給定模型下即時分類遺失部份資訊的資料

(Real-time classification with missing data given models)

一、摘要

隨著大數據時代的來臨，機器學習（Machine Learning）已經逐漸成為顯學而被應用在各大領域裏，但是將機器學習理論套用到現實世界問題中的時候卻難免遇到很多問題，其中一個最直接、嚴重且難以避免的問題就是「資料遺失（data missing)」，不論是偵測裝置的異常或是儲存設備的遺漏，資料一旦出現遺失就很難再被復原成原本的模樣，如此一來將會導致諸多機器學習演算法應用上的困難，例如：類似支持向量機（Support Vector Machine, SVM）的演算法都是嘗試在一高維空間中使用一高維平面來切分資料並進行分類，但是一旦資料出現遺漏，就無法得知該資料會落在高維平面的哪一側，也就無法給出準確的答案。

這次要解決的問題將針對於「一般性（general)」的資料，即是我對於原問題本身不需要有任何的背景知識（domain knowledge)，這樣將有助於我提出的解決方法有效套用到各個領域之中。除此之外，在我的設計模型之下，我將會著重於速度以及效能之間的取捨，使得最後的演算法可以應用到現實問題中 —完成即時填補遺失資訊。

二、研究動機與研究問題

(一) 研究動機：

很多現實世界中的現象或結果都有既定的模式、規律，換句話說，只要我們能夠從中擷取出關鍵性的因素，就能夠準確地預測出未來發生類似事件的結果，機器學習就是一個研究如何能夠更精準地透過機器來抓取抽象模式再進而預測尚未發生事件的結果的一個領域。儘管現今已經有非常多良好、簡易使用的模型被設計出來，例如類支持向量機模型、隨機森林（Random Forest)、GBDT（Gradient Boosting Decision Tree)、類神經網路 (Neural Network) 等，但是一旦資料搜集不完整，這些模型將會很難甚至無法正常運作，這時就需要額外地先做一些預處理（preprocessing)，即預先填補缺失資料。

在現實世界的應用上，資料遺漏的情形可能出現在兩種地方，其一是在過去蒐集來的資料裏，這類資料大多都有觀測到的目標值（target value)，又被稱為已標記資料（labelled data)，一般而言，這類型的問題比較容易解決，因為我們可以離線（off-line）利用高複雜度、高精確度地作法預先整理、歸納這些資料，之後再建立相關的模型來填補遺失的資訊；其二則是出現在將要或是正在搜集到的資料裡，這些資料沒有目標值，又被稱為未標記資料（unlabelled data)，這部分比較困難的原因是：對於每筆新觀測

到的資料我們必須兼顧即時預測的效率以及表現結果，如果採用太過複雜的預處理，將無法做到即時預測，但是相對的，如果使用過於簡單的模型，將會使得預測出來的結果出現很大的偏差，因此如何權衡表現以及時間將是本次研究很重視的一環。

在以往的文獻裡，大多都是專注在解決第一個問題 —如何處理過去蒐集來的資料，鮮少有人研究後者，可是在實務上兩者其實是同樣重要的；即便已經建立高精度的預測模型，如果沒有辦法適當的即時填補未標示資料的遺失資訊，最後模型預測出來的結果仍然會很難令人滿意。

資料遺失的情形不論在什麼領域都很常見, 發生的原因也非常多種, 因此我的研究目的即是找到一個高效率、高精準度的一般性演算法來處理這類型的問題。

(二) 研究問題：

我的研究目標在於找出一個兼顧高效率以及高精準度的演算法來處理未標記資料的資料遺失問題，值得注意的是，原題目敘述中的「分類」是泛指所有追求目標值準確度的問題，包含推估類別以及推估實際數值兩種。在這裡我會作出以下假設與定義：

- 標記與未標記的特徵向量（feature vector）為固定長度
- 遺失的資料之間不存在著相依關係（dependency）
- 建立模型時的資料並無遺失，或遺失的資料已經準確地填補完成；更明確地說，模型已經給定，並忽略建構模型時可能遇到的問題

資料部分包含已標記資料以及未標記資料，在這裡為了機器學習一貫的命名方式，統一稱做訓練資料 (training data) 以及測試資料（testing data)。

