

Dec 19, 21 23:08

tiny.lua

Page 1/4

```

1 local the=require"tiny0"[[
2 lua hint.lua [OPTIONS]
3
4 A small sample multi-objective optimizer / data miner.
5 (c)2021 Tim Menzies <tim@ieee.org> unlicense.org
6
7 OPTIONS:
8 -best X Best examples are in 1.best*size(all) = .2
9 -debug X run one test, show stackdumps on fail = ing
10 -epsilon X ignore differences under epsilon*stdev = .35
11 -file X Where to read data = ../data/auto93.csv
12 -h Show help = false
13 -seed X Random number seed; = 10019
14 -Stop X Create subtrees while at least 2*stop eggs = 4
15 -Tiny X Min range size = size(egs)*tiny = .5
16 -todo X Pass/fail tests to run at start time = ing
17 If "X=all", then run all.
18 If "X=ls" then list all. ]]
19
20 --
21 --
22 local _require="tinylib"
23 local say,fmt,color,out,shout= _say, _fmt, _color, _out, _shout, _csv -- strings
24 local map,copy,keys,push = _map, _copy, _keys, _push -- tables
25 local sort, firsts, seconds = _sort, _firsts, _seconds -- sorting
26 local norm, sum = _norm, _sum -- maths
27 local randi,rand = _randi, _rand -- randoms
28 local same = _same -- meta
29 local csv = _csv -- files
30
31 local ent,mode
32 function ent(t, n,e)
33 n=0; for _,n1 in pairs(t) do n = n + n1 end
34 e=0; for _,n1 in pairs(t) do e = e + n1/n*math.log(n1/n,2) end
35 return e,n end
36
37 function mode(t, most,out)
38 most = 0
39 for x,n in pairs(t) do if n > most then most,out = n,x end end
40 return out end
41
42 --
43 -- Sample
44 --
45 -- [5] Returns a sample, initialized, updated
46 -- [1] Self initialize (if nil, then create).
47 -- [2] Read from disc file
48 -- [3] First item is special (contains names of columns)
49 -- [4] Other rows are the actual examples. Use these to update column headers
50 -- [6] Numeric columns have an "num[n]" entry that tracks the
51 -- "num[n].lo" and "num[n].hi" range for each variable.
52 -- [7] Columns to be minimized or maximized are dependent (listed in "ys")
53 -- [8] All other columns are the independent (listed in "xs")
54 -- [9] Dependent variables are minimized,maximized at weights -1,1
55 -- if their name contains "-" or "+". The number of dependents ins "nys"
56 -- [10] Columns contain ":" are ignored
57 -- [11] Each example will be discretized (later) so each example holds the
58 -- "raw" values (not discretized) and the "cooked" examples (discretized).
59 local slurp,sample,ordered,clone
60 function slurp(out)
61 for eg in csv(the.file) do out=sample(eg,out) end --[2]
62 return out end
63
64 function clone(i, inits, out)
65 out = sample(i.heads)
66 for _,eg in pairs(inits or {}) do out = sample(eg,out) end
67 return out end
68
69 function sample(eg,i)
70 local numeric,independent,dependent,head,data,datum
71 function head(n,x)
72 function numeric(i) i.num[n]= (hi=-math.huge,lo=math.huge) end -- [6]
73 function independent(i) i.xs[n]= x end -- [8]
74 function dependent(i)
75 i.num[n].w = x:find="-" and -1 or 1 -- [9]
76 i.ys[n] = x
77 i.nys = i.nys+1 end
78 if not x:find":" then -- [10]
79 if x:match("[A-Z]") then numeric(i) end
80 if x:find="-" or x:find"+" then dependent(i) else independent(i) end end --[7,8]
81 return x end
82 function data(eg) return {raw=eg, cooked=copy(eg)} end --[11]
83 function datum(n,x) -- [4]
84 if x ~= "?" then
85 local num=i.num[n]
86 if num then
87 num.lo = math.min(num.lo,x) -- [6]
88 num.hi = math.max(num.hi,x) end end -- [6]
89 return x end
90 eg = eg.raw and eg.raw or eg
91 if i then push(i.egs, data(map(eg,datum))) else -- [4]
92 i = {xs={},nys=0,ys={},num={},egs={},divs={},heads={}} -- [1] [3]
93 i.heads = map(eg,head) end -- [3]
94 return i end -- [5]
95
96 -- [14] Returns the sample, examples sorted by their goals, each example
97 -- tagged with "eg.klass=best" or "eg.klass=rest" if "eg" is in the top
98 -- "the.best" in the sort.
99 -- [12] Sort each example by exploring all goals (dependent variables).
100 -- [15] The direction that losses the most points to best example.
101 -- e.g. a.b=-7,.6 and a-b is .1 (small loss) and b-a is -.1
102 -- (much smaller than a or b) so a is more important than b.
103 -- [13] Goal differences are amplified by raising them to a power (so normalize
104 -- the goals first so you that calculation does not explode.
105 function ordered(i)
106 local function better(eg1,eg2, a,b,s1,s2)
107 s1,s2=0,0
108 for n,_ in pairs(i.ys) do -- [12]
109 local num = i.num[n]
110 a = norm(num.lo, num.hi, eg1.raw[n]) -- [13]
111 b = norm(num.lo, num.hi, eg2.raw[n]) -- [13]
112 s1 = s1 - 2.71828*(num.w * (a-b)/i.nys) -- [13] [15]
113 s2 = s2 - 2.71828*(num.w * (b-a)/i.nys) end -- [13] [15]
114 return s1/i.nys < s2/i.nys end -- [15]
115 for j,eg in pairs(sort(i.egs,better)) do
116 if j < the.best*#i.egs then eg.klass="best" else eg.klass="rest" end end
117 return i end -- [14]
118
119

