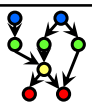


Lecture 33 of 42

Machine Learning: Version Spaces & Decision Trees Discussion: Inductive Learning Hypothesis

William H. Hsu
Department of Computing and Information Sciences, KSU

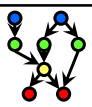
KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:

Chapter 18, Russell and Norvig

Lecture Outline

- **Learning Algorithms and Models**
  - Models: decision trees, winnow, artificial neural networks, naïve Bayes, genetic algorithms (GAs) and genetic programming (GP), instance-based learning (nearest-neighbor), inductive logic programming (ILP)
  - Algorithms: for decision trees (ID3/C4.5/J48), ANNs (backprop), etc.
  - Methodologies: supervised, unsupervised, reinforcement; knowledge-guided
- **Theory of Learning**
  - Computational learning theory (COLT): complexity, limitations of learning
  - Probably Approximately Correct (PAC) learning
  - Probabilistic, statistical, information theoretic results
- **Multistrategy Learning: Combining Techniques, Knowledge Sources**
- **Data: Time Series, Very Large Databases (VLDB), Text Corpora**
- **Applications**
  - Performance element: classification, decision support, planning, control
  - Database mining and knowledge discovery in databases (KDD)
  - Computer inference: learning to reason



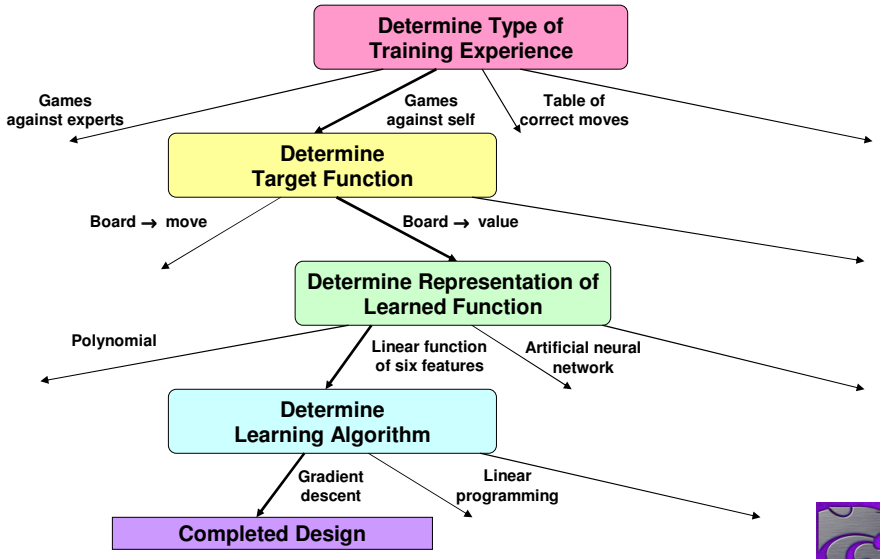
Why Machine Learning?

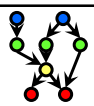
- **New Computational Capability**
  - Database mining: converting records into knowledge
  - Self-customizing programs: learning news filters, adaptive monitors
  - Learning to act: robot planning, control optimization, decision support
  - Applications that are hard to program: automated driving, speech recognition
- **Better Understanding of Human Learning and Teaching**
  - Cognitive science: theories of knowledge acquisition (e.g., through practice)
  - Performance elements: reasoning (inference) and *recommender* systems
- **Time is Right**
  - Recent progress in algorithms and theory
  - Rapidly growing volume of online data from various sources
  - Available computational power
  - Growth, interest in learning-based industries (e.g., data mining/KDD)

Rule and Decision Tree Learning

- **Example: Rule Acquisition from Historical Data**
- **Data**
  - Patient 103 (time = 1): Age 23, First-Pregnancy: no, Anemia: no, Diabetes: no, Previous-Premature-Birth: no, Ultrasound: *unknown*, Elective C-Section: *unknown*, Emergency-C-Section: *unknown*
  - Patient 103 (time = 2): Age 23, First-Pregnancy: no, Anemia: no, Diabetes: yes, Previous-Premature-Birth: no, Ultrasound: *unknown*, Elective C-Section: no, Emergency-C-Section: *unknown*
  - Patient 103 (time = n): Age 23, First-Pregnancy: no, Anemia: no, Diabetes: yes, Previous-Premature-Birth: no, Ultrasound: abnormal, Elective C-Section: no, Emergency-C-Section: YES
- **Learned Rule**
  - IF *no previous vaginal delivery*, AND *abnormal 2nd trimester ultrasound*, AND *malpresentation at admission*, AND *no elective C-Section*
    THEN *probability of emergency C-Section is 0.6*
  - Training set: 26/41 = 0.634
  - Test set: 12/20 = 0.600

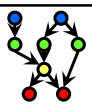
Learning to Play Checkers [3]: Design Choices





What to Learn?

- **Classification Functions**
  - Learning hidden functions: estimating ("fitting") parameters
  - Concept learning (e.g., chair, face, game)
  - Diagnosis, prognosis: risk assessment, medical monitoring, security, ERP
- **Models**
  - Map (for navigation)
  - Distribution (query answering, *aka* QA)
  - Language model (e.g., automaton/grammar)
- **Skills**
  - Playing games
  - Planning
  - Reasoning (acquiring representation to use in reasoning)
- **Cluster Definitions for Pattern Recognition**
  - Shapes of objects
  - Functional or taxonomic definition
- ***Many Problems Can Be Reduced to Classification***



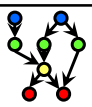
How to Learn It?