- 輸入: 訓練資料, 給予過去蒐集而來的資料 $D = \{d_1, d_2, ...\}$ ，每筆資料 d_i 都由一長度為 L 的特徵向量 v_i 以及一個標記 s_i 構成，且預設蒐集而來的資料都符合上述假設，即所有元素值都是已知確定，不會有遺失資訊的情形。
- 查詢: 測試資料, 給予一筆蒐集到的資料, 該筆詢問 q_i 由一長度為 L 的特徵向量 u_i 構成，元素值可能會有部分的遺失。
- 輸出: 對於每筆詢問都要立即輸出一個最有可能的數值 t_i, 作為對於該筆詢問的預測值。

三、文獻回顧與探討

過去的文獻裡所使用的方法大致上可以被區分為機率模型以及特徵模型；前者是以機率為基礎，利用一些已知條件建構出機率模型的架構，再嘗試最大化整體架構的聯合

機率（joint probability），著名的方法包括：貝氏推估法 (Bayesian Inference)[1]、EM 交替學習演算法（Expectation-Maximization）[2]、多重插補法（Multiple Imputation, MI）[3][4]—傾向分數法（Propensity Score Method）以及蒙地卡羅 - 馬可夫鏈法（Monte Carlo Marcov Chain, MCMC）、連續插補法（Sequential Imputation, SI）[5] 等；後者是以特徵值為基礎，嘗試直接從蒐集來的資料中學到某種適當的加權方式，並推估出未標記資料最接近的答案，著名的方法包含：分類樹（Classification Tree）[6][9]—猜測數值插值（Predictive Value Imputation）以及分佈插值（Distribution-based Imputation）、限縮特徵模型 (Reduced-feature Model) 等。

由於機率模型都需要都需要假設先驗機率（prior probability）屬於（follows）某種分佈 —通常都是高斯分佈（Gaussian distribution）或是常態分佈（Normal distribution），但是這樣的假設卻不見得適用於所有的資料類型，此外，機率為基礎的做法通常都擁有較高的複雜度，很難做到即時的資料處理，EM 演算法雖然可以提早終止最佳化的過程，但是少量迭代的結果卻往往不太理想；因此，在之後模型的建構中我將會傾向使用機率以外的做法。

四、研究方法及步驟

此次的研究目的為如何即時的填補資料遺失的情形，因為我設計了多個模型，分別基於不同的理論假設，大致上可以分為以下兩種類別：

- 協同過濾模型（Collaborative Filtering Model）
- 特徵基礎模型（Feature-based Model）

前者的假設是基於兩筆類似的資料應該會在特徵值上面有類似的分佈，同時兩列類似的特徵向量也會在不同筆的資料中有相近的數值；而後者則是直接將原本的填補遺失資訊問題轉換成一個以特徵為基礎的機器學習問題，透過模型學到的參數比重來推測出適合的預測值。

為了陳述方便，圖 1 是我對於這次研究的系統架構圖，後續則會詳細介紹各個子環節的細節內容：

1. **資料搜集 (Data collection)：**
   我採用來自機器學習競賽 Loan Default Prediction - Imperial College London 的資料做為此次研究的資料集（dataset），有以下主要兩個原因：第一，我曾經參與過那場比賽並且在該比賽中得到不錯的名次，因此我對該資料的本質有比較深刻的瞭解，第二，該資料的真實答案（ground truth）是實際數值（real value），在實驗數據的呈現上會有較好的效果；此外，有效的特徵向量並不多（詳見資料降維章節），適合用來比較不同複雜度做法所需要花費的時間。

