

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

```

## act.lua

Page 2/3

```

101 --## Methods
102
103 -- add(i,v,inc) --> v ;; Update 'i' with 'v' (incrementing by 'inc').
104 local function add(i,v, inc)
105   if v == "" then return v end
106   inc = inc or 1
107   i.n = i.n + inc
108   if i.has then i.has[v] = inc + (i.has[v] or 0)
109   elseif i.mu then
110     if inc < 0 and i.n < 2 then i.sd, i.m2, i.mu, i.n = 0,0,0 else
111       local d = v - i.mu
112       i.mu = i.mu + inc * d / i.n
113       i.m2 = i.m2 + inc * d * (v - i.mu)
114       i.sd = i.n<2 and 0 or sqrt((max(0,i.m2)/(i.n - 1))) end
115   elseif i.rows then
116     if not i.cols then i.cols = COLS(v) else
117       i.mid = nil
118       for _,col in pairs(i.cols.all) do add(col, v[col.at], inc) end
119       if inc > 0 then i.rows[l+#i.rows] = v end end end
120   return v end
121
122 -- sub(i,v) --> v ;; Decrement 'v' from 'i'.
123 local function sub(i,v) return add(i,v,-1) end
124
125 -- adds(src,it) --> it ;; Update 'it' with all items from 'src'.
126 function adds(src, it)
127   it = it or NUM()
128   if type(src) == "string"
129     then for row in csv(src) do add(it,row) end
130     else for _,row in pairs(src or {}) do add(it,row) end end
131   return it end
132
133 -- mid(i) --> v,row ;; Return central tendency of 'i'.
134 local function mid(i) --> a | v ;; Expected value for 'i'.
135   if i.mu then return i.mu
136   elseif i.has then return mode(i.has)
137   elseif i.rows then
138     if not i.mid then
139       local t={} ; for _,col in pairs(i.cols.all) do t[l+#t] = mid(col) end
140       i.mid = t end
141   return i.mid end end
142
143 -- norm(i,v) --> n ;; Normalize 'v' 0..1 using 'i'.
144 local function norm(i,v)
145   return (i.has or v=="") and v
146   or 1/(1 + math.exp(-1.7 * (v - i.mu)/(i.sd + 1e-32))) end
147
148 -- aha(col,v1,v2) --> n ;; Return distance between 'v1' and 'v2'.
149 local function aha(col,v1,v2)
150   if v1=="*" and v2=="*" then return 1 end
151   if col.has then return v1==v2 and 0 or 1 end
152   v1,v2 = norm(col,v1), norm(col,v2)
153   v1 = v1 == "*" and v1 or (v2 > 0.5 and 0 or 1)
154   v2 = v2 == "*" and v2 or (v1 > 0.5 and 0 or 1)
155   return abs(v1 - v2) end
156
157 -- distx(i,row1,row2) --> n ;; Return distance 'row1' to 'row2' (using X cols).
158 local function distx(i,row1,row2, d)
159   d=0; for _,x in pairs(i.cols.x) do d = d + aha(x, row1[x.at],row2[x.at])^2 end
160   return sqrt(d/#i.cols.x) end
161
162 -- disty(i,row) --> n ;; Return distance of 'row' to best goal (using Y cols).
163 local function disty(i,row, d)
164   d=0; for _,y in pairs(i.cols.y) do d = d + (norm(y, row[y.at]) - y.best)^2 end
165   return sqrt(d/#i.cols.y) end
166
167 -- distys(i,rows) --> rows ;; Sort 'rows' by their distance to heaven.
168 local function distys(i, rows, y)
169   y = function(row) return disty(i, row) end
170   return sort(rows or i.rows, function(r1,r2) return y(r1) < y(r2) end) end
171
172 --## Think
173
174 -- two(data) --> t ;; Incrementally cluster 'data' into 'best' and 'rest'.
175 local function two(data)
176   local train,test,start,todo,seen,best,rest,d
177   shuffle(data.rows)
178   train,test = cut(data.rows, data.n//2)
179   start,todo = cut(train, 4)
180   seen = clone(data, start)
181   best,rest = cut(distys(seen),2,data)
182   d = function(row,what) return distx(seen, row, mid(what)) end
183   for n,row in pairs(todo) do
184     if n>256 then break end; --say("n")
185     if d(row,best) < d(row,rest) then
186       add(seen, add(best, row)) ; --say(best.n)
187       if best.n > sqrt(seen.n) then -- print("n")
188         add(rest, sub(best, table.remove(distys(best)))) end end end
189     distys(best)
190   return {best=best, rest=rest, seen=seen, test=test,
191           model = it(function(row,v) return d(row,best) - d(row,rest) end) end} end

```

## act.lua

Page 3/3

```

192 --## Demos
193 local eggs={}
194
195 eggs["-h"] = function(_) print("\n".help.."") end
196 eggs["-s"] = function(n) math.randomseed(n); the.seed =n end
197 eggs["-l"] = function(_) print(o(the)) end
198 eggs["-csv"] = function(_) for row in csv(the.file) do print(o(row)) end end
199 eggs["--shuffle"] = function(_) print(o(shuffle(10,20,30,40,50))) end
200 eggs["--mode"] = function(_) print(mode(d=2,f=10,g=1)) end
201
202 eggs["--cut"] = function(_, b,c)
203   b,c=cut((10,20,30,40,50),2); print(o(b),o(c))
204   for _,l,100 do b,c=cut((10,20,30,40,50),2) end end
205
206 eggs["--num"] = function(_, num)
207   num=NUM()
208   for _,l,1000 do add(num, box_muller(10,5))
209   print(fmt("%.3f%.3f", num.mu, num.sd)) end end
210
211 eggs["--data"] = function(_)
212   for n,col in pairs(DATA(the.file).cols.x) do
213     print(n,o(col)) end end
214
215 eggs["--distb"] = function(_, data,t,u)
216   data = DATA(the.file)
217   t,rows = {}, shuffle(data.rows)
218   for n = 2,#rows do t[l+#t] = distx(data,rows[n-1],rows[n]) end
219   print(o(sort(t))) end
220
221 eggs["--disty"] = function(_, data,num)
222   data,t = DATA(the.file), {}
223   distys(data)
224   for n,row in pairs(data.rows) do t[n]=disty(data,row) end
225   print(o(t)) end
226
227 eggs["--inc"] = function(_, data,data2)
228   data1 = DATA(the.file)
229   print(o(mid(data1)))
230   data2 = clone(data1)
231   for _,row in pairs(data1.rows) do
232     add(data2,row)
233     if data2.n==50 then print(o(mid(data2))) end end
234   while data2.rows do
235     sub(data2, table.remove(data2.rows))
236     if data2.n==50 then print(o(mid(data2)));break end end end
237
238 eggs["--two"] = function(_, data,out,t)
239   t,data = {}, DATA(the.file)
240   for _,l,20 do
241     out = two(data)
242     t[l+#t] = (100*disty(out.seen, sort(out.test, out.model)[1]))//1 end
243   print(o(sort(t))) end
244
245 -- cli(d,funs) --> nil ;; Update 'd' with flags from command-line; run 'funs'.
246 local function cli(d,funs)
247   for i,s in pairs(arg) do
248     if funs[s]
249       then funs[s](coerce(arg[i+1]))
250       else for k_,_ in pairs(d) do
251         if k:sub(1,1)==s:sub(2) then d[k]=coerce(arg[i+1]) end end end end
252   if arg[0]:find"two.lua" then cli(the,egs) end

```