Introduction to Artificial Intelligence

CS171, Winter Quarter, 2019
Introduction to Artificial Intelligence
Prof. Richard Lathrop

Read Beforehand: All assigned reading so far

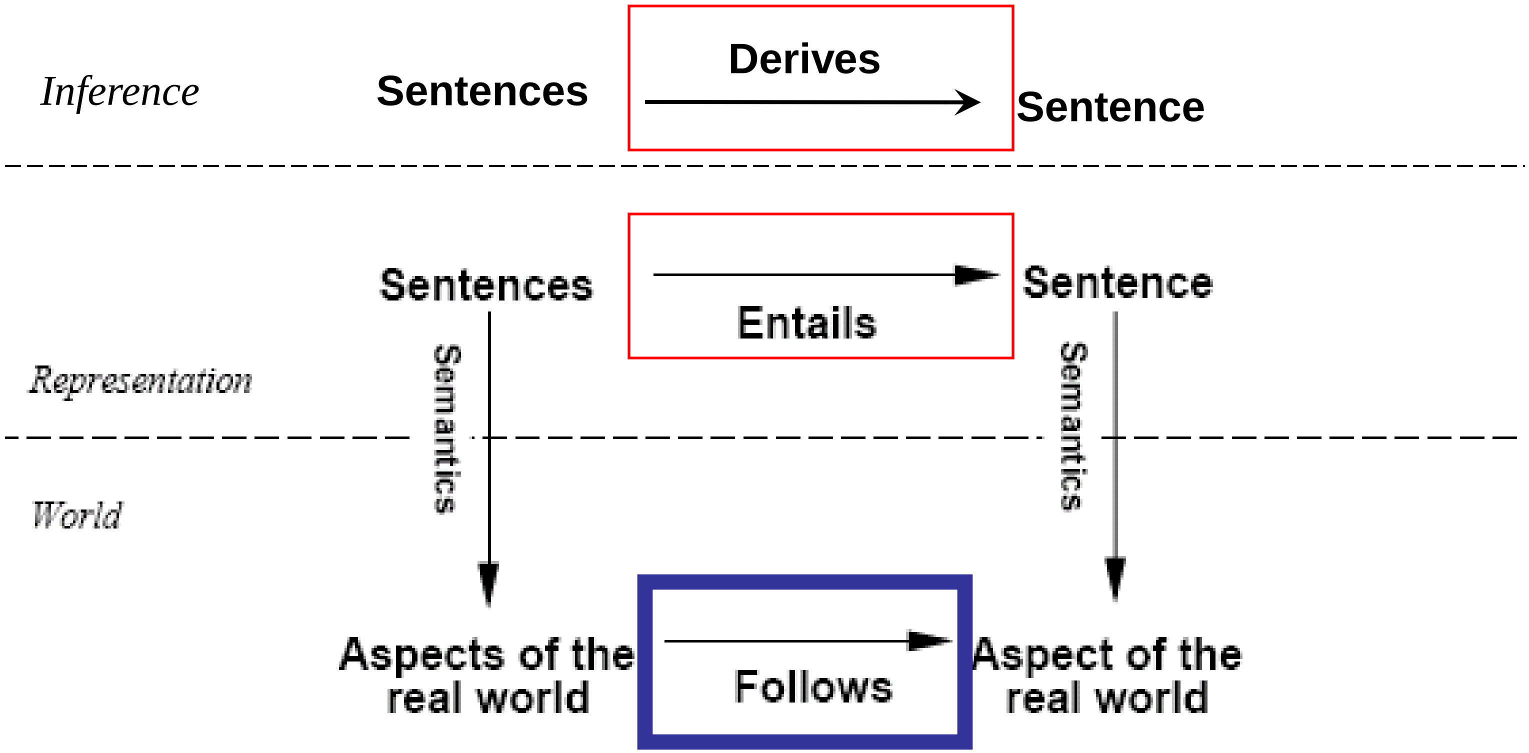
Final Exam Review

- Propositional Logic B: R&N Chap 7.1-7.5
- Predicate Logic, Knowledge Representation: R&N Chap 8.1-8.5, 9.1-9.2
- Probability: R&N Chap 13
- Bayesian Networks: R&N Chap 14.1-14.5
- Intro Machine Learning: R&N Chap 18.1-18.4

Review Propositional Logic
Chapter 7.1-7.5; Optional 7.6-7.8

- Definitions:
  - Syntax, Semantics, Sentences, Propositions, Entails, Follows, Derives, Inference, Sound, Complete, Model, Satisfiable, Valid (or Tautology)
- Syntactic & Semantic Transformations:
  - E.g., $(A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$
  - E.g., $(KB \models \alpha) \equiv (\models (KB \Rightarrow \alpha)$
- Truth Tables:
  - Negation, Conjunction, Disjunction, Implication, Equivalence (Biconditional)
- Inference:
  - By Resolution (CNF)
  - By Backward & Forward Chaining (Horn Clauses)
  - By Model Enumeration (Truth Tables)

Review: Schematic for Follows, Entails, and Derives



If KB is true in the real world,

then any sentence α ***entailed*** *by KB*

and any sentence α ***derived*** *from KB*

by a sound inference procedure

is also true in the real world.

Recap propositional logic: Validity and satisfiability

A sentence is valid if it is true in all models,

e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is false in all models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models A$ if and only if $(KB \wedge \neg A)$ is unsatisfiable

(there is no model for which KB is true and A is false)

Inference Procedures

- *KB* $\vdash_i$ A means that sentence A can be derived from *KB* by procedure *i*
- Soundness: *i* is sound if whenever *KB* $\vdash_i \alpha$, it is also true that *KB* $\models \alpha$
  - *(no wrong inferences, but maybe not all inferences)*
- Completeness: *i* is complete if whenever *KB* $\models \alpha$, it is also true that *KB* $\vdash_i \alpha$
  - *(all inferences can be made, but maybe some wrong extra ones as well)*
- Entailment can be used for inference (Model checking)
  - enumerate all possible models and check whether α is true.
  - For *n* symbols, time complexity is $O(2^n)$...
- Inference can be done directly on the sentences
  - Forward chaining, backward chaining, resolution (see FOPC, later)

Inference by Resolution

- KB is represented in CNF
  - KB = AND of all the sentences in KB
  - KB sentence = clause = OR of literals
  - Literal = propositional symbol or its negation
- Find two clauses in KB, one of which contains a literal and the other its negation
  - Cancel the literal and its negation
  - Bundle everything else into a new clause
  - Add the new clause to KB
  - Repeat

Example: Conversion to CNF

Example: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $= (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$ and simplify.

   $= (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and simplify.

   $$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta), \neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$$

   $= (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributive law ($\wedge$ over $\vee$) and simplify.

   $= (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Example: Conversion to CNF

Example: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

From the previous slide we had:

$= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

5. KB is the conjunction of all of its sentences (all are true), so write each clause (disjunct) as a sentence in KB:

KB =

...

$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$ → $(\neg B_{1,1} \quad P_{1,2} \quad P_{2,1})$

$(\neg P_{1,2} \lor B_{1,1})$ → $(\neg P_{1,2} \quad B_{1,1})$

$(\neg P_{2,1} \lor B_{1,1})$ → $(\neg P_{2,1} \quad B_{1,1})$

(same)

...

<u>Often, Won't Write "$\lor$" or "$\land$"</u>

(we know they are there)

Resolution = Efficient Implication

Recall that (A => B) = ((NOT A) OR B)
and so:
(Y OR X) = ((NOT X) => Y)
((NOT Y) OR Z) = (Y => Z)
which yields:
((Y OR X) AND ((NOT Y) OR Z)) = ((NOT X) => Z) = (X OR Z)

(OR A B C D)	->Same ->	(NOT (OR B C D)) => A
(OR ¬A E F G)	->Same ->	A => (OR E F G)
----------------------------		--
(OR B C D E F G)		(NOT (OR B C D)) => (OR E F G)
		--
	→	(OR B C D E F G)

Recall: All clauses in KB are conjoined by an implicit AND (= CNF representation).

Resolution Examples

- Resolution: inference rule for CNF: sound and complete! *

$(A \vee B \vee C)$

$(\neg A)$

"If A or B or C is true, but not A, then B or C must be true."

$-----------$

$\therefore (B \vee C)$

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

"If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true."

$-----------$

$\therefore (B \vee C \vee D \vee E)$

$(A \vee B)$

$(\neg A \vee B)$

"If A or B is true, and not A or B is true, then B must be true."

$--------$

$\therefore (B \vee B) \equiv B$ ← Simplification is done always.

* Resolution is "refutation complete" in that it can prove the truth of any entailed sentence by refutation.

More Resolution Examples

- (P Q ¬R S) with (P ¬Q W X) yields (P ¬R S W X)
  - Order of literals within clauses does not matter.
- (P Q ¬R S) with (¬P) yields (Q ¬R S)
- (¬R) with (R) yields ( ) or FALSE
- (P Q ¬R S) with (P R ¬S W X) yields (P Q ¬R R W X) or (P Q S ¬S W X) or TRUE
- (P ¬Q R ¬S) with (P ¬Q R ¬S) yields None possible
- (P ¬Q ¬S W) with (P R ¬S X) yields None possible
- ((¬ A) (¬ B) (¬ C) (¬ D)) with ((¬ C) D) yields ((¬ A) (¬ B) (¬ C))
- ((¬ A) (¬ B) (¬ C)) with ((¬ A) C) yields ((¬ A) (¬ B))
- ((¬ A) (¬ B)) with (B) yields (¬ A)
- (A C) with (A (¬ C)) yields (A)
- (¬ A) with (A) yields ( ) or FALSE

Only Resolve <u>ONE</u> Literal Pair!

If more than one pair, result always = TRUE.

<u>Useless!!</u> Always simplifies to TRUE!!

No!

(OR A B C D)
(OR ¬A ¬B F G)

(OR C D F G)

No! This is wrong!

No!

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR D)

No! This is wrong!

Yes! (but = TRUE)

(OR A B C D)
(OR ¬A ¬B F G)

(OR B ¬B C D F G)

Yes! (but = TRUE)

Yes! (but = TRUE)

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR A ¬A B ¬B D)

Yes! (but = TRUE)

Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- Generate all new sentences from KB and the (negated) query.
- One of two things can happen:

1. We find $P \wedge \neg P$ which is unsatisfiable. I.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \wedge \neg\alpha$ (non-trivial) and hence we cannot entail the query.

Resolution example

Resulting Knowledge Base stated in CNF

- "Laws of Physics" in the Wumpus World:

  $(\neg B_{1,1} \quad P_{1,2} \quad P_{2,1})$

  $(\neg P_{1,2} \quad B_{1,1})$

  $(\neg P_{2,1} \quad B_{1,1})$

- Particular facts about a specific instance:

  $(\neg B_{1,1})$

- <u>Negated</u> goal or query sentence:

  $(P_{1,2})$

Resolution example

A Resolution proof ending in ( )

- Knowledge Base at start of proof:

$(\neg B_{1,1} \quad P_{1,2} \quad P_{2,1})$

$(\neg P_{1,2} \quad B_{1,1})$

$(\neg P_{2,1} \quad B_{1,1})$

$(\neg B_{1,1})$

$(P_{1,2})$

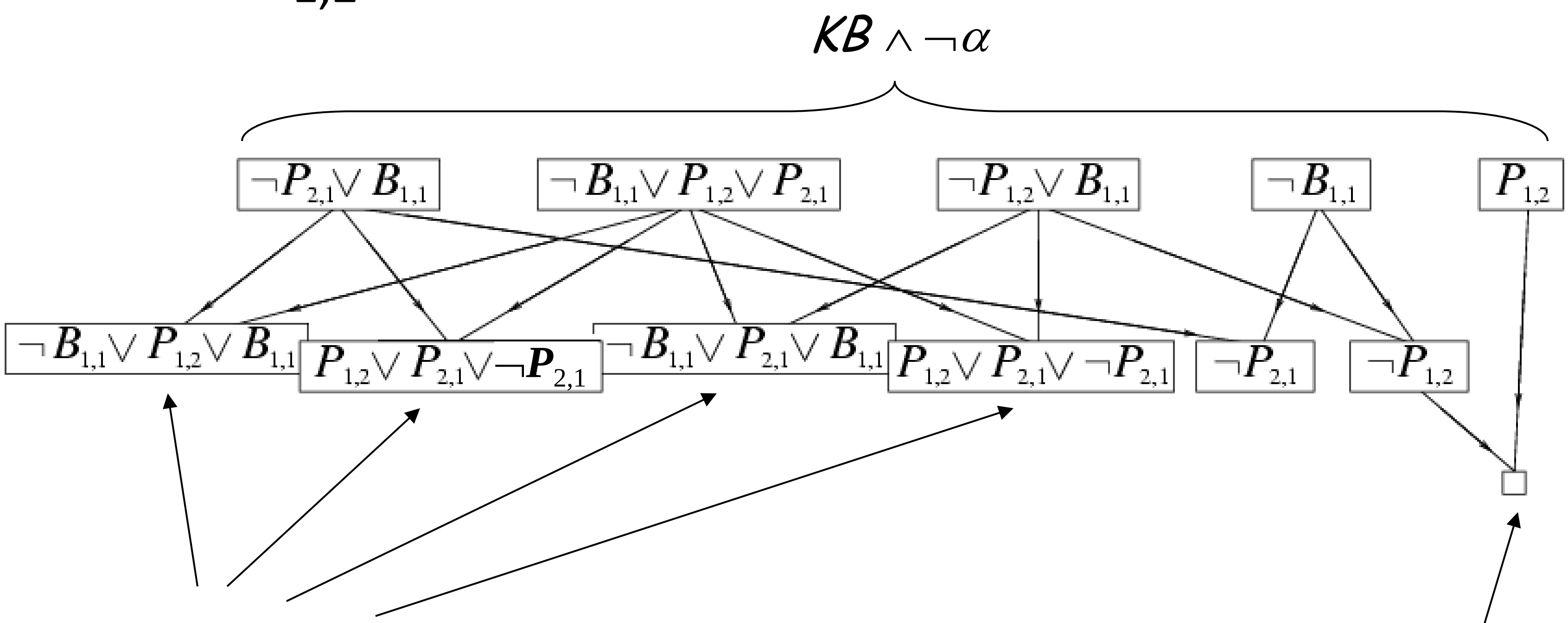
A resolution proof ending in ( ):

- Resolve $(\neg P_{1,2} \quad B_{1,1})$ and $(\neg B_{1,1})$ to give $(\neg P_{1,2})$
- Resolve $(\neg P_{1,2})$ and $(P_{1,2})$ to give ( )

- Consequently, the goal or query sentence is entailed by KB.
- Of course, there are many other proofs, which are OK iff correct.