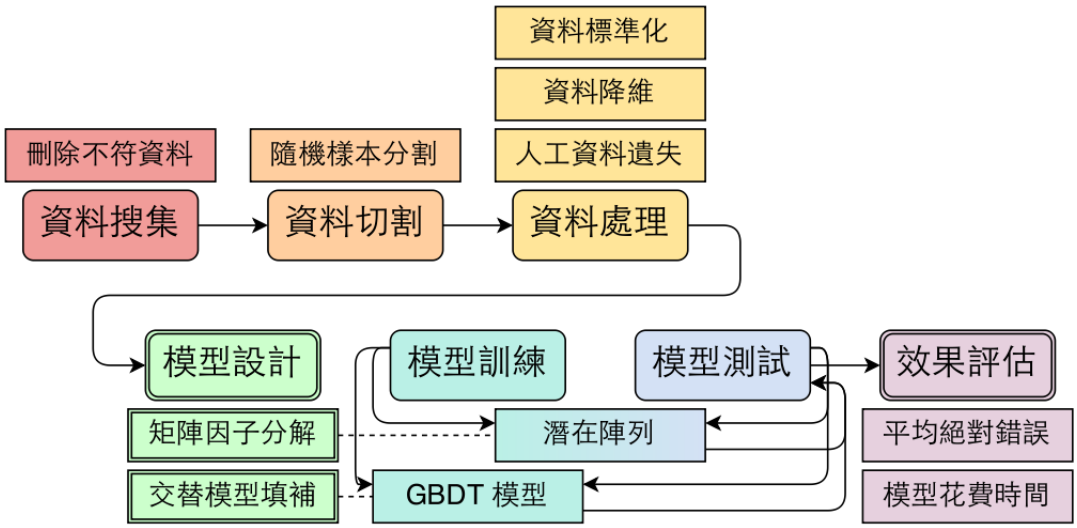


Figure 1: 實驗架構圖

原比賽中帶有目標值的資料有 105472 筆，每筆資料有 770 維特徵值以及 1 維目標值，而資料特徵值有部分遺失的情形發生；為了符合此次研究假設，我決定採用部份刪除法（Partial Deletion）中的條列刪除（Listwise Deletion）方式來處理違背假設的資料，刪除之後剩餘的 51940 筆資料即為我後續實驗的資料集。

整個實驗的評估方式將採用平均絕對錯誤 (mean absolute error, MAE)，即計算每筆資料目標值的預測數值（predicted value）與實際數值（real value）之間的絕對錯誤 (absolute error)，將每筆錯誤的數值相加之後再取平均。

2. **資料切割 (Data partition)：**
為了實驗的公正性，我採用隨機樣本 (random sample) 的方式，將刪除遺失資訊之後所得到的 51940 筆資料以 0.632 、0.368 的比例 [10] 切割成訓練集（training set）— 32826 筆資料與測試集（testing set）— 19114 筆資料；此外，訓練集將會預做隨機排序，如此一來可以避免模型學習到原始資料中可能存在的順序關係。

3. **資料處理 (Data preprocessing)：**

   - **資料標準化（data normalization）：** 為了避免模型對於數字的處理誤差，以及一些方法使用上的假設，我會預先將所有特徵值標準化，即每一特徵值都減去該特徵向量的平均，再除以標準差。
   - **資料降維（data dimension reduction）：** 此外，由於原始資料的特徵向量較多，如果直接讓模型學習整份資料將會大幅降低模型的預測速度以及準確度，因此我預先進行特徵向量降維的操作；一般而言，特徵向量降維常用的方法有主成份分析（Principal Component Analysis, PCA）以及 K-近鄰演算法（K Nearest Neighbor, KNN），兩者的核心概念均是合併並消除多餘的特徵向量來達到降維的目的。

然而，這兩種方法並沒有直接對最後的目標值進行優化，換句話說，假設原始資料中有意義的特徵向量並不多，合併降維之後還是沒有辦法提升模型的準確度；因此在這裡我提出一個特設（Ad-hoc）演算法：枚舉任意兩個特徵向量當作特徵集並算出其預測值和實際數值之間的誤差，利用累計誤差的方式計算出哪些特徵向量能夠最有效率的降低誤差值，最後選取最好的 20 維特徵向量當作之後訓練時的特徵集。

- **人工資料遺失（human-made data missing）：** 為了題目的一般性，我假設所有的資訊遺失都是隨機遺失 (missing at random, MAR)，即遺失的資料之間並不存在著相依關係；此外，為了提高實驗的顯著性，我刻意設定測試資料的隨機遺失比例，之後的實驗裡將分析不同遺失率對於不同作法的影響。測試集的遺失機率為 r，即代表該資料集的每個特徵值均有 r 的機率出現遺失，而在測試資料中，目標值保證未知。

4. **模型設計 (Model design)：**
針對此次實驗，除了一些基準演算法 (baseline algorithm) 外，我提出兩個新穎的解決方法來填補遺失資訊，第一種是將協同過濾（Collaborative filtering）中的一種著名算法 —矩陣因子分解（Matrix Factorization）應用到此問題中，第二種則是使用交替模型填補 (alternating model imputation) 的方式來解決資料缺失問題。以下會對於模型的設計動機、設計假設、整體框架做出詳細的解釋。

