

act.lua

Page 1/3

act.lua

Page 2/3

act.lua

Page 3/3

```

1  #!/usr/bin/env lua
2  local help = [
3    act.lua : stochastic incremental XAI
4    (c) 2023, 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    -e era=30   Update model every 'era' number of rows.
10   -r ruleMax=3 Max conditions in a rule.
11   -s seed=42  Random number seed.
12   -f file.../luat/auto93.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 _M={}; for k,v in help:gmatch("(%)=(%)") 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=o.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,...,function(a,b) return f(a)< f(b) end end
33 local cat = function(a,...) if type(a) == "string" then return a else return table.concat(a, " ") .. end end
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(a,...,u) u=a end}
41   for _,v in ipairs(a) do u[1+#u]=o(v) end; return cat(u) end
42   dict = function(d,...,u)
43     for k,v in pairs(d) do u[1+#u]= fmt("%s %s", k, o(v)) end
44   end
45   return cat(sort(u))
46 end
47 return type(v) == "number" and fmt(v==l=0 and "%,.0f" or "%,.3f", v) or
48 type(v) == "table" and tostring(v) or (v==0 and list or dict)(v,...)
49
50 -- s2a(s) --> a ;; Return array of words from string 's', split on "\".
51 local function s2a(s,...,a)
52   a={}; for sl in s:match("([^\"]+)") do a[1+#a]= coerce(sl) end; return a end
53
54 -- csv(file) --> f ;; Iterator that returns rows from 'file'.
55 local function csvfile(...,src)
56   src = assert(io.open(file))
57   return function(...,s)
58     s = src:read(); if s then return s2a(s) else src:close() end end end
59
60 -- shuffle(t) --> t ;; Randomly shuffle the order of elements in 't'.
61 local shuffle = function(t,...,n)
62   for m=t,2,-1 do n=math.random(m); t[m],t[n]=t[n],t[m] end; return t end
63
64 -- cut(a0,n,data) --> t,t ;;Split 'a0' at 'n' (if 'data' exists,split that too).
65 local function cut(a0,...,n,...,data)
66   local al,a2={},{},()
67   for jv,y in ipairs(a0) do if j <= n then al[1+#al]=y else a2[1+#a2]=y end end
68   return data and clone(data,al),clone(data,a2) or al,a2 end
69
70 -- mode(d) --> v ;; Return the most frequent key in 'd'.
71 local function mode(d,...,v,n)
72   v,n = nil,...,nil
73   for yl,nl in pairs(d) do if nl>n then v,yl,nl end end
74   return v end
75
76 -- box_muller(mu,sd) --> n ;; Return a random number from a Gaussian 'mu','sd'.
77 function box_muller(mu,sd,...,n)
78   return mu + sd * sqrt(-2 * log(rand())) * cos(6.28 * rand()) end
79
80 --## Classes
81
82 -- DATA(src) --> DATA ;; Create a new DATA, populated with 'src'.
83 function DATA(...,src) return adds(src, {n=0,rows={},cols=nil}) end
84
85 -- clone(i,src) --> DATA ;; Return a new DATA with same structure as 'i'.
86 function clone(...,src) return adds(src, DATA(i.cols.names)) end
87
88 -- NUM(NUM,at,s) --> NUM ;; Create a NUM object to summarize numbers.
89 function NUM(...,at,s) return {at=at,of=s, n=0, mu=0, m2=0, sd=0,
90   best=(tostring(s) or ""):find"%$%" and 1 or 0} end
91
92 -- SYM(at,s) --> SYM ;; Create a SYM object to summarize symbols.
93 function SYM(...,at,s) return {at=at,of=s, n=0, has={} } end
94
95 -- COLS(row) --> COLS ;; Create a COLS object from a list of column names.
96 function COLS(...,row,...,t,x,y,all,col)
97   x,y,all = {},{},{},()
98   for n,s in ipairs(row) do
99     col = s:match('([A-Z]+)') and NUM or SYM(n,s)
100    col[n] = col
101    if not s:match'^X$%' then
102      t = s:find'^[!-]' and y or x
103      t[1+#t] = col end end
104   return {all=all, x=x, y=y, names=row} end

