

Some Ideas From Machine Learning

Nicholas Roy
16.410/13
Session 24

Slides adapted from:
Andrew Moore &
Tom Mitchell, CMU

Machine Learning

- Learning = improving with experience
- Improve over task T (e.g, Classification, control tasks)
- with respect to performance measure P (e.g., accuracy, speed, etc.)
- based on experience E (direct, indirect, teacher-provided, from exploration).
- What can we learn?
 - Discrete classifications (a.k.a. discriminative classifiers)
 - Probability of classifications (a.k.a. generative classifiers)
 - Continuous Functions (a.k.a. regression)
 - Control policies
 - Model parameters
- You have already seen some of these ideas in Baum-Welch learning of Hidden Markov Models and Q-Learning in Reinforcement Learning

Notation

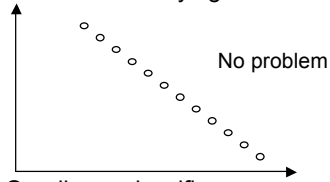
- Instances: x_1, x_2, \dots
- Target concept C labels instance x with label $c(x)$
- Labelled data D is a set of pairs $\langle x, c(x) \rangle$
- Hypothesis h is a possible target concept that also labels (correctly or incorrectly) each instance x with a label $h(x)$

The inductive learning hypothesis

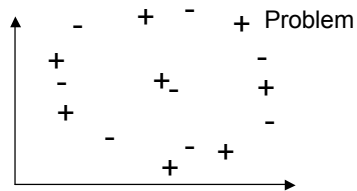
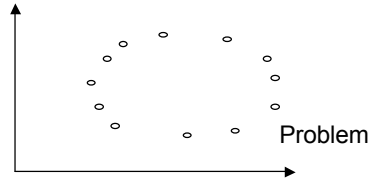
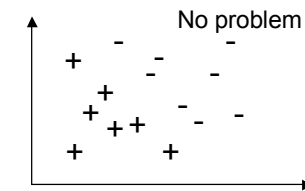
- Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.
- A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D .
 - $\text{Consistent}(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) h(x) = c(x)$
- The **version space**, $VS_{H,D}$, with respect to hypothesis space H and training examples D , is the subset of hypotheses from H consistent with all training examples in D .
 - $VS_{H,D} \equiv \{h \in H \mid \text{Consistent}(h, D)\}$

The Assumption of the Inductive Hypothesis

- We assume the target concept is in the hypothesis space
- Assume we're trying to learn the parameters of a line:



Or a linear classifier



What can be done about this problem? Not much.
Just be careful when you set out to use machine learning.
Like everything in life, it'll usually work as well as you want
on your training data, and as poorly as you'd expect on the test cases.

List-Then-Eliminate Algorithm

- *VersionSpace* \leftarrow a list containing every hypothesis in H
- For each training example, $\langle x, c(x) \rangle$
 - remove from *VersionSpace* any hypothesis h for which $h(x) \neq c(x)$
 - Output the list of hypotheses in $\$VersionSpace\$$

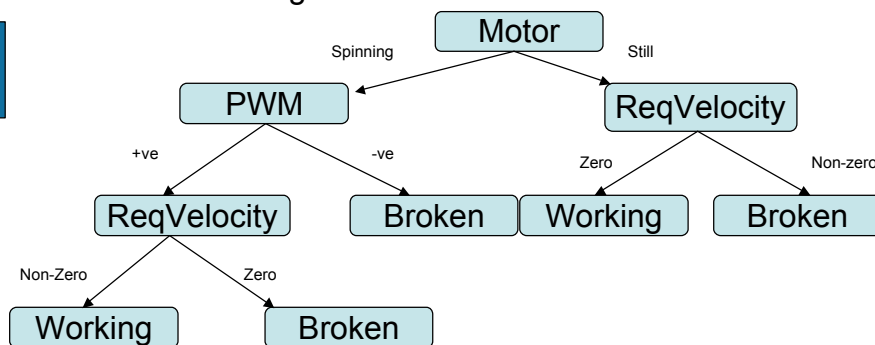
What's wrong with this algorithm?

Decision Trees

- Assumptions:
 - Instances describable by attribute-value pairs
 - Target function is discrete valued
 - Disjunctive hypothesis may be required
 - Possibly noisy training data
- Structure:
 - Each internal node tests an attribute
 - Each branch corresponds to attribute value
 - Each leaf node assigns a classification

Decision Trees

- Consider the following instance
 $x = (\text{Motor} = \text{Spinning}, \text{Pwm} = +, \text{AngularVel} = +, \text{ReqVelocity} = +)$
- And the following decision tree:



- What should label $c(x)$ should be given to x ?

