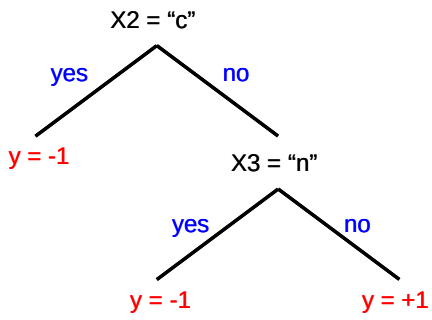
整體學習*(Ensemble Learning)*入門

1. 簡介

“監督學習*(Supervised Learning)*”是各種統計學習方法中最單純，最容易理解的形式。一般而言，監督學習的正規定義可以這樣來描述。每筆資料點*(data point)*是由一個特徵向量，我們以 X 表示之，和一個類別標籤*(class label)* y 所組成；同時，假定有一個未知*(underlying)*的函式 f 存在，對於每一筆訓練的資料點 (X, y) 來說，$y = f(\mathrm{X})$ 是恆成立的。於是學習演算法的目標就是要找一個令人滿意的近似函式 h，並使得針對任何一筆新增的特徵向量 X_{new} 所求得的類別標籤 y_{new} 可以愈接近原始函式 f 計算的結果。這個近似函式 h，我們就稱爲分類器*(classifier)*，如此命名的原因，是因爲它可以將輸入的特徵向量分發或歸類到某一個真實或接近真實的類別。監督學習能被應用於很多的問題上，包括手寫辨識、醫學診斷和部分語音或文字的標籤處理。

傳統機器學習運算的原理是靠搜尋*(search)*，透過搜尋所有可能函式構成的空間集合，找出一個最逼近於未知函式 f 的近似函式 h，也就是假說*(hypotheses)*。無可避免的，一個學習演算法在搜尋的過程中爲了決定哪一個近似函式是比較接近於真實的函式，因此必須具備有測量近似函式 h 和未知函式 f 在訓練的資料點上有多相稱*(match)*，或者能以任何與問題相關的知識進行檢定近似函式 h 和未知函式 f 有多一致*(consistent)*的能力。

搜尋一個由決策樹*(decision tree)*構成的假說空間*(hypotheses space)*可算是最有效率和使用最廣泛的學習演算法之一。圖(1)是一個決策樹的範例，這個決策樹能以下列的步驟將一個新的資料點分類。起初，從所謂的"根部*(root)*"開始，我們首先檢查 x_2 ="c" 是否爲真。如果檢查的結果成立，則隨著左側成立的分枝*(branch)*到達 $y = -1$ 的"葉部*(leaf)*"。如果不成立，則隨著右側反對的分枝，並進行另一次檢驗，即 x_3 ="n" 是否成立。若檢查的結果成立，同樣的，則隨著左側分枝到另一 $y = -1$ 的葉部。反之若不成立，就隨著右側的分枝到 $y = +1$ 的葉部。



圖(1) 決策樹

決策樹學習演算法透過第一個考慮到的特徵，搜尋預先定義好的樹空間。每次試驗，其實只檢查選擇的那個特徵（上述例子選取到的特徵是 x_2 ）並且立即分類。接著透過使用另一個特徵的試驗，繼續延伸樹空間的搜尋（上述例子，右側分支的試驗就是使用特徵 x_3 替代之）。在一次又一次迭代試驗的過程中，各種各樣的啓發式探索法*(heuristics)*常被用於選擇哪項特徵比較適合於試驗，以及什麼時候適合應該停止搜尋樹空間，目前有諸多的研究方法皆能針對不同的問題改善搜尋假說空間的速度與效果。

