Lecture 18: Learning -2

Victor Lesser

CMPSCI 683
Fall 2004

Announcement

- Homework due Tuesday at 5 on November 29

Today's Lecture

- Continuation of Decision-tree Algorithms
- The Version Space Algorithm
- Neural Networks

Hypothesis Space Search in Decision Tree

- Complete space of finite discrete-valued functions relative to available attributes
- Maintains only a single current hypothesis (decision tree)
- Performs no backtracking in its search
- Uses all training examples at each step in the search to make statistically-based decisions regarding how to refine current hypothesis

Inductive Bias in Decision Tree Construction

- **Selects in favor of shorter trees over longer ones**
- **Selects trees that place the attributes with highest information gain closest to the root**

Overfitting in Decision Trees

A hypothesis *overfits* the training examples if there is some other hypothesis that fits the training examples less well, yet actually performs better over the entire distribution of instances

- **Causes**
  - Noisy Data — construct tree to explain noisy data
  - Lack of Examples — small number of examples associated with leaf
    - Coincidental irregularities cause the construction of more detail tree than warranted

Avoiding Overfitting

- **Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data**
- **Post-prune the tree**
  - Use non-training instances to evaluate based on a statistical test to estimate whether pruning a particular node is likely to produce an improvement beyond the training set

Broadening the applicability - Missing Data

- Add new attribute value - “unknown”
- Estimate missing value based on other examples for which this attribute has a known value
  - **Assign value that is most common among training examples at parent node**
- Instantiated example with all possible values of missing attribute but assign weights to each instance based on likelihood of missing value being a particular value given the distribution of examples in the parent node
  - **Modify decision tree algorithm to take into account weighting**

Broadening the applicability - Multi-valued Attributes

- Handling multivalued (large) attributes and classification
  - **Need another measure of information gain**
  - **Information gain measure gives inappropriate indication of attributed usefulness because of likelihood of singleton values**
  - **Gain ratio**
    - **Gain over intrinsic information content**

Broadening the Applicability - Continuous-Valued attributes

- Continuous-valued attributes
  - **Discretize**
    - **Example $, $$, $$$**
  - **Preprocess to find out which ranges give the most useful information for classification purposes**

Preprocessing for Continuous-Valued Attributes

- **Sort instances based on value of an attribute (e.g. temperature)**
- **Identify adjacent examples that differ in their target classification**
- **Generate a set of candidate thresholds midway between corresponding examples**
- **Use information gain to decide appropriate threshold**

Decision Tree as a Logical Sentence

WillWait(r) ⇔

Patron(r,Some) ∨
(Patron(r,Full) ∧ ¬Hungry(r) ∧ Type(r,French)) ∨
(Patron(r,Full) ∧ ¬Hungry(r) ∧ Type(r,Thai) ∧ Fri/Sat(r)) ∨
(Patron(r,Full) ∧ ¬Hungry(r) ∧ Type(r,Burger))

Each example is a logical sentence:

Alt(r) ∧ ¬Bar(r) ∧ ¬Fri(r) ∧ ... ⇒ WillWait(r)

A decision tree is consistent with the data iff the corresponding KB is consistent.

Inductive Learning : Incremental Learning of Logical Expressions

- **Can use Simpler Approach than Decision Tree Algorithm**
  - Assuming Complete Consistency
- **Incrementally present examples**
- **Incrementally refine hypothesis**

Current-best Hypothesis in Search

- **Maintain single hypothesis**
- **Adjust to new example in order to maintain consistency**
  - **An example can be consistent with the current hypothesis, or it can be:**

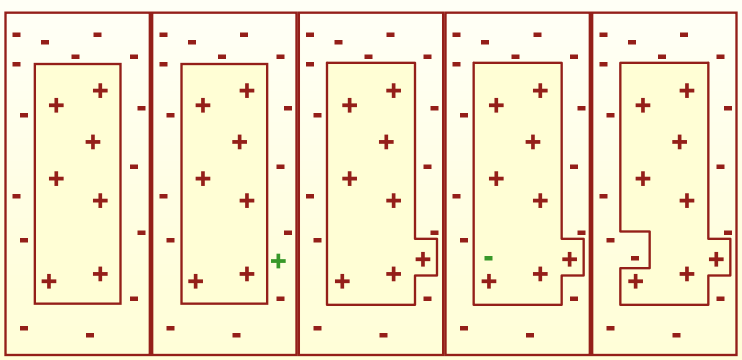
    false negative **if the hypothesis says it is negative but in fact it is positive, or**

    false positive **if the hypothesis says it is positive but in fact it is negative.**
- **Generalization/specialization**
  - Dropping conditions
  - Adding conditions