- **Supervised**
  - **What is learned? Classification function; other models**
  - **Inputs and outputs? Learning:** examples $\langle x, f(x) \rangle \rightarrow$ approximation $\hat{f}(x)$
  - **How is it learned? Presentation of examples to learner (by teacher)**
- **Unsupervised**
  - **Cluster definition, or *vector quantization* function (*codebook*)**
  - **Learning:** observations $x \times$ distance metric $d(x_1, x_2) \rightarrow$ discrete codebook $f(x)$
  - **Formation, segmentation, labeling of clusters based on observations, metric**
- **Reinforcement**
  - **Control policy (function from states of the world to actions)**
  - **Learning:** state/reward sequence $\{\langle s_i, r_i \rangle : 1 \leq i \leq n\} \rightarrow$ policy $p : s \rightarrow a$
  - **(Delayed) feedback of reward values to agent based on actions**
  - **Model updated based on reward, (partially) observable state**

Learning and Types [1]: Generic Supervised Learning Problem

Example	x_1	x_2	x_3	x_4	y
0	0	1	1	0	0
1	0	0	0	0	0
2	0	0	1	1	1
3	1	0	0	1	1
4	0	1	1	0	0
5	1	1	0	0	0
6	0	1	0	1	0

- **Input x_i: t_i, desired output y: t, "target" function f: $(t_1 \times t_2 \times t_3 \times t_4) \rightarrow t$**
- **Learning function: Vector $(t_1 \times t_2 \times t_3 \times t_4 \times t) \rightarrow (t_1 \times t_2 \times t_3 \times t_4) \rightarrow t$**



Learning and Types [2]: Unrestricted Hypothesis Space

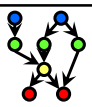
- $|A \rightarrow B| = |B|^{|A|}$ – proof?
- $|H^4 \rightarrow H| = |\{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\} \rightarrow \{0,1\}| = 2^{2^4} = 65536$ functions
- Complete Ignorance: Is Learning Possible?
  - Need to see every possible input/output pair
  - After 7 examples, still have $2^9 = 512$ possibilities (out of 65536) for *f*

Example	x_1	x_2	x_3	x_4	y
0	0	0	0	0	?
1	0	0	0	1	?
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	?
8	1	0	0	0	?
9	1	0	0	1	1
10	1	0	1	0	?
11	1	0	1	1	?
12	1	1	0	0	0
13	1	1	0	1	?
14	1	1	1	0	?
15	1	1	1	1	?

Training Examples for Concept *EnjoySport*

- Specification for Examples
  - Similar to a data type definition
  - 6 attributes: Sky, Temp, Humidity, Wind, Water, Forecast
  - Nominal-valued (symbolic) attributes - enumerative data type
- Binary (Boolean-Valued or H -Valued) Concept
- Supervised Learning Problem: *Describe the General Concept*

Example	Sky	Air Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
0	Sunny	Warm	Normal	Strong	Warm	Same	Yes
1	Sunny	Warm	High	Strong	Warm	Same	Yes
2	Rainy	Cold	High	Strong	Warm	Change	No
3	Sunny	Warm	High	Strong	Cool	Change	Yes



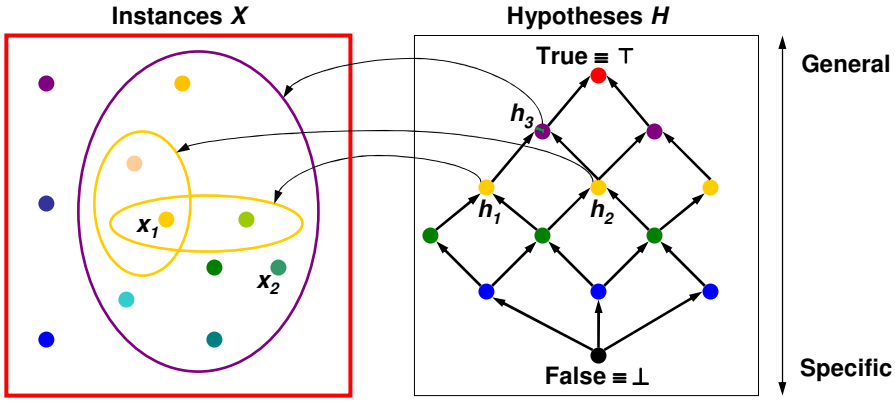
Representing Hypotheses

- Many Possible Representations
- Hypothesis *h*: Conjunction of Constraints on Attributes
- Constraint Values
  - Specific value (e.g., *Water = Warm*)
  - Don't care (e.g., *"Water = ?"*)
  - No value allowed (e.g., "Water = Ø")
- Example Hypothesis for *EnjoySport*

Sky	AirTemp	Humidity	Wind	Water	Forecast
<Sunny	*?*	*?*	*Strong*	*?*	*Same>*

  - Is this consistent with the training examples?
  - What are some hypotheses that are consistent with the examples?

Instances, Hypotheses, and the Partial Ordering *Less-General-Than*

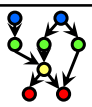


x_1 = <Sunny, Warm, High, Strong, Cool, Same>
x_2 = <Sunny, Warm, High, Light, Warm, Same>

h_1 = <Sunny, ?, ?, Strong, ?, ?>
h_2 = <Sunny, ?, ?, ?, Cool, ?>
h_3 = <Sunny, ?, ?, ?, ?, ?>

$\leq_p \equiv$ *Less-General-Than*
(corresponding set of instances: *Subset-Of*)

$h_1 \leq_p h_3$
$h_2 \leq_p h_3$



FIND-S ALGORITHM

1. Initialize h to the most specific hypothesis in *H*

   H: the *hypothesis space*

   (partially ordered set under relation *Less-Specific-Than)*

2. For each positive training instance *x*

   For each attribute constraint a_i in *h*

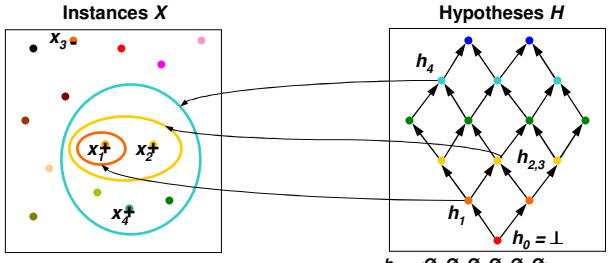
   IF constraint a_i in *h* is satisfied by *x*