   為了討論方便，先在這裡統一符號定義：一共 N_{tn} 筆訓練資料、N_{tt} 筆測試資料，每筆資料有 M 維特徵值，以及 1 維目標值。

   - **矩陣因子分解 (Matrix Factorization)：**
     - **設計動機：** 缺失的資料資訊只能透過其他未遺失資料取得，因此最直覺的解決方法即去歸納、擷取資料背後的潛在模式（latent pattern），透過將一個稀疏、低秩的特徵矩陣拆解成兩個較低維的矩陣相乘，遺失的資訊可以重新被推測、填補出來。
     - **設計假設：** 使用矩陣因子分解有以下幾個假設：第一，該特徵矩陣必須足夠稀疏以及低秩，如此一來才能使用較低維的隱藏陣列描繪出隱藏在資料背後的模式；第二，兩個相似的使用者應該會有類似的表現行為，即他們在某個特定的特徵上面會有相近的數值；第三，兩個相似的特徵應該會有類似的分佈情形，即在將特徵數值標準化（normalization）之後，他們在某個特定的使用者上面會有相近的數值。
     - **整體框架：** 原始資料符合前兩項假設，但是由於不同的特徵向量可能會有不同的數值規模，因此資料標準化在這裡扮演著重要的角色；在標準化後，我會將每一筆的測試資料接附在完整的訓練資料後面，並把該新構造出來的矩陣拆解成多個長度為 L 的潛在使用者陣列 $P \in \mathbf{R}^{N_{tn} \times L}$ 以及長度同為 L 的潛在特徵陣列 $Q \in \mathbf{R}^{L \times M}$。

- **交替模型填補 (Alternating Model Imputation)：**
  - **設計動機：** 類似矩陣因子分解的概念，不同點在於局部資訊的填補是使用特徵為基礎的模型來解決，如果某一維的特徵可以藉由其他維的資訊來做推測，在遺失狀況並不嚴重的時候，此種方法應該會有良好的表現。
  - **設計假設：** 此方法的核心概念是藉由其他特徵值填補遺失特徵值，故資料遺失情況不能過於嚴重；如果一開始就有過多的資訊遺失，填補出來的效果將不如預期。此外，由於每一維特徵向量都需要訓練出一個預測模型，大量的訓練資料才能保證足夠的準確度，因此我假設：$N_{tt} \gg M$。
  - **整體框架：** 首先，替每個特徵向量都訓練出一個模型，即對於第 i 個特徵向量，利用第 1 到 $i-1$ 以及 $i+1$ 到 M 維的特徵向量當作新的特徵集，並將第 i 維的特徵向量當作目標值，進行模型訓練；在 M 次的操作之後，我可以得到 M 個預測不同特徵集的模型。

    接著對於每一筆測試資料 j 都紀錄遺失特徵值的位置，並填上一隨意初始數值，我選用的初始數值是該遺失值在原訓練集裡特徵向量的平均值；之後再進行 T 次的迭代填值，預期數值將逐漸收斂到一個合理的狀態。有兩種可能的做法，其一是將目標值也視為一種特徵值來填補，如此一來在填補過程後也就得到了目標參數的推測值，其二是先將特徵值做適當的填補之後再去推估目標值，在後面的實驗中會比較此兩種做法的結果差異。為了更精準的闡述我的想法，以下是虛擬碼（pesudo code）：

Algorithm 1 Model Mutual Imputation

$Learn(DataX, Label) : Model$
$Guess(DataX, Model) : Prediction$
function LeaveOneOut_ModelBuild(D_{tn}) ▷ $D_{tn} \in \mathbf{R}^{N_{tn} \times M}$
    models = []
    for $i = 1$ to M **do**
        $models[i] = Learn(D_{tn}[:, 1 : i-1] + D_{tn}[:, i+1 : M], D_{tn}[:, i])$
function Iter_Impute(D_{tn}, $models$, D_{tt}) ▷ $D_{tt} \in \mathbf{R}^{N_{tt} \times M}$
    for $i = 1$ to N_{tt} **do**
        $MissID = FindMissID(D_{tt}[i, :])$
        for j in $MissID$ **do** ▷ Initialize missing value
            $D_{tt}[i, j] = mean(D_{tn}[:, j])$
        while not converge **do** ▷ Iteratively update value
            for $j = 1$ in $MissID$ **do**
                $D_{tt}[i, j] = Guess(D_{tn}[:, 1 : i-1] + D_{tn}[:, i+1 : M], models[j])$

