距離的總和

結果 2: 三線最佳解

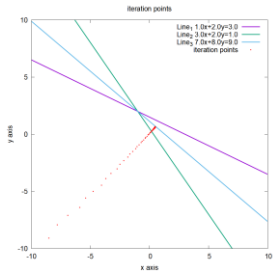

圖 四、三線最佳解路徑

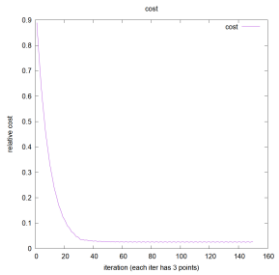

圖 五、最佳解與三線距離的總和

結果 3: 縮放

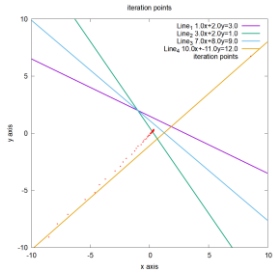

圖 六、四線最佳解路徑

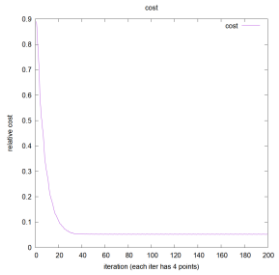

圖 七、最佳解與四線距離的總和

實驗二: 測試多條線不同起始點的最佳解

預設環境: 請至程式開發文件確認所有軟體的需求

測試方法: 將設定檔中的"line_cnt"設為目前要測試的線條數，接著完成各線的"line_title", "line_symbol", "line_param1", "line_param2", 及"line_param3"即可設定各線為$line_{param1}x + line_{param2}y = line_{param3}$。最後設定"initial_x"及"initial_y"即可設定起始點。