```

Dec 19, 21 23:08

tiny.lua

Page 2/4

```

120 --
121 --
122 --
123 local discretize, xys_sd, bin, div
124 function bin(z,divs)
125 if z=="?" then return "?" end
126 for n,x in pairs(divs) do
127 if x.lo<= z and z<= x.hi then return n end end end
128
129 function discretize(i)
130 function xys_sd(col,egs, out,p)
131 out={}
132 for _,eg in pairs(egs) do
133 local x=eg.raw[col]
134 if x=="?" then push(out, {x=x, y=eg.klass}) end end
135 out = sort(out, function(a,b) return a.x < b.x end)
136 p = function(z) return out[z*#out//10].x end
137 return out, math.abs(p(.9) - p(.1))/2.56
138 end
139 for col,name in pairs(i.xs) do
140 if i.num[col] then
141 local xys,sd = xys_sd(col, i.egs)
142 i.divs[col] = div(col,name,xys, (#xys)^the.Tiny, the.epsilon*sd)
143 for _,eg in pairs(i.egs) do
144 eg.cooked[col]= bin(eg.raw[col], i.divs[col]) end end end
145 return i end
146
147 local function showDiv(i,at,val, out)
148 out="?"
149 if i.num[at] then
150 for k,div in pairs(i.divs[at]) do
151 if k=val then out =fmt("%s<=%s<=%s",div.lo, i.xs[at], div.hi) end end
152 else out=fmt("%s=%s", i.xs[at], val) end
153 return out end
154
155 function div(col,name,xys,tiny,epsilon, one,all,merged,merge)
156 function merged(a,b,an,bn, c)
157 c={}
158 for x,v in pairs(a) do c[x] = v end
159 for x,v in pairs(b) do c[x] = v + (c[x] or 0) end
160 if ent(c*.99 <= (an*ent(a) + bn*ent(b))/(an+bn) then return c end
161 end
162 function merge(b4)
163 local j,tmp = 0,{}
164 while j < #b4 do
165 j = j + 1
166 local now, after = b4[j], b4[j+1]
167 if after then
168 local simpler = merged(now.has,after.has, now.n,after.n)
169 if simpler then
170 now = {col=col,name=name, lo=now.lo, hi=after.hi,
171 n=now.n+after.n, has=simpler}
172 j = j + 1 end end
173 return #tmp==#b4 and b4 or merge(tmp) -- recurse until nothing merged
174 end
175 one = {col=col,name=name,lo=xys[1].x, hi=xys[1].x, n=0, has={}}
176 all = {one}
177 for j,xy in pairs(xys) do
178 local x,y = xy.x, xy.y
179 if j<#xys-tiny and x== xys[j+1].x and one.n> tiny and one.hi-one.lo>epsilon
180 then one = push(all, {col=col,name=name,lo=one.hi, hi=x, n=0, has={}})
181 end
182 one.n = 1 + one.n
183 one.hi = x
184 one.has[y] = 1 + (one.has[y] or 0); end
185 return merge(all) end
186
187