從決策樹學習演算法的觀點來看，整體學習演算法*(ensemble learning algorithms)*是一種很不同的方法。整體學習演算法不會像決策樹學習演算法只是去尋找一個最好的假說來解釋數據，而是建造一組由多個假說組合而成的整體假說（有時也稱爲一個"委員會"），然後再採用一些模式讓這些多個假說進行"投票"，以推測*(predict)*新增資料點的合理標籤。更確切的說，整體學習的方法是建造一組假說 $\{ h_1, h_2, ..., h_k \}$ ，並選擇一組權重 $\{ w_1, w_2, ..., w_k \}$ 。接著，再依據假說與權重產生一個"投票決定"的整體分類器 $H(\mathrm{X}) = w_1 h_1(\mathrm{X}) + ... + w_k h_k(\mathrm{X})$ 。整體分類器最後決定的結果，等於是結合了整體假說中每一個別假說的結果與個別的權重。如果 $H(\mathrm{X}) \geq 0$ ，就是說整體分類器選擇了 $+1$ 類別標籤，反之就是 -1 。很多實驗結果與研究報告都已顯示了整體方法經常比任何單一的假說要來得準確。例如，Freund和Schapire在1996年針對22個基準問題*(benchmark)*進行測試，其中一個問題的執行結果差不多，四個問題的結果比較差，但其它問題的表現結果都有明顯的改進。

2. 整體方法爲什麼能起作用

一般來說，輸出結果只產生一個假說的學習演算法普遍都會遭遇三個嚴重的問題：統計問題、計算問題和代表性問題。然而，這些問題通常是可以透過整體學習的方法加以解決的。

當學習演算法搜尋一個訓練資料*(train data)*數量過於龐大的假說空間時，就會產生所謂的統計問題。在這種情況下，由於可取得的資訓練料過多以致於可能會有數個不同的假說皆提供訓練資料數據上相當程度的準確度，但學習演算法卻又被強迫必須冒著風險從這些可能性極高的假說中挑選一個。於是，被選取的假說就具有某程度以上的偏頗，因此將導致可能無法準確地預測未來每一個新的資料點。所以，一種針對所有分類器所進行之簡單、平等的投票機制，將可有效的降低這種風險。

計算問題常出現的時機，常常是因爲在學習演算法不能保證能從假說空間裡找到一個最好的假說的時候。例如在神經網路*(neural network)*和決定樹演算法

中，要尋找一個最符合*(best fit)*訓練資料數據的假說，從計算機計算能力的角度來評估，絕對是不可能實現的*(intractable)*，因此勢必要採用所謂的啓發式探索方法來達成。然而，有一些像坡度降下*(gradient descent)*的探索方法，又常會被卡在本地最小量*(local minima)*的地方，也就是說，很多啓發式探索方法，理論上是無法找到最好的假說。就像統計問題一樣，如果利用加權方式，將多種不同的本地最小量結合起來作爲輸出，事實上是可以有效降低選擇錯誤而陷入本地最小量的危險。

最後，當假說空間不包含任何近似真實函式 *f* 的好假說時，代表性問題就出現了。有時候，給予個別假說不同的權重，透過加總的效果所擴大的函式空間，是可以產生具有代表性的假說。換句話說，假若提供不同權重的投票方式給每一個假說，整體學習演算法是很有機會在一個沒有代表性的假說空間裡，找到一個非常準確且逼近真實函式 *f* 的近似值。

若學習演算法有統計問題時，我們就說這個方法具有高的"變異*(variance)*"。如果有計算問題的話，我們則描述它是一個有高的"計算量變異*(computational variance)*"。另外，若是學習演算法有代表性問題，我們即稱它具有高的"偏差*(bias)*"。大致來說，多數的研究報告中證實了整體方法能降低學習演算法的偏差和變異。

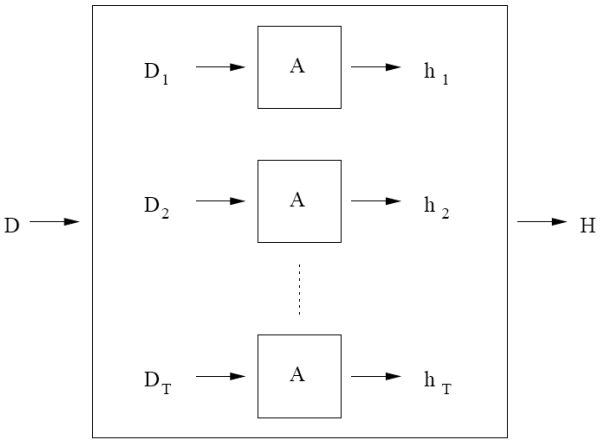
3. Ensemble 演算法回顧

整體學習演算法的運作是透過多次執行"基礎學習演算法"，並且針對每次產生的假說進行投票，最後整合投票的結果構成一致同意的假說。設計整體學習演算法的技巧有兩種主要的方法。