```

```

104
105  ---## Methods
106
107  -- add(i,v,inc) --> v ; Update 'i' with 'v' (incrementing by 'inc').
108  local function add(i,v, inc)
109    if v == "?" then return v end
110    inc = inc or 1
111    i.v = i.v + inc
112    if i.has then i.has[v] = inc + (i.has[v] or 0)
113  elseif i.mu then
114    if inc < 0 and i.n < 2 then i.sd, i.m2, i.mu, i.n = 0,0,0,0 else
115      local d = v - i.mu
116      i.mu = i.mu + inc * d / i.n
117      i.m2 = i.m2 + inc * d * (v - i.mu)
118      i.sd = i.sd*2 > 0 or sqrt((max(0,i.m2)/(i.n - 1))) end
119  elseif i.rows then
120    if not i.cols then i.cols = COLS(v) else
121      i._mid = nil
122      for _col in pairs(i.cols.all) do add(col, v[col..at], inc) end
123      if inc > 0 then i.rows[i + #i.rows] = v end end end
124  return v end
125
126  -- sub(i,v) --> v ; Decrement 'v' from 'i'.
127  local function sub(i,v) return add(i,-v,-1) end
128
129  -- adds(src,it) --> it ; Update 'it' with all items from 'src'.
130  function adds(src, it)
131    it = it or NUM()
132    if type(src) == "string"
133      then for row in csv(src) do add(it,row) end
134    else for _,row in pairs(src or {}) do add(it,row) end end
135    return it end
136
137  -- mid(i) --> v | row ; Return central tendency of 'i'.
138  local function mid(i) --> a | v;; Expected value for 'i'.
139    if i.mu then return i.mu
140    elseif i.mode then return mode(i.has)
141    elseif i.rows then
142      if not i._mid then
143        local t=(); for _col in pairs(i.cols.all) do t[1#+t] = mid(col) end
144        i._mid = t end
145      return i._mid end end
146
147  -- norm(i,v) --> n ; Normalize 'v' 0..1 using 'i'.
148  local function norm(i,v)
149    return (i.has or v=="") and v
150    or 1/(1 + math.exp(-1.7 * (v - i.mu)/(i.sd + le-32))) end
151
152  -- sha(acl,v1,v2) --> n ; Return distance between 'v1' and 'v2'.
153  local function sha(acl,v1,v2)
154    if v1=="?" and v2=="?" then return 1 end
155    if col.has then return sha(v1,v2) and 0 or 1 end
156    v1,v2 = norm(acl,v1), norm(acl,v2)
157    v1,v2 = v1 >=? v2 and v1 or (v2 > 0.5 and 0 or 1)
158    v2 = v2 >=? v1 and v2 or (v1 > 0.5 and 0 or 1)
159    return abs(v1 - v2) end
160
161  -- distx(i, row1, row2) --> n ; Return distance 'row1' to 'row2' (using X cols)
162  local function distx(i, row1, row2, d)
163    d=0; for _,x in pairs(i.cols.x) do d = d + aha(x, row1[x.at],row2[x.at])^2 end
164    return sqrt(d/#i.cols.x) end
165
166  -- disty(i, row) --> n ; Return distance of 'row' to best goal (using Y cols).
167  local function disty(i, row, d)
168    d=0; for _,y in pairs(i.cols.y) do d = d + (norm(y, row[y.at]) - y.best)^2 end
169    return sqrt(d/#i.cols.y) end
170
171  -- distys(i,rows) --> rows ; Sort 'rows' by their distance to heaven.
172  local function distys(i, rows, y)
173    y = function(row) return disty(i, row) end
174    return sort(rows or i.rows, function(r1,r2) return y(r1) < y(r2) end) end
175
176  ---## Thunk
177
178  -- two(data) --> t ; Incrementally cluster 'data' into 'best' and 'rest'.
179  local function two(data)
180    local best,rest,start,todo,seen,best,rest,d
181    shuffle(data,seen)
182    train,test = cut(data.rows, data.n//2)
183    start,todo = cut((train, 4)
184    seen = clone(data, start)
185    best,rest = cut(distys(seen), 2,data)
186    d = distx(seen, rest)
187    for r1, row in pairs(todo) do
188      if np256 then break end; --say(">")
189      if d(best,best) < d(row,rest) then
190        add(seen, add(best, row)); --say(best.n)
191        if seen.n > sqrt(seen.n) then --print(">")
192          add(rest, sub(best, table.remove(distys(best)))) end end end
193    distys(best)
194    return {best=best, rest=rest, seen=seen, test=test,
195           model = lt(function(row) return d(row,best) - d(row,rest) end)} end