```

Dec 19, 21 23:08

tiny.lua

Page 3/4

```

188 -- T ( , , )
189 --
190 local splitter, worth, tree, count, keep, tree
191
192 function count(t, at) t=t or {}; t[at]=1+(t[at] or 0); return t end
193 function keep(t, at, x) t=t or {}; t[at]=t[at] or {}; push(t[at], x); return t end
194
195 function splitter(xs, eggs)
196   function worth(at, xy, n, x, xpect)
197     xy, n = {}, 0
198     for _, eg in pairs(egs) do
199       x = eg.cooked[at]
200       if x ~= "?" then
201         n=n+1
202         xy[x] = count(xy[x] or {}, eg.klass) end end
203     return (at, sum(xy, function(t) local e, n1=ent(t); return n1/n* e end)) end
204   return sort(map(xs, worth), seconds)[1][1] end
205
206 function tree(xs, eggs, lvl)
207   local here, at, splits, counts
208   for _, eg in pairs(egs) do counts=count(counts, eg.klass) end
209   here = {mode=mode(counts), n=#egs, kids={}}
210   if #egs > the.Stop then
211     splits, at = {}, splitter(xs, eggs)
212     for _, eg in pairs(egs) do splits=keep(splits, eg.cooked[at], eg) end
213     for val, split in pairs(splits) do
214       if #split < #egs and #split > the.Stop then
215         push(here.kids, {at=at, val=val,
216           sub=tree(xs, split, (lvl or "").."|..")}) end end end
217   return here end
218
219 local function show(i, tree)
220   local vals=function(a,b) return a.val < b.val end
221   local function showl(tree, pre)
222     if #tree.kids==0 then io.write(fmt("==> %s[%s]", tree.mode, tree.n)) end
223     for _, kid in pairs(sort(tree.kids, vals)) do
224       io.write("\n"..fmt("%s%s", pre, showDiv(i, kid.at, kid.val)))
225       showl(kid.sub, pre.."|..") end
226   end
227   showl(tree, ""); print("") end
228
229

```

Dec 19, 21 23:08

tiny.lua

Page 4/4

```

230 -- T ( , , )
231 --
232 local go={}
233
234 function go.ls()
235   print("ulua", .arg[0].." -todo ACTION\n\nACTIONS:")
236   for _, k in pairs(keys(go)) do print(" -todo", k) end end
237   function go.the() shout(the) end
238   function go.bad( s) assert(false) end
239   function go.ing() return true end
240   function go.ordered( s, n)
241     s = ordered(slurp())
242     n = #s.egs
243     shout(s.heads)
244     for i=1,15 do shout(s.egs[i].raw) end
245     print("#")
246     for i=n,n-15,-1 do shout(s.egs[i].raw) end
247     n={}; for _, eg in pairs(s.egs) do n=count(n, eg.klass) end
248     shout(n)
249   end
250
251   function go.bins( s)
252     s= discretize(ordered(slurp()))
253     for m, div in pairs(s.divs) do
254       print("")
255       for n, div1 in pairs(div) do print(m, n, out(div1)) end end
256   end
257
258   function go.tree( s, t)
259     s = discretize(ordered(slurp()))
260     show(s, tree(s.xs, s.egs))
261   end
262
263 -- Start - tip
264 --
265 --
266 --
267 the(go)

```

Dec 19, 21 18:10

tinylib.lua

Page 1/1

```

1 local lib={}
2
3 --
4 -- Strings
5
6 lib.fmt = string.format
7
8 function lib.say(...) print(lib.fmt(...)) end
9 function lib.color(n,s) return lib.fmt("\27[1m\27[%sm%s\27[0m",n,s) end
10 function lib.shout(x) print(lib.out(x)) end
11
12 function lib.out(t, u,key,val)
13   function key(_,k) return string.format("%s %s", k, lib.out(t[k])) end
14   function val(_,v) return lib.out(v) end
15   if type(t) ~= "table" then return tostring(t) end
16   u = #t>0 and lib.map(t, val) or lib.map(lib.keys(t), key)
17   return "{ "..table.concat(u," ").."}" end
18
19 --
20 -- Tables
21
22 function lib.push(t,x) t[1+#t]=x; return x end
23 function lib.copy(t, u) u={};for k,v in pairs(t) do u[k]=v end; return u end
24
25 function lib.map(t,f, u)
26   u,f={},f or same; for k,v in pairs(t) do u[1+#u] = f(k,v) end; return u end
27
28 function lib.keys(t,u)
29   u={}; for k,_ in pairs(t) do u[1+#u]=k end;return lib.sort(u);end
30
31 --
32 -- Sorting
33
34 function lib.sort(t,f) table.sort(t,f); return t end
35 function lib.firsts(x,y) return x[1] < y[1] end
36 function lib.seconds(x,y) return x[2] < y[2] end
37
38 --
39 -- Maths
40
41 function lib.norm(lo,hi,x)
42   return math.abs(lo-hi)<1E-32 and 0 or (x-lo)/(hi-lo) end
43
44 function lib.sum(t,f, n)
45   n,f=0,f or same; for _,v in pairs(t) do n = n + f(v) end; return n end
46
47 --
48 -- Random
49
50 function lib.randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end
51
52 function lib.rand(lo,hi)
53   lo, hi = lo or 0, hi or 1
54   the.seed = (16807 * the.seed) % 2147483647
55   return lo + (hi-lo) * the.seed / 2147483647 end
56
57 --
58 -- Maths
59
60 function lib.same(x,...) return x end
61
62 --
63 -- Files
64
65 function lib.csv(file, x)
66   file = io.input(file)
67   return function() t,tmp
68     x = io.read()
69     if x then
70       t={}
71       for y in x:gsub("[\t]", ""):gmatch("[^\t]+") do t[1+#t]=tonumber(y) or y end
72       if #t>0 then return t end
73     else io.close(file) end end end
74
75 --
76 -- Return
77
78 return lib
79
```