第一種方法是用獨立的模式去建造每一個假說，這些假說形成的組合具有多樣化的特性，而且每一假說的準確度也是完全不同。換句話說，每一單獨的假說對於新資料點的預測，具有某一個合理低的出錯率，但是假說和假說之間，在大多數預測裡常常是彼此不一致的。如果能夠統合單獨假說的預測，並建立一個具有整體性、一致同意的假說時，比起任何一個單獨或個別的分類器來說，一定具有更高準確度的預測表現。理由很簡單，因爲那些對於準確度具有影響力的爭議點，將比較容易互相被取消掉。於是最後的結果，就是大家都較爲同意的部分。很明顯的，這樣的整體學習方法不就是可以順利解決上面所提到的統計和計算問題嗎。

第二種設計整體學習的方法是用採用連接模式來建造假說。連接模式的意思是說，把權重高的票投給和實際資料誤差小的假說，然後把權重低的票投給和實

際資料誤差大的假說，藉由不同權重的投票方式結合所有的假說，並產生一個比任何單獨假說都逼近實際資料的整體假說。這種方法所生成的整體假說，不就可直接解決上面討論所提及到的代表性問題嗎。



圖(2) 引導總計(Bagging)方法圖解

3.1 Methods for Independently Constructing Ensembles

改變學習演算所需的訓練資料是一種能讓學習演算法強迫建造多個不同假說最簡單的方法。過程中，雖然每次執行相同的"基礎學習演算法"，但因爲提供的訓練資料不同，最後所產生的假說，彼此之間或多或少都具有某種程度的差異。例如，在1996年Breiman提出的Bagging*(Bootstrap Aggregating)*方法，其運算過程介紹如圖(2)。給一組包含 *m* 個用以訓練的資料點，Bagging方法在每次迭代過程中，透過來自 *m* 個訓練的資料點均勻取樣所得到的資料點替換原先的資料點，而建立新的、重新取樣過的訓練資料。很顯然地，這些新造的訓練資料中，可能有些資料點是重複出現許多次，也可能有一些原本的資料點根本不再出現。如果學習演算法具有不穩定*(unstable)*的性質的話，不穩定性質的意思是指，如果稍微改變少許訓練的資料點的話，就將對假說產生的結果造成很大的變化。在這種情況下，使用Bagging的方法將會產生一個包含多樣化特性的整體假說（即單一假說和假說之間的差異大）。

第二種迫使學習演算法產生多樣化特性的方法是在每次呼叫學習演算法時，都採用一個具有不同輸入特徵的子集合。例如，在維納斯*(Venus)*上鑑定火山的一項工程裡，Cherkauer(1996)訓練一個由32項神經網路構成的整體系統。這32項單元網路分別由具有119可取得輸入特徵組成的8個不同子集合及4個不同的網路規模所構成。利用不同的影像處理操作方法（例如主成份分析和快速傅立葉變換）去選取輸入特徵的子集合，最後形成一個群體性的特徵，最後導致的整體性的分類結果比任何個別神經網路處理的結果更爲準確。

強迫產生具有多樣化特性學習演算法的第三種方法是去操作訓練的資料點的

輸出標籤。Dietterich和Bakiri(1995)描述一種稱爲改正錯誤的輸出編碼*(error-correcting output coding)*技術，假設類別的數量C是一個很大的數，接著，把C個類別隨機分割成二個子集合A_k和B_k，若原來的資料點被分在子集合A_k裡，其類別標籤就重新標示成-1，反之若是在子集合B_k中就重新標示爲$+1$。於是，透過重新賦予新類別標籤的資料點與重新執行學習演算法，即可建造一個新的分類器h_k。若一直重複這過程K次，就可以取得$h_1, h_2, ..., h_k$等K個分類器結合成的整體分類機制。現在給予一個向量特徵爲X的新資料點，我們如何將它分類呢？答案其實很簡單，只要讓每分類器h_k根據向量特徵X進行分類，比較多分類器同意的類別標籤就是新的資料點應該被指定的類別標籤。也就是說，如果$h_k(\mathrm{X}) = -1$，則屬於子集合A_k裡所有的原始類別皆可取得一票；如果$h_k(\mathrm{X}) = +1$，則原先被歸納在子集合B_k裡所有的類別皆可獲得一票。重複這樣的程序，最後經由分類器全體投票的結果，總共獲取最高票的類別就是該新增資料點的類別標籤。