5. **模型訓練 (Model training)：**

- **矩陣因子分解 (Matrix Factorization)：** 除了潛在陣列的長度之外，此種做法並不需要額外的參數，唯迭代時候所使用的梯度距離（gradient distance）不能過大，因此在訓練過程中也相對簡單。

- **交替模型填補 (Alternating Model Imputation)：** 此作法需要花費的訓練時間以及參數選擇上的複雜度與所使用的模型種類高度相關，在此次實驗中，我使用 GBDT 作為我的模型，該模型是一個樹狀模型，比較顯著的參數有樹的數量、交互深度（interaction depth）以及收斂速度（shrinkage）；為了得到更為精準的預測，我使用格子狀搜索（grid search）在交互驗證（cross validation）上找到一組表現最佳的參數。
  最後的參數為：樹的數量 5000 棵，交互深度 8，收斂速度 0.025 。

6. **模型測試 (Model testing)：**
   以平均絕對錯誤作為估計錯誤的方式，除了模型表現之外，模型測試的時間也會被納入考慮，期望能夠提供實務上可行的填補缺值框架（framework）。

五、實驗數據與討論

除了建構出前述的兩種模型—矩陣因子分解以及交替模型填補外，我還會將「GBDT 演算法本身接受遺失資訊的機制（後稱 $GBDT_0$）」、「缺失資訊由平均值填補（後稱 $MEAN$）」、「最後目標值只輸出 0（後稱 $ZERO$）」等三種基準作法納入我的實驗比較中。我將比較資料在不同資訊缺失比例下的表現結果（遺失率從 0% 到 50% ），以下將詳述各個圖表以及數據。

為統一實驗結果，設備一致假設為 2.30GZ 單核 CPU；時間估計可能會有些微誤差。

- **矩陣因子分解 (Matrix Factorization)：** 在這個方法中，我遇到我當初設計架構時預期以外的結果—該模型收斂速度比我想像中的還慢得多；由於在此次實驗之中訓練資料佔有超過六成的比例，對於每一筆資訊都做完整的矩陣因子分解顯然是一件非常花費時間的工作，因此在這裡我提出一種與時間妥協的做法，即每次只採用 10% 隨機抽樣的訓練資料，如此一來可以大幅降低模型的時間複雜度。

  - **模型表現：** 該模型所得到的結果跟我想像的有很大的落差，如圖 3(a)，甚至比 $MEAN$、$ZERO$ 的表現還差；推測可能的原因是：一般所謂的矩陣因子問題的資料都是低秩的，但是這與我研究題目的假設–訓練資料完整並無遺失–是相違背的。第二個原因是為了符合即時預測的要求，並沒有辦法拆解完整的訓練資料，因此潛在陣列所隱含的資訊量不足以預測出準確的目標值。
  - **模型花費時間：** 即便只有 10% 的訓練資料拆解成潛在陣列，由圖 3(b) 可以發現，它所花費的時間仍遠超過其他模型，卻無較好的預測結果。
    * 遺失率 0% 時：平均時間：5.28892 秒，平均絕對錯誤：1.428092。
    * 遺失率 25% 時：平均時間：5.22684 秒，平均絕對錯誤：1.45111。
    * 遺失率 50% 時：平均時間：5.20526 秒，平均絕對錯誤：1.467771。

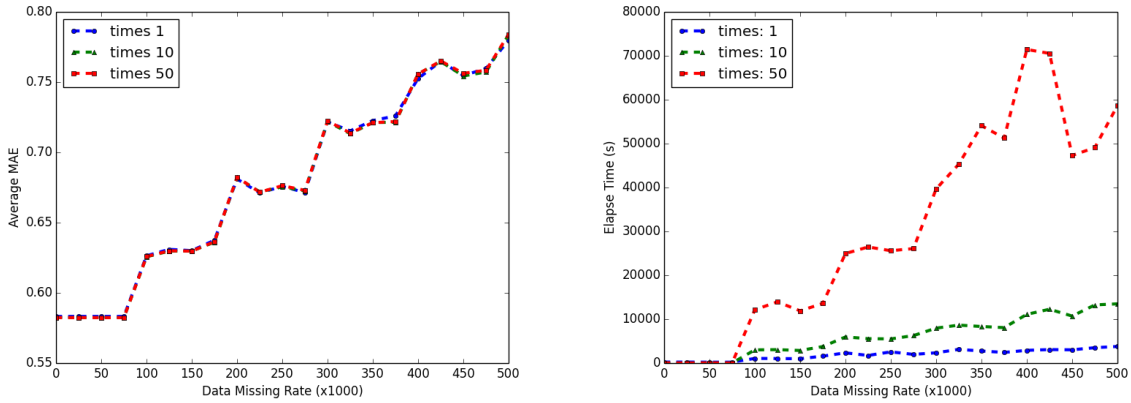
- **交替模型填補 (Alternating Model Imputation)：**
  - **模型表現：** 在這裡有三種不同的模型需要比較，除了 GBDT 演算法本身接受遺失資訊的機制（$GBDT_0$）外，我提出的兩種做法分別稱作 $GBDT_1$ 以及 $GBDT_2$。

