Dec 04, 21 19:57

line = io.read()
return function(

```
Page 1/5
 #!/usr/bin/env lua
-- vim: filetype=lua ts=2 sw=2 et:
local b4={}; for k,v in pairs(_ENV) do b4[k]=v end
                                    Hearning)
                             -]] local options=
  \label{eq:what} \begin{array}{ll} \mbox{what} &= \mbox{"Small sample multi-objective optimizer.",} \\ \mbox{usage= "(c) 2021 Tim Menzies < timm@ieee.org> unlicense.org",} \\ \end{array}
  about= [[
Sort N examples on multi-goals using a handful of 'hints'; i.e.
 - Evaluate and rank, a few examples (on their y-values);
- Sort other examples by x-distance to the ranked ones;
- Recurse on the better half (so we sample more and more
from the better half, then quarter, then eighth...).
 A regression tree learner then explores the examples (sorted left to right, worst to best). By finding branches that reduce the variance of the index of those examples, this tree reports what attribute ranges select for the better (or worse) examples. ]],
how= { "file", "-f", ".j./data/auto/3.csv", "read data from file" }, "how { "file", "-f", false , "show help" }, "help", "-h", false , "show help" }, "how h
 function cli(opt, u)
       u={} for _t in pairs(opt.how) do u[t[1]] = t[3] for n, word in ipairs(arg) do if word==t[2] then u[t[1]] = t[3] and (tonumber(arg[n+1]) or arg[n+1]) or true end end if u.help then help(opt) end math.randomseed(u.seed or 100019) return u end
 the = cli(options) -- e.g. the = {seed=10019, help=false, p=2...}
-- table tricks
local cat,map,lap,keys, copy,pop,push,sort,firsts,first,second,shuffle,bchop cat = table.concat sort = function(t,f) table.sort(t,f); return t end push = table.insert pop = table.remove first = function(t) return t[1] end second = function(t) return t[2] end firsts = function(a,b) return first(a) < first(b) end
         unction shuffle(t, j)
for i=#t,2,-1 do j=math.random(1,i); t[i],t[j]=t[j],t[i] end; return t end
 function lap(t.f) return map(t.f.1) end
function map(t,f,one, u)
u=(); for x,y in pairs(t) do
if one then x,y=f(y) else x,y=f(x,y) end
if x ~= nil then
if y then u[x]=y else u[1+#u]=x end end end
return u end
 -- binary chop (assumes sorted lists)
function bchop(t,val,lt,lo,hi, mid)
lt = lt or function(x,y) return x < y end
lo,hi = lo or 1, hi or #t
while lo <= hi do
mid = (lo+hi) // 2
if lt(tmid],val) then lo=mid+1 else hi= mid-1 end end
return math.min(lo,#t) end
          maths tricks
  local abs, norm, sum, rnd, rnds
abs = math.abs
  function rnd(x,d, n)
  n=10^(d or 0); return math.floor(x*n+0.5) / n end
 function rnds(t,d)
  return lap(t, function(x) return rnd(x,d) end ) end
  function norm(x,lo,hi)
if x=="" then return x end
return abs(hi - lo) < 1E-32 and 0 or (x - lo)/(hi - lo) end</pre>
 function sum(t,f)
f= f or function(x) return x end
out=0; for _, x in pairs(f) do out = out + f(x) end; return out end
 -- printing tricks
local out, shout, red, green, yellow, blue
function red(s) return "27[im27[3]m"..s.."27[0m" end
function green(s) return "27[im27[3]m"..s.."27[0m" end
function yellow(s) return "27[im27[3]m"..s.."27[0m" end
function blue(s) return "27[im27[3]m"..s.."27[0m" end
  shout= function(x) print(out(x)) end
 function out(t, seen, u, key, value, public)
function key(k) return fmt(".%% %%", blue(k), out(t[k], seen)) end
function value(v) return out(v, seen) end
if type(t) == "function" then return "(...)" end
if type(t) -= "table" then return tostring(t) end
seen = seen or ()
if seen[t] then return "..." else seen[t] = t end
u = #t>0 and lap(t, value) or lap(keys(t), key)
return red((t._is or"").."[")..cat(u,"")..red(")") end
  --- file i/o tricks
local csv
function csv(file, line)
file = io.input(file)
```

