Midterm Exam

- Thursday, October 24, 7:15 – 9:15 p.m.
  Last name A - Lin in room B130 Van Vleck
  Last name Liou - Z in room 3650 Humanities
- Covers topics through Decision Trees, Random Forests, and *k*-Nearest-Neighbors
- Closed book except 8.5″ x 11″ sheet with notes on both sides (typed or handwritten)
- Bring student ID number, pencil, eraser, calculator (not on a phone)

1

- Covers lecture notes, readings in textbook (except Chapters 1 and 2), and 1 paper (intro to machine learning)
- True/False and multiple choice questions

2

Uninformed Search Methods

- Problem solving as search, problem representation in terms of states, goal test, operators, state-space graph search formulation, closed world assumption, Frontier and Explored lists, expanding a node, partial solution path, solution path, search tree
- Properties and computations
  - completeness, optimality, time and space complexity
- Methods
  - breadth-first search
  - depth-first search
  - uniform-cost search
  - iterative-deepening search
  - bidirectional search

3

Informed Search Methods

- Heuristic functions, evaluation function, admissible heuristic ($h \leq h^*$), consistent heuristic ($h(n) \leq c(n, n') + h(n')$), better informed heuristic, devising heuristics, completeness, admissibility, optimality
- Methods
  - best-first search
  - greedy best-first search
  - beam search
  - algorithm A
  - algorithm A*
  - IDA*

4

Local Search Methods

- Local search problem formulation, operators, neighborhood (aka move set), local optima problem
- Methods:
  - hill-climbing algorithm
  - hill-climbing with random restarts,
  - Simulated annealing (stochastic hill-climbing)
    escaping local optima, Boltzman's equation ($p = e^{\Delta E / T}$), temperature, cooling schedule

5

Game Playing

- Zero-sum games, perfect information games, deterministic vs. stochastic games, game playing as search, search tree, branching factor, ply, static evaluation function, horizon effect
- Methods
  - Minimax algorithm
    - Minimax principle, optimal playing strategy
  - Alpha-beta pruning algorithm
    - Cutoff tests: If $v \leq \alpha$ for *some* MAX node ancestor, **don't visit** any more of the current MIN node's children. If $v \geq \beta$ for *some* MIN node ancestor, **don't visit** any more of the current MAX node's children
  - iterative-deepening search with alpha-beta pruning
  - non-deterministic games
    - chance nodes, expectiminimax value
  - Monte Carlo tree search
    - Pure MCTS, Selection, expansion, simulation, backpropagation, playout, exploitation, exploration

7

MCTS Algorithm

Recursively build search tree, where each iteration consists of:

1. Selection: Starting at root, successively select best child nodes using scoring method until leaf node *L* reached
2. Expansion: Create and add best (or random) new child node, *C*, of *L*
3. Simulation: Perform a (random) playout from *C*
4. Backpropagation: Update score at *C* and all of *C*'s ancestors in search tree based on playout results

8

Unsupervised Learning

- Inductive learning problem, feature space, feature, attribute, examples (aka instances), labels, classes, training set, tuning set, testing set, classification problems, decision boundaries
- Methods
  - Hierarchical Agglomerative Clustering (HAC)
    - single linkage, complete linkage, average linkage, Euclidean distance, Manhattan distance, dendrogram
  - *k*-Means Clustering
    - cluster center (centroid), distortion cluster quality measure: For all clusters, sum of squared distances from each point to its cluster center
  - Nothing on mean-shift clustering

9

Hierarchical Agglomerative Clustering Algorithm

Input: a training sample $\{\mathbf{x}_i\}_{i=1}^{n}$; a distance function $d()$.
1. Initially, place each instance in its own cluster (called a singleton cluster).
2. *while* (number of clusters > 1) *do*:
3. Find the closest cluster pair A, B, i.e., they minimize $d(A, B)$.
4. Merge A, B to form a new cluster.

Output: a binary tree showing how clusters are gradually merged from singletons to a root cluster, which contains the whole training sample.