Resolution example

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



True!

A sentence in KB is not "used up" when it is used in a resolution step. It is true, remains true, and is still in KB.

False in all worlds

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  ((NOT Y) (NOT R)) **(M Y)** **(R Y)** **(H (NOT M))**
  (H R) **((NOT H) G)** **((NOT G) (NOT H))**

- **Fourth, produce a resolution proof ending in ( ):**
- Resolve (¬H ¬G) and (¬H G) to give (¬H)
- Resolve (¬Y ¬R) and (Y M) to give (¬R M)
- Resolve (¬R M) and (R H) to give (M H)
- Resolve (M H) and (¬M H) to give (H)
- Resolve (¬H) and (H) to give ( )

- Of course, there are many other proofs, which are OK iff correct.

Propositional Logic --- Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions

- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences
  - valid: sentence is true in every model (a tautology)

- Logical equivalences allow syntactic manipulations

- Propositional logic lacks expressive power
  - Can only state specific facts about the world.
  - Cannot express general rules about the world
    (use First Order Predicate Logic instead)

Review First-Order Logic
Chapter 8.1-8.5, 9.1-9.2, 9.5.1-9.5.5

- Syntax & Semantics
  - Predicate symbols, function symbols, constant symbols, variables, quantifiers.
  - Models, symbols, and interpretations
- De Morgan's rules for quantifiers
- Nested quantifiers
  - Difference between "$\forall$ x $\exists$ y P(x, y)" and "$\exists$ x $\forall$ y P(x, y)"
- Translate simple English sentences to FOPC and back
  - $\forall$ x $\exists$ y Likes(x, y) $\Leftrightarrow$ "Everyone has someone that they like."
  - $\exists$ x $\forall$ y Likes(x, y) $\Leftrightarrow$ "There is someone who likes every person."
- Unification and the Most General Unifier
- Inference in FOL
  - By Resolution (CNF)
  - By Backward & Forward Chaining (Horn Clauses)
- Knowledge engineering in FOL

Syntax of FOL: Basic elements

- Constants KingJohn, 2, UCI,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Quantifiers $\forall$, $\exists$
- Connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ (standard)
- Equality = (but causes difficulties….)

Syntax of FOL: Basic syntax elements are symbols

- **Constant** Symbols (correspond to English nouns)
  - Stand for objects in the world.
    - E.g., KingJohn, 2, UCI, ...
- **Predicate** Symbols (correspond to English verbs)
  - Stand for relations (maps a tuple of objects to a **truth-value**)
    - E.g., Brother(Richard, John), greater_than(3,2), ...
  - P(x, y) is usually read as "x is P of y."
    - E.g., Mother(Ann, Sue) is usually "Ann is Mother of Sue."
- **Function** Symbols (correspond to English nouns)
  - Stand for functions (maps a tuple of objects to an **object**)
    - E.g., Sqrt(3), LeftLegOf(John), ...
- **Model** (world) = set of domain objects, relations, functions
- **Interpretation** maps symbols onto the model (world)
  - Very many interpretations are possible for each KB and world!
  - The KB is to rule out those inconsistent with our knowledge.

Syntax of FOL: Terms

- **Term** = logical expression that **refers to an object**

- **There are two kinds of terms:**

  - **Constant Symbols** stand for (or name) objects:
    - E.g., KingJohn, 2, UCI, Wumpus, ...

  - **Function Symbols** map tuples of objects to an object:
    - E.g., LeftLeg(KingJohn), Mother(Mary), Sqrt(x)
    - This is nothing but a complicated kind of name
      - No “subroutine” call, no “return value”

Syntax of FOL: Atomic Sentences

- **Atomic Sentences** state facts (logical truth values).
  - An **atomic sentence** is a Predicate symbol, optionally followed by a parenthesized list of any argument terms
  - E.g., *Married(Father(Richard), Mother(John))*
  - An **atomic sentence** asserts that some relationship (some predicate) holds among the objects that are its arguments.

- An **Atomic Sentence is true** in a given model if the relation referred to by the predicate symbol holds among the objects (terms) referred to by the arguments.

Syntax of FOL: Connectives & Complex Sentences

- **Complex Sentences** are formed in the same way, using the same logical connectives, as in propositional logic

- The **Logical Connectives**:
  - $\Leftrightarrow$ biconditional
  - $\Rightarrow$ implication
  - $\wedge$ and
  - $\vee$ or
  - $\neg$ negation

- **Semantics** for these logical connectives are the same as we already know from propositional logic.

Syntax of FOL: Variables

- **Variables** range over objects in the world.
- A **variable** is like a **term** because it represents an object.
- A **variable** may be used wherever a **term** may be used.
  - **Variables** may be arguments to functions and predicates.
- (A **term with NO variables** is called a **ground term**.)
- (A **variable not bound by a quantifier** is called **free**.)
  - All variables we will use are bound by a quantifier.

Syntax of FOL: Logical Quantifiers

- There are two **Logical Quantifiers:**
  - **Universal:** $\forall$ x P(x) means "For all x, P(x)."
    - The "upside-down A" reminds you of "ALL."
    - Some texts put a comma after the variable: $\forall$ x, P(x)
  - **Existential:** $\exists$ x P(x) means "There exists x such that, P(x)."
    - The "backward E" reminds you of "EXISTS."
    - Some texts put a comma after the variable: $\exists$ x, P(x)

- You can ALWAYS convert one quantifier to the other.
  - $\forall x\, P(x) \equiv \neg\exists x\, \neg P(x)$
  - $\exists x\, P(x) \equiv \neg\forall x\, \neg P(x)$
  - **RULES:** $\forall \equiv \neg\exists\neg$ and $\exists \equiv \neg\forall\neg$

- **RULES:** To move negation "in" across a quantifier,

  Change the quantifier to "the other quantifier"
  and negate the predicate on "the other side."
  - $\neg\forall x\, P(x) \equiv \neg\, \neg\exists x\, \neg P(x) \equiv \exists x\, \neg P(x)$
  - $\neg\exists x\, P(x) \equiv \neg\, \neg\forall x\, \neg P(x) \equiv \forall x\, \neg P(x)$

Universal Quantification ∀

- ∀ x means "for all x it is true that…"

- Allows us to make statements about all objects that have certain properties

- Can now state general rules:

  ∀ x King(x) => Person(x) "All kings are persons."
  ∀ x Person(x) => HasHead(x) "Every person has a head."
  ∀ i Integer(i) => Integer(plus(i,1)) "If i is an integer then i+1 is an integer."

- **Note: ∀ x King(x) ∧ Person(x) is not correct!**

  This would imply that all objects x are Kings and are People (!)

  ∀ x King(x) => Person(x) is the correct way to say this

- **Note that => (or ⇔) is the natural connective to use with ∀ .**

Existential Quantification ∃

- ∃ x means "there exists an x such that…."
  - There is in the world at least one such object x

- Allows us to make statements about some object without naming it, or even knowing what that object is:

  $\exists x$ King(x) "Some object is a king."
  $\exists x$ Lives_in(John, Castle(x)) "John lives in somebody's castle."
  $\exists i$ Integer(i) $\wedge$ Greater(i,0) "Some integer is greater than zero."

- **Note: $\exists i$ Integer(i) $\Rightarrow$ Greater(i,0) is not correct!**

  It is vacuously true if anything in the world were not an integer (!)

  $\exists i$ Integer(i) $\wedge$ Greater(i,0) is the correct way to say this

- **Note that $\wedge$ is the natural connective to use with $\exists$.**

Combining Quantifiers --- Order (Scope)

The order of "unlike" quantifiers is important.

Like nested variable scopes in a programming language.

Like nested ANDs and ORs in a logical sentence.

$\forall$ x $\exists$ y Loves(x,y)

- For everyone ("all x") there is someone ("exists y") whom they love.
- There might be a different y for each x (y is inside the scope of x)

$\exists$ y $\forall$ x Loves(x,y)

- There is someone ("exists y") whom everyone loves ("all x").
- Every x loves the same y (x is inside the scope of y)

Clearer with parentheses: $\exists$ y ($\forall$ x Loves(x,y))

The order of "like" quantifiers does not matter.

Like nested ANDs and ANDs in a logical sentence

$\forall x\ \forall y\ P(x, y) \equiv \forall y\ \forall x\ P(x, y)$

$\exists x\ \exists y\ P(x, y) \equiv \exists y\ \exists x\ P(x, y)$

De Morgan’s Law for Quantifiers

De Morgan’s Rule

$P \wedge Q \equiv \neg (\neg P \vee \neg Q)$
$P \vee Q \equiv \neg (\neg P \wedge \neg Q)$

$\neg (P \wedge Q) \equiv (\neg P \vee \neg Q)$
$\neg (P \vee Q) \equiv (\neg P \wedge \neg Q)$

Generalized De Morgan’s Rule

$\forall x\, P(x) \equiv \neg \exists x \neg P(x)$
$\exists x\, P(x) \equiv \neg \forall x \neg P(x)$

$\neg \forall x\, P(x) \equiv \exists x \neg P(x)$
$\neg \exists x\, P(x) \equiv \forall x \neg P(x)$

AND/OR Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (¬ OR → AND ¬ ; ¬ AND → OR ¬).

QUANTIFIER Rule is similar: if you bring a negation inside a universal or existential, always switch between them (¬ ∃ →∀ ¬ ; ¬ ∀ → ∃ ¬).

Fun with sentences

Brothers are siblings

$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$

"Sibling" is symmetric

$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x).$

One's mother is one's female parent

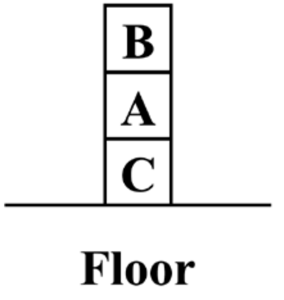
$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \land Parent(x, y)).$

A first cousin is a child of a parent's sibling

$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow\; \exists\, p, ps \;\; Parent(p, x) \land Sibling(ps, p) \land Parent(ps, y)$

Semantics: Interpretation

- An interpretation of a sentence is an assignment that maps
  - Object constants to objects in the worlds,
  - n-ary function symbols to n-ary functions in the world,
  - n-ary relation symbols to n-ary relations in the world
- Given an interpretation, an atomic sentence has the value "true" if it denotes a relation that holds for those individuals denoted in the terms. Otherwise it has the value "false"
  - Example: Block world:
    - A, B, C, floor, On, Clear
  - World:
  - On(A,B) is false, Clear(B) is true, On(C,Floor) is true…
    - Under an interpretation that maps symbol A to block A, symbol B to block B, symbol C to block C, symbol Floor to the floor



Semantics: Models and Definitions

- An interpretation and possible world **satisfies** a wff (sentence) if the wff has the value “true” under that interpretation in that possible world.

- Model: A domain and an interpretation that satisfies a wff is a **model** of that wff

- Validity: Any wff that has the value “true” in all possible worlds and under all interpretations is **valid.**

- Any wff that does not have a model under any interpretation is inconsistent or **unsatisfiable.**

- Any wff that is true in at least one possible world under at least one interpretation is **satisfiable**.

- If a wff w has a value true under all the models of a set of sentences KB then KB logically **entails** w.

