Inductive Decision Tree Learning (the basic ML)

The classification problem

- given a finite set of examples of concepts expressed as values for features, where the examples are classified into different groups
- Discover rules or reasons for the classification that accurately and efficiently classifies the training examples, and
- Generalizes to unseen examples in the same domain.
  - Can be seen as a form of Knowledge-compression

Approaches

- symbolic approaches would like to qualify the rules or reasons so that they are compact, understandable to a human performing the same task, executable rapidly
- Superproblem is to allow non-feature based approaches, where the inputs are described in a less constrained language
- Superproblem is to not be given the classes
  - This is clustering or "unsupervised learning"
- Superproblem is to work with raw input, e.g. pixels or soundwaves, not features
- Subproblem is to have clean boolean features

Historical Milestones

- Quine-McClusky Logic Minimization
- Michaelski's AQ11 - predicate logic
- * Quinlan's/Fisher's decision trees
- Mitchell's LEX - refined concept spaces
- * RHW Back Propagation - Layered NN's

Sample Problem

	tenfloors	reception	heliport	parking	sat.dish	signage
hospital	0	1	1	1	0	0
hospital	0	1	0	1	1	0
office	1	1	0	0	1	0
office	1	0	1	0	0	0
mall	0	0	0	1	1	1
mall	0	1	0	1	0	1

3 Different Approaches?

- Storing the whole database?
  - Just storing the table does not enable generalization, nor is it compact
- Combined with Similarity function however, can generalize
- Similarity functions include
  - Distance (hamming, euclidean, etc)

3 Different Approaches?

- Extract Logical rules from the data which apply to each category.
  - ~Parking -> Office
  - Sat.dish &~Heliport -> Mall
  - Parking & ~signage -> Hospital
- Minimal Logic related to the NP-Complete Vertex-covering problem!

3 Different Approaches?

- Extract a Decision Tree

```
If building is tall,
   then it is office building
   else if it has signage,
           then it is a mall,
           else it is a hospital
```

Algorithm for Decision Tree

```
If all data is in the same class,
return that class,
Else
Select variable to make decision

Split into (two) subproblems by
removing that column, and breaking
apart rows by value of the variable

Recursively solve those problems
```

File format --> internal data format

- 3
- 2 2 2
- 6 6
- 0 1 1 1 0 0
- 0 1 0 1 1 0
- 1 1 0 0 1 0
- 1 0 1 0 0 0
- 0 0 0 1 1 1
- 0 1 0 1 0 1

- ((1 0 1 1 1 0 0)
- (1 0 1 0 1 1 0)
- (2 1 1 0 0 1 0)
- (2 1 0 1 0 0 0)
- (3 0 0 0 1 1 1)
- (3 0 1 0 1 0 1))

Read data from a file to lisp form

```
(defun readin (file &aux nc inclass classes nv ne)
 (with-open-file (fp file :direction :input)
  (setq nc (read fp))
  (setq inclass (loop for i from 1 to nc collect (read fp)))
  (setq classes (loop for x in inclass as i from 1
                      nconc (loop for j from 1 to x collect i)))
  (setq ne (read fp))
  (setq nv (read fp))
  (loop for i from 1 to ne as c in classes
        collect (cons c (loop for j from 1 to nv collect (read fp))))))
```

Induce a Tree with random choice

```
(defun dumb-decision-tree (data &optional columns)
 (or columns (setq columns (loop for i from 0 to (1- (length (car data)))
                                 collect i)))
 ;; if all cases are in the same class, stop and return that class
 (if (loop for x in (cdr data) always (eq (car x) (caar data)))
     (caar data)
     ;; else split on random variable
     (let ((col (random (length columns))))
       (list (nth col columns)
             (dumb-decision-tree (split data col 1)
                                 (remove (nth col columns) columns))
             (dumb-decision-tree (split data col 0)
                                 (remove (nth col columns) columns))))))
```

Split is called twice or more

- ;returns submatrix along colnum returning rows where col=val
- (defun split (data colnum val)
- (loop for line in data when (eq val (nth colnum line))
- collect (loop for x in line as i from 0
- unless (= i colnum)
- collect x)))

Demo...

- with random column selection, we get a LARGE decision tree...

How to select variable?

- Need to compute a function for each attribute that predicts how well it discriminates the data

Mutual Information

- A variable provides perfect information if the classes can be discriminated, some information if it is unbalanced, and none it it doesn't change or is balanced.

	PERFECT	GOOD	BAD	BAD
A	0	0	0	0
A	0	0	1	0
A	0	0	0	0
A	0	1	1	0
B	1	0	1	0
B	1	1	0	0
B	1	1	1	0
B	1	1	0	0

Computing Mutual Information of Variables

p(hospital) = .33, p (office) = .33, p (mall) = .33

p (10f) = .33, p(sign) = .33,p(recept) = .66
p(heliport) = .33, p(parking) = .66, p(satellite) = .5,

p(hospital,10F) = 0, p(hospital,~10f) = .33
p (office,10f)=.33 p(office,~10f)=0
p (mall,10f)=0 p(mall,~10f) = .33
how good is TENFLOORS?
p(w,x) ln p(w,x) /p(w) * p(x) ***(when p(w,x) aint 0!!!)***

hosp/~10f = .33 ln .33/(.33*.66)+
office/10f = .33 ln .33/(.33*.33)+
mall/~10f = .33 ln .33/(.33*.66)= .63

