

act.lua

Page 1/3

```

1 #!/usr/bin/env lua
2 local help = {}
3 act.lua : stochastic incremental XAI
4 (c) 2025, Tim Menzies, timm@ieee.org, mit-license.org
5
6 Options:
7   -h          Show help.
8   -b bins=7   Number of bins for discretization.
9   -s str=30   Update model every 'str' number of rows.
10  -r ruleMax=3 Max conditions in a rule.
11  -s seed=42   Random number seed.
12  -f file=../lua6/aut93.csv ]]
13
14 -- coerce(s) --> v ;; Return int or float or bool or string from 's'.
15 local function coerce(s)
16   if s then return tonumber(s) or s:match("'%s'*(-)%s*$") end end
17
18 local the={} ; for k,v in help:match("(%%$+)=%%$+)" do the[k] = coerce(v) end
19 math.randomseed(the.seed)
20
21 local DATA, NUM, SYM, COLS, clone, adds
22
23 --## Lib
24
25 local abs,exp,sqrt,log = math.abs, math.exp, math.sqrt, math.log
26 local max,rand,cos = math.max, math.random, math.cos
27
28 local say=io.write
29 local fmt = string.format
30
31 -- sort(t,f) --> t ;; Sort 't' using function 'f'.
32 local sort = function(t,f) table.sort(t,f); return t end
33 -- it(f) --> f ;; Return a function that sorts 'a' and 'b' on 'f'.
34 local it = function(f) return function(a,b) return f(a) < f(b) end end
35 -- cat(a) --> s ;; Return a string representation of array 'a'.
36 local cat = function(a) return "[" .. table.concat(a, ",") .. "]" end
37
38 -- o(v) --> s ;; Return a string representation of 'v'.
39 local function o(v, list,dict)
40   list = function(s, u)
41     for k,v in ipairs(s) do u[l+#u] = o(v) end; return cat(u) end
42   dict = function(d, u)
43     for k,v in pairs(d) do u[l+#u] = fmt("%s%s", k, o(v)) end
44     return cat(sort(u)) end
45   type(v) == "number" and fmt(v%1==0 and "%00" or "%3", v) or
46   type(v) == "table" and tostring(v) or (#v>0 and list or dict)(v, {}) end
47
48 -- s2a(s) --> a ;; Return array of words from string 's', split on " ".
49 local function s2a(s, a)
50   a={} ; for s1 in s:match("(A)+") do a[l+#a] = coerce(s1) end; return a end
51
52 -- csv(file) --> f ;; Iterator that returns rows from 'file'.
53 local function csv(file, src)
54   src = assert(io.open(file))
55   return function()
56     s = src:read(); if s then return s2a(s) else src:close() end end end
57
58 -- shuffle(t) --> t ;; Randomly shuffle the order of elements in 't'.
59 local shuffle = function(t, n)
60   for m=#t,2,-1 do n=math.random(m); t[m],t[n]=t[n],t[m] end; return t end
61
62 -- cut(a0,n,data) --> t,t ;; Split 'a0' at 'n' (if 'data' exists, split that too).
63 local function cut(a0,n, data)
64   local a1,a2 = {},{}
65   for j,v in ipairs(a0) do if j <= n then a1[l+#a1]=v else a2[l+#a2]=v end end
66   return data and clone(data,a1),clone(data,a2) or a1,a2 end
67
68 -- mode(d) --> v ;; Return the most frequent key in 'd'.
69 local function mode(d, v,n)
70   v,n = nil,0
71   for v1,n1 in pairs(d) do if n1>n then v,n=v1,n1 end end
72   return v end
73
74 -- box_muller(mu,sd) --> n ;; Return a random number from a Gaussian 'mu','sd'.
75 function box_muller(mu,sd)
76   return mu + sd * sqrt(-2 * log(rand())) * cos(6.28 * rand()) end
77
78 --## Classes
79
80 -- DATA(src) --> DATA ;; Create a new DATA, populated with 'src'.
81 function DATA( src) return adds(src, {n=0,rows={},cols=nil}) end
82
83 -- clone(i,src) --> DATA ;; Return a new DATA with same structure as 'i'.
84 function clone(i, src) return adds(src, DATA(i.cols.names)) end
85
86 -- NUM(at,s) --> NUM ;; Create a NUM object to summarize numbers.
87 function NUM(at,s)
88   return {at=at or 0, of=s, n=0, mu=0, m2=0, sd=0,
89     best=(tostring(s) or ""):find"%S" and 1 or 0} end
90
91 -- SYM(at,s) --> SYM ;; Create a SYM object to summarize symbols.
92 function SYM(at,s) return {at=at, of=s, n=0, has={}} end
93
94 -- COLS(row) --> COLS ;; Create a COLS object from a list of column names.
95 function COLS(row, t,x,y,all,col)
96   x,y,all = {}, {}, {}
97   for n,s in ipairs(row) do
98     col = (s:match("[A-Z]" and NUM or SYM)(n,s)
99     all[n] = col
100     if not s:match"%X%" then
101       t = s:find"%[+-%]" and y or x
102       t[l+#t] = col end end
103   return {all=all, x=x, y=y, names=row} end