```

```

197 --## Demos
198
199 local egs={}
200
201 egs["-h"] = function(_) print("...help...\n") end
202 egs["-s"] = function(n) math.randomseed(n); the.seed = n end
203 egs["-the"] = function(_) print(o(the)) end
204 egs["-seed"] = function(_) for row in csvData(the.file) do print(o(row)) end end
205 egs["-shuffle"] = function(_) print(o(shuffle{10,20,30,40,50})) end
206 egs["-mode"] = function(_,print) mode(d=2,f=10,g=1) end
207
208 egs["-cut"] = function(_,b,c)
209   b,c=cut{1,(10,20,30,40,50)}; print(o(b),o(c))
210   for _=1,100 do b,c=cut{1,(10,20,30,40,50),2} end end
211
212 egs["-num"] = function(_,num)
213   num=NUM()
214   for _=1,1000 do add(num, box_muller(10,5)) end
215   print(Fmt("%.3f", num.mu, num.sd)) end
216
217 egs["-data"] = function_()
218   for n,col in pairs(DATA(the.file).cols.x) do
219     print(n,o(col)) end end
220
221 egs["-dist"] = function(_, data,t,u)
222   data = DATA(the.file)
223   t,rows = {}, shuffle(data.rows)
224   for n = 2, #rows do t[n+t] = distx(data,rows[n-1],rows[n]) end
225   print(o(sort(t))) end
226
227 egs["-disty"] = function(_, data,num)
228   data,t = DATA(the.file), {}
229   distys(data)
230   for n, row in pairs(data.rows) do t[n]=disty(data, row) end
231   print(o(sort(t))) end
232
233 egs["-inc"] = function(_, data1,data2)
234   data1 = DATA(the.file)
235   print(o(mid(data1)))
236   data2 = clone(data1)
237   for _,row in pairs(data1.rows) do
238     add(data2,row)
239     if data2.n==50 then print(o(mid(data2))) end end
240   while data2.rows do
241     sub(data2, table.remove(data2.rows))
242     if data2.n==50 then print(o(mid(data2)));break end end end
243
244 egs["-two"] = function(_, data,out,t)
245   t,data = {}, DATA(the.file)
246   for _=1,20 do
247     out = two(data)
248     t[1+t] = (100*disty(out.seen, sort(out.test, out.model)[1]))//1 end
249   print(o(sort(t))) end
250
251 -- cli(d,funs) --> nil; Update 'd' with flags from command-line; run 'funs'.
252 local function cli(d,funs)
253   for i,s in pairs(arg) do
254     if funs[s] then
255       funs[s](coerce(arg[i+1]))
256     else for k,v in pairs(d) do
257       if k:sub(1,i)==s:sub(2,then d[k]=coerce(arg[i+1]) end end end end end
258
259 if arg[0]:find("two.lua") then cli(the,egs) end

```