Current-best-hypothesis search

Add Unknown Example (positive + or negative -) and adjust Current Hypothesis

Monotonic View of Evolution of Current Best Hypothesis, never modify to eliminate any example



(a) (b) (c) (d) (e)

Current-best-hypothesis cont.

function CURRENT-BEST-LEARNING(*examples*) **returns** a hypothesis

$H \leftarrow$ any hypothesis consistent with the first example in *examples*
for each remaining example in *examples* **do**
  if e is false positive for H **then**
    $H \leftarrow$ **choose** a specialization of H consistent with *examples*
  else if e is false negative for H **then**
    $H \leftarrow$ **choose** a generalization of H consistent with *examples*
  if no consistent specialization/generalization can be found **then fail**
end
return H

Inducing Decision Trees from Examples

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	Yes
X_2	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	No
X_3	No	Yes	No	No	Some	$	No	No	Burger	0–10	Yes
X_4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10–30	Yes
X_5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	Yes
X_7	No	Yes	No	No	None	$	Yes	No	Burger	0–10	No
X_8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	Yes
X_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	No
X_{11}	No	No	No	No	None	$	No	No	Thai	0–10	No
X_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	Yes

Restaurant Dining Example

- X1 is positive. Alternate(X1) is true
  - **H1: initial hypothesis Vx WillWait (x) = Alternate(x)**
- X2 is negative, false positive
  - **H2: WillWait (x) = Alternate(x) and Patrons(x,Some)**
- X3 is positive, false negative
  - **H3:WillWait (x) = Patrons(x,Some)**
- X4 is positive, false negative
  - **H4:WillWait (x) = Patrons(x,Some) v (Patrons(x,Full) and Fri/Sat(x))**
- Other Hypotheses
  - **H4':WillWait (x) = Patrons(x,Some) v (Patrons(x,Full) and WaitEstimate(x,10-30))**
  - **H4'':WillWait (x) = not WaitEstimate(x,30-60))**

Problems with Current-best Hypothesis

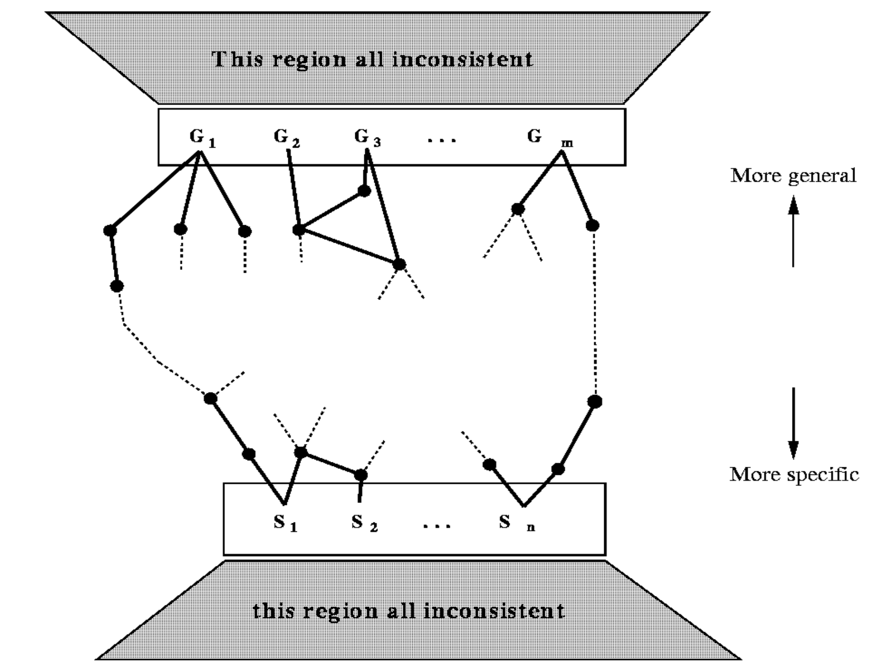
- **Very large search space**
  - No good heuristics
  - Non-deterministic search
  - May need to backtrack
- **Updating/checking hypothesis is expensive in terms of number of examples**
  - Need to re-evaluate every modified hypothesis on all examples presented

The Version-Space Strategy

- A *least commitment* approach — keep all the hypotheses that are consistent with all the examples so far.
  - **No backtracking**
- Problem: how to represent the current set of remaining hypotheses (the version space) efficiently ?