Conversion to CNF

- Everyone who loves all animals is loved by someone:

$\forall x\ [\forall y\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x,y)] \Rightarrow [\exists y\ \mathit{Loves}(y,x)]$

1. Eliminate biconditionals and implications

$\forall x\ [\neg\forall y\ \neg\mathit{Animal}(y) \lor \mathit{Loves}(x,y)] \lor [\exists y\ \mathit{Loves}(y,x)]$

2. Move $\neg$ inwards:

$\neg\forall x\ p \equiv \exists x\ \neg p,\ \ \neg\ \exists x\ p \equiv \forall x\ \neg p$

$\forall x\ [\exists y\ \neg(\neg\mathit{Animal}(y) \lor \mathit{Loves}(x,y))] \lor [\exists y\ \mathit{Loves}(y,x)]$
$\forall x\ [\exists y\ \neg\neg\mathit{Animal}(y) \land \neg\mathit{Loves}(x,y)] \lor [\exists y\ \mathit{Loves}(y,x)]$
$\forall x\ [\exists y\ \mathit{Animal}(y) \land \neg\mathit{Loves}(x,y)] \lor [\exists y\ \mathit{Loves}(y,x)]$

Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$

4. Skolemize: a more general form of existential instantiation.

Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

5. Drop universal quantifiers:

$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

6. Distribute $\vee$ over $\wedge$:

$[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Unification

- Recall: Subst(θ, p) = result of substituting θ into sentence p

- Unify algorithm: takes 2 sentences p and q and returns a unifier if one exists

  Unify(p,q) = θ   where Subst(θ, p) = Subst(θ, q)

  where θ is a list of variable/substitution pairs
  that will make p and q syntactically identical

- Example:
  p = Knows(John,x)
  q = Knows(John, Jane)

  Unify(p,q) = {x/Jane}

Unification examples

- simple example: query = Knows(John,x), i.e., who does John know?

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	{fail}

- Last unification fails: only because x can't take values John and OJ at the same time
  - But we know that if John knows x, and everyone (x) knows OJ, we should be able to infer that John knows OJ
- Problem is due to use of same variable x in both sentences
- Simple solution: Standardizing apart eliminates overlap of variables, e.g., Knows(z,OJ)

Unification examples

- UNIFY(Knows(John, x), Knows(John, Jane)) { x / Jane }
- UNIFY(Knows(John, x), Knows(y, Jane)) { x / Jane, y / John }
- UNIFY(Knows(y, x), Knows(John, Jane)) { x / Jane, y / John }
- UNIFY(Knows(John, x), Knows(y, Father (y))) { y / John, x / Father (John) }
- UNIFY(Knows(John, F(x)), Knows(y, F(F(z)))) { y / John, x / F (z) }
- UNIFY(Knows(John, F(x)), Knows(y, G(z))) None
- UNIFY(Knows(John, F(x)), Knows(y, F(G(y)))) { y / John, x / G (John) }

Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

- Prove that Col. West is a criminal

Example knowledge base (Horn clauses)

... it is a crime for an American to sell weapons to hostile nations:

American(x) $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*

Nono … has some missiles, i.e., $\exists$x Owns(Nono,x) $\wedge$ Missile(x):

Owns(Nono,M_1*)* $\wedge$ *Missile(*M_1*)*

… all of its missiles were sold to it by Colonel West

Missile(x) $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*

Missiles are weapons:

Missile(x) $\Rightarrow$ *Weapon(x)*

An enemy of America counts as "hostile“:

Enemy(x,America) $\Rightarrow$ *Hostile(x)*

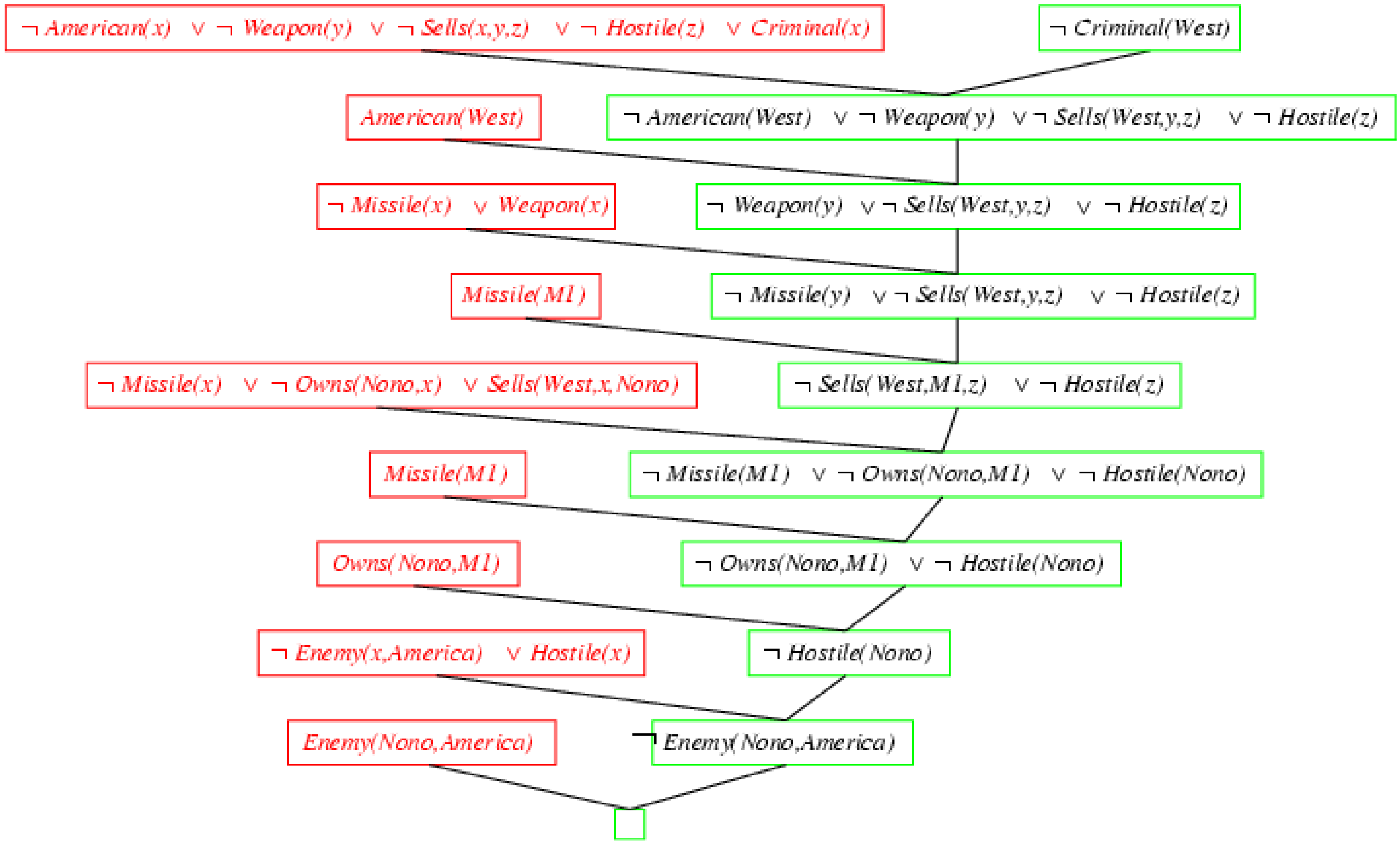
West, who is American …

American(West)

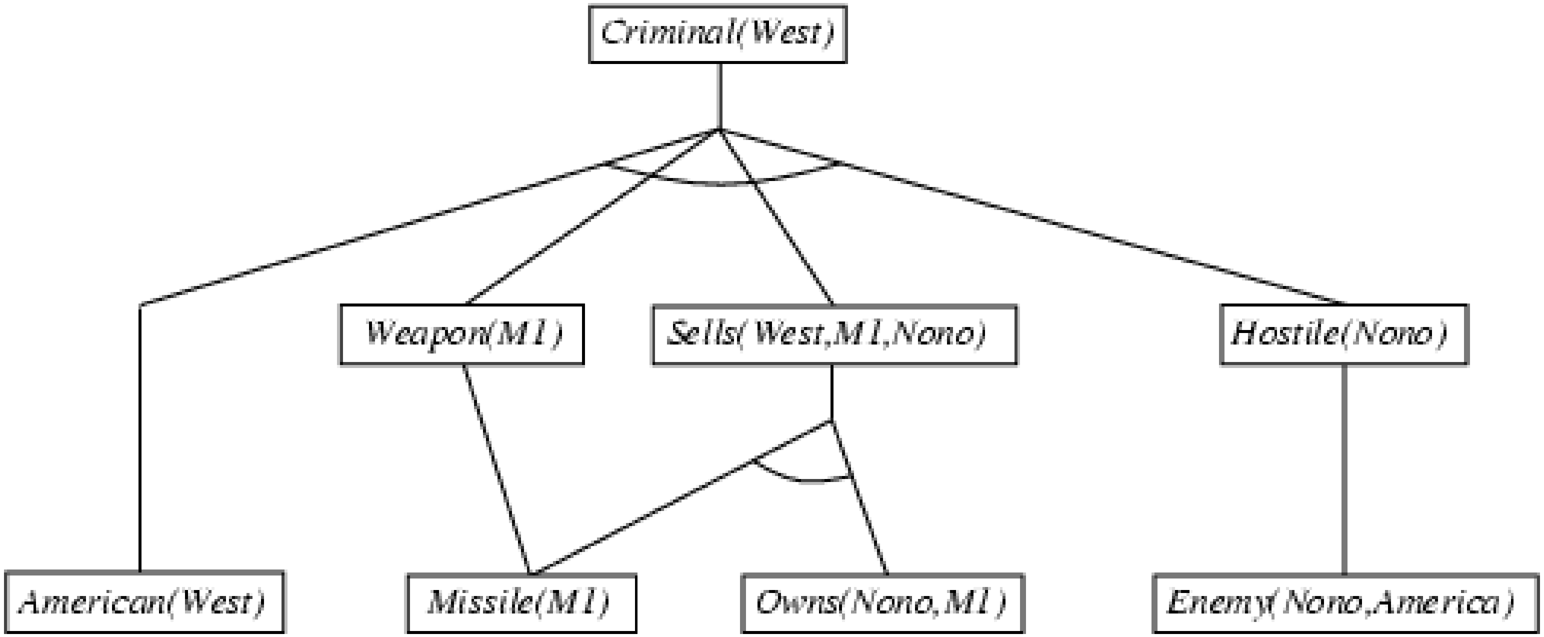
The country Nono, an enemy of America …

Enemy(Nono,America)

Resolution proof:

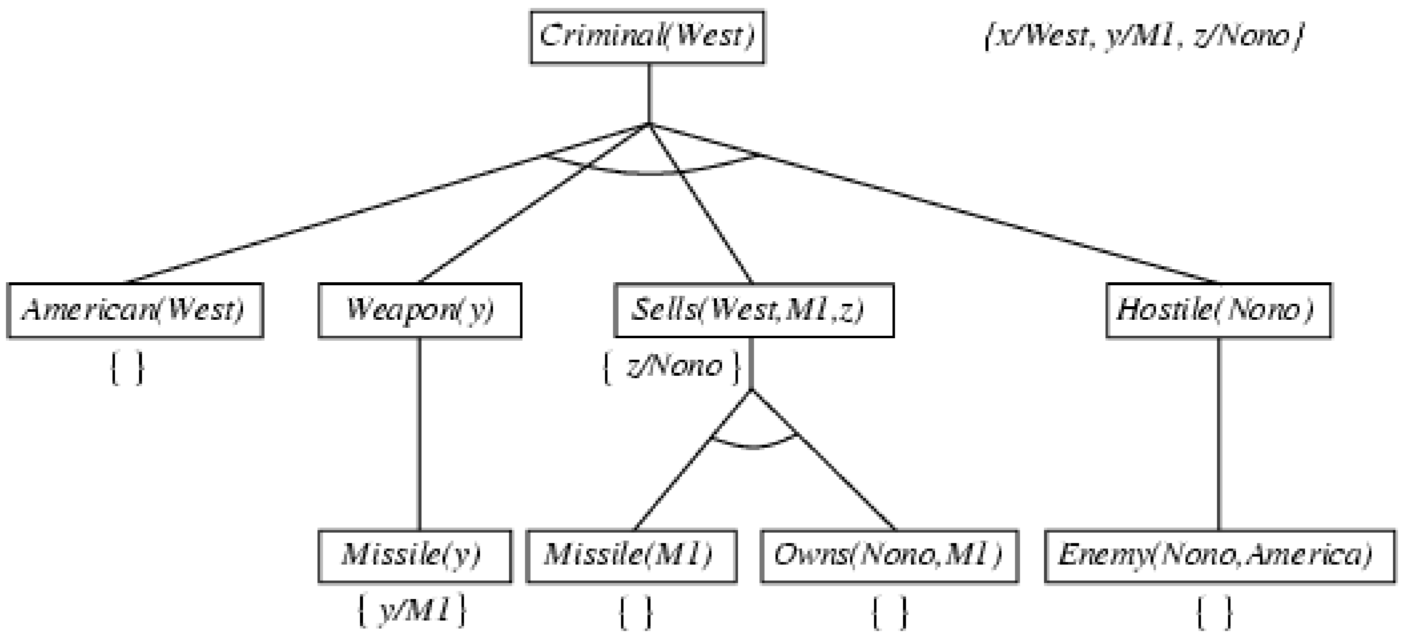


Forward chaining proof (Horn clauses)



*American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x,y,z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x)
*Owns(Nono,M1) and Missile(M1)
*Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)
*Missile(x) $\Rightarrow$ Weapon(x)
*Enemy(x,America) $\Rightarrow$ Hostile(x)
*American(West)
*Enemy(Nono,America)

Backward chaining example (Horn clauses)