   THEN do nothing

   ELSE replace a_i in *h* by next more general constraint satisfied by *x*

3. Output hypothesis *h*

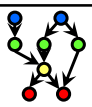
HYPOTHESIS SPACE SEARCH BY FIND-S



x_1 = <Sunny, Warm, Normal, Strong, Warm, Same>, +
x_2 = <Sunny, Warm, High, Strong, Warm, Same>, +
x_3 = <Rainy, Cold, High, Strong, Warm, Change>, -
x_4 = <Sunny, Warm, High, Strong, Cool, Change>, +

h_0 = <Ø, Ø, Ø, Ø, Ø, Ø>
h_1 = <Sunny, Warm, Normal, Strong, Warm, Same>
h_2 = <Sunny, Warm, ?, Strong, Warm, Same>
h_3 = <Sunny, Warm, ?, Strong, Warm, Same>
h_4 = <Sunny, Warm, ?, Strong, ?, ?>

- **Shortcomings of *Find-S***
  - **Can't tell whether it has learned concept**
  - **Can't tell when training data inconsistent**
  - **Picks a maximally specific *h* (why?)**
  - **Depending on *H*, there might be several!**



Version Spaces

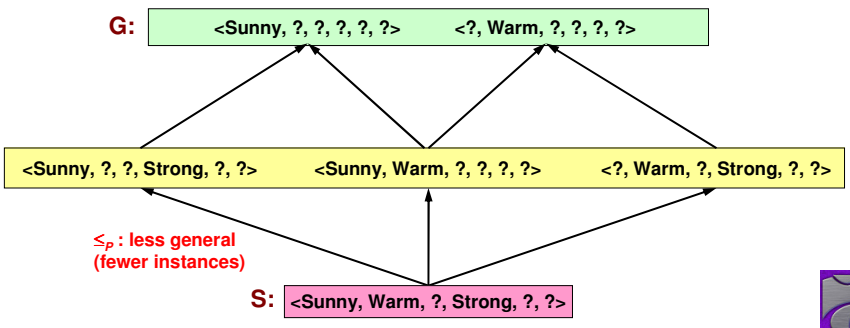
- **Definition: Consistent Hypotheses**
  - A hypothesis h is consistent with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D.
  - $Consistent(h, D) \equiv \forall \langle x, c(x) \rangle \in D \cdot h(x) = c(x)$
- **Given**
  - Hypothesis space H
  - Data set D: set of training examples
- **Definition**
  - Version space $VS_{H,D}$ with respect to H, D
  - Subset of hypotheses from H consistent with all training examples in D
  - $VS_{H,D} \equiv \{ h \in H \mid Consistent(h, D) \}$

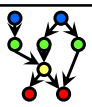
List-Then-Eliminate Algorithm

1. Initialization: *VersionSpace* ← list containing every hypothesis in H
2. For each training example $\langle x, c(x) \rangle$

   Remove from *VersionSpace* any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

Example Version Space





Hypothesis Spaces As Lattices

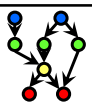
- **Meet Semilattice**
  - Every pair of hypotheses h_i and h_j has *greatest lower bound* (GLB) $h_i \vee h_j$
  - Is *H* meet semilattice?
  - $\perp \equiv$ some ø
- **Join Semilattice**
  - Every pair of hypotheses h_i and h_j has *least upper bound* (LUB) $h_i \wedge h_j$

  Is *H* join semilattice?
  - $\top \equiv$ all ?
- **(Full) Lattice**
  - Every pair of hypotheses has GLB $h_i \vee h_j$ and LUB $h_i \wedge h_j$
  - Both meet semilattice and join semilattice
  - Partial ordering Less-General-Than

Representing Version Spaces As Lattices

- **Definition: General (Upper) Boundary**
  - General boundary *G* of version space $VS_{H,D}$: set of most general members
  - Most general ≡ *maximal* elements of $VS_{H,D}$ ≡ "set of necessary conditions"
- **Definition: Specific (Lower) Boundary**
  - $\top$
  - Specific boundary S of version space $VS_{H,D}$: set of least general members
  - Most specific ≡ *minimal* elements of $VS_{H,D}$ ≡ "set of sufficient conditions"
- **Version Space**
  - $VS_{H,D}$ ≡ consistent poset (partially-ordered subset of *H*)
  - Every member of version space lies between *S* and *G*
  - $VS_{H,D} \equiv \{ h \in H \mid \exists\ s \in S, g \in G\ .\ s \leq_P h \leq_P g \}$, $\leq_P \equiv$ Less-General-Than

    "Version space is defined as set of hypotheses sandwiched between specific *s* and general *g* (given data)"



Candidate Elimination Algorithm [1]

1. Initialization

$G_0 \leftarrow \top \equiv$ most general hypothesis in H, denoted {<?, … , ?>}

$S_0 \leftarrow \bot \equiv$ least general hypotheses in H, denoted {<Ø, … , Ø>}

2. For each training example *d*

If *d* is a positive example (*Update-S*) // generalize

Remove from *G* any hypotheses inconsistent with *d*

For each hypothesis *s* in *S* that is not consistent with *d*

Remove *s* from *S* // "move *S* upwards"

Add to *S* all minimal generalizations *h* of *s* such that

1. *h* is consistent with *d*
2. Some member of *G* is more general than *h*

(These are least upper bounds, or *joins*, $s \wedge d$, in $VS_{H,D}$)