Using boundary sets:

- S-set = most specific (consistent) hypotheses
  - **every member of S is consistent with all observations so far and there are no consistent hypotheses that are more specific**
- G-set = most general (consistent) hypotheses
  - **every member of G is consistent with all observations so far and there are no consistent hypotheses that are more general**



The Version-Space Algorithm

Initialize the sets *S* and *G* to the sets of maximally specific and maximally general hypothesis that are consistent with the first observed positive training instance.

- G -- set of hypotheses that represent disjunction of each single attribute/value pair
- S -- the hypothesis which is the conjunction of the attribute/value pairs in the training instance

For each subsequent instance, *i*, do

The Version-Space Algorithm cont.

If *i* is negative

- Remove from *S* the hypotheses which match *i*
  - **False positive for Sj, too general**
- Make hypotheses in *G* that match *i* more specific, only to the extent required so that they no longer match *i*
  - **False positive for Gk, too general**
- Remove from *G* any element that is no longer more general than some member of *S*
- Remove from *G* any element that is more specific than some other member in *G*

The Version-Space Algorithm cont.

If *i* is positive

- Remove from *G* the hypotheses which do not match *i*
  - **False negative for G_k, too specific**
- Make hypotheses in *S* that do not match *i* more general, only to the extent required so that they match *i*
  - **False negative for Sj, too specific, replace by immediate generalizations**
- Remove from *S* any element that is no longer more specific than some member of *G*
- Remove from *S* any element that is more general than some other member in *S*

Version-Space Algorithm cont.

Termination:

- One hypothesis is left in the version space indicating that it is the desired concept definition.
- The version space collapses with either *S* or *G* becoming empty indicating that there are no consistent hypothesis for the given training set.
- The algorithm runs out of examples with more than one hypothesis left — can use the result for classification (if all agree fine, otherwise can use majority vote).

Positive and Negative Examples of the Concept "Japanese economy car"

origin: *Japan*
mfr: Honda
color: blue
decade: 1980
type: *Economy*
(+)

origin: Japan
mfr: Toyota
color: green
decade: 1970
type: Sports
(-)

origin: Japan
mfr: Toyota
color: blue
decade: 1990
type: Economy
(+)

origin: USA
mfr: Chrysler
color: blue
decade: 1980
type: *Economy*
(-)

origin: Japan
mfr: Honda
color: white
decade: 1980
type: Economy
(+)

A Version Space Example
(Rich/Knight 1991)

- E += {(*Japan, Honda, Blue*, 1980, *Economy*)}
  $G = \{(x_1, x_2, x_3, x_4, x_5)\}$
  S = {(*Japan, Honda, Blue*, 1980, *Economy*)}
- E- = {(*Japan, Toyota,Green*, 1970, *Sport*)}
  G = {(x_1, *Honda*, x_3, x_4, x_5), (x_1, x_2, *Blue*, x_4, x_5),
  (x_1, x_2, x_3, 1980, x_5), (x_1, x_2, x_3, x_4, *Economy*)}
- Note did not include {(Japan, x_2, x_3, x_4, x_5)}
  S = {(*Japan, Honda, Blue*, 1980, *Economy*)}
- E+ = {(*Japan, Toyota, Blue*, 1990, *Economy*)}
- *G* = {(x_1, x_2, *Blue*, x_4, x_5), (x_1, x_2, x_3, x_4, *Economy*)}
  S = {(*Japan*, x_2, *Blue*, x_4, *Economy*)}

Example Continued

- *G* = {(x_1, x_2, *Blue*, x_4, x_5), (x_1, x_2, x_3, x_4, *Economy*)}
- *S* = {(*Japan*, x_2, *Blue*, x_4, *Economy*)}
- E- = {(*USA, Chrysler, Blue*, 1980, *Economy*)}
- *G* = {(*Japan*, x_2, *Blue*, x_4, x_5), (*Japan*, x_2, x_3, x_4, *Economy*)}
- *S* = {(*Japan*, x_2, *Blue*, x_4, *Economy*)}
- E+ ={(*Japan, Honda, White*, 1980, *Economy*)}
- *G* = {(*Japan*, x_2, x_3, x_4, *Economy*)}
- *S* = {(*Japan*, x_2, x_3, x_4, *Economy*)}

Conclusions on Version Space

- **Elegant Algorithm**
- **Limited Applicability**
  - There are no errors in the training examples
    - Will remove correct hypothesis from set as soon as encounters false negative hypothesis
  - Does not handle unlimited disjunctions in hypothesis space
    - Extensions allow limited forms of disjunction
    - Generalization Hieararchy or more general predicates (that represent disjunction)

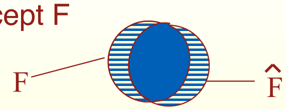
Why does Learning Work — Computational Learning Theory

- **How do we know the hypothesis *h* is close to the target function *f* if we don't know what *f* is?**
  - Sample Complexity -- Can we decide how many examples we need to train on
- **Underlying principle:**
  - An *h* that is seriously wrong will almost certainly be "found out" with high probability after a small number of examples
  - An *h* that is consistent with a large set of training examples is unlikely to be seriously wrong
- **Probably Approximately Correct (PAC) Learning:**
  - Stationary assumption: training and test data drawn randomly from same population of examples using same distribution

How Good is a Hypothesis?

