Artificial Intelligence Term Project2 Report

Wei Li 李韡

May 31, 2017

Contents

1 Overview

在提交的檔案中，資料夾code中包括cv_tools.cpp 與 naive_bayes.cpp兩個程式碼，以及4個shell檔案，分別用於：compile程式碼，對data set進行k-fold切割，執行 NB classifier以及執行 C4.5 演算法。

詳細的說明見code資料夾下的readme.txt檔案。

請務必在Windows系統下使用std=c++11 -O3參數進行compile

1.1 Architecture of cv_tools.cpp Program

我們的任務是實現一個a k-fold stratified cross validation tool, 其中k可以自用指定。

首先，我們需要讀取xx.names這個檔案，通過parser，得到class的總數量。

然後，我們再使用若干個vector，將每一種class的資料存入相對應的vector中。

再來，我們對每一個vector，進行shuffle操作，保證數據的亂序，這很重要，也會對最後validate的結果產生很大影響。這樣做的目的，是為了保證stratified。

最後，我們我們在每一個vector中選取一份，作為test data，剩下的k-1份，作為train data。 在這裡，我的k設定是10。同時，我們輸出編號次序連續的10份names檔案，以方便C4.5 演算法的執行。

1.2 Architecture of naive_bayes.cpp Program

同樣，在執行演算法之前，我們需要對data進行適當的處理。

具體包括，讀取names檔案，確定class的數量和種類，確定feature的數量以及種類，特別需要注意的是對continuous數據的處理，這在後面我會再次提到。

讀取完names檔案後，再讀取data檔案，做相應的處理，存儲必要的資訊。

最後，我們便可以開始執行我們的naive bayes演算法，對test檔案的data進行預測。

這裡，强調一下我對資料的處理：

每次，我只讀入一行，讀入後，刪除該行所有的空格，若行尾還有"."存在，也一併刪除。 對於xx.names檔案下features的處理，先用分號進行分割，分開後再用逗號進一步分開。 對於xx.data檔案下features的處理，過濾掉空格後，直接用逗號進行分割。

2 Detailed Description

2.1 Deal with numeric attributes

解決連續性數據的方法可以分為兩種，其中一種是把每一個連續型的數據離散化，然後用相應的離散區間替換連續數值。但這種方法不好控制離散區間劃分的粒度，如果粒度太細，就會因為每一個區間的訓練量太少而不能對機率做出可靠的估計，如果粒度太粗，那麼有些區間就會含有來自不同類的記錄，因此失去了正確的決策邊界。

介於五種data set中粒度難以確定，比如年齡，大致的範圍是在0-100之間，而温度，很顯然則是在0-40度之間，這是的我們難以確定一個合適的分割粒度，所以我沒有使用離散化這種方法，而是使用了第二種方法，假設連續的變量服從某種分佈，然後使用訓練數據估計分佈的參數，這裡我使用的高斯分佈(Gaussian distribution)，反觀年齡，温度，確實也是遵循高斯分佈或者説近似能夠滿足高斯分佈，這使得方法二來得更加準確。

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

使用高斯分佈，我們需要計算mean 和 vairance，所以，在前面處理xx.data檔案的同時，我們一併計算每一個feature所對應的mean和vairance，這樣，就可以方便後面naive bayes計算機率了，詳細的過程，可以見code的load_data()這一函數。

另外，使用高斯分佈在程式的耗時上，也優與使用離散化的方法。特別是如果我們的分割區間很小，則計算的次數會明顯增加，記憶體的消耗也會增加。

2.2 Test Result

下圖是在對數據進行亂序分割，並使用高斯分佈處理連續型數據，拉普拉斯修正的情況下得到的結果：

dataset		cv1	cv2	cv3	cv4	cv5	cv6	cv7	cv8	cv9	cv10	avg	p-value
adult	NB_Acc	0.82	0.82	0.83	0.84	0.83	0.83	0.83	0.82	0.83	0.83	0.83	2.7E-07
	c4.5_Acc	0.87	0.85	0.87	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	
car	NB_Acc	0.90	0.87	0.89	0.88	0.87	0.89	0.85	0.85	0.88	0.87	0.87	3.1E-05
	c4.5_Acc	0.90	0.94	0.94	0.95	0.90	0.94	0.91	0.91	0.95	0.92	0.93	
isolet	NB_Acc	0.18	0.17	0.21	0.22	0.19	0.21	0.17	0.17	0.21	0.21	0.19	3.94E-13
	c4.5_Acc	0.75	0.78	0.79	0.80	0.83	0.77	0.81	0.79	0.75	0.80	0.79	
page-blocks	NB_Acc	0.92	0.88	0.92	0.93	0.90	0.92	0.91	0.91	0.90	0.92	0.91	5.88E-07
	c4.5_Acc	0.96	0.97	0.98	0.97	0.96	0.96	0.97	0.97	0.97	0.99	0.97	
winequality	NB_Acc	0.45	0.41	0.44	0.45	0.47	0.46	0.44	0.44	0.43	0.44	0.44	4.68E-08
	c4.5_Acc	0.61	0.62	0.58	0.59	0.63	0.56	0.56	0.60	0.63	0.58	0.59	