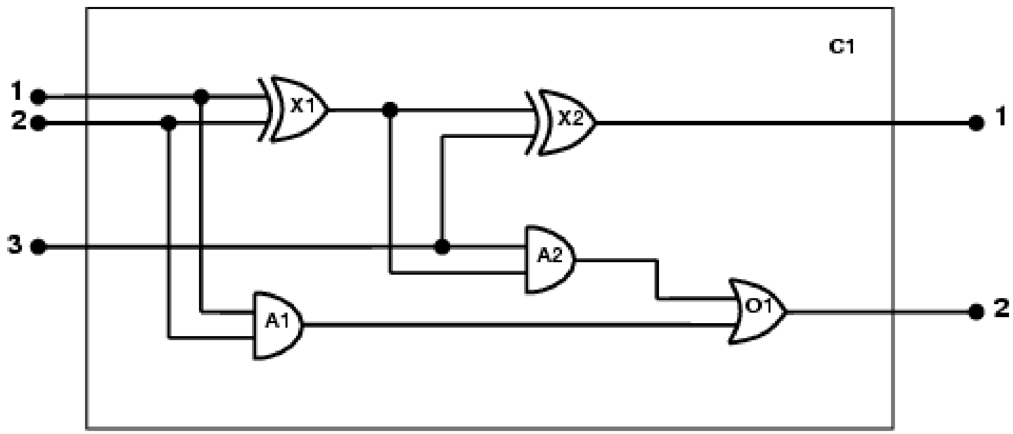


Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder



Possible queries:

- does the circuit function properly?
- what gates are connected to the first input terminal?
- what would happen if one of the gates is broken?

and so on

The electronic circuits domain

1. Identify the task
   – Does the circuit actually add properly?

2. Assemble the relevant knowledge
   – Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   –
   – Irrelevant: size, shape, color, cost of gates
   –

3. Decide on a vocabulary
   – Alternatives:
   –

   Type(X_1) = XOR (function)
   Type(X_1, XOR) (binary predicate)
   XOR(X_1)
   (unary predicate)

The electronic circuits domain

4. Encode general knowledge of the domain
   - $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Signal(t_1) = Signal(t_2)
   - $\forall t$ Signal(t) = 1 $\vee$ Signal(t) = 0
   - $1 \neq 0$
   - $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Connected(t_2, t_1)
   - $\forall g$ Type(g) = OR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 1
   - $\forall g$ Type(g) = AND $\Rightarrow$ Signal(Out(1,g)) = 0 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 0
   - $\forall g$ Type(g) = XOR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ Signal(In(1,g)) $\neq$ Signal(In(2,g))
   - $\forall g$ Type(g) = NOT $\Rightarrow$ Signal(Out(1,g)) $\neq$ Signal(In(1,g))

The electronic circuits domain

5. Encode the specific problem instance

$Type(X_1) = XOR$
$Type(A_1) = AND$
$Type(O_1) = OR$

$Type(X_2) = XOR$
$Type(A_2) = AND$

$Connected(Out(1,X_1),In(1,X_2))$
$Connected(Out(1,X_1),In(2,A_2))$
$Connected(Out(1,A_2),In(1,O_1))$
$Connected(Out(1,A_1),In(2,O_1))$
$Connected(Out(1,X_2),Out(1,C_1))$
$Connected(Out(1,O_1),Out(2,C_1))$

$Connected(In(1,C_1),In(1,X_1))$
$Connected(In(1,C_1),In(1,A_1))$
$Connected(In(2,C_1),In(2,X_1))$
$Connected(In(2,C_1),In(2,A_1))$
$Connected(In(3,C_1),In(2,X_2))$
$Connected(In(3,C_1),In(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure:

What are the possible sets of values of all the terminals for the adder circuit?

$$\exists i_1, i_2, i_3, o_1, o_2 \; Signal(In(1,C_1)) = i_1 \land Signal(In(2,C_1)) = i_2 \land Signal(In(3,C_1)) = i_3 \land Signal(Out(1,C_1)) = o_1 \land Signal(Out(2,C_1)) = o_2$$

7. Debug the knowledge base

May have omitted assertions like $1 \neq 0$

Review Probability
Chapter 13

- Basic probability notation/definitions:
  - Probability model, unconditional/prior and conditional/posterior probabilities, factored representation (= variable/value pairs), random variable, (joint) probability distribution, probability density function (pdf), marginal probability, (conditional) independence, normalization, etc.
- Basic probability formulae:
  - Probability axioms, sum rule, product rule, Bayes' rule.
- How to use Bayes' rule:
  - Naïve Bayes model (naïve Bayes classifier)

Syntax

- Basic element: random variable
- Similar to propositional logic: possible worlds defined by assignment of values to random variables.

- Booleanrandom variables

  e.g., *Cavity (= do I have a cavity?)*
- Discreterandom variables

  e.g., *Weather is one of <sunny,rainy,cloudy,snow>*
- Domain values must be exhaustive and mutually exclusive

- Elementary proposition is an assignment of a value to a random variable:

  e.g., *Weather = sunny; Cavity = false(abbreviated as ¬cavity)*

- Complex propositions formed from elementary propositions and standard logical connectives :

  e.g., *Weather = sunny* $\lor$ *Cavity = false*

Probability

- P(a) is the probability of proposition "a"
  - e.g., P(it will rain in London tomorrow)
  - The proposition a is actually true or false in the real-world

- **<u>Probability Axioms</u>**:
  - $0 \leq P(a) \leq 1$
  - $P(NOT(a)) = 1 - P(a)$ => $\Sigma_A P(A) = 1$
  - $P(true) = 1$
  - $P(false) = 0$
  - $P(A \text{ OR } B) = P(A) + P(B) - P(A \text{ AND } B)$

- Any agent that holds degrees of beliefs that contradict these axioms will act irrationally in some cases

- **Rational agents <u>cannot</u> violate probability theory.**
  - Acting otherwise results in irrational behavior.

Conditional Probability

- P(a|b) is the conditional probability of proposition a, conditioned on knowing that b is true,
  - E.g., P(rain in London tomorrow | raining in London today)
  - P(a|b) is a "posterior" or conditional probability
  - The updated probability that a is true, now that we know b
  - $P(a|b) = P(a \wedge b) / P(b)$
  - Syntax: P(a | b) is the probability of a given that b is true
    - a and b can be any propositional sentences
    - e.g., p(John wins OR Mary wins | Bob wins AND Jack loses)

- P(a|b) obeys the same rules as probabilities,
  - E.g., P(a | b) + P(NOT(a) | b) = 1
  - All probabilities in effect are conditional probabilities
    - E.g., P(a) = P(a | our background knowledge)

Concepts of Probability

- **Unconditional Probability**
  - **P(a)**, the probability of "a" being true, or **P(a=True)**
  - Does not depend on anything else to be true (**unconditional**)
  - Represents the probability prior to further information that may adjust it (**prior**)

- **Conditional Probability**
  - **P(a|b)**, the probability of "a" being true, given that "b" is true
  - Relies on "b" = true (**conditional**)
  - Represents the prior probability adjusted based upon new information "b" (**posterior**)
  - Can be generalized to more than 2 random variables:
    - e.g. P(a|b, c, d)

- **Joint Probability**
  - **P(a, b) = P(a ∧ b)**, the probability of "a" and "b" both being true
  - Can be generalized to more than 2 random variables:
    - e.g. P(a, b, c, d)

Basic Probability Relationships

- **$P(A) + P(\neg A) = 1$**
  - Implies that $P(\neg A) = 1 - P(A)$
- **$P(A, B) = P(A \wedge B) = P(A) + P(B) - P(A \vee B)$**
  - Implies that $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- **$P(A \mid B) = P(A, B) / P(B)$**
  - Conditional probability; "Probability of A **given** B"
- **$P(A, B) = P(A \mid B)\, P(B)$**
  - Product Rule (Factoring); applies to any number of variables
  - $P(a, b, c, \ldots z) = P(a \mid b, c, \ldots z)\, P(b \mid c, \ldots z)\, P(c \mid \ldots z) \ldots P(z)$
- **$P(A) = \sum_{B,C} P(A, B, C) = \sum_{b \in B, c \in C} P(A, b, c)$**
  - Sum Rule (Marginal Probabilities); for any number of variables
  - $P(A, D) = \sum_{B} \sum_{C} P(A, B, C, D) = \sum_{b \in \mathbf{B}} \sum_{c \in \mathbf{C}} P(A, b, c, D)$
- **$P(B \mid A) = P(A \mid B)\, P(B) / P(A)$**
  - Bayes' Rule; for any number of variables

You need to know these !

Summary of Probability Rules

- **Product Rule**:
  – **P(a, b) = P(a|b) P(b) = P(b|a) P(a)**
  – Probability of "a" and "b" occurring is the same as probability of "a" occurring given "b" is true, times the probability of "b" occurring.
    - e.g., *P(rain, cloudy) = P(rain | cloudy) * P(cloudy)*

- **Sum Rule**: (AKA **Law of Total Probability**)
  – **P(a) = Σ_b P(a, b) = Σ_b P(a|b) P(b),** where B is any random variable
  – Probability of "a" occurring is the same as the sum of all joint probabilities including the event, provided the joint probabilities represent all possible events.
  – Can be used to "marginalize" out other variables from probabilities, resulting in prior probabilities also being called marginal probabilities.
    - e.g., *P(rain) = $\Sigma_{Windspeed}$ P(rain, Windspeed)*
      where Windspeed = {0-10mph, 10-20mph, 20-30mph, etc.}

- **Bayes' Rule**:
  - **P(b|a) = P(a|b) P(b) / P(a)**
  - Acquired from rearranging the product rule.
  - Allows conversion between conditionals, from P(a|b) to P(b|a).
    - e.g., b = disease, a = symptoms
      More natural to encode knowledge as P(a|b) than as P(b|a).

Full Joint Distribution

- We can fully specify a probability space by constructing a **full joint distribution**:
  - A full joint distribution contains a probability for every possible combination of variable values.
  - E.g., P(J=f, M=t, A=t, B=t, E=f)

- From a full joint distribution, the product rule, sum rule, and Bayes' rule can create any desired joint and conditional probabilities.

Computing with Probabilities: Law of Total Probability

Law of Total Probability (aka "summing out" or marginalization)

$P(a) = \Sigma_b \; P(a, b)$

$= \Sigma_b \; P(a \mid b) \, P(b)$ where B is any random variable

Why is this useful?

Given a joint distribution (e.g., P(a,b,c,d)) we can obtain any "marginal" probability (e.g., P(b)) by summing out the other variables, e.g.,

$$P(b) = \Sigma_a \, \Sigma_c \, \Sigma_d \; P(a, b, c, d)$$

We can compute <u>any conditional probability</u> given a joint distribution, e.g.,

$$P(c \mid b) = \Sigma_a \, \Sigma_d \, P(a, c, d \mid b)$$

$$= \Sigma_a \, \Sigma_d \, P(a, c, d, b) \,/\, P(b)$$

where P(b) can be computed as above

Computing with Probabilities: The Chain Rule or Factoring

We can always write

$P(a, b, c, \ldots z) = P(a \mid b, c, \ldots. z) P(b, c, \ldots z)$

(by definition of joint probability)

Repeatedly applying this idea, we can write

$P(a, b, c, \ldots z) = P(a \mid b, c, \ldots. z) P(b \mid c,.. z) P(c \mid .. z)..P(z)$

This factorization holds for any ordering of the variables

This is the chain rule for probabilities

Independence

- Formal Definition:
  - 2 random variables A and B are independent iff:

    P(a, b) = P(a) P(b), for all values a, b

- Informal Definition:
  - 2 random variables A and B are independent iff:

    P(a | b) = P(a) OR P(b | a) = P(b), for all values a, b

  - P(a | b) = P(a) tells us that knowing b provides no change in our probability for a, and thus b contains no information about a.

- Also known as marginal independence, as all other variables have been marginalized out.

- In practice true independence is very rare:
  - "butterfly in China" effect
  - Conditional independence is much more common and useful

Conditional Independence

- Formal Definition:
  - 2 random variables A and B are conditionally independent given C iff:

    P(a, b|c) = P(a|c) P(b|c),   for all values a, b, c

- Informal Definition:
  - 2 random variables A and B are conditionally independent given C iff:

    P(a|b, c) = P(a|c)   OR   P(b|a, c) = P(b|c),   for all values a, b, c
  - P(a|b, c) = P(a|c) tells us that learning about b, given that we already know c, provides no change in our probability for a, and thus b contains no information about a beyond what c provides.

- Naïve Bayes Model:
  - Often a single variable can directly influence a number of other variables, all of which are conditionally independent, given the single variable.
  - E.g., k different symptom variables X_1, X_2, … X_k, and C = disease, reducing to:

    $$\mathbf{P(X_1, X_2, \ldots. X_K \mid C) = P(C) \prod P(X_i \mid C)}$$

Examples of Conditional Independence

- **H=Heat, S=Smoke, F=Fire**
  - P(H, S | F) = P(H | F) P(S | F)
  - P(S | F, S) = P(S | F)
  - If we know there is/is not a fire, observing heat tells us no more information about smoke
- **F=Fever, R=RedSpots, M=Measles**
  - P(F, R | M) = P(F | M) P(R | M)
  - P(R | M, F) = P(R | M)
  - If we know we do/don't have measles, observing fever tells us no more information about red spots
- **C=SharpClaws, F=SharpFangs, S=Species**
  - P(C, F | S) = P(C | S) P(F | S)
  - P(F | S, C) = P(F | S)
  - If we know the species, observing sharp claws tells us no more information about sharp fangs

Review Bayesian Networks
Chapter 14.1-5

- **Basic concepts and vocabulary of Bayesian networks.**
  - Nodes represent random variables.
  - Directed arcs represent (informally) direct influences.
  - Conditional probability tables, P(Xi | Parents(Xi)).

- **Given a Bayesian network:**
  - Write down the full joint distribution it represents.