Remove from *S* any hypothesis that is more general than another hypothesis in *S* (remove any dominating elements)

Candidate Elimination Algorithm [2]

(continued)

If *d* is a negative example (*Update-G*) // specialize

Remove from *S* any hypotheses inconsistent with *d*

For each hypothesis *g* in *G* that is not consistent with *d*

Remove *g* from *G* // "move *G* downwards"

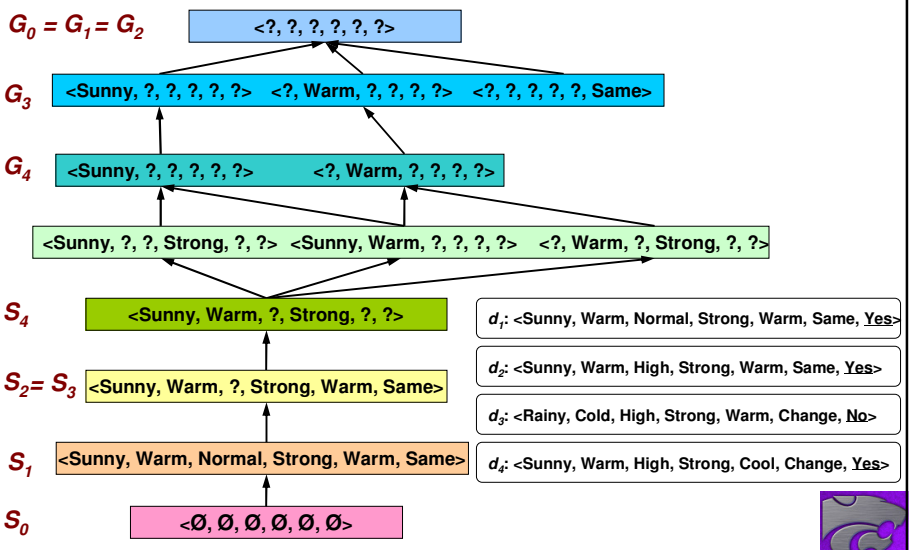
Add to *G* all minimal specializations *h* of *g* such that

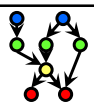
1. *h* is consistent with *d*
2. Some member of *S* is less general than *h*

(These are greatest lower bounds, or *meets*, $g \vee d$, in $VS_{H,D}$)

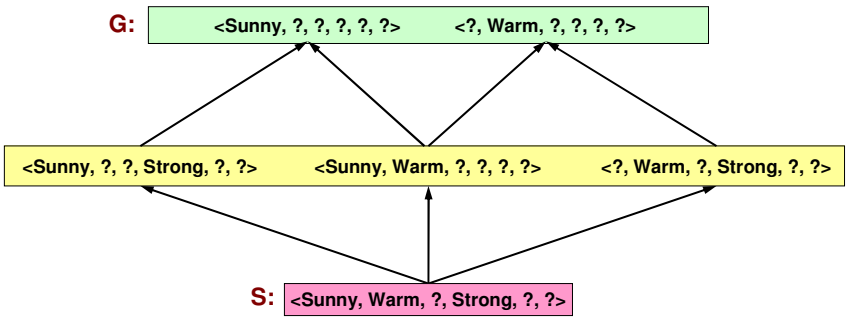
Remove from *G* any hypothesis that is less general than another hypothesis in *G* (remove any dominated elements)

Example Trace

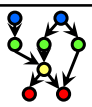




What Next Training Example?



- Active Learning: What Query Should The Learner Make Next?
- How Should These Be Classified?
  - <Sunny, Warm, Normal, Strong, Cool, Change>
  - <Rainy, Cold, Normal, Light, Warm, Same>
  - <Sunny, Warm, Normal, Light, Warm, Same>



What Justifies This Inductive Leap?

- **Example: Inductive Generalization**
  - Positive example: <Sunny, Warm, Normal, Strong, Cool, Change, Yes>
  - Positive example: <Sunny, Warm, Normal, Light, Warm, Same, Yes>
  - Induced *S*: <Sunny, Warm, Normal, ?, ?, ?>
- **Why Believe We Can Classify The Unseen?**
  - e.g., <Sunny, Warm, Normal, Strong, Warm, Same>
  - When is there enough information (in a new case) to make a prediction?

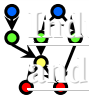
An Unbiased Learner

- **Inductive Bias**
  - Any preference for one hypothesis over another, *besides* consistency
  - Example: $H \equiv$ *conjunctive* concepts with don't cares
  - What concepts can H not express? (Hint: what are its syntactic limitations?)
- **Idea**
  - Choose unbiased H': expresses every teachable concept (i.e., power set of X)
  - Recall: $| A \rightarrow B | = | B |^{| A |}$ ($A = X$; $B = \{\text{labels}\}$; $H' = A \rightarrow B$)
  - {{Rainy, Sunny, Cloudy} × {Warm, Cold} × {Normal, High} × {None-Mild, Strong} × {Cool, Warm} × {Same, Change}} → {0, 1}
- **An Exhaustive Hypothesis Language**
  - Consider: H' = disjunctions (∨), conjunctions (∧), negations (¬) over H
  - $| H' | = 2^{(2 \cdot 2 \cdot 2 \cdot 3 \cdot 2 \cdot 2)} = 2^{96}$; $| H | = 1 + (3 \cdot 3 \cdot 3 \cdot 4 \cdot 3 \cdot 3) = 973$
- **What Are S, G For The Hypothesis Language H'?**
  - S ← *disjunction of all positive examples*
  - G ← *conjunction of all negated negative examples*