結果 1: 兩線以$(-10, -10)$為起始點

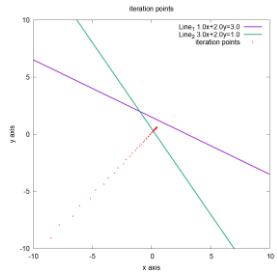

圖 八、
兩線最佳解路徑

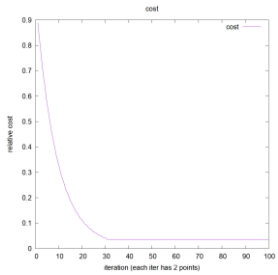

圖 九、
最佳解與兩線距離的總和

結果 2: 兩線以$(10, -10)$為起始點

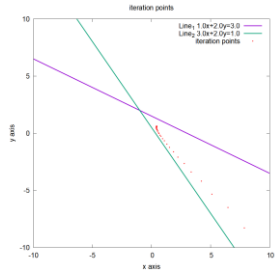

圖 十、
兩線最佳解路徑

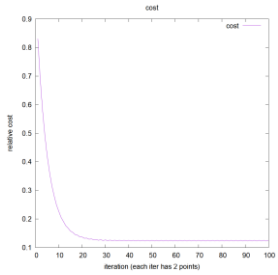

圖 十一、
最佳解與兩線距離的總和

結果 3: 兩線以$(-10, 10)$為起始點

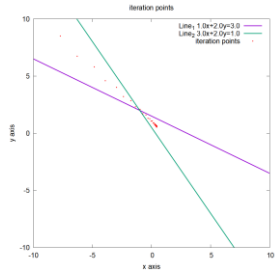

圖 十二、
兩線最佳解路徑

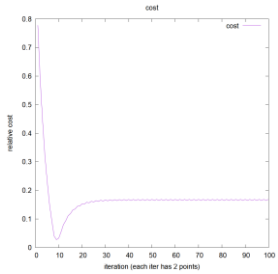

圖 十三、
最佳解與兩線距離的總和

結果 4: 兩線以$(10, 10)$為起始點

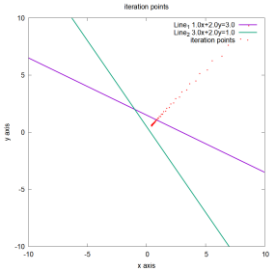

圖 十四、
兩線最佳解路徑

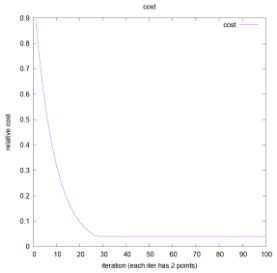

圖 十五、
最佳解與兩線距離的總和

實驗三: 測試不同 step size

預設環境: 請至程式開發文件確認所有軟體的需求

測試方法: 將設定檔中的"line_cnt"設為目前要測試的線條數，接著完成各線的"line_title", "line_symbol", "line_param1", "line_param2", 及"line_param3"即可設定各線為$line_{param1}x + line_{param2}y = line_{param3}$。最後修改"initial_step"即可調整"step size"。接下來三個結果的"max_iter"皆設為 50。

結果 1: $step\ size = 0.01$

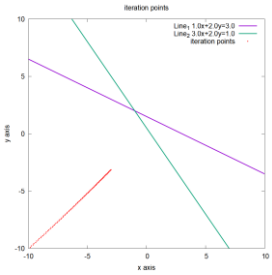

圖 十六、
兩線最佳解路徑

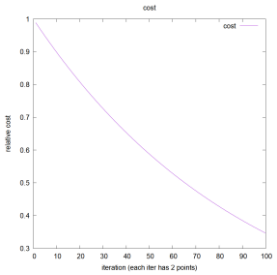

圖 十七、
最佳解與兩線距離的總和

結果 2: $step\ size = 0.1$

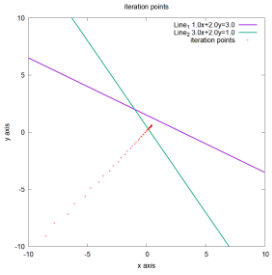

圖 十八、
兩線最佳解路徑

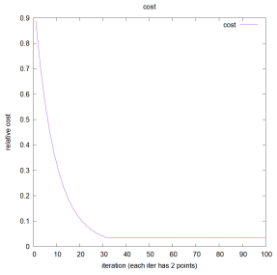

圖 十九、
最佳解與兩線距離的總和

結果 3: $step\ size = 1$

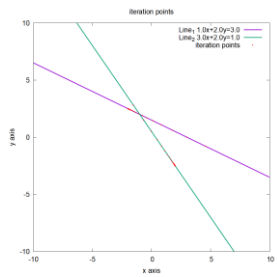

圖 二十、
兩線最佳解路徑

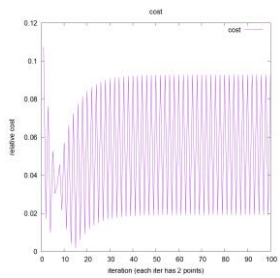

圖 二十一、
最佳解與兩線距離的總和

4. 結果

在實驗一中可以見到不管使用幾條線，演算法都會最終收斂，可能因為四線互相的交點都十分相近，所以最佳解的位置都十分相似。但是同時可以發現這個演算法無法準確的趨近線與線的交點。

實驗二中可以見到從不同方向出發趨近最佳解在兩線的情況中都會趨近相同的最佳解。

實驗三中可以見到若是把每步的距離設得太小，會需要很多次運算，而設的太大則永遠無法趨近最佳解。

5. 討論

若是線條數量非常多，會造成計算量非常大，這時候以幾何的方式計算趨近方向與座標會過於浪費運算效能，希望能夠有更快速及準確的方式進行運算。

6. 結論

由於對於使用幾何計算問題的經驗幾乎都是已經有標準答案的題目，因此在以幾何的方式解這個問題時很難確認是否正確，也不確定計算方式對不對。因此只能依照趨近方向大致判斷演算法的正確性。

7. 參考文獻

甲、https://bytes.com/topic/c/answers/611313-how-open-files-fopen-passed-parameter

乙、https://itecnote.com/tecnote/passing-char-array-as-argument-to-fopen/

丙、https://stackoverflow.com/questions/14785442/typcasting-a-character-array-to-a-const-char

丁、https://www.geogebra.org/graphing?lang=en