152 153	<pre>line = io.read() return function(t,tmp) if line then</pre>
154 155	t={} for cell in line:gsub("[\text{\text{tr}}]*",""):gsub("#.*",""):gmatch("([^,]+)") do
156 157	<pre>push(t, tonumber(cell) or cell) end line = io.read()</pre>
158 159	<pre>if #t>0 then return t end else io.close(file) end end end</pre>
160 161	oo tricks
162 163 164	local has,obj function has(mt,x) return setmetatable(x,mt) end
165	function obj(s, o,new)
167 168	<pre>o = {_is=s,tostring=out} oindex = o return setmetatable(o, {call = function(_,) return o.new() end}) end</pre>
169 170	local Nums=obj"Nums"
171 172	<pre>function Nums.new(inits, self) self= has(Nums,{has={}}, n=0, ready=true})</pre>
173 174 175	<pre>for _,one in pairs(inits or {}) do self:add(one) end return self end</pre>
176	<pre>function Nums:add(x) push(self.has,x); self.n=self.n+1; self.ready=false end</pre>
178 179	<pre>function Nums:all(x)</pre>
180 181	<pre>if not self.ready then table.sort(self.has) end self.ready = true</pre>
182 183 184	return self.has end function Nums:per(p, here,t)
185	function here(x) x=x*#t//1; return x < 1 and 1 or x>#t and #t or x end t=self:all()
187 188	<pre>return #t <2 and t[1] or t[here(p or .5)] end</pre>
189 190	<pre>function Nums:sd() return (self:per(.9) - self:per(.1))/ 2.56 end</pre>
191	function Nums:merge(other, new) new = Nums.new(self.has)
193 194 195	<pre>for _,x in pairs(other.has) do new:add(x) end return new end</pre>
196 197	<pre>function Nums:mergeable(other, new,b4) new = self:merge(other) b4 = (self.n*self:sd() + other.n*other:sd()) / new.n</pre>
198 199	<pre>b4 = (self.n*self:sd() + other.n*other:sd()) / new.n if b4 >= new:sd() then return new end end</pre>
200	decompliant in twister
202 203 204	doscretization tricks local splits={} function splits.best(sample, best,tmp,xpect,out)
205	<pre>best = maths.huge for .x in pairs(sample.xs) do</pre>
207 208	<pre>tmp, xpect = splits.whatif(x.at,self) if xpect < best</pre>
209 210	<pre>then out,best = tmp,xpect end end return out end</pre>
211 212 213	<pre>function splits.whatif(col, sample, out = splits.spans(col, sample)</pre>
214 215	<pre>xpect = sum(out, function(x) return x.has.n*x:sd() end)/#sample.egs out = map(out, function(_,x) x.has=x.has:all(); x.col= col end)</pre>
216 217	return out, xpect end
218 219 220	<pre>function splits.spans(col,sample, xs,xys, symbolic,x) xys,xs, symbolic = {}, Nums(), sample.nums[col] for rank,eq in pairs(sample.egs) do</pre>
221	x = eg[col] if x ~= "?" then
223 224	xs:add(x) if symbolic
225 226	<pre>then in symbolic columns, xys are the indexes seen with each symbol xys[x] = xys[x] or {)</pre>
227 228 229	<pre>push(xys[x], rank) else in numeric columns, xys are each number paired with its row id push(xys, (x=x,y=rank)) end end</pre>
230	end if symbolic
233	then return map(xys, function(x,t) return {lo=x, hi=x, has=Nums(t)} end) else return splits.merge(
232 233 234 235 236	<pre>else return splits.merge(</pre>
233 234 235 236 237 238	<pre>else return splits.merge(</pre>
233 234 235 236 237 238 239 240	<pre>else return splits.merge(</pre>
233 234 235 236 237 238 239 240 241 242	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249	<pre>else return splits.merge(</pre>
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 255 256 257 258 260	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 250 251 252 253 254 255 256 257 257 258 259 260 261 262	<pre>else return splits.merge(</pre>
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 265 265 265 265 265 265 265 265 265	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 250 251 252 253 254 255 256 257 258 259 261 262 263 264 267	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 250 251 255 256 256 266 265 266 266 266 266 266	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 251 255 256 257 258 259 260 267 268 269 267 268 269 270 271 272	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 250 251 252 253 254 255 255 265 266 267 268 269 270 271 272 273	else return splits.merge(
233 234 235 236 237 238 2240 241 242 244 245 246 247 2248 250 251 252 253 254 256 257 258 266 267 268 269 270 271 272 273 274	else return splits.merge(
233 234 235 236 237 238 240 241 242 242 243 244 245 246 250 251 252 253 255 266 267 268 269 270 271 272 277 278	else return splits.merge(
233 234 235 236 237 238 244 245 246 247 248 250 251 252 253 254 255 256 257 258 260 261 277 278 277 278 277 278 279 280	else return splits.merge(
233 234 235 236 237 240 241 255 256 256 256 266 267 277 278 277 278 277 278 281 282 283 283 283 283 284 285 286 286 286 286 286 286 286 286 286 286	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 245 246 247 248 245 255 256 257 262 263 264 265 266 267 268 269 277 278 278	else return splits.merge(
233 234 235 236 237 238 239 239 239 239 240 241 242 245 251	else return splits.merge(
232 (234) 234 (234) 235 (234)	else return splits.merge(
233 (234) 234 (235) 235 (236) 237 (237) 238 (237) 240 (247)	else return splits.merge(
233 234 235 235 235 235 235 235 235 235 235 235	else return splits.merge(
233 234 235 236 237 238 239 240 241 242 243	else return splits.merge(
223 234 24 24 24 24 24 24 24 24 24 24 24 24 24	else return splits.merge(
333 333 333 333 333 333 333 333 333 33	else return splits.merge(