10

K-Means Clustering Algorithm

- Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n, k$ where each $\mathbf{x}_i$ is a point in a d-dimensional feature space
- **Step 1**: Select k cluster centers, $\mathbf{c}_1, \ldots, \mathbf{c}_k$
- **Step 2**: For each point $\mathbf{x}_i$, determine its cluster: Find the closest center (using, say, Euclidean distance)
- **Step 3**: Update all cluster centers as the centroids

$$\mathbf{c}_i = \frac{1}{num\_pts\_in\_cluster\_i} \sum_{\mathbf{x} \in \text{cluster } i} \mathbf{x}$$

- Repeat steps 2 and 3 until cluster centers no longer change

11

Supervised Learning

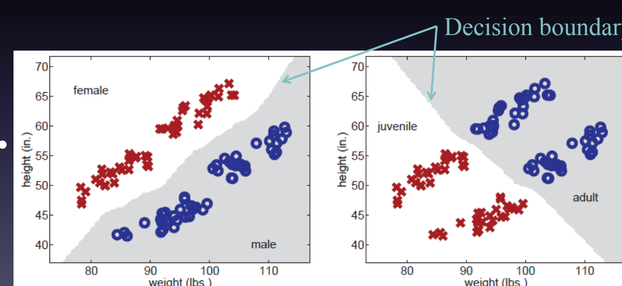
- Inductive bias, preference bias, decision boundaries, training set, tuning set, testing set, overfitting, noisy data, setting parameters
- Methods
  - *K*-nearest neighbor (k-NN)
  - Decision Trees
    - Ockham's razor, entropy, conditional entropy (aka remainder), information gain, categorical attributes, real-valued attributes, rule extraction, methods to avoid overfitting, pruning algorithm
  - Random Forests
    - Ensemble methods, bagging, bootstrap sampling, randomized node optimization, majority/mode classifier
  - Performance evaluation
    - Training set accuracy, training set error, testing set accuracy, testing set error, *k*-fold cross validation, leave-one-out cross validation

12

k-Nearest-Neighbors (k-NN)

Input: Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$; *distance function* $d()$; *number of neighbors* k; *test instance* $\mathbf{x}^*$

1. Find the k training instances $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ closest to $\mathbf{x}^*$ under distance $d()$.
2. Output y^* as the majority class of $y_{i_1}, \ldots, y_{i_k}$. Break ties randomly.



(a) classification by gender (b) classification by age

13

- Entropy: $$H(Y) = \sum_{i=1}^{k} -p_i \log_2 p_i$$
- Conditional Entropy:

$$H(Y \mid X = v) = \sum_{i=1}^{k} -\Pr(Y = y_i \mid X = v) \log_2 \Pr(Y = y_i \mid X = v)$$

$$H(Y \mid X) = \sum_{v:\text{values of } X} \Pr(X = v) H(Y \mid X = v)$$

- Information Gain:

$$I(Y;X) = H(Y) - H(Y \mid X)$$

14

Decision-Tree-Learning Algorithm

buildtree(*examples*, *attributes*, *default-label*)
if empty(*examples*) then return *default-label*
if (*examples* all have same label *y*) then return *y*
if empty(*attributes*) then return majority-class of *examples*
q = best_attribute(*examples*, *attributes*)
tree = create-node with attribute *q*
foreach value *v* of attribute *q* do
  v-ex = subset of examples with *q* == *v*
  subtree = **buildtree**(*v-ex*, *attributes* - {*q*}, *majority-class*(*examples*))
  add arc from *tree* to *subtree*
return *tree*

15

Pruning using a Greedy Algorithm

Prune(tree T, TUNE set)

1. Compute T's accuracy on TUNE; call it Acc(T)
2. For every internal node N in T:
   a) New tree T_N = copy of T, but prune (delete) the subtree under N
   b) N becomes a leaf node in T_N. The class is the majority vote of TRAIN examples reaching N
   c) Acc(T_N) = T_N's accuracy on TUNE
3. Let T* be the tree (among the T_N's and T) with the largest Acc()
   Set T = T* /* prune */
4. If no improvement then Return T else Goto Step 1

16

Random Forests

For each tree,

1. Build a training set by choosing *n* times with replacement from all *N* available training examples (aka "taking a bootstrap sample")
2. At each node of decision tree during construction, choose a *random subset* of *m* ***attributes*** from the total number, *M*, of possible attributes (*m* << *M*)
3. Select the best attribute at node using Max-Gain

17

K-Fold Cross Validation

1. Divide all examples into K disjoint subsets $E = E_1, E_2, \ldots, E_K$
2. For each $i = 1, \ldots, K$
   - let TEST set = E_i and TRAIN set = $E - E_i$
   - build decision tree using TRAIN set
   - determine accuracy Acc_i using TEST set
3. Compute ***K*-fold cross-validation** estimate of performance = mean accuracy = $(Acc_1 + Acc_2 + \ldots + Acc_K)/K$

18

Leave-One-Out Cross Validation

For $i = 1$ to N do // N = number of examples

1. Let (x_i, y_i) be the i^{th} example
2. Remove (x_i, y_i) from the dataset
3. Train on the remaining N-1 examples
4. Compute accuracy on i^{th} example

- Accuracy = mean accuracy on all N runs

19