- **Given a full joint distribution in factored form:**
  - Draw the Bayesian network that represents it.

- **Given a variable ordering and background assertions of conditional independence among the variables:**
  - Write down the factored form of the full joint distribution, as simplified by the conditional independence assertions.
- **Use the network to find answers to probability questions about it.**

Bayesian Networks

- Represent dependence/independence via a directed graph
  - Nodes = random variables
  - Edges = direct dependence
- Structure of the graph ⇔ Conditional independence

- Recall the chain rule of repeated conditioning:

$$P(X_1, X_2, X_3 ..., X_N) = P(X_1|X_2, X_3 ..., X_N)P(X_2|X_3, ..., X_N) \cdots P(X_N)$$

$$P(X_1, X_2, X_3 ..., X_N) = \prod_{i=1}^{n} P(X_i|parents(X_i))$$

The full joint distribution — The graph-structured approximation

- Requires that graph is acyclic (no directed cycles)
- 2 components to a Bayesian network
  - The graph structure (conditional independence assumptions)
  - The numerical probabilities (of each variable given its parents)

Bayesian Network

- A Bayesian network specifies a joint distribution in a structured form:

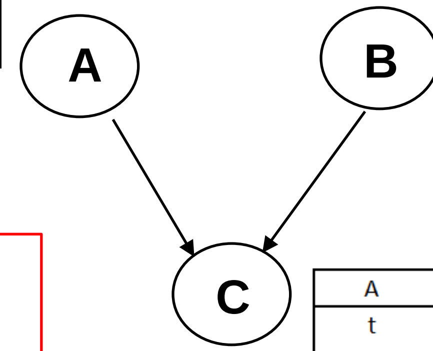
Full factorization

$$p(A,B,C) = p(C|A,B)p(A|B)p(B)$$
$$= p(C|A,B)p(A)p(B)$$

After applying conditional independence from the graph

P(A)
0.33

P(B)
0.67



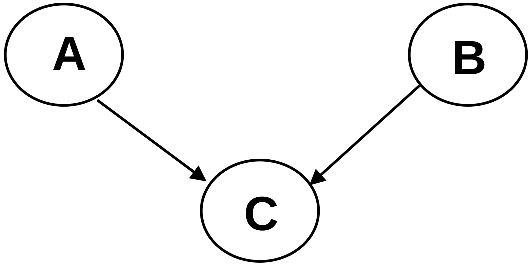
A	B	P(C)
t	t	0.2
t	f	0.4
f	t	0.3
f	f	0.3

- Dependence/independence represented via a directed graph:
  - Node = random variable
  - Directed Edge = conditional dependence
  - Absence of Edge = conditional independence

- Allows concise view of joint distribution relationships:
  - Graph nodes and edges show conditional relationships between variables.
  - Tables provide probability data.

Examples of 3-way Bayesian Networks

Independent Causes
A Earthquake
B Burglary
C Alarm



Independent Causes:
p(A,B,C) = p(C|A,B)p(A)p(B)

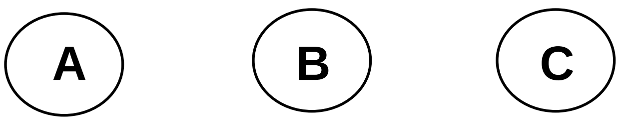
"Explaining away" effect:
Given C, observing A makes B less likely
e.g., earthquake/burglary/alarm example

A and B are (marginally) independent
but become dependent once C is known

You heard alarm, and observe Earthquake
…. It explains away burglary

Nodes: Random Variables
A, B, C
Edges: P(Xi | Parents) → Directed edge from parent nodes to Xi
A → C
B → C

Examples of 3-way Bayesian Networks

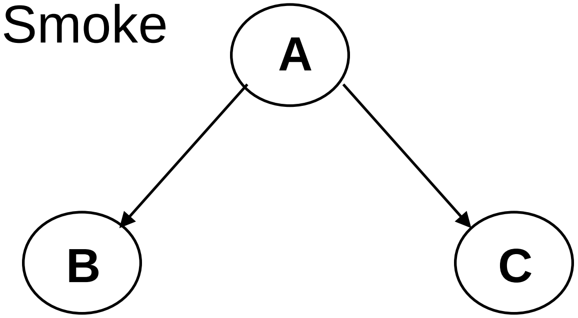


Marginal Independence:
p(A,B,C) = p(A) p(B) p(C)

Nodes: Random Variables
A, B, C
Edges: P(Xi | Parents) → Directed edge from parent nodes to Xi
No Edge!

Extended example of 3-way Bayesian Networks

Common Cause
A : Fire
B: Heat
C: Smoke



Conditionally independent effects:
$p(A,B,C) = p(B|A)p(C|A)p(A)$

B and C are conditionally independent Given A

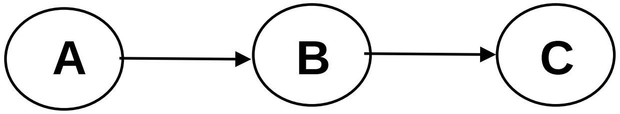
"Where there's Smoke, there's Fire."

If we see Smoke, we can infer Fire.

If we see Smoke, observing Heat tells us very little additional information.

Examples of 3-way Bayesian Networks

Markov Dependence
A Rain on Mon
B Ran on Tue
C Rain on Wed



Markov dependence:
p(A,B,C) = p(C|B) p(B|A)p(A)

A affects B and B affects C
Given B, A and C are independent

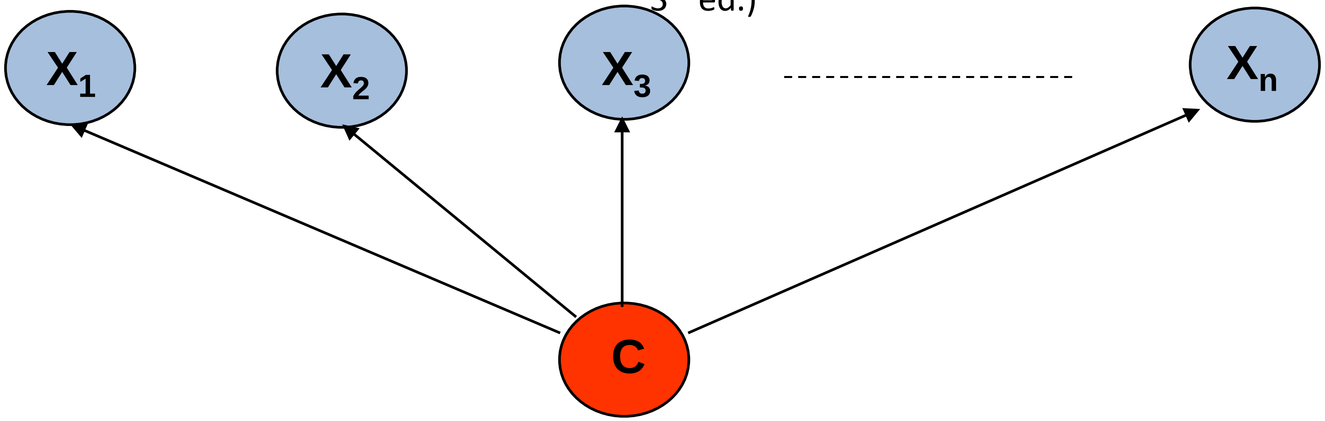
e.g.
If it rains today, it will rain tomorrow with 90%

On Wed morning…
If you know it rained yesterday,
it doesn't matter whether it rained on Mon

Nodes: Random Variables
A, B, C
Edges: P(Xi | Parents) → Directed edge from parent nodes to Xi
A → B
B → C

Naïve Bayes Model

(section 20.2.2 R&N 3rd ed.)



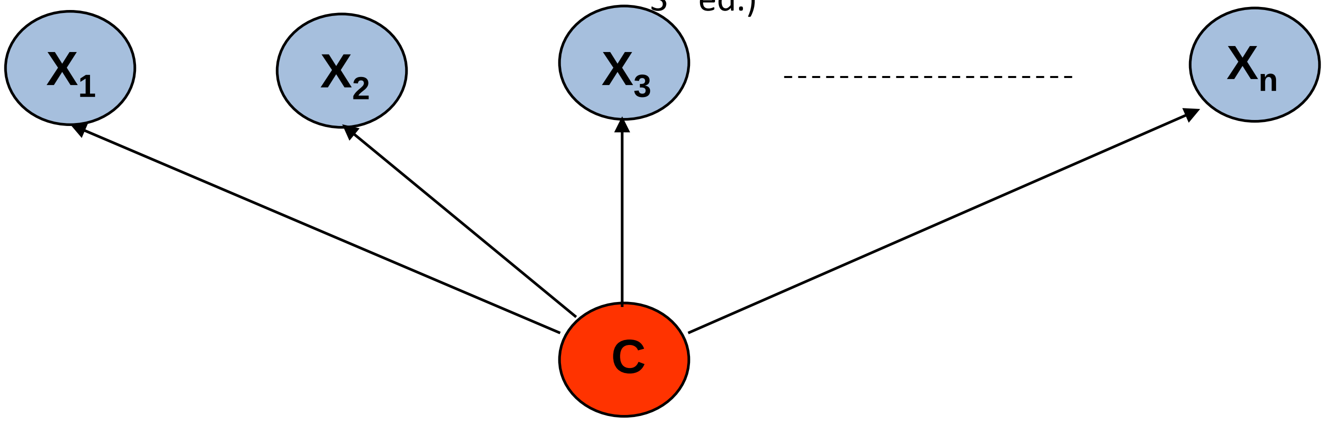
Basic Idea: We want to estimate $P(C \mid X_1,\ldots X_n)$, but it's hard to think about computing the probability of a class from input attributes of an example.

Solution: Use Bayes' Rule to turn $P(C \mid X_1,\ldots X_n)$ into a proportionally equivalent expression that involves only $P(C)$ and $P(X_1,\ldots X_n \mid C)$. Then assume that feature values are conditionally independent given class, which allows us to turn $P(X_1,\ldots X_n \mid C)$ into $\Pi_i\ P(X_i \mid C)$.

We estimate $P(C)$ easily from the frequency with which each class appears within our training data, and we estimate $P(X_i \mid C)$ easily from the frequency with which each X_i appears in each class C within our training data.

Naïve Bayes Model

(section 20.2.2 R&N 3rd ed.)



Bayes Rule: $P(C \mid X_1, \ldots X_n)$ is proportional to $P(C) \; \Pi_i \; P(X_i \mid C)$
[note: denominator $P(X_1, \ldots X_n)$ is constant for all classes, may be ignored.]

Features Xi are conditionally independent given the class variable C

- choose the class value c_i with the highest $P(c_i \mid x_1, \ldots, x_n)$
- simple to implement, often works very well
- e.g., spam email classification: X's = counts of words in emails

Conditional probabilities $P(X_i \mid C)$ can easily be estimated from labeled date

- Problem: Need to avoid zeroes, e.g., from limited training data
- Solutions: Pseudo-counts, beta[a,b] distribution, etc.

Naïve Bayes Model (2)

$$P(C \mid X_1,\ldots X_n) = \alpha \; P(C) \; \Pi_i \; P(X_i \mid C)$$

Probabilities $P(C)$ and $P(X_i \mid C)$ can easily be estimated from labeled data

$P(C = c_j) \approx$ #(Examples with class label $C = c_j$) / #(Examples)

$P(X_i = x_{ik} \mid C = c_j)$
≈ #(Examples with attribute value $X_i = x_{ik}$ and class label $C = c_j$)
/ #(Examples with class label $C = c_j$)

Usually easiest to work with logs

$$\log [\, P(C \mid X_1,\ldots X_n) \,] = \log \alpha + \log P(C) + \Sigma \log P(X_i \mid C)$$

DANGER: What if ZERO examples with value $X_i = x_{ik}$ and class label $C = c_j$?
An unseen example with value $X_i = x_{ik}$ will NEVER predict class label $C = c_j$!

Practical solutions: Pseudocounts, e.g., add 1 to every #() , etc.
Theoretical solutions: Bayesian inference, beta distribution, etc.

Bigger Example

- Consider the following 5 binary variables:
  - B = a burglary occurs at your house
  - E = an earthquake occurs at your house
  - A = the alarm goes off
  - J = John calls to report the alarm
  - M = Mary calls to report the alarm
- Sample Query: What is P(B|M, J) ?
- Using full joint distribution to answer this question requires
  - 2^5 - 1= 31 parameters
- Can we use prior domain knowledge to come up with a Bayesian network that requires fewer probabilities?

Constructing a Bayesian Network: Step 1

- Order the variables in terms of influence (may be a partial order)

  e.g., {E, B} -> {A} -> {J, M}

  Generally, order variables to reflect the assumed causal relationships.