Compare Hypothesis $\hat{F}$

To correct concept F



Probability of misclassifying an instance ≡

Probability of instance being in

$\hat{F} \oplus F$; $\hat{F}(x)$ not equal F(x)

Hypothesis is good to extent it classifies instances correctly

How Good is a Hypothesis? cont.

Hypothesis $\hat{F}$ approximately correct

IF

$$u \in \hat{F} \oplus F, P(u) \leq \in \qquad \text{(Valiant)}$$

Accuracy parameter

Hypothesis $\hat{F}$ Probably Approximately Correct (PAC)

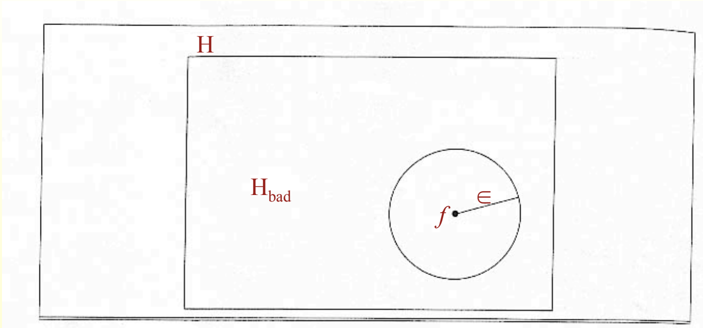
IF

$$P(u \in \hat{F} \oplus F, P(u) > \in) < \delta$$

Confidence parameter

Schematic Diagram of Hypothesis Space

- **Hypothesis space, showing the "∈-ball" around the true function f.**



What is the Probability of a Hypothesis Agreeing with all of M examples?

Assume worst case – All $h \in H = \{h_{bad} = H\}$ Have Error > ∈

Space of possible hypotheses

Upper bound is:

P(h_b agrees with M examples) $\leq (1-\in)^M$

For |H| hypotheses, probability of some $h \in H$ being consistent with all M examples

P(h_{bad} contains a consistent hypothesis with M examples) $\leq |H| \cdot (1-\in)^M$

What is the Probability of a Hypothesis Agreeing with all of M examples? cont.

To guarantee that $\hat{F}$ is PAC

$$|H|(1-\in)^m \leq \delta$$

Because $\in \delta, |H|$ are known, can solve for M (Blumer et al)

$$M \geq \frac{1}{\in}(LN\frac{1}{\delta} + LN|H|)$$ Given (1- ∈) <= ∈ $^{-\in}$

Any $h \in H$ consistent with M examples, $M \geq$ ---- , is PAC!!

By looking at H for various representations, can determine corresponding M_1 giving bound on sample complexity for PAC learning.

Decision Tress an PAC

- **Space of H is $2^{2\exp(n)}$, n attributes**
- **Sample complexity of space grows as 2^n**
- **Number of examples is at most 2^n**
- **Learning algorithm will no better than a lookup table in terms of PAC**
- **Problems occurs because of worst-case complexity analysis and size of H**
  - Do not necessarily reflect the average-case sample complexity
- **Can we reduce the size of H and still learn reasonable Boolean functions**

Decision Lists

- **Series of Tests, each with conjunction of literals**
  - Patrons(x,Some) ----> yes
  - Patrons(x,full) and Fri/Sat(x) ----> yes
  - Nil ---> no
- ***k*-DL, restrict size of test to *k* literals**
  - More expressive power than depth *k* decision tree
- **PAC-learn in a reasonable number of examples for small *k***

Biological Inspiration Learning: The Brain

- Approximately 10^{11} neurons, 10^4 synapses (connections) per neuron.
- Neuron “fires” when its inputs exceed a threshold.
- Inputs are weighted and can have excitory or inhibitory effect.
- Individual firing is slow (≈ .001 second) but bandwidth is very high (≈ 10^{14} bits/sec).
- The brain performs many tasks much faster than a computer (Scene recognition time ≈ .1 second).
- Learning and graceful degradation.

What is Connectionist Computation?