Dec 19, 21 0:20

tiny0.lua

Page 1/1

```

1 -- standard load and start functions
2 -- first line of code should be a help string (e.g. see tiny.lua)
3 -- last line of code should call this code, pass in table of actions
4 -- e.g
5 --     the(go)
6
7 --
8 -- Rogues
9
10 -- at load time, remember the current globals
11 local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
12 -- after start time, complain if code has created rogue globals
13 local function rogues()
14   for k,v in pairs(_ENV) do if not b4[k] then print("?:",k,type(v)) end end end
15
16 --
17 -- Misc
18
19 -- Table keys, in sorted order
20 local function keys(t,u)
21   u={}; for k,_ in pairs(t) do u[1+#u]=k end; table.sort(u); return u end
22
23 -- pretty colors, n=(31,32),=(red,green)
24 local function color(n,s) return string.format("\27[1m\27[%sm%s\27[0m",n,s) end
25
26 -- shallow copy of a list
27 local function copy(t, u)
28   u={}; for k,v in pairs(t) do u[k]=v end ; return u end
29
30 --
31 -- Start-up
32
33 local help = ""
34
35 -- All the start-up actions:
36 -- [1] keep a copy of the options as "defaults"
37 -- [2] maybe just show the help text
38 -- [3] maybe run an action in verbose mode (show stackdump; halt on error)
39 -- [4] before actions, reset options to defaults
40 -- [5] before actions, reset random number seed
41 -- [6] maybe run an action in fast mode (no stackdumps; no halts one errors)
42 -- [7] for fast mode, count the number of failures
43 -- [8] return to the operating system the count of failures
44 -- [9] light now, we just print rogue globals)
45 local function what2doAtLastLine(options, actions)
46   local fails, defaults = 0, copy(options) -- [1]
47   if options.h then return print(help) end -- [2]
48   if options.debug then actions[options.debug]() end -- [3]
49   local todos = options.todos == "all" and keys(actions) or {options.todo}
50   for _todo in pairs(todos) do
51     if type(actions[_todo]) ~= "function"
52     then print(color(31,"NOFUN"),_todo)
53     else for k,v in pairs(defaults) do options[k]=v end -- [4]
54     options.seed = options.seed or 10019 -- [5]
55     local ok,msg = pcall(actions[_todo]) -- [6]
56     if ok then print(color(32,"PASS"),_todo)
57     else print(color(31,"FAIL"),_todo,msg)
58     fails=fails+1 end end -- [7]
59   end
60   os.exit(fails) end -- [8]
61
62 --
63 -- Lib 0, Lib 0
64
65 -- In paragraph of the text that starts with "Options", all lines that start with
66 -- "flag" have a default value as the last word on that line.
67 -- [1] Build the "options" array from those flags and defaults
68 -- [2] Check if we can update those defaults from command line arguments).
69 -- [3] Anything on the command line is a string. Check if these can become nums
70 -- For the sake of brevity:
71 -- [4] command line flags need only match the start of the flag;
72 -- [5] for boolean values, -flag flips the default boolean
73 -- [6] add in the ability to call "what2doAtLastLine"
74 local function what2doAtFirstLine(txt)
75   local options={}
76   help = txt
77   txt:gsub("^.*OPTIONS:", ""):gsub("%s*-[^\s%+]+[^\n]*%s*([^\s%+]+)",
78     function(flag,x)
79       for n,word in ipairs(arg) do -- [2]
80         if flag:match("^"..word:sub(2).."") then -- [4]
81           x=(x=="false" and "true") or (x=="true" and "false") or arg[n+1] end end
82         if x=="true" then x=true
83         elseif x=="false" then x=false -- [4]
84         else x=tonumber(x) or x -- [3]
85       end
86       options[flag] = x end) -- [1]
87   return setmetatable(options,{__call=what2doAtLastLine}) end -- [6]
88
89 --
90 -- Return
91
92 return what2doAtFirstLine
93
94
```