Logistic Regression

Code

```
6   sigmoid = lambda z: 1/(1+np.exp(-z))

104 def LogReg(D, GDpara):
105     print 'Training...'
106     x, y = D['x'], D['y']
107     nD, dimx = x.shape
108     print '\tTraining Data #', nD, ', Dim x:', dimx
109     w = np.zeros(dimx, dtype='float')
110     itr = GDpara['itr']
111     eta = GDpara['eta']
112     reg = GDpara['reg']
113     G = 0
114     if itr>0:
115         itr = int(itr)
116         for i in range(itr):
117             f = sigmoid( np.dot(x,w) )
118             g = np.dot(y-f,x) + 2*reg*w
119             G += g**2
120             w = w+eta*g/(G**0.5)
121             diff = y - np.rint(f)
122             acc = len( diff[diff==0] )/ float(len(diff)) * 100
123             if i%100 == 0:
124                 print '\t', i, ', acc:', '%.2f' % acc

143     print 'Done.\n'
144     return w
```

itr: iteration
eta: learning rate
reg: regularizaion

f: 由 model 計算的 y
y: 實際的 class (ŷ)
g: gradient
G: AdaGrad
Acc: accuracy (%)

Description

- 將 Capital 相關的屬性取 log、其他屬性開根號作為 feature，總共有 58 維。
- 增加一個維度都是 1，以同時計算 w 與 b。
- 使用 Adagrad 做 training，iteration 取 3000-20000 之間，有做 regularization。
- 使用 4-fold validation 來評價 model 的好壞。
- 因為 f 在 0-1 之間，所以直接使用 rint 選擇最後的預測。

Another Method: Decision tree

Code

```
143  def trainTree(tree, idx, D, attrs, stopUnity=0.9):
144      unity1 = len ( D[ D['y']==1 ] ) / float( len(D) )
145      print '[Y/N] Ratio %.2f : %.2f' % (unity1*100, (1-unity1)*100 )
146      if unity1 > stopUnity:
147          tree[idx] = 1
148          print 'Reach leaf node 1 with %d data' % len(D)
149      elif ( 1-unity1 ) > stopUnity:
150          tree[idx] = 0
151          print 'Reach leaf node 0 with %d data' % len(D)
152
153      else:
154          attr, c, splitD = findAttr(D, attrs)
155
156          lidx, ridx = len(tree), len(tree)+1
157          tree.append({}), tree.append({})
158
159          node = { 'attr': attr,
160                   'c': c,
161                   'son': [lidx, ridx], }
162          tree[idx] = node
163
164          trainTree(tree, lidx, splitD['left'], attrs, stopUnity)
165          trainTree(tree, ridx, splitD['right'], attrs, stopUnity)
```

Description

Decision tree 是一種 Greedy 的演算法，使用資料中不同的 attribute 作為分類依據。演算法每一輪會選擇可以將目前資料分類得最好的 attribute, value pair 當作新的 node 加到 Decision tree，再遞迴尋找分類後的新 node，直到只剩下單一類別的資料時，就以這個類別當成 leaf node 並不再遞迴。

Verification

在這次作業中，我以 max information gain 來選擇分類的 attribute, value pair，information gain 的定義如下：

$$\mathrm{Gain}(S, A) = H(S) - \Sigma_{j=1}^{v} \frac{|S_j|}{|S|} H(S_j)$$

S: 分割前的所有資料

S_j: 分割後第 j 塊資料

每次分割前先將資料依各個 attribute 排序好，在每個 attribute 中以 Dynamic Programming 選出 max information gain 所在的 value。

Pruning

根據 Decision tree 的結束條件，可知這個方法很容易 overfitting，所以必須做 Pruning。一般來說，Decision tree 的 pruning 分為兩種，一種是在長樹的時候就修剪，另一種是長完之後再做修剪，在這次作業中我只有做了第一種，也用了最簡單的方法：當某個 class 在分割後的 data set 中超過了特定的濃度就視為 leaf node，不再生長。

Performance

Submission	Public	Private
LR with Regularization	0.93667	0.93667
LR without Regularization	0.94	0.9333
Decision Tree	0.86000	0.82333

Discussions

Logistic Regression

- 若用 std 和 mean 做 scaling、其他參數相同，training 的結果反而不會比較好。可能是因為原始數據大多就是介於 0-1 之間，再用這個方法做 scaling 反而讓彼此之間的差異更大。
- 將 Capital 相關的屬性取 log、其他屬性開根號作為 feature，可以得到較好的結果。可能是因為 capital 的原始數值和其他類型差異太大，在取 log 之後會比較相近，而其他數值開根號也可以增大。

Decision Tree

- 調整停止生長的濃度對結果產生的影響很大，若太大會有 overfitting 的問題，但是太小又沒辦法正確的做出分類（假設調整成超過 0.9 就停止，那結果就很難比 0.9 更好）。
- 可能是因為使用 Greedy 演算法的關係，沒辦法得到太好的結果（無法在每次選擇當中看到最後哪一種分割會最好）。相較而言，Logistic Regression 為 end-to-end approach，所以能得到較好的結果。