Computational architectures and cognitive models that are neurally-inspired:

- **Faithful to coarse neural constraints — not neural models**
- **Large numbers of simple (neuron-like) processing units interconnected through weighted links**
- **They do not compute by transmitting symbolically coded messages**
- **“program” resides in the structure of the interconnections**
- **“massive parallelism” and no centralized control**

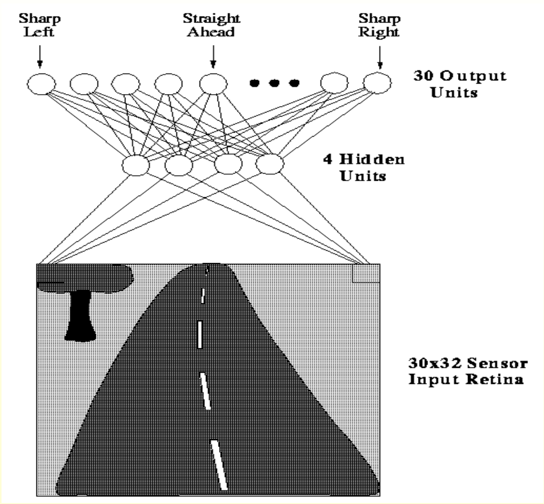
Some Properties of Connectionist Systems

- Ability to bring large numbers of interacting constraints to bear on problem solving (soft constraints)
- Noise resistance, error tolerance, graceful degradation
- Ability to do complex multi-layer recognition with a large number of inputs/outputs (quickly)
- Learning with generalization
- Biological plausibility
- Potential for speed of processing through fine-grained parallelism

Applications of neural networks

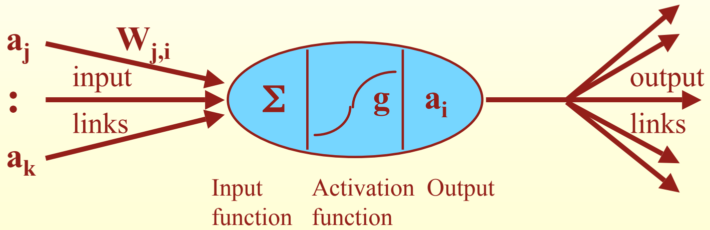
- Automobile automatic guidance systems
- Credit application evaluation, mortgage screening, real estate appraisal
- Object recognition (faces, characters)
- Speech recognition and voice synthesis
- Market forecasting, automatic bond trading
- Robot control, process control
- Breast cancer cell analysis
- Oil and gas exploration
- Image and data compression

ALVINN drives 70 mph on highways

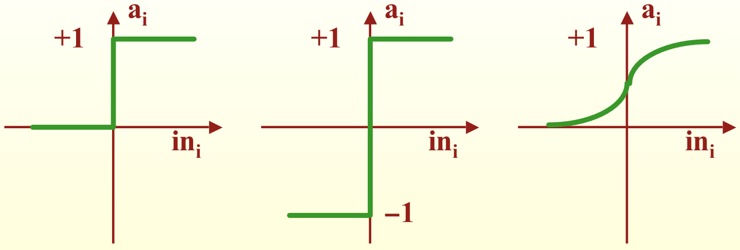


Artificial Neural Networks

Processing units compute weighted sum of their inputs, and then apply a threshold function.



Sample activation functions



(a) Step function (b) Sign function (c) Sigmoid function

Representation of Boolean Functions

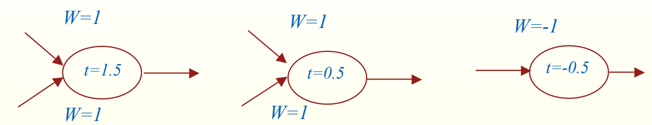
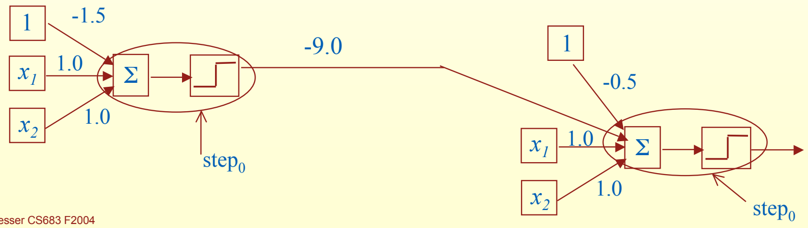


Figure 19.6 Units with a step function for the activation function can act as logic gates, given appropriate thresholds and weights.

> XOR requires multi-layer network



Next Lecture

- **Continuation of Neural Networks**