    GBDT 本身就能夠接受缺失部份資訊的資料，這也是我決定選用這個模型當作特徵模型的原因之一，它所採用的解決方法是：對於樹上每一個點都會切分成左子樹、右子樹以及缺失樹 (missing tree)，之後再使用決策樹用替代分裂點（surrogate splits）的方式來處理缺失特徵值 [11]；即便 GBDT 在實務上是一個能容忍缺失資料並擁有良好表現的模型，實驗證明我提出的做法在各種缺失資訊的情形下皆有較好的表現。

    第一個我採取的做法是將目標向量也當作一列缺失的特徵向量，由圖 $2(a)$ 可以看見實驗得到的結果非常糟糕，甚至當遺失率稍高時，兩個最基本樸素（naïve）都有較好的表現；我推測可能的原因是：該目標值本身數值就是個比較難以預測，是一個比較困難的問題，在其他數值尚未確定前就去預測該數值可能會有很大的偏差，而該偏差又會繼續導致之後的數值產生更多的偏差，即錯誤傳遞（error propagation）現象。

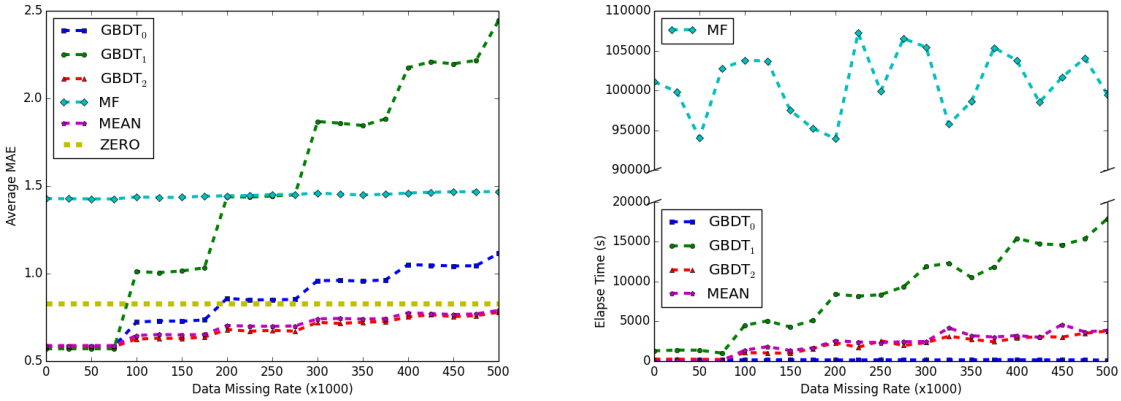
    第二個做法是先填補所有缺失的特徵向量，之後再利用這些完整的特徵集去預測目標向量，從圖 $2(a)$ 可以發現：即便整體的資料缺失比例已經高達 50%，最後得到的平均絕對錯誤才僅有 0.78 左右。另外，由圖 $2(a)$、$2(b)$ 可以看出迭代次數的增加並沒有讓表現有明顯的進步，但是卻造成極大的時間負擔，為了討論簡便，在往後的討論中我將只使用迭代次數為一次的版本。