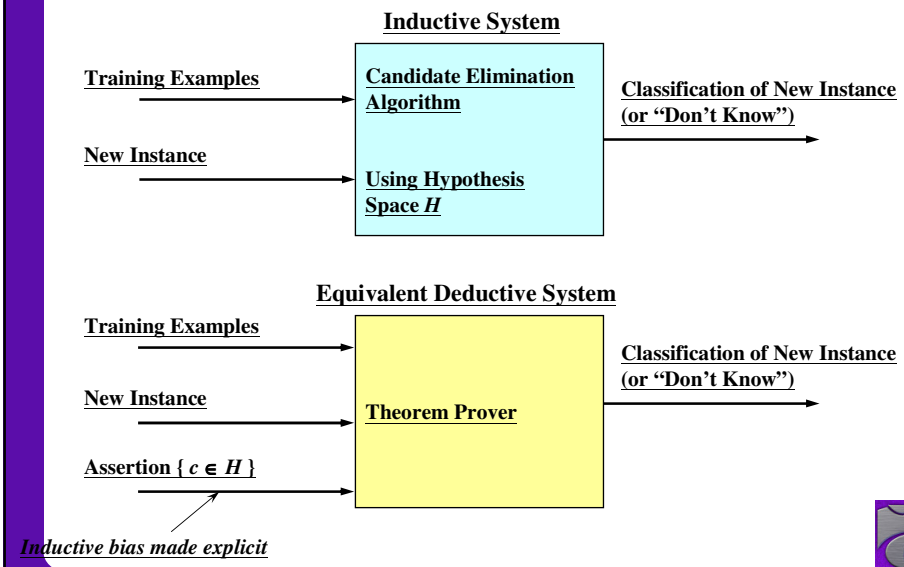
Inductive Bias

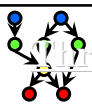
- **Components of An Inductive Bias Definition**
  - Concept learning algorithm L
  - Instances X, target concept c
  - Training examples $D_c = \{\langle x, c(x)\rangle\}$
  - $L(x_i, D_c)$ = classification assigned to instance x_i by L after training on D_c
- **Definition**
  - The inductive bias of L is any minimal set of assertions B such that, for any target concept c and corresponding training examples D_c,

    $\forall x_i \in X . [(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$ where A $\vdash$ B means A *logically entails* B
  - Informal idea: preference for (i.e., restriction to) certain hypotheses by structural (syntactic) means
- **Rationale**
  - Prior assumptions regarding target concept
  - Basis for inductive generalization



Inductive Systems and Equivalent Deductive Systems



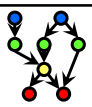


Three Learners with Different Biases

- **Rote Learner**
  - Weakest bias: anything seen before, i.e., no bias
  - Store examples
  - Classify *x if and only if* it matches previously observed example
- **Version Space Candidate Elimination Algorithm**
  - Stronger bias: concepts belonging to conjunctive *H*
  - Store extremal generalizations and specializations
  - Classify *x if and only if* it "falls within" *S* and *G* boundaries (all members agree)
- **Find-S**
  - Even stronger bias: most specific hypothesis
  - Prior assumption: any instance *not observed to be positive* is negative
  - Classify *x* based on *S* set

Views of Learning

- **Removal of (Remaining) Uncertainty**
  - Suppose unknown function was *known* to be *m-of-n* Boolean function
  - Could use training data to infer the function
- **Learning and Hypothesis Languages**
  - Possible approach to *guess a good, small hypothesis language*:
    - Start with a very small language
    - Enlarge until it contains a hypothesis that fits the data
  - Inductive bias
    - Preference for certain languages
    - Analogous to data compression (removal of redundancy)
    - Later: coding the "model" versus coding the "uncertainty" (error)
- **We Could Be Wrong!**
  - Prior knowledge could be wrong (e.g., $y = x_4 \wedge$ one-of (x_1, x_3) also consistent)
  - If guessed language was wrong, errors will occur on new cases



- **Develop Ways to Express Prior Knowledge**
  - Role of prior knowledge: guides search for hypotheses / hypothesis languages
  - Expression languages for prior knowledge
    - Rule grammars; stochastic models; etc.
    - Restrictions on computational models; other (formal) specification methods
- **Develop Flexible Hypothesis Spaces**
  - Structured collections of hypotheses
    - Agglomeration: nested collections (hierarchies)
    - Partitioning: decision trees, lists, rules
    - Neural networks; cases, etc.
  - Hypothesis spaces of adaptive size
- **Either Case: Develop Algorithms for Finding A Hypothesis That Fits Well**
  - Ideally, will generalize well
- **Later: Bias *Optimization* (Meta-Learning, Wrappers)**

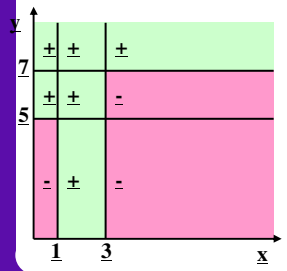
- **What General Laws Constrain Inductive Learning?**
- **What Learning Problems Can Be Solved?**
- **When Can We Trust The Output of A Learning Algorithm?**
- **We Seek Theory To Relate:**
  - Probability of successful learning
  - Number of training examples
  - Complexity of hypothesis space
  - Accuracy to which target concept is approximated
  - Manner in which training examples are presented

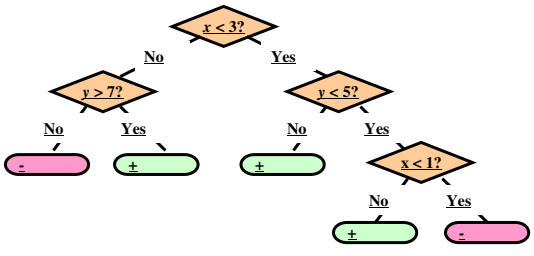
When to Consider Using Decision Trees

- Instances Describable by Attribute-Value Pairs
- Target Function Is Discrete Valued
- Disjunctive Hypothesis May Be Required
- Possibly Noisy Training Data
- Examples
  - Equipment or medical diagnosis
  - Risk analysis
    - Credit, loans
    - Insurance
    - Consumer fraud
    - Employee fraud
  - Modeling calendar scheduling preferences (predicting quality of candidate time)