Dec 04, 21 19:57 I5.lua Page 3/5

```
function Sample:tree(min, node,min,sub)
node = {node=self, kids={}}
min = min or (#self.egs)^the.small
if #self.egs >= 2*min then
           --- here
for _,span in pairs(splits.best(sample)) do
    sub = self:clone()
    for _,at in pairs(span.has) do sub:add(self.egs[at]) end
    push(node.kids, span)
    span.has = sub:tree(min) end end
return node end
            - at node
unction Sample:where(tree,eg, max,x,default)
if #kid.has==0 then return tree end
          if #kid.has==U then return tree end
max = 0
for _kid in pairs(tree.node) do
if #kid.has > max then default,max = kid,#kid.has end
x = eg[kid.col]
if x ~= "" then
if x <= kid.hi and x >= kid.lo then
    return self:where(kid.has.eg) end end end
return self:where(default, eg) end
 -- geometry tricks
-- y column rankings
local dist, better,betters
function dist(egl,eg2,sample, a,b,d,n,inc,dist1)
function dist(egl,eg2,sample, a,b,d,n,inc,dist1)
function of the column then return a==b and 0 or 1 end
if a= "" then b=norm(b, num.lo, num,hi); a = b>.5 and 0 or 1
elseif b= " then a=norm(a, num.lo, num,hi); b = a>.5 and 0 or 1
elseif b= " then a=norm(a, num.lo, num.hi); norm(b, num.lo, num.hi)
end
return abs(a-b)
end
d,n=0.0
     -- ordered object
-- per sd add sort here. mergabe
           end - d, n=0, 0 for col_{n-1} in pairs(sample.xs) do a, b=egl[col], eg2[col] inc a==*?* and b==?* and b==?* and 1 or distl(sample.nums[col],a,b) d=d+inc^{+}tb=p n=n+1 end n=n+1 en
     function betters(egs,sample)
  return sort(egs,function(a,b) return better(a,b,sample) end) end
  function better(a,f), each better(a,f), set of the first 
   -- sample sample sorting local hints={} function hints.default(eg) return eg end
  return evals, sample:clone(train), sample:clone(test) end

function hints.recurse(sample, egs, evals, scorefun, out, small, worker)

if #egs < small then
for i=1, #egs do push(out, pop(egs)) end
return evals, out
end
local scoreds = {}
function worker(eg) return hints.locate(scoreds, eg, sample) end
for j=1, the.hints do evals=evals+1;
push(scoreds, scorefun(pop(egs))) end
scoreds = betters(scoreds, sample)
egs = lap(sort(lap(egs, worker), firsts), second)
for i=1, #egs//2 do push(out, pop(egs)) end
return hints.recurse(sample, egs, evals, scorefun, out, small)
end
  closest, rank, tmp)
  function eg.norm()
  assert(norm(5,0,10)==.5,"small") end
   function eg.map()
  assert(3==map({1,2},function(_,x) return x+1 end)[2]) end
     function eg.tables()
  assert(20==sort(shuffle({{10,20},{30,40},{40,50}}),firsts)[1][2]) end
     function eg.csv( n,z)
            for eg in csv(the.file) do n=n+1; z=eg end assert(n==399 and z[#z]==50) end
   function eg.nums( n1,n2,n3,n4)
n1=Nums(10,20,30,40,50,10,20,30,40,50,10,20,30,40,50)
n2=Nums(10,20,30,40,50,10,20,30,40,50,10,20,30,40,50)
n2=Nums(10,20,30,40,50,10,20,30,40,50,10,20,30,40,50)
n3=Nums(10,20,30,40,50,10,20,30,40,50,10,20,30,40,50)
n4=Nums(10,20,30,40,50,10,20,30,40,50,10,20,30,40,50)
assert(n3:mergeable(n4)==ni1) end
```

Printed by Tlm Menzies Dec 04, 21 19:57 I5.lua Page 4/5

476 --[[
477 needs stats on samples
478
479 teaching:
480 - sample is v.useful
481
482
483 --]]