    從圖 $3(a)$ 可以看見，我所提出的 $GBDT_1$ 以及 $GBDT_2$ 所得到的結果都比 GBDT 本身解決缺失資訊資料的表現還要好。
  - **模型花費時間：** 圖 $3(b)$ 可以看出來 $GBDT_0$ 需要花費的時間最少，且幾乎與缺失資料程度無關，每筆測試資料所花費的時間大約是 0.009207 秒；而 $GBDT_1$、$GBDT_2$ 雖然需要花費較多的時間，但是在應用上仍是可被接受的。
    * 遺失率 0% 時：$GBDT_0$ 平均錯誤：0.583252；
      $GBDT_1$ 平均時間：0.068360 秒，平均絕對錯誤：0.570968；
      $GBDT_2$ 平均時間：0.012672 秒，平均絕對錯誤：0.583252。
    * 遺失率 25% 時：$GBDT_0$ 平均錯誤：0.850592；
      $GBDT_1$ 平均時間：0.436940 秒，平均絕對錯誤：1.443071；
      $GBDT_2$ 平均時間：0.134019 秒，平均絕對錯誤：0.675757。
    * 遺失率 50% 時：$GBDT_0$ 平均錯誤：1.115385；
      $GBDT_1$ 平均時間：0.937377 秒，平均絕對錯誤：2.444178；
      $GBDT_2$ 平均時間：0.198244 秒，平均絕對錯誤：0.779937。



(a) 模型結果表現　　(b) 模型花費時間

Figure 2: $GBDT_2$ 在不同迭代次數下的行為



(a) 模型結果表現　　(b) 模型花費時間

Figure 3: 各種模型之間的行為比較

六、結論

在這次的研究中我提出三種不同的做法，他們分別是：MF、$GBDT_1$ 以及 $GBDT_2$，MF 是協同過濾模型中的一種，其嘗試透過分解訓練資料找到有效的潛在向量，來預測那些遺失的特徵值；後兩者都是根據 GBDT 模型來捕捉可能存在的資料模式，差異在於 $GBDT_1$ 將目標值也視為一種遺失的特徵向量，在填補缺失特徵值是也會一併填補目標值，$GBDT_2$ 則會先將特徵集填補完成之後再去推估目標值。

從實驗結果可以看出，我提出的 $GBDT_2$ 在任何資料缺失的情形下均擁有最好的表現，甚至打敗 GBDT 模型內預設處理缺失資訊的做法。以缺失 25% 資訊的情形為例，其平均絕對錯誤為 0.676，僅是完全無缺失資訊的預測結果錯誤率 0.583 的 1.158 倍。除此之外，對於每一筆新增的詢問，該模型在單核運算能力上平均只需要 0.134 秒就能夠給出結果，完全符合我們在現實生活中對於「即時」的期待。

我提出的做法僅是一個泛用的框架，GBDT 模型可以被抽換成任何其他以特徵為基礎的模型；這份研究成果將大大提升實際生活中對於缺失部份資訊的資料即時預測的品質，相信將可以很快地在各大領域中被廣泛使用。

七、參考文獻

[1]: George E. P. Box & George C. Tiao(1973). Bayesian Inference in Statistical Analysis, Wiley Classics Library Edition Pulished 1992

[2]: Arthur Dempster & Nan Laird & Donald Rubin(1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No. 1 (1977), pp. 1-38

[3]: Donald B. Rubin(1987). Multiple Imputation for Nonresponse in Surveys, Department of Statistics Harvard University

[4]: Joseph L. Schafer & Maren K. Olsen(1998). Multiple Imputation for Multivariate Missing-Data Problems: A Data Analyst's Perspective, Multivariate Behavioral Research, 33:4, 545-571, DOI: 10.1207/s15327906mbr3304_5

[5]: Augustine Kong & Jun S. Liu & Wing Hung Wong(1994). Sequential Imputation for missing values, Computational Biology and Chemistry 31 (2007) 320–327

[6]: Saar-Tsechansky, Maytal & Provost, Foster(2007). Handling Missing Values when Applying Classification Models, Journal of Machine Learning Research. 7/1/2007, Vol. 8 Issue 7, p1625-1657. 33p. 1 Diagram, 8 Charts, 13 Graphs

[7]: Steffen L. Lauritzen(1995). The EM algorithm for graphical association models with missing data, Computational Statistics and Data Analysis, Volume 19, Issue 2, February 1995, Pages 191–201

[8]: Martin A. Tanner & Wing Hung Wong(1987). The Calculation of Posterior Distributions by Data Augmentation, Journal of the American Statistical Association, Computational Statistics and Data Analysis, Volume 82, Issue 398, 1987

[9]: Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[10]: Bradley Efron , Robert Tibshirani. Improvements on Cross-Validation: The 632+ Bootstrap Method. Journal of the American Statistical Association

[11]: Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. Classification and Regression Trees. January 1, 1984