- Now, apply the chain rule, and simplify based on assumptions

- P(J, M, A, E, B) = P(J, M | A, E, B) P(A| E, B) P(E, B)

  ≈ P(J, M | A) P(A| E, B) P(E) P(B)

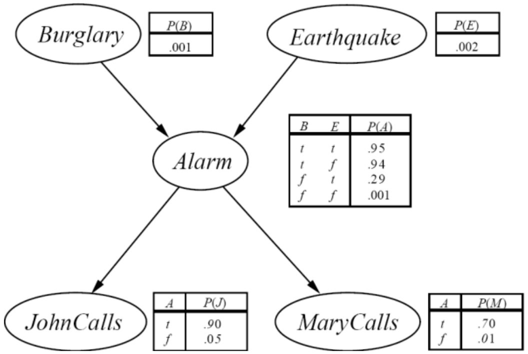
  ≈ P(J | A) P(M | A) P(A| E, B) P(E) P(B)

These conditional independence assumptions are reflected in the graph structure of the Bayesian network

Constructing this Bayesian Network: Step 2

- P(J, M, A, E, B) =
  P(J | A) P(M | A) P(A | E, B) P(E) P(B)

  <u>**Parents in the graph ⇔ conditioning variables (RHS)**</u>

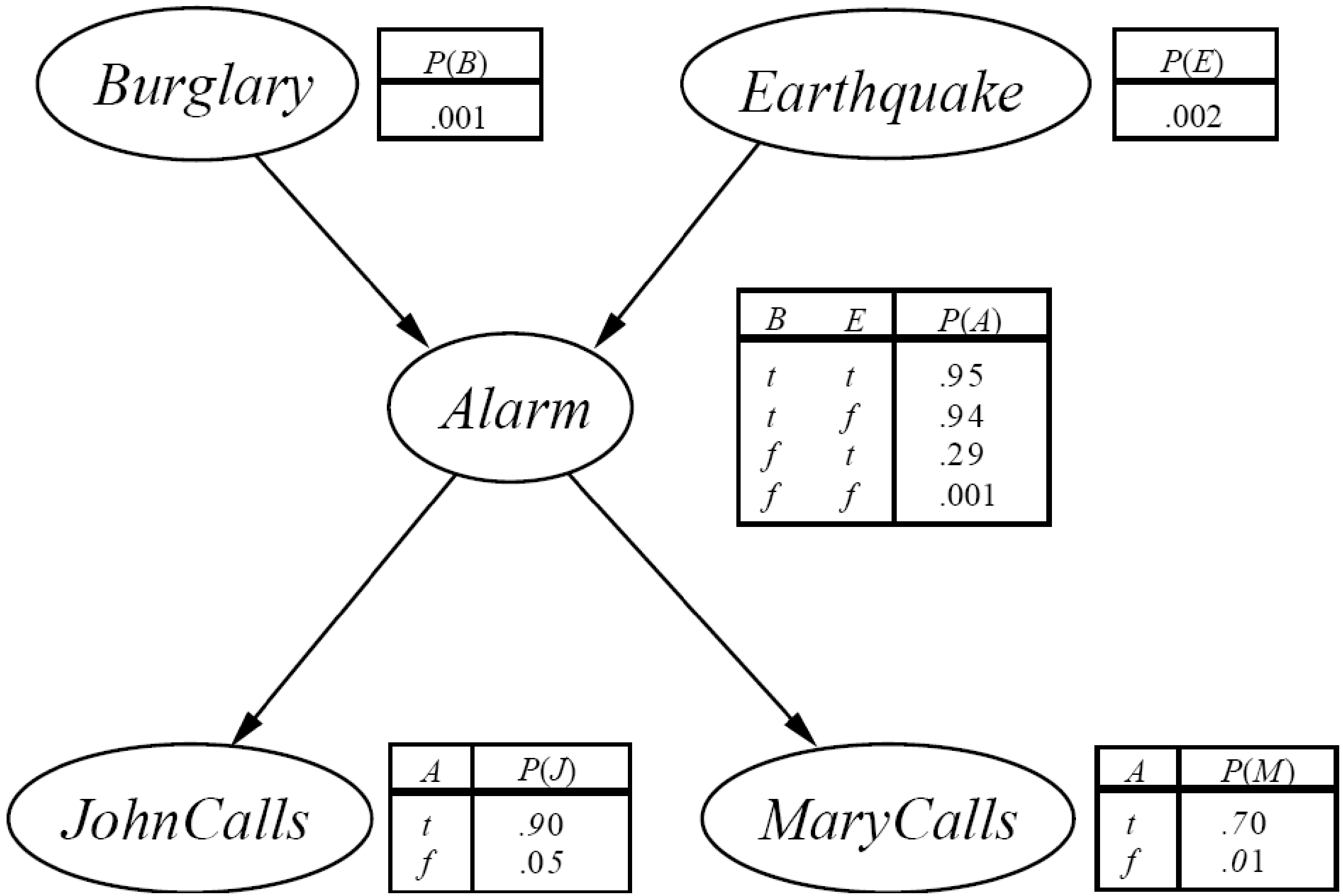


- There are 3 conditional probability tables (CPDs) to be determined: P(J | A), P(M | A), P(A | E, B)
  – Requiring 2 + 2 + 4 = 8 probabilities

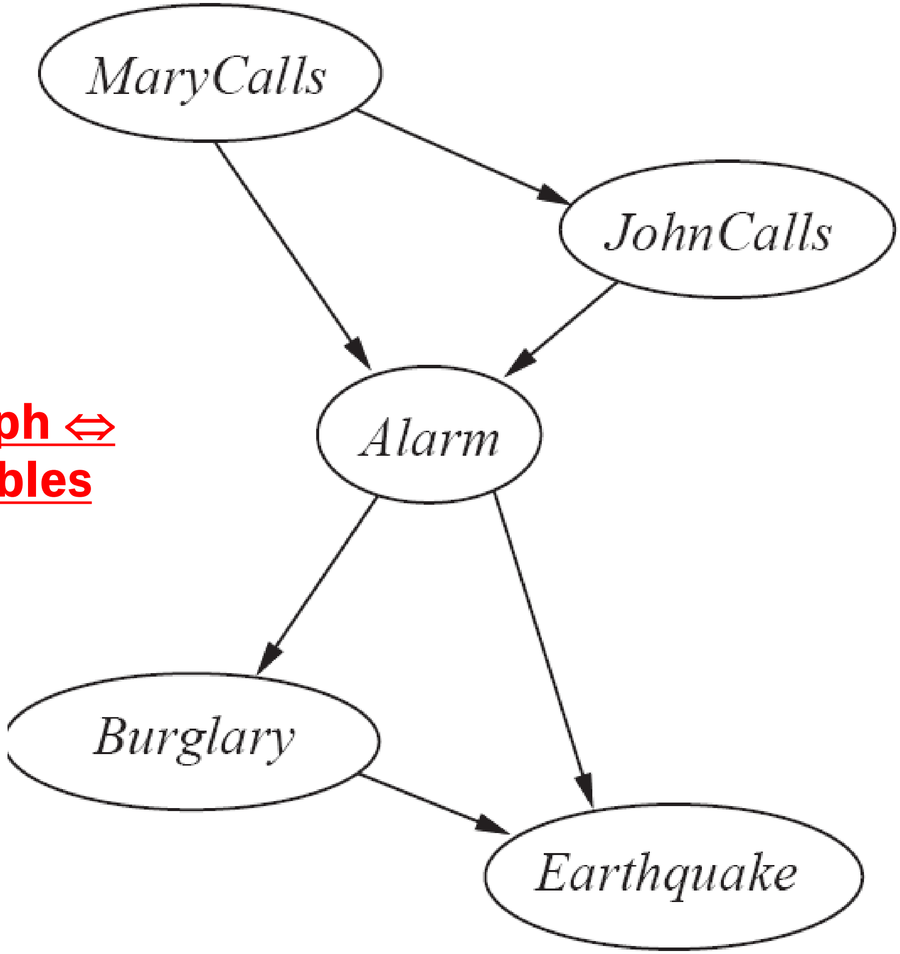
- And 2 marginal probabilities P(E), P(B) -> 2 more probabilities

- Where do these probabilities come from?
  – Expert knowledge
  – From data (relative frequency estimates)
  – Or a combination of both - see discussion in Section 20.1 and 20.2 (optional)

The Resulting Bayesian Network



The Bayesian Network From a Different Variable Ordering



Parents in the graph ⇔ conditioning variables (RHS)

$$P(J, M, A, E, B) = P(E \mid A, B)\ P(B \mid A)\ P(A \mid M, J)\ P(J \mid M)\ P(M)$$

Generally, order variables so that resulting graph reflects assumed causal relationships.

Example of Answering a Simple Query

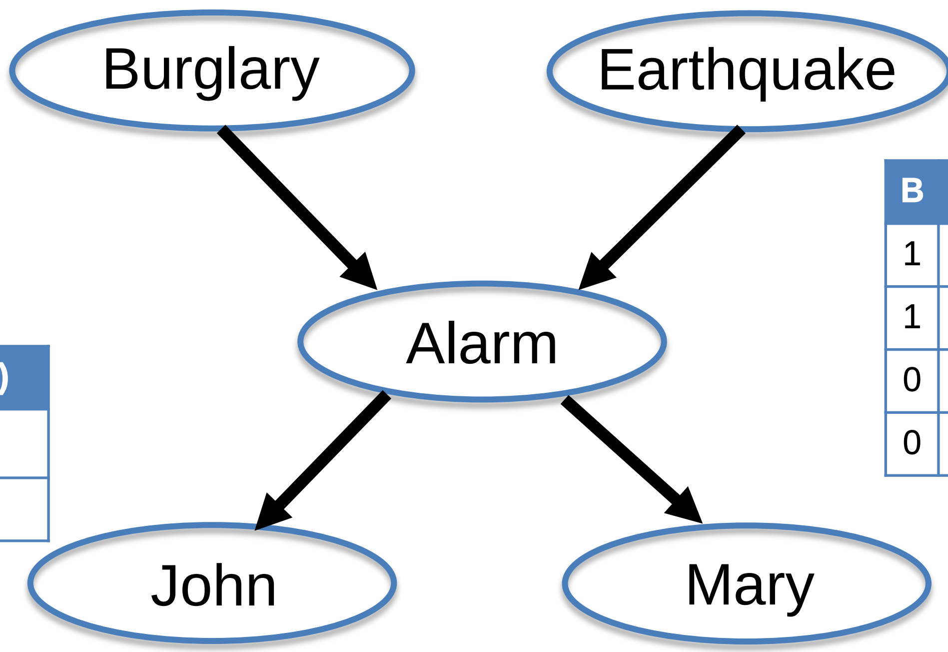
- What is $P(\neg j, m, a, \neg e, b) = P(J = \text{false} \wedge M{=}\text{true} \wedge A{=}\text{true} \wedge E{=}\text{false} \wedge B{=}\text{true})$

$P(J, M, A, E, B) \approx P(J \mid A)\, P(M \mid A)\, P(A \mid E, B)\, P(E)\, P(B)$; by conditional independence

$$P(\neg j, m, a, \neg e, b) \approx P(\neg j \mid a)\, P(m \mid a)\, P(a \mid \neg e, b)\, P(\neg e)\, P(b)$$
$$= 0.10 \times 0.70 \times 0.94 \times 0.998 \times 0.001 \approx .0000657$$

P(B)
0.001

P(E)
0.002



B	E	P(A|B,E)
1	1	0.95
1	0	0.94
0	1	0.29
0	0	0.001

A	P(J|A)
1	0.90
0	0.05

A	P(M|A)
1	0.70
0	0.01

Inference in Bayesian Networks

- **X** *= { X1, X2, …, Xk }* = **query variables** of interest
- **E** *= { E1, …, El }* = **evidence variables** that are observed
- **Y** = *{ Y1, …, Ym }* = **hidden variables** (nonevidence, nonquery)

- **What is the posterior distribution of X, given E?**
  - $\mathbf{P}(\mathbf{X} \mid \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(\mathbf{X}, \mathbf{y}, \mathbf{e})$

    Normalizing constant $\alpha = \sum_{\mathbf{x}} \sum_{\mathbf{y}} \mathbf{P}(\mathbf{X}, \mathbf{y}, \mathbf{e})$

- **What is the most likely assignment of values to X, given E?**
  - $\mathbf{argmax}_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{e}) = \mathbf{argmax}_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}, \mathbf{e})$

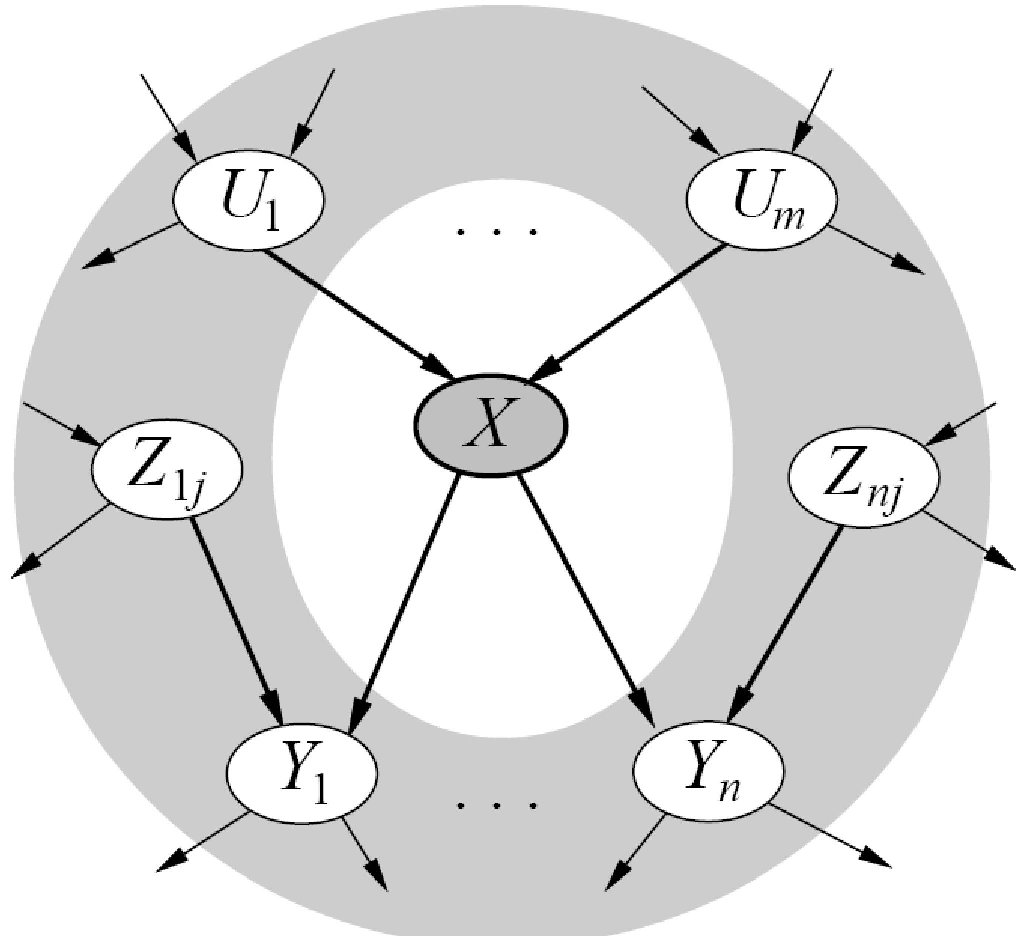
Given a graph, can we "read off" conditional independencies?