第四種產生較高準確度及具有多樣性特質的整體分類方法是在學習演算法中加入偶然性*(randomness)*。例如，執行backpropagation演算法時，每次都從不同隨機設定的權重開始；其它像決定樹演算法在選擇分支時，也可以在特徵向量的選取與判斷的門檻上添加一些隨機的成份。Dietterich(2000)的研究結果指出隨機化排列樹的方法可以讓33項測試基準項目中14項的效能有明顯改善，另外其它19項的效能並沒有明顯的變化。Hao(1998)針對成長的決策樹收集*(grow collections of decision tree)*，即森林決定*(decision forests)*，提出"random subspace"方法。這種方法在樹的每個節點上利用隨機選擇一個特徵的子集合，並且經由成長樹的演算法*(tree-growing algorithm)*從這個子集合之中選擇決定分支的規則。研究報告中指出，在16個測試基準項目有明顯的性能改進。Breiman(2001)結合Bagging和random subspace方法去處理漸增隨機森林決定*(grow random decision forests)*的問題，也獲得極好的性能改善。

3.2 Methods for Coordinated Construction of Ensembles

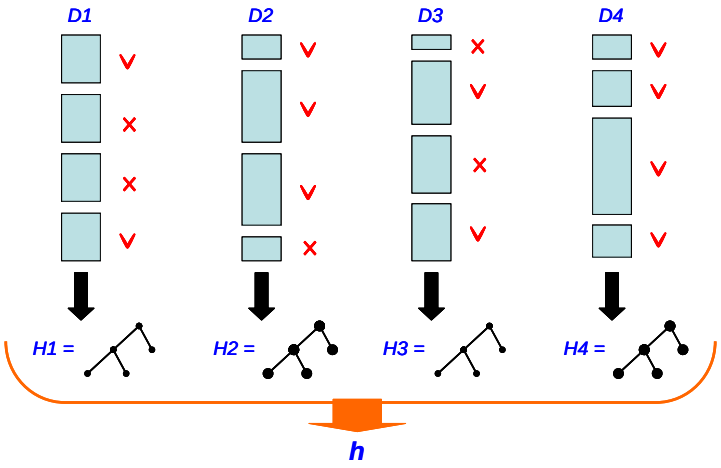
綜合以上所述的全部方法，採用整體學習產生的假說，無論是透過獨立改變輸入、輸出或特徵，或者是利用注入偶然性的方式，最後決定的分類結果都是由一個沒有賦予權重的投票機制所確定下來的。

從另一個角度來看，整體學習也可以是一種附加模型*(additive model)*。所謂的附加模型通常是指一個新增的資料點，最後所指定的類別標籤，是由部分或所有的附屬模型*(component model)*經由賦予不等的權重後，再加總所得到的結果。這種作法，對於演算法來說，其實只要選擇適當的附屬模型以及適當的權重，就可利用加總權重的效果達到符合數據*(fit data)*的目的。然而，選擇一個組成的附屬模型或者一個看似微不足道的權重，其實就可能足以影響到選擇其它的附屬模型

與最後附加模型的結果，用統計的術語來解釋，這種現象就是通稱的一般化附加模型*(generalized additive model)*。

Freund和Schapire(1996，1997)提出的Adaboost演算法，可以說是建造附加模型極有效的方法。透過學習演算法，極盡可能地將分類錯誤減少到最小的方式去產生一個假說，每次增加一個假說到整體學習之中，分類錯誤就相對的降低。一直重複這個步驟並逐次累積假說，最後建造一個經由加權總數所得到的整體假說 $H(\mathrm{X}_i)=\sum_k w_k h_k(\mathrm{X}_i)$（這函式計算結果的正負號即是標籤 y_i ），理論上該整體假說可以使得分類錯誤減少到最小。