Decision Trees and Decision Boundaries

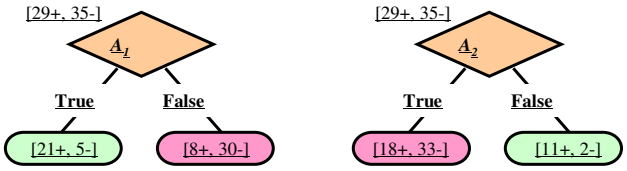
- Instances Usually Represented Using Discrete Valued Attributes
  - Typical types
    - Nominal ({red, yellow, green})
    - Quantized ({low, medium, high})
  - Handling numerical values
    - Discretization, a form of vector quantization (e.g., histogramming)
    - Using thresholds for splitting nodes
- Example: Dividing Instance Space into Axis-Parallel Rectangles





Decision Tree Learning: Top-Down Induction (*ID3*)

- **Algorithm *Build-DT* (*Examples*, *Attributes*)**
  - IF all examples have the same label THEN RETURN (leaf node with *label*)
  - ELSE
    - IF set of attributes is empty THEN RETURN (leaf with *majority label*)
    - ELSE
      - Choose best attribute A as root
      - FOR each value v of A
        - Create a branch out of the root for the condition $A = v$
        - IF $\{x \in Examples: x.A = v\} = \emptyset$ THEN RETURN (leaf with *majority label*)
        - ELSE *Build-DT* ($\{x \in Examples: x.A = v\}$, Attributes ~ {A})
- **But Which Attribute Is Best?**



Broadening the Applicability of Decision Trees

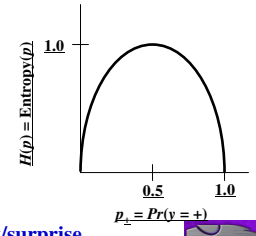
- **Assumptions in Previous Algorithm**
  - Discrete *output*
    - Real-valued outputs are possible
    - Regression trees [Breiman *et al*, 1984]
  - Discrete *input*
    - Quantization methods
    - *Inequalities* at nodes instead of equality tests (see rectangle example)
- **Scaling Up**
  - Critical in knowledge discovery and database mining (KDD) from very large databases (VLDB)
  - Good news: efficient algorithms exist for processing many *examples*
  - Bad news: much harder when there are too many *attributes*
- **Other Desired Tolerances**
  - Noisy data (classification noise ≡ incorrect labels; attribute noise ≡ inaccurate or imprecise data)
  - Missing attribute values

Choosing the "Best" Root Attribute

- **Objective**
  - Construct a decision tree that is a small as possible (Occam's Razor)
  - Subject to: consistency with labels on training data
- **Obstacles**
  - Finding the *minimal* consistent hypothesis (i.e., decision tree) is NP-hard (D'oh!)
  - Recursive algorithm (*Build-DT*)
    - A greedy heuristic search for a simple tree
    - Cannot guarantee optimality (D'oh!)
- **Main Decision: Next Attribute to Condition On**
  - Want: attributes that split examples into sets that are relatively pure in one label
  - Result: closer to a leaf node
  - Most popular heuristic
    - Developed by J. R. Quinlan
    - Based on information gain
    - Used in *ID3* algorithm

Entropy: Intuitive Notion

- **A Measure of Uncertainty**
  - The Quantity
    - Purity: how close a set of instances is to having just one label
    - Impurity (disorder): how close it is to total uncertainty over labels
  - The Measure: Entropy
    - Directly proportional to impurity, uncertainty, irregularity, surprise
    - Inversely proportional to purity, certainty, regularity, redundancy
- **Example**
  - For simplicity, assume H = {0, 1}, distributed according to $Pr(y)$
    - Can have (more than 2) discrete class labels
    - Continuous random variables: differential entropy
  - Optimal purity for y: either
    - $Pr(y = 0) = 1, Pr(y = 1) = 0$
    - $Pr(y = 1) = 1, Pr(y = 0) = 0$
  - What is the least pure probability distribution?
    - $Pr(y = 0) = 0.5, Pr(y = 1) = 0.5$
    - Corresponds to maximum impurity/uncertainty/irregularity/surprise
    - Property of entropy: concave function ("concave downward")



Entropy: Information Theoretic Definition

- **Components**
  - D: a set of examples $\{\langle x_1, c(x_1)\rangle, \langle x_2, c(x_2)\rangle, \ldots, \langle x_m, c(x_m)\rangle\}$
  - $p_+ = Pr(c(x) = +)$, $p_- = Pr(c(x) = -)$
- **Definition**
  - H is defined over a probability density function p
  - D contains examples whose frequency of + and - labels indicates p_+ and p_- for the observed data
  - The entropy of D relative to c is:

$$H(D) \equiv -p_+ \log_b(p_+) - p_- \log_b(p_-)$$

- **What Units is H Measured In?**
  - Depends on the base b of the log (bits for $b = 2$, nats for $b = e$, etc.)
  - A single bit is required to encode each example in the worst case ($p_+ = 0.5$)
  - If there is less uncertainty (e.g., $p_+ = 0.8$), we can use less than 1 bit each

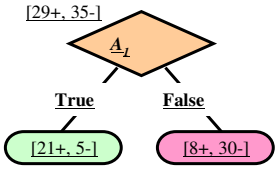
Information Gain: Information Theoretic Definition