Figure 1: result

2.2.1 Shuffle vs. No Shuffle

從這裡我們能夠很明顯地看出，如果在做k-fold CV的時候，沒有對data 進行shuffle，結果會差甚遠。

dataset		cv1	cv2	cv3	cv4	cv5	cv6	cv7	cv8	cv9	cv10	avg	p-value
adult	NB_Acc	0.82	0.83	0.82	0.82	0.83	0.83	0.82	0.83	0.83	0.83	0.83	2.05E-07
	c4.5_Acc	0.85	0.86	0.86	0.85	0.86	0.86	0.86	0.86	0.87	0.85	0.86	
car	NB_Acc	0.71	0.64	0.64	0.72	0.76	0.68	0.86	0.87	0.84	0.82	0.75	0.004303
	c4.5_Acc	0.77	0.77	0.73	0.76	0.86	0.85	0.87	0.88	0.84	0.84	0.82	
isolet	NB_Acc	0.22	0.15	0.19	0.18	0.18	0.21	0.14	0.22	0.20	0.16	0.18	1.51E-09
	c4.5_Acc	0.74	0.79	0.77	0.83	0.80	0.73	0.55	0.78	0.65	0.75	0.74	
page-blocks	NB_Acc	0.88	0.92	0.92	0.81	0.91	0.91	0.89	0.87	0.85	0.91	0.89	9.72E-05
	c4.5_Acc	0.95	0.98	0.96	0.97	0.97	0.97	0.97	0.98	0.95	0.94	0.96	
winequality	NB_Acc	0.48	0.43	0.44	0.42	0.41	0.49	0.43	0.40	0.41	0.45	0.44	0.382873
	c4.5_Acc	0.44	0.42	0.35	0.42	0.42	0.48	0.45	0.44	0.48	0.44	0.43	

Figure 2: result without CV data shuffle

2.2.2 About Data Set

從 figure 1 我們可以看到，除了isolet外，其他data set所預測的結果，NB ACC和C4.5 ACC還算接近，但都是c4.5更好一些，這可能和我的NB實現有一定

的關係，當然，更多的原因應該是傳統的Naive Bayes將各個feature視為獨立不相干的，但在實際生活中，各個feature往往是存在聯繫的，有些甚至存在緊密的連續，這樣計算出的概率也許會有不准確的時候。

而我們來看看 isolet 這個 data set，總共有600多個feature，全部為連續型，這相對與NB來說是比較不利的，feature間是否存在聯繫，feature是否是滿足高斯分佈的，有是否獨立，在如此高緯度的features下，均難以保證，而decision tree則不會受到很大的影響。

這導致的結果就是NB 和 C45兩種演算法結果差異巨大。

2.2.3 About Continuous and Discretization

從結果中還可以發現，離散型feature居多的data set，如 car 這個 data set，兩種算法的表現均比較理想，而在連續型feature居多的data set，如isolet 和 winequality， 兩種算法的表現就稍微差一些。

原因仍然在於，想處理好連續型的數據，並非是一件很容易的事，需要經過一些轉化，從而喪失掉一些精準度。

3 Broblem Encountered

3.1 Save Running Time

Naive Bayes計算機率的部分，是程式最耗時的地方，如何節省時間，這是一個問題？

為了避免每一次計算機率都要重新計算個數，我使用了map將已經計算過的記錄存起來，當下次訪問的時候，直接拿出來用，這樣可以加速程序的運行。

但是仍然存在一個小問題，在處理離散型的feature時，就無法這樣做，這也是整個程式最耗時的地方，因為每一次均要重新計算，不過也還算可以接受。

3.2 Using scripts

手動地進行測試是一件很繁瑣的事，我使用了4個shell腳本來簡化操作。另外，C4.5的原程式輸出太多無用的資訊，我均將他們刪去，因為我只需要得到test acc即可。