The "Markov Blanket" of X (the gray area in the figure)

X is conditionally independent of everything else, GIVEN the values of:

* X's parents
* X's children
* X's children's parents

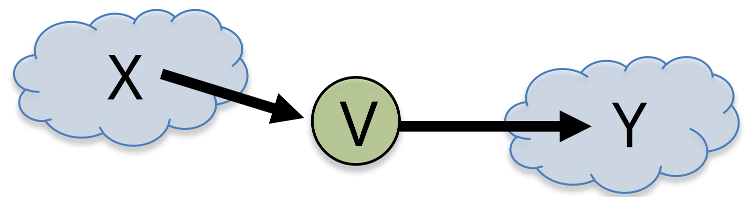
X is conditionally independent of its non-descendants, GIVEN the values of its parents.



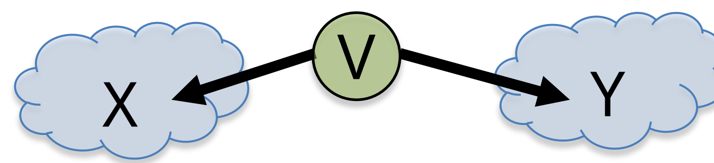
D-Separation

- Prove sets X,Y independent given Z?
- Check all *undirected* paths from X to Y
- A path is "inactive" if it passes through:

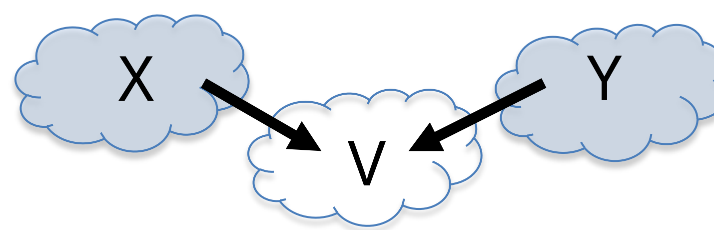
(1) A "chain" with an observed variable



(2) A "split" with an observed variable



(3) A "vee" with **only unobserved** variables below it



- If all paths are inactive, conditionally independent!

Summary

- Bayesian networks represent a joint distribution using a graph
- The graph encodes a set of conditional independence assumptions
- Answering queries (or inference or reasoning) in a Bayesian network amounts to computation of appropriate conditional probabilities
- Probabilistic inference is intractable in the general case
  - Can be done in linear time for certain classes of Bayesian networks (polytrees: at most one directed path between any two nodes)
  - Usually faster and easier than manipulating the full joint distribution

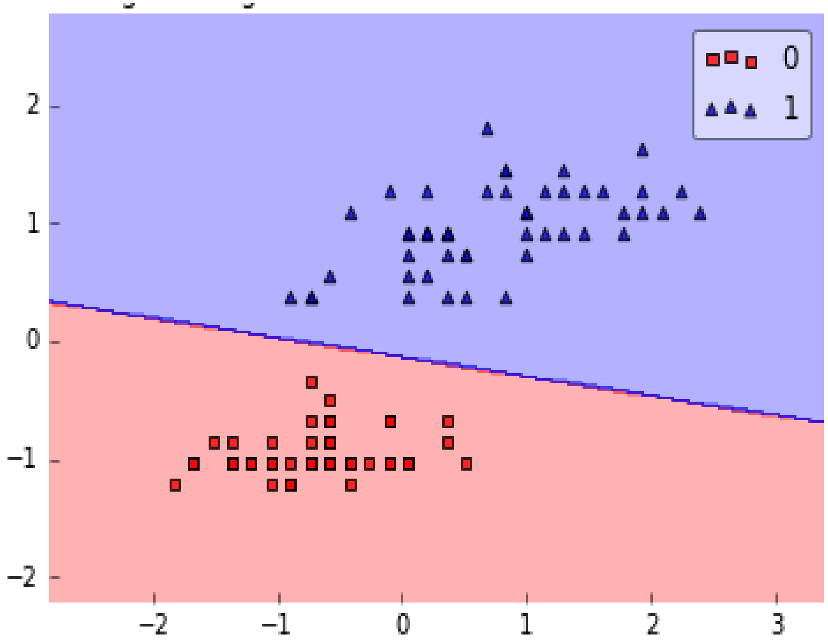
Review Intro Machine Learning
Chapter 18.1-18.4

- Understand Attributes, Target Variable, Error (loss) function, Classification & Regression, Hypothesis (Predictor) function
- What is Supervised Learning?
- Decision Tree Algorithm
- Entropy & Information Gain
- Tradeoff between train and test with model complexity
- Cross validation

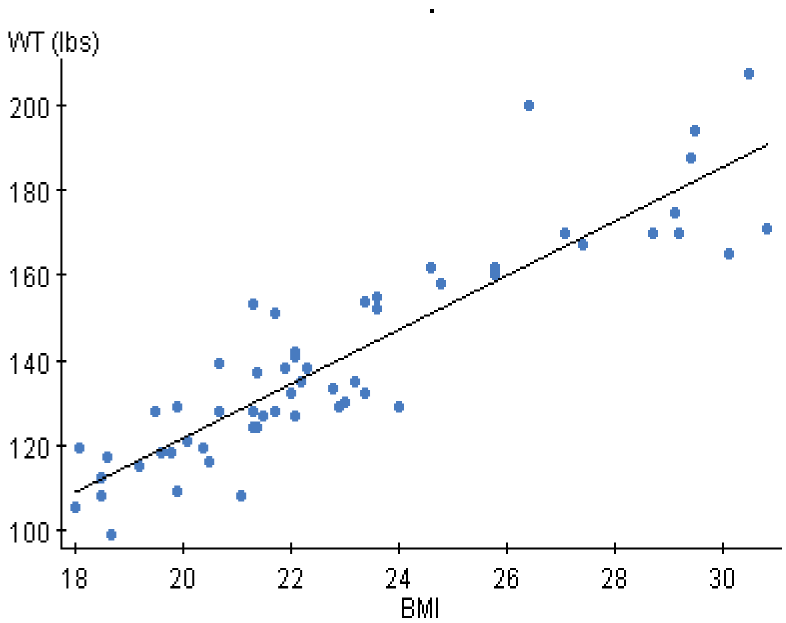
Supervised Learning

- Use supervised learning – training data is given with correct output
- We write program to reproduce this output with new test data
- Eg : face detection
- Classification : face detection, spam email
- Regression : Netflix guesses how much you will rate the movie

Classification Graph



Regression Graph



Terminology

- Attributes
  - Also known as features, variables, independent variables, covariates
- Target Variable
  - Also known as goal predicate, dependent variable, ...
- Classification
  - Also known as discrimination, supervised classification, ...
- Error function
  - Also known as objective function, loss function, ...

Inductive or Supervised learning

- Let x = input vector of attributes (feature vectors)

- Let f(x) = target label
  - The implicit mapping from x to f(x) is unknown to us
  - We only have training data pairs, D = {**x**, **f(x)**} available

- We want to learn a mapping from x to f(x)
  - Our hypothesis function is $h(x, \theta)$
  - $h(x, \theta) \approx f(x)$ for all training data points x
  - θ are the parameters of our predictor function h

- Examples:
  - $h(x, \theta) = \text{sign}(\theta_1 x_1 + \theta_2 x_2 + \theta_3)$ (perceptron)
  - $h(x, \theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ (regression)
  - $h_k(x) = (x_1 \wedge x_2) \vee (x_3 \wedge \neg x_4)$

Empirical Error Functions

- $E(h) = \Sigma_x$ distance[$h(x, \theta)$, $f(x)$]

Sum is over all training pairs in the training data D

Examples:

distance = squared error if h and f are real-valued (regression)

distance = delta-function if h and f are categorical (classification)

In learning, we get to choose

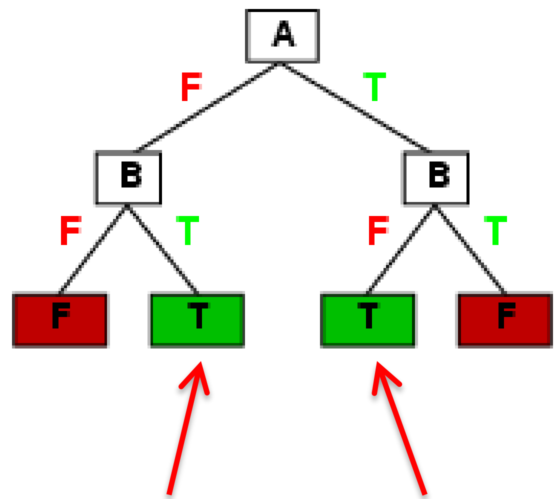
1. what class of functions h(..) we want to learn
   – potentially a huge space! ("hypothesis space")

2. what error function/distance we want to use
   - should be chosen to reflect real "loss" in problem
   - but often chosen for mathematical/algorithmic convenience

Decision Tree Representations

- Decision trees are fully expressive
  - Can represent any Boolean function (in DNF)
  - Every path in the tree could represent 1 row in the truth table
  - Might yield an exponentially large tree
    - Truth table is of size 2^d, where d is the number of attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



A xor B = $(\neg A \wedge B) \vee (A \wedge \neg B)$ in DNF

Decision Tree Representations

- Decision trees are DNF representations
  - often used in practice → often result in compact approximate representations for complex functions
  - E.g., consider a truth table where most of the variables are irrelevant to the function

- Simple DNF formulae can be easily represented
  - E.g., $f = (A \wedge B) \vee (\neg A \wedge D)$
  - DNF = disjunction of conjunctions

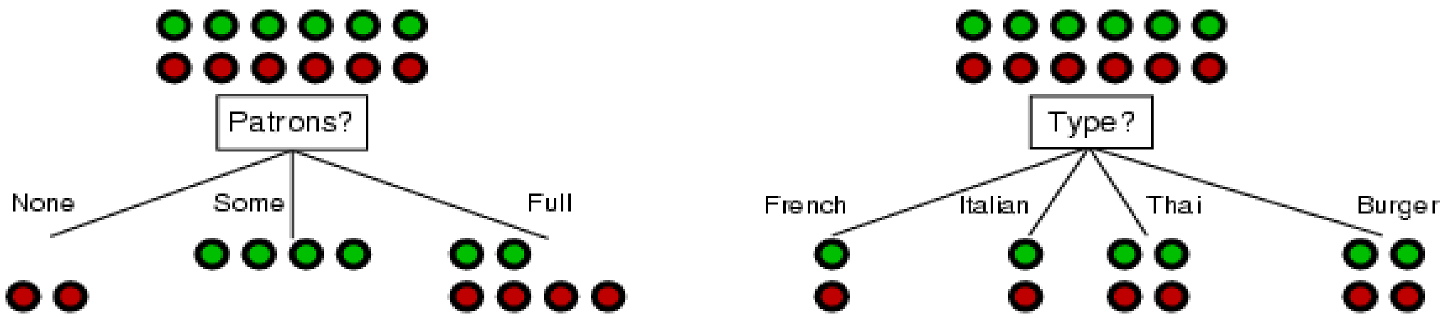
- Trees can be very inefficient for certain types of functions
  - Parity function: 1 only if an even number of 1's in the input vector
    - Trees are very inefficient at representing such functions
  - Majority function: 1 if more than ½ the inputs are 1's
    - Also inefficient