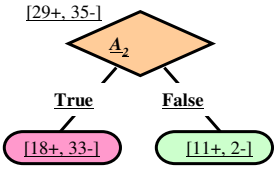
- **Partitioning on Attribute Values**
  - Recall: a partition of D is a collection of disjoint subsets whose union is D
  - Goal: *measure the uncertainty removed* by splitting on the value of attribute A
- **Definition**
  - The information gain of D relative to attribute A is the expected reduction in entropy due to splitting ("sorting") on A:

$$Gain(D, A) \equiv -H(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \bullet H(D_v) \right]$$

where D_v is $\{x \in D: x.A = v\}$, the set of examples in D where attribute A has value v

  - Idea: partition on A; scale entropy to the size of each subset D_v
- **Which Attribute Is Best?**



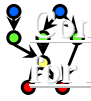


Illustrative Example

- **Training Examples for Concept *PlayTennis***

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

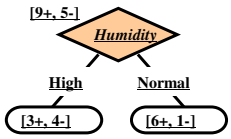
- ***ID3* ≡ *Build-DT* using *Gain*(•)**
- **How Will *ID3* Construct A Decision Tree?**

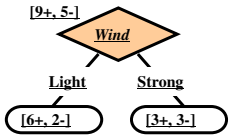


Constructing A Decision Tree for *PlayTennis* using *ID3* [1]

- **Selecting The Root Attribute**

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No





- **Prior (unconditioned) distribution: 9+, 5-**
  - ***H*(*D*) = -(9/14) lg (9/14) - (5/14) lg (5/14) *bits* = 0.94 bits**
  - ***H*(*D*, *Humidity* = High) = -(3/7) lg (3/7) - (4/7) lg (4/7) = 0.985 bits**
  - ***H*(*D*, *Humidity* = Normal) = -(6/7) lg (6/7) - (1/7) lg (1/7) = 0.592 bits**
  - ***Gain*(*D*, *Humidity*) = 0.94 - (7/14) * 0.985 + (7/14) * 0.592 = 0.151 bits**
  - **Similarly, *Gain* (*D*, *Wind*) = 0.94 - (8/14) * 0.811 + (6/14) * 1.0 = 0.048 bits**

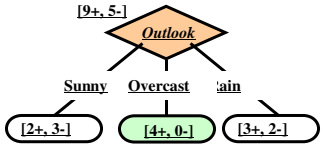
$$Gain(D, A) \equiv -H(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \bullet H(D_v) \right]$$

Constructing A Decision Tree for *PlayTennis* using *ID3* [2]

- **Selecting The Root Attribute**

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

  - *Gain*(*D*, *Humidity*) = 0.151 bits
  - *Gain*(*D*, *Wind*) = 0.048 bits
  - *Gain*(*D*, *Temperature*) = 0.029 bits
  - *Gain*(*D*, *Outlook*) = 0.246 bits



- **Selecting The Next Attribute (Root of Subtree)**
  - Continue until every example is included in path or purity = 100%
  - What does purity = 100% mean?
  - Can *Gain*(*D*, *A*) < 0?

Constructing A Decision Tree for *PlayTennis* using *ID3* [3]

- **Selecting The Next Attribute (Root of Subtree)**

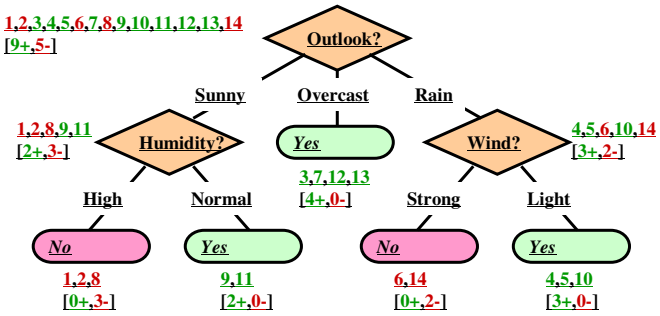
Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

  - Convention: lg (0/a) = 0
  - $Gain(D_{Sunny}, Humidity) = 0.97 - (3/5) * 0 - (2/5) * 0 = 0.97$ bits
  - $Gain(D_{Sunny}, Wind) = 0.97 - (2/5) * 1 - (3/5) * 0.92 = 0.02$ bits
  - $Gain(D_{Sunny}, Temperature) = 0.57$ bits

- **Top-Down Induction**
  - For discrete-valued attributes, terminates in $O(n)$ splits
  - Makes at most one pass through data set at each level (why?)

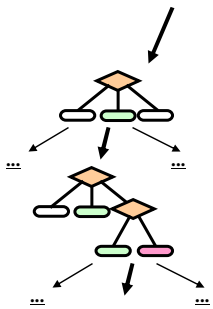
Constructing A Decision Tree for *PlayTennis* using *ID3* [4]

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



Hypothesis Space Search by *ID3*

- **Search Problem**
  - Conduct a search of the *space of decision trees*, which can represent all possible discrete functions
    - Pros: expressiveness; flexibility
    - Cons: computational complexity; large, incomprehensible trees (next time)
  - Objective: to find the best decision tree (minimal consistent tree)
  - Obstacle: finding this tree is NP-hard
  - Tradeoff
    - Use heuristic (figure of merit that guides search)
    - Use greedy algorithm
    - *Aka* hill-climbing (gradient "descent") without backtracking
- **Statistical Learning**
  - Decisions based on statistical descriptors p_+, p_- for subsamples D_v
  - In ID3, *all data used*
  - Robust to noisy data



Inductive Bias in *ID3*

- **Heuristic : Search :: Inductive Bias : Inductive Generalization**
  - ***H* is the power set of instances in *X***
  - **⇒ Unbiased? Not really…**
    - **Preference for short trees (termination condition)**
    - **Preference for trees with high information gain attributes near the root**
    - ***Gain*(•): a heuristic function that *captures the inductive bias of ID3***
  - **Bias in *ID3***
    - **Preference for some hypotheses is encoded in heuristic function**
    - **Compare: a restriction of hypothesis space *H* (previous discussion of propositional normal forms: *k*-CNF, etc.)**
- **Preference for Shortest Tree**
  - **Prefer shortest tree that fits the data**
  - **An Occam's Razor bias: shortest hypothesis that explains the observations**