```

act.lua

Page 2/3

```

104 --## Methods
105
106 -- add(i,v,inc) --> v ;; Update 'i' with 'v' (incrementing by 'inc').
107 local function add(i,v, inc)
108   if v == "" then return v end
109   inc = inc or 1
110   i.n = i.n + inc
111   if i.has then i.has[v] = inc + (i.has[v] or 0)
112   elseif i.mu then
113     if inc < 0 and i.n < 2 then i.sd, i.m2, i.mu, i.n = 0,0,0 else
114       local d = v - i.mu
115       i.mu = i.mu + inc * d / i.n
116       i.m2 = i.m2 + inc * d * (v - i.mu)
117       i.sd = i.n<2 and 0 or sqrt((max(0,i.m2)/(i.n - 1))) end
118   elseif i.rows then
119     if not i.cols then i.cols = COLS(v) else
120       i.mid = nil
121       for _,col in pairs(i.cols.all) do add(col, v[col.at], inc) end
122       if inc > 0 then i.rows[l+#i.rows] = v end end end
123   return v end
124
125 -- sub(i,v) --> v ;; Decrement 'v' from 'i'.
126 local function sub(i,v) return add(i,v,-1) end
127
128 -- adds(src,it) --> it ;; Update 'it' with all items from 'src'.
129 function adds(src, it)
130   it = it or NUM()
131   if type(src) == "string"
132     then for row in csv(src) do add(it,row) end
133     else for _,row in pairs(src or {}) do add(it,row) end end
134   return it end
135
136 -- mid(i) --> v,row ;; Return central tendency of 'i'.
137 local function mid(i) --> a | v ;; Expected value for 'i'.
138   if i.mu then return i.mu
139   elseif i.has then return mode(i.has)
140   elseif i.rows then
141     if not i.mid then
142       local t={} ; for _,col in pairs(i.cols.all) do t[l+#t] = mid(col) end
143       i.mid = t end
144     return i.mid end end
145
146 -- norm(i,v) --> n ;; Normalize 'v' 0..1 using 'i'.
147 local function norm(i,v)
148   return (i.has or v=="") and v
149   or 1/(1 + math.exp(-1.7 * (v - i.mu)/(i.sd + 1e-32))) end
150
151 -- aha(col,v1,v2) --> n ;; Return distance between 'v1' and 'v2'.
152 local function aha(col,v1,v2)
153   if v1=="*" and v2=="*" then return 1 end
154   if col.has then return v1==v2 and 0 or 1 end
155   v1,v2 = norm(col,v1), norm(col,v2)
156   v1 = v1 == "*" and v1 or (v2 > 0.5 and 0 or 1)
157   v2 = v2 == "*" and v2 or (v1 > 0.5 and 0 or 1)
158   return abs(v1 - v2) end
159
160 -- distx(i,row1,row2) --> n ;; Return distance 'row1' to 'row2' (using X cols).
161 local function distx(i,row1,row2, d)
162   d=0; for _,x in pairs(i.cols.x) do d = d + aha(x, row1[x.at],row2[x.at])^2 end
163   return sqrt(d/#i.cols.x) end
164
165 -- disty(i,row) --> n ;; Return distance of 'row' to best goal (using Y cols).
166 local function disty(i,row, d)
167   d=0; for _,y in pairs(i.cols.y) do d = d + (norm(y, row[y.at]) - y.best)^2 end
168   return sqrt(d/#i.cols.y) end
169
170 -- distys(i,rows) --> rows ;; Sort 'rows' by their distance to heaven.
171 local function distys(i, rows, y)
172   y = function(row) return disty(i, row) end
173   return sort(rows or i.rows, function(r1,r2) return y(r1) < y(r2) end) end
174
175 --## Think
176
177 -- two(data) --> t ;; Incrementally cluster 'data' into 'best' and 'rest'.
178 local function two(data)
179   local train,test,start,todo,seen,best,rest,d
180   shuffle(data.rows)
181   train,test = cut(data.rows, data.n//2)
182   start,todo = cut(train, 4)
183   seen = clone(data, start)
184   best,rest = cut(distys(seen),2,data)
185   d = function(row,what) return distx(seen, row, mid(what)) end
186   for n,row in pairs(todo) do
187     if n>256 then break end; --say("n")
188     if d(row,best) < d(row,rest) then
189       add(seen, add(best, row)) ; --say(best.n)
190       if best.n > sqrt(seen.n) then -- print("n")
191         add(rest, sub(best, table.remove(distys(best)))) end end end
192     distys(best)
193   return {best=best, rest=rest, seen=seen, test=test,
194     model = it(function(row,v) return d(row,best) - d(row,rest) end) end} end