Decision Tree Induction

```
(defun build-decision-tree (data &optional columns)
  (or columns (setq columns (loop for i from 0 to (1- (length (car data)))
                                  collect i)))
  ;; if all cases are in the same class, stop and return that class
  (if (loop for x in (cdr data) always (eq (car x) (caar data)))
      (caar data)
      ;; else split on the "best" variable and recurse
      (let ((col (find-best-variable data)))
        (list (nth col columns)
              (build-decision-tree (split data col 1)
                                   (remove (nth col columns) columns))
              (build-decision-tree (split data col 0)
                                   (remove (nth col columns) columns))))))
```

FIND BEST VARIABLE

- which variable to split on is a bit of a heuristic.
- Mutual Information is a set of equations about how well the remaining features/variables/columns divide up the training set.
- read paper.. too complex for a lecture.
- pr(class), pr(feature) (pr class given feature), etc. are just counts.

calculating mutual info

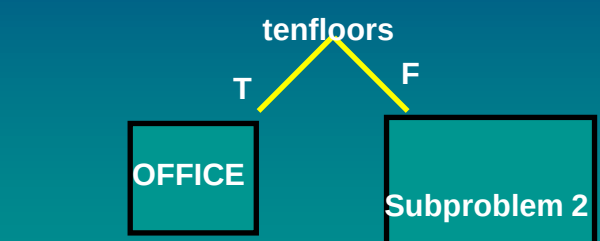
```
(defun calcentropy (data)
 (let ((numevents (length data))
       (classes (remove-duplicates (loop for x in data collect (car x))))
       (numvars (1- (length (car data)))))
  (let ((pclass (loop for x in classes
                      collect (/ (loop for y in data
                                       count (eq x (car y))) numevents)))
        (pvar0 (loop for i from 1 to numvars
                     collect (/ (loop for y in data
                                      count (zerop (nth i y)))
                                numevents)))
        (pjoint (loop for v from 1 to numvars collect
                      (loop for c in classes collect
                            (loop for i in '(0 1) collect
                                  (/ (loop for x in data count
                                           (and (eq (car x) c)
                                                (= (nth v x) i)))
                                     numevents))))))
```

(continued)

```
(loop for v from 1 to numvars as pv in pvar0 as pjl in pjoint
          collect (loop for c in classes as pc in pclass as pj in
 pjl
                    ;; oops fix for floatingpoin
                    sum (+ (if (zerop (car pj)) 0
                               (* (car pj)
                                  (log (/ (car pj)(* pv pc)))))
                           (if (zerop (cadr pj)) 0
                               (* (cadr pj)
                                  (log (/ (cadr pj)(* (- 1.0 pv)
                                                      pc)))))))))
```

Pick a good feature, then recurse

- Tenfloors, parking, and signage (.63) all discriminate better than reception and heliport (.17), which are better than satellite dish (0.0)!



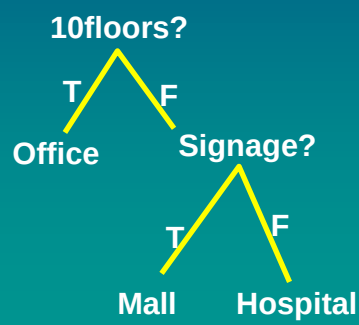
If a subproblem is entirely in a single class, that is a terminates recursion

Subproblem (~ 10floors)

	.21 reception	.21 heliport	0 parking	0 sat.dish	.69 signage
Hospital	1	1	1	0	0
hospital	1	0	1	1	0
mall	0	0	1	1	1
mall	1	0	1	0	1

Information from Features change!

- Full problem, signage and parking were equally useful; now that we selected 10floors, parking has become useless.
- Final Decision Tree:



CONVERTING NUMBERS TO SYMBOLS

```
(defparameter classes '(hospital office mall))

(defparameter features '(tenfloors reception heliport parking
 sattelite signage))

(defun labelit (tree features classes)
  (if tree
      (if (atom tree) (nth (- tree 1) classes)
          (list (nth (- (car tree) 1) features)
                (labelit (cadr tree) features classes)
                (labelit (caddr tree) features classes)))
   (print "ERROR: NIL in tree"))) ;;ELSE
```

OPERATIONAL

```
(dumb-decision-tree mydata)
(6 3 (4 1 2))

* (labelit '(6 3 (4 1 2)) features classes)

(SIGNAGE MALL (PARKING HOSPITAL
 OFFICE))
*
```

interptree

```
(defun interptree (tree namefile  &optional classes vars)
  ;; if we've read the file, then
  (if classes
     ;; if the tree is atom, it is a class, look up in classes
     (if (atom tree) (nth (1- tree) classes)
        ;;else look up which variable, and recurse
        (cons 'if (cons (nth (1- (car tree)) vars)
              (list 'then (interptree (cadr tree) namefile classes vars)
                    'else (interptree (caddr tree) namefile classes vars)))))
     ;;else read the files and call again
     (with-open-file (fp namefile :direction :input)
       (setq classes (loop for i from 1 to (read fp) collect (read fp)))
       (setq vars (loop for i from 1 to (read fp) collect (read fp)))
       (interptree tree namefile classes vars))))
```

What about Generalization?

- The universe of decision trees is large
- Smaller decision trees are thought to generalize better
  - Occam's Razor - simpler or more compact description is better.

What about Generalization?

- The universe of decision trees is large
- Smaller decision trees are *thought* to generalize better
  - –Occam's Razor - simpler or more compact description is better.
  - –therefore, we coded an inductive bias to find smaller decision trees into the design of the greedy algorithm.

Discussion

- Non Binary Data
  - –Continuous Values
    - –use average as threshold?
  - –more than binary features
    - –ternary features – use case statement
    - –many features – heuristics for subsets

Discussion

- Relation to Neural Networks
  - back propagation can be used for classification
  - convert features to floats
  - use 1-of-n outputs for classes
  - much slower