MLC++: Machine Learning Library

- ***MLC++***
  - **http://www.sgi.com/Technology/mlc**
  - **An object-oriented machine learning library**
  - **Contains a suite of inductive learning algorithms (including *ID3*)**
  - **Supports incorporation, reuse of other DT algorithms (*C4.5*, etc.)**
  - **Automation of statistical evaluation, cross-validation**
- **Wrappers**
  - **Optimization loops that iterate over inductive learning functions (inducers)**
  - **Used for performance tuning (finding subset of *relevant* attributes, etc.)**
- **Combiners**
  - **Optimization loops that iterate over or interleave inductive learning functions**
  - **Used for performance tuning (finding subset of *relevant* attributes, etc.)**
  - **Examples: bagging, boosting (later in this course) of *ID3*, *C4.5***
- **Graphical Display of Structures**
  - **Visualization of DTs (AT&T *dotty*, SGI *MineSet TreeViz*)**
  - **General logic diagrams (projection visualization)**

Using *MLC++*

- **Refer to *MLC++* references**
  - Data mining paper (Kohavi, Sommerfeld, and Dougherty, 1996)
  - *MLC++* user manual: Utilities 2.0 (Kohavi and Sommerfeld, 1996)
  - *MLC++* tutorial (Kohavi, 1995)
  - Other development guides and tools on SGI *MLC++* web site
- **Online Documentation**
  - Consult class web page after Homework 2 is handed out
  - *MLC++* (Linux build) to be used for Homework 3
  - Related system: *MineSet* (commercial data mining edition of *MLC++*)
    - http://www.sgi.com/software/mineset
    - Many common algorithms
    - Common DT display format
    - Similar data formats
- **Experimental Corpora (Data Sets)**
  - UC Irvine Machine Learning Database Repository (MLDBR)
  - See http://www.kdnuggets.com and class "Resources on the Web" page

Terminology

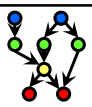
- **Decision Trees (DTs)**
  - Boolean DTs: target concept is binary-valued (i.e., Boolean-valued)
  - Building DTs
    - Histogramming: a method of vector quantization (encoding input using bins)
    - Discretization: converting continuous input into discrete (e.g.., by histogramming)
- **Entropy and Information Gain**
  - Entropy $H(D)$ for a data set D relative to an implicit concept c
  - Information gain $Gain(D, A)$ for a data set partitioned by attribute A
  - Impurity, uncertainty, irregularity, surprise versus purity, certainty, regularity, redundancy
- **Heuristic Search**
  - Algorithm *Build-DT*: greedy search (hill-climbing without backtracking)
  - *ID3* as *Build-DT* using the heuristic $Gain(\bullet)$
  - Heuristic : Search :: Inductive Bias : Inductive Generalization
- ***MLC++* (Machine Learning Library in C++)**
  - Data mining libraries (e.g., *MLC++*) and packages (e.g., *MineSet*)
  - Irvine Database: the Machine Learning Database Repository at UCI

Summary Points

- Decision Trees (DTs)
  - Can be boolean ($c(x) \in \{+, -\}$) or range over multiple classes
  - When to use DT-based models
- Generic Algorithm *Build-DT*: Top Down Induction
  - Calculating best attribute upon which to split
  - Recursive partitioning
- Entropy and Information Gain
  - Goal: to measure *uncertainty removed* by splitting on a candidate attribute *A*
    - Calculating information gain (change in entropy)
    - Using information gain in construction of tree
  - *ID3* ≡ *Build-DT* using *Gain*(•)
- ID3 as Hypothesis Space Search (in State Space of Decision Trees)
- Heuristic Search and Inductive Bias
- Data Mining using *MLC++* (Machine Learning Library in C++)
- Next: More Biases (Occam's Razor); Managing DT Induction

SUMMARY

- Reading: Sections 18.1 – 18.2, Russell & Norvig
- Suggested Exercises: 2.2, 2.3, 2.4, 2.6
- Taxonomy of Learning Systems
- Today: Overview, Learning from Examples
  - (Supervised) concept learning framework
  - Simple approach: assumes no noise; illustrates key concepts
- Today & Next Class: Hypothesis Learning and Inductive Bias
  - Sources: Mitchell (1997) – online notes and handout
  - Wednesday: inductive learning, version space
  - Friday: candidate elimination algorithm, active learning, inductive bias
  - Background concepts: partially-ordered set (poset) formalism
- Next Class: Decision Trees
- Outside References
  - Han & Kamber 2nd edition (2006)
  - Witten & Frank 2nd edition (2005)
  - Mitchell (1997), Chapters 1 – 3



Terminology

- Learning: Improving at Task given Performance Measure, Experience
- Performance Element: Part of System that Applies Result of Learning
- Types of Learning
  - Supervised: with "teacher" (often, classification from labeled examples)
  - Unsupervised: from data, using similarity measure (unlabeled instances)
  - Reinforcement: "by doing", with reward/penalty signal
- Supervised Learning: Target Functions
  - Target function – function *c* or *f* to be learned
  - Target – desired value y to be predicted (sometimes "target function")
  - Example / labeled instance – tuples of the form <*x, f(x)*>
  - Classification function, classifier – nominal-valued *f* (enumerated return type)
- Clustering: Application of Unsupervised Learning
- Concepts and Hypotheses
  - Concept – function *c* from observations to TRUE or FALSE (membership)
  - Class label – output of classification function
  - Hypothesis – proposed function *h* believed to be similar to *c* (or *f*)