Adaboost演算法運作的過程敘述如下，圖(3)：假設在第 k 次迭代*(iteration)*時，$d_k(\mathrm{X}_k)$ 是資料點 X_k 的權重。一開始，全部訓練的資料點 X_i 都給予 $d_1(\mathrm{X}_k)=1/m$ 權重，其中 m 是資料點的數量。在迭代 k 的過程中，基礎學習演算法根據訓練錯誤加權總和減少到最小的目標建造假說，其中加權錯誤的計算公式爲 $r=\sum_i d_i(\mathrm{X}_i)y_i h_k(\mathrm{X}_i)$，公式中的 $h_k(\mathrm{X}_i)$ 是指假說 h_k 預測的標籤。假說所指派的權重爲 $w_k=\frac{1}{2}\ln\frac{1+r}{1-r}$，爲了計算下一次迭代的權重，訓練資料點 X_i 的權重則必須調整成 $d_{k+1}=d_k(\mathrm{X}_i)\frac{\exp(-w_k y_i h_k(\mathrm{X}_i))}{z_k}$。



圖(3) Adaboost演算法流程示意圖

Breiman(1997)證明了這演算法是函數空間中的一種坡度最佳化*(gradient optimization)*形式，其所採用以最佳化的目標函數*(objective function)*可以寫成 $J(H)=\sum_i \exp(-y_i H(\mathrm{X}_i))$。其中 $y_i H(\mathrm{X}_i)$ 稱爲邊緣*(margin)*，實際代表的意義是指 X_i 正確被分類的數量。如果邊緣的值爲正，則 $H(\mathrm{X}_i)$ 的正負號和 y_i 的正負號將會一致。另外，如果將目標函數 J 的結果可以減少到最小，此時將會產生最大的邊緣的值。

在多數的研究實驗中（弗羅因德& Schapire，1996； 鮑爾& Kohavi，1999；Dietterich，2000）都說明了Adaboost確實可以提供大部分數據資料最好的表現結果。但若針對包含較多貼錯標籤*(mislabeled)*的訓練資料來說，最後的表現結果就不是那麼順利了。在這種情形下，Adaboost把非常高的權重放在雜訊的資料點上，然後生成一個非常差的整體分類器。目前確實有許多的研究工作，著重在如何延伸Adaboost的功能，使之能夠在處理較高雜訊的訓練資料。

4. 討論

多數在整體方法的研究普遍集中在如何建造一組適當的整體決策樹。眾所周知，決策樹學習演算法因爲使用了搖擺式*(cascade)*的選擇方法，因此常有高的差異之困擾。也就是說，一旦當決策樹在內部選擇了錯誤的試驗節點後，接下來有非常高的可能足以影響未來全盤的決定。另外，加上決策樹內部的試驗節點只測試單一個的特徵，因此也可能在矩形的決定地區範圍造成極高偏差。總之，整體性的決策樹構成的分類器可以說是比個別的決策樹運作的效果更好。

如果在基礎學習演算法所產生的假說，不比決策樹有表現力的情況下，Adaboost的方法是非常值得推薦與使用的。很多實驗已經引進所謂的"decision stumps"，也就是只有一個內部節點的決策樹。爲了使用”decision stumps”進而學習較爲複雜的函數，如何適當地利用Adaboost強大的能力直接建造一個附加的模型是很重要的，這通常也會得到比Bagging或其他方法更爲準確與多樣化的結果。

對多重標籤*(multiclass)*的問題來說，改正錯誤的生產編碼演算法是能夠建造一個不錯的整體分類器。不過，因爲輸出編碼可能會造成另一個難度高的雙重類別學習問題*(two-class learning problem)*，所以基礎學習者*(base learner)*的表現力就顯得格外重要。

5. 參考資料

(1) Thomas G. Dietterich. ”Ensemble learning”, In The Handbook of Brain Theory and Neural Networks, Second Edition, 2002.

(2) S. Russell & P. Norvig, “Learning From Observations”, in The Textbook of Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall, 2003

(3) Eric Bauer, Ron Kohavi, “An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”, Machine Learning, 1999.

(4) Robert E. Schapire. “The boosting approach to machine learning: An overview”, In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, editors, Nonlinear Estimation and Classification. Springer, 2003.

(5) Robert E. Schapire. “A brief introduction to boosting”, In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999.
(6) Yoav Freund, Robert Schapire, “A Decision-Theoretic eneralization of On-Line Learning and an Application to Boosting”, European Conference on Computational Learning Theory 1995.
(7) Paul Viola, Michael Jones, Robust Real-Time Face