Top-Down Induction of Decision Trees

- $A \leftarrow$ the "best" decision attribute for next node n
- Assign A as decision attribute for node n
- \forall value a of A , create new descendant of node n
- Assign training examples to each descendent
- If training examples are perfectly classified, then stop, else iterate over new leaf nodes

ID3 (Quinlan 1986)

- "Best" decision attribute maximizes information gain
- S is a sample of data taken from \mathcal{D}
- $Entropy(S)$ = expected number of bits needed to encode the label $c(x)$ of randomly drawn members of s (under the optimal, shortest-length code)
- Information theory (Shannon 1951): optimal length code assigns $-\log_2 p$ bits to message having probability p . Expected number of bits to encode $c(x)$ of random member x of S :

$$H(x) \equiv E[\text{number of bits}]$$
$$E[-\log_2 p(c(x))]$$
$$-\sum_{c_i} p(c_i(x)) \log_2 p(c_i(x))$$

ID3 (Quinlan 1986)

- Information theory (Shannon 1951): information gain is the expected reduction in entropy that results from partitioning data (e.g., using attribute A).

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- ID3 starts with a hypothesis space that includes all possible target concepts mapping discrete attributes to a discrete target
 - Outputs a single hypothesis
 - Statistically-based search choices
 - Robust to noisy data...
 - Inductive bias: approx "prefer shortest tree"

Example Data

Instance	MotorSpinning	PWM +ve	Encoder +ve	AngularVel +ve	Motor Working
A	F	F	F	F	F
B	F	F	T	F	F
C	T	T	F	T	F
D	T	F	F	T	T
E	F	T	T	F	T
F	F	F	T	T	T
G	F	F	F	T	T
H	T	T	F	F	T

ID3

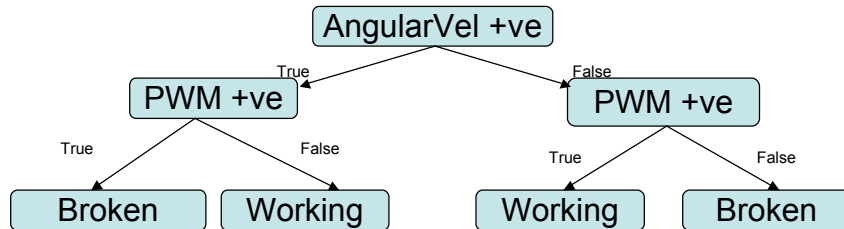
- Initial entropy:
 $-(3/8)\log_2(3/8)-(5/8)\log_2(5/8)=.954434$
- Split on Motor:
 $.954434-(5/8)(.97095)-(3/8)(.918296)=.003229$

		# False	# True	Entropy	Gain
Initial Entropy		3/8	5/8	.954434	
Split					
Motor	5/8	2/5	3/5	0.97095	0.003229
	3/8	1/3	2/3	0.918296	
PWM	5/8	2/5	3/5	0.97095	0.003229
	3/8	1/3	2/3	0.918296	
Encoder	5/8	2/5	3/5	0.97095	.003229
	3/8	1/3	2/3	0.918296	
Angular Velocity	4/8	2/4	2/4	1	0.048795
	4/8	1/4	3/4	0.811278	

ID3

AngularVel = F (4 instances) : Entropy = 1					
Motor	3/4	2/3	1/3	0.918296	0.311278
	1/4	1	0	0	
PWM	2/4	1	0	0	1
	2/4	0	1	0	
Encoder	2/4	1/2	1/2	1	0
	2/4	1/2	1/2	1	
AngularVel = T (4 instances) : Entropy is 0.811278					
Motor	2/4	1/2	1/2	1	0.311278
	2/4	1	0	0	
PWM	3/4	1	0	0	0.811278
	1/4	0	1	0	
Encoder	3/4	1/3	2/3	0.918296	0.122556
	1/4	0	1	0	

Resulting Tree:



- At this point, all training examples are classified correctly.
- We can also extract rules from the tree:
 - Each path from root to leaf constitutes a single rule.
 - If AngularVel is +ve, and PWM is +ve, Motor is not working.
 - If AngularVel is -ve, and PWM is +ve, Motor is working.

Overfitting

- Consider error of hypothesis h over
 - training data: $error_{train}(h)$
 - entire distribution D of data: $error_D(h)$
- Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that
$$error_{train}(h) < error_{train}(h')$$
- and
$$error_D(h) > error_D(h')$$
- How can we avoid overfitting?
 - stop growing when data split not statistically significant
 - grow full tree, then post-prune
- How to select "best" tree:
 - Measure performance over training data
 - Measure performance over separate validation data set
 - MDL: minimize $size(tree) + size(misclassifications(tree))$

Reduced-Error Pruning

- Split data into *training* and *validation* set
- Do until further pruning is harmful:
 - Evaluate impact on *validation* set of pruning each possible node (plus those below it)
 - Greedily remove the one that most improves *validation* set accuracy
- This produces smallest version of most accurate subtree
- What if data is limited?

- Rule Post-Pruning
 - Convert tree to equivalent set of rules
 - Prune each rule independently of others
 - Sort final rules into desired sequence for use
 - Perhaps most frequently used method (e.g., C4.5)

Some Issues in Machine Learning

- What algorithms can approximate functions well (and when)?
- How does number of training examples influence accuracy?
- How does complexity of hypothesis representation impact it?
- How does noisy data influence accuracy?
- What are the theoretical limits of learnability?
- How can prior knowledge of learner help?
- What clues can we get from biological learning systems?
- How can systems alter their own representations?