```

act.lua

Page 3/3

```

196 --## Demos
197
198 local eggs={}
199
200 eggs["-h"] = function(_) print("\n".help.."") end
201 eggs["-s"] = function(n) math.randomseed(n); the.seed =n end
202 eggs["-l"] = function(_) print(o(the)) end
203 eggs["-csv"] = function(_) for row in csv(the.file) do print(o(row)) end end
204 eggs["-shuffle"] = function(_) print(o(shuffle(10,20,30,40,50))) end
205 eggs["-mode"] = function(_) print(mode(d=2,f=10,g=1)) end
206
207 eggs["-cut"] = function(_, b,c)
208   b,c=cut((10,20,30,40,50),2); print(o(b),o(c))
209   for _,l,100 do b,c=cut((10,20,30,40,50),2) end end
210
211 eggs["-num"] = function(_, num)
212   num=NUM()
213   for _,l,1000 do add(num, box_muller(10,5))
214     print(fmt("%.3f%.3f", num.mu, num.sd)) end
215
216 eggs["-data"] = function(_)
217   for n,col in pairs(DATA(the.file).cols.x) do
218     print(n,o(col)) end end
219
220 eggs["-distx"] = function(_, data,t,u)
221   data = DATA(the.file)
222   t,rows = {}, shuffle(data.rows)
223   for n = 2,#rows do t[l+#t] = distx(data,rows[n-1],rows[n]) end
224   print(o(sort(t))) end
225
226 eggs["-disty"] = function(_, data,num)
227   data,t = DATA(the.file), {}
228   distys(data)
229   for n,row in pairs(data.rows) do t[n]=disty(data,row) end
230   print(o(t)) end
231
232 eggs["-inc"] = function(_, data1,data2)
233   data1 = DATA(the.file)
234   print(o(mid(data1)))
235   data2 = clone(data1)
236   for _,row in pairs(data1.rows) do
237     add(data2,row)
238     if data2.n==50 then print(o(mid(data2))) end end
239   while data2.rows do
240     sub(data2, table.remove(data2.rows))
241     if data2.n==50 then print(o(mid(data2)));break end end end
242
243 eggs["-two"] = function(_, data,out,t)
244   t,data = {}, DATA(the.file)
245   for _,l,20 do
246     out = two(data)
247     t[l+#t] = (100*disty(out.seen, sort(out.test, out.model)[1]))//1 end
248   print(o(sort(t))) end
249
250 -- cli(d,funs) --> nil ;; Update 'd' with flags from command-line; run 'funs'.
251 local function cli(d,funs)
252   for i,s in pairs(arg) do
253     if funs[s]
254       then funs[s](coerce(arg[i+1]))
255       else for k_,_ in pairs(d) do
256         if k:sub(1,1)==s:sub(2) then d[k]=coerce(arg[i+1]) end end end end
257   if arg[0]:find"two.lua" then cli(the,egs) end

```