Pseudocode for Decision tree learning

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

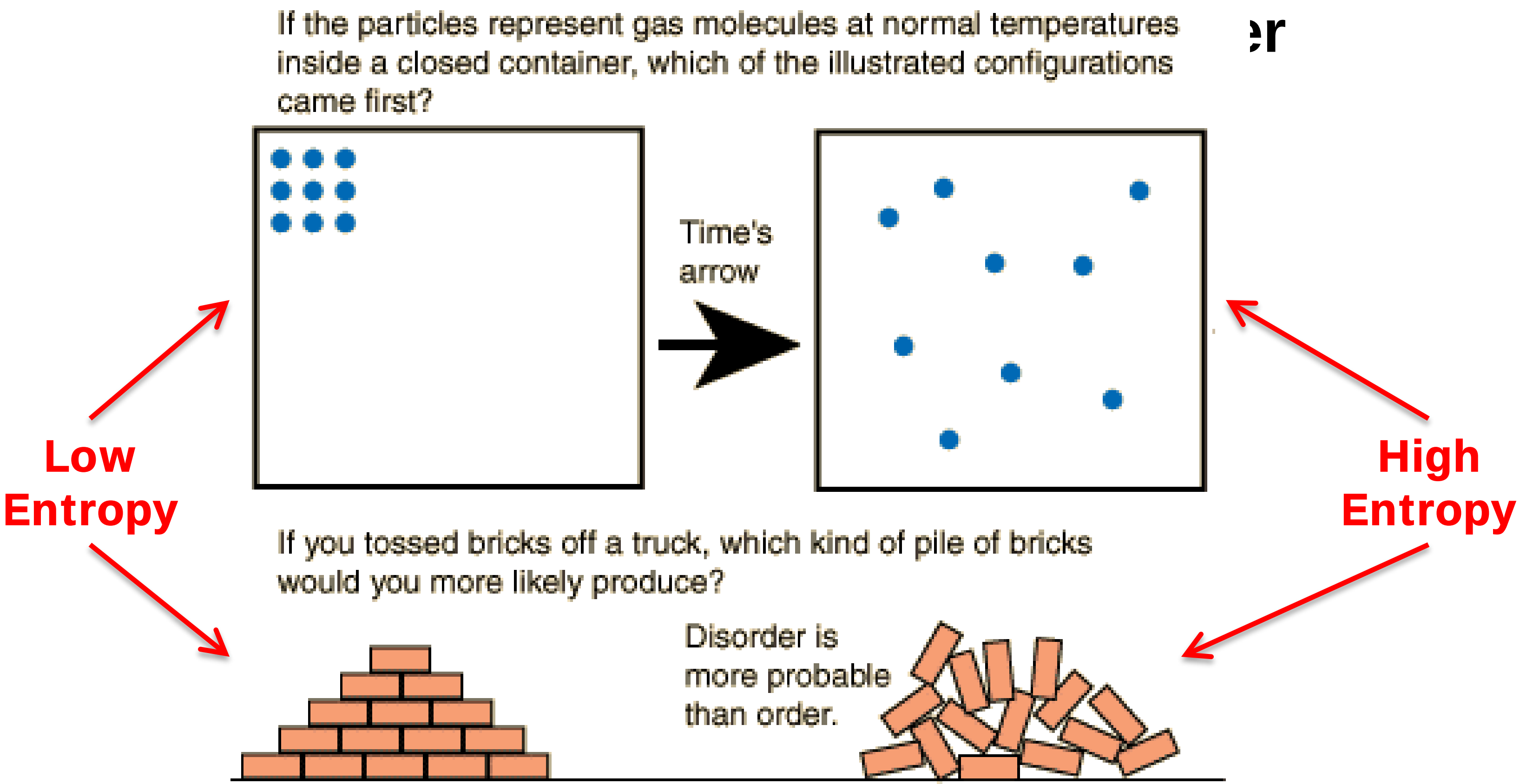


- *Patrons?* is a better choice
  - How can we quantify this?
  - One approach would be to use the classification error E directly (greedily)
    - Empirically it is found that this works poorly
  - **Much better is to use information gain (next slides)**
  - Other metrics are also used, e.g., Gini impurity, variance reduction
    - Often very similar results to information gain in practice

Entropy and Information

- "Entropy" is a measure of randomness

er



https://www.youtube.com/watch?v=ZsY4WcQOrfk

Entropy, H(p), with only 2 outcomes

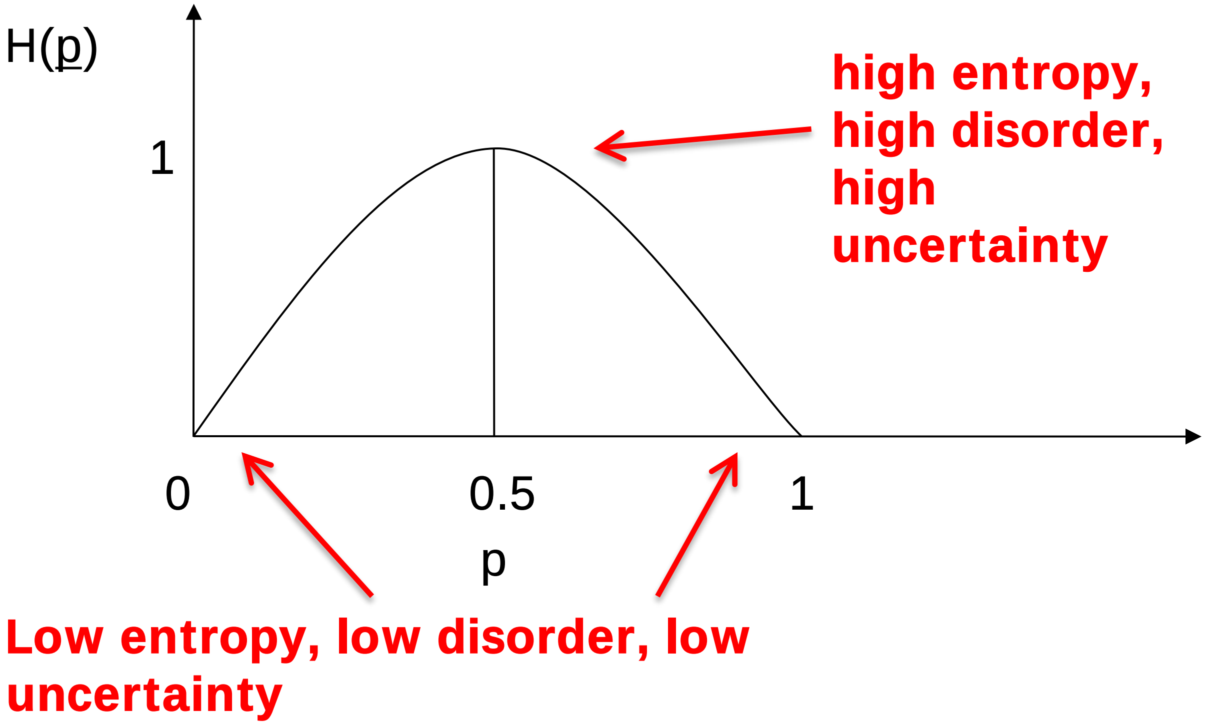
Consider 2 class problem:

p = probability of class #1,

$1 - p$ = probability of class #2

In binary case:

$$H(p) = -p \log p - (1-p) \log (1-p)$$

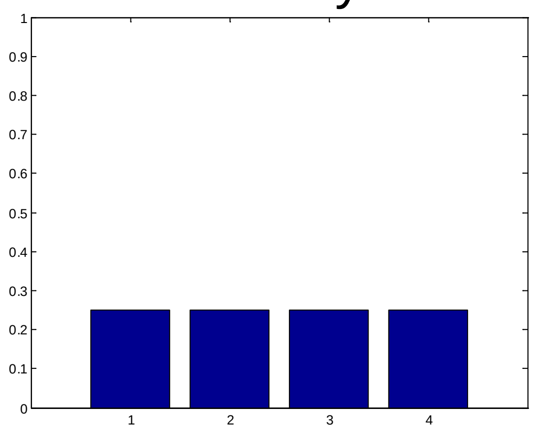


Entropy and Information

- **Entropy $H(X) = E[\log 1/P(X)] = \sum_{x \in X} P(x) \log 1/P(x)$**

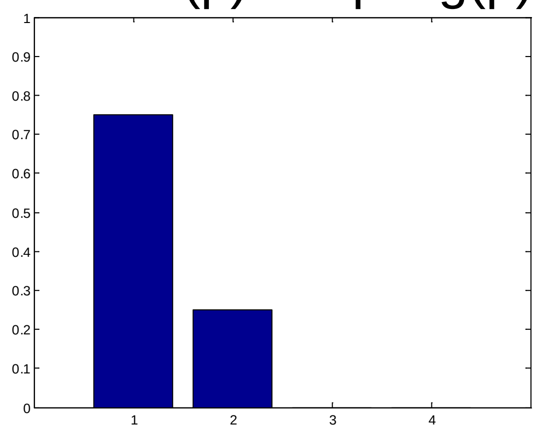
  $= -\sum_{x \in X} P(x) \log P(x)$

  - Log base two, units of entropy are "bits"
  - If only two outcomes: $H(p) = -p \log(p) - (1-p) \log(1-p)$

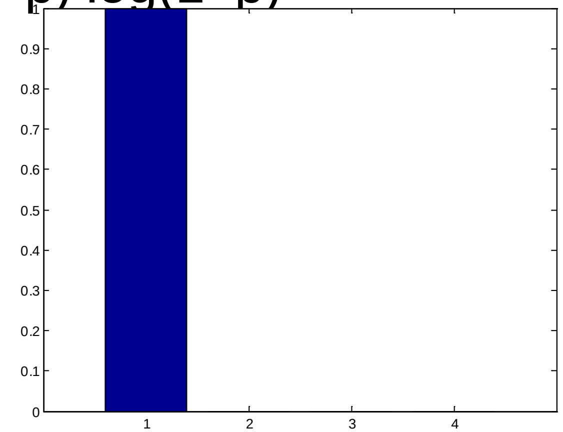


H(x) = .25 log 4 + .25 log 4 + .25 log 4 + .25 log 4 = log 4 = 2 bits

Max entropy for 4 outcomes



H(x) = .75 log 4/3 + .25 log 4 = 0.8133 bits



H(x) = 1 log 1 = 0 bits

Min entropy

Information Gain

- H(P) = current entropy of class distribution P at a particular node,

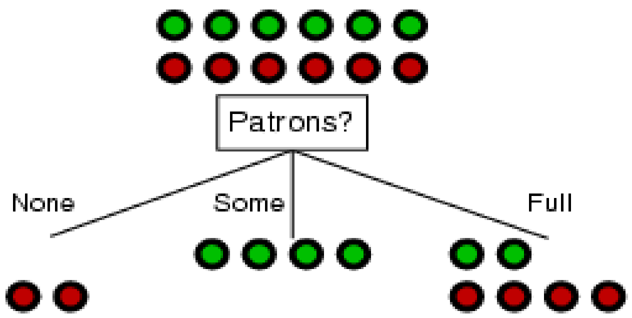
  before further partitioning the data

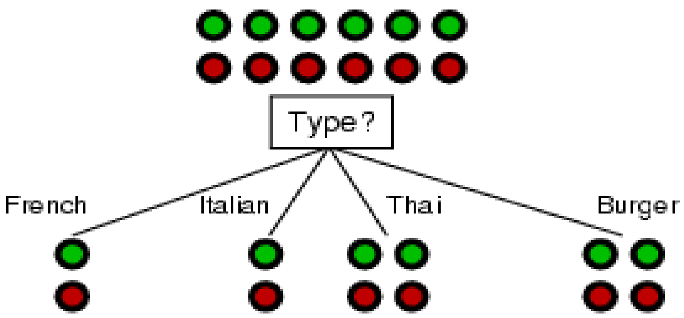
- H(P | A) = conditional entropy given attribute A

  = weighted average entropy of conditional class distribution,

  after partitioning the data according to the values in A

Choosing an attribute





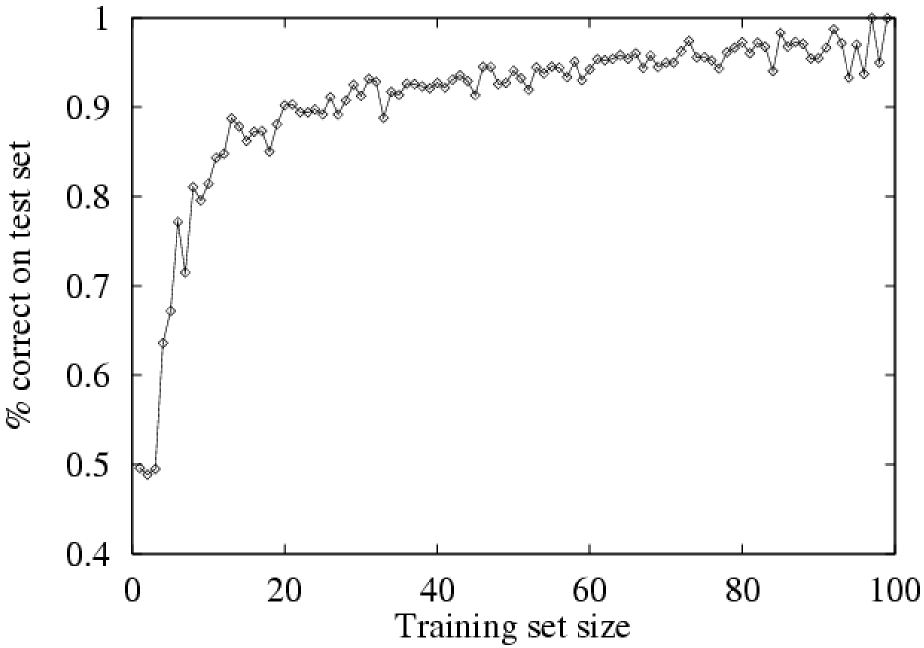
IG(Patrons) = 0.541 bits

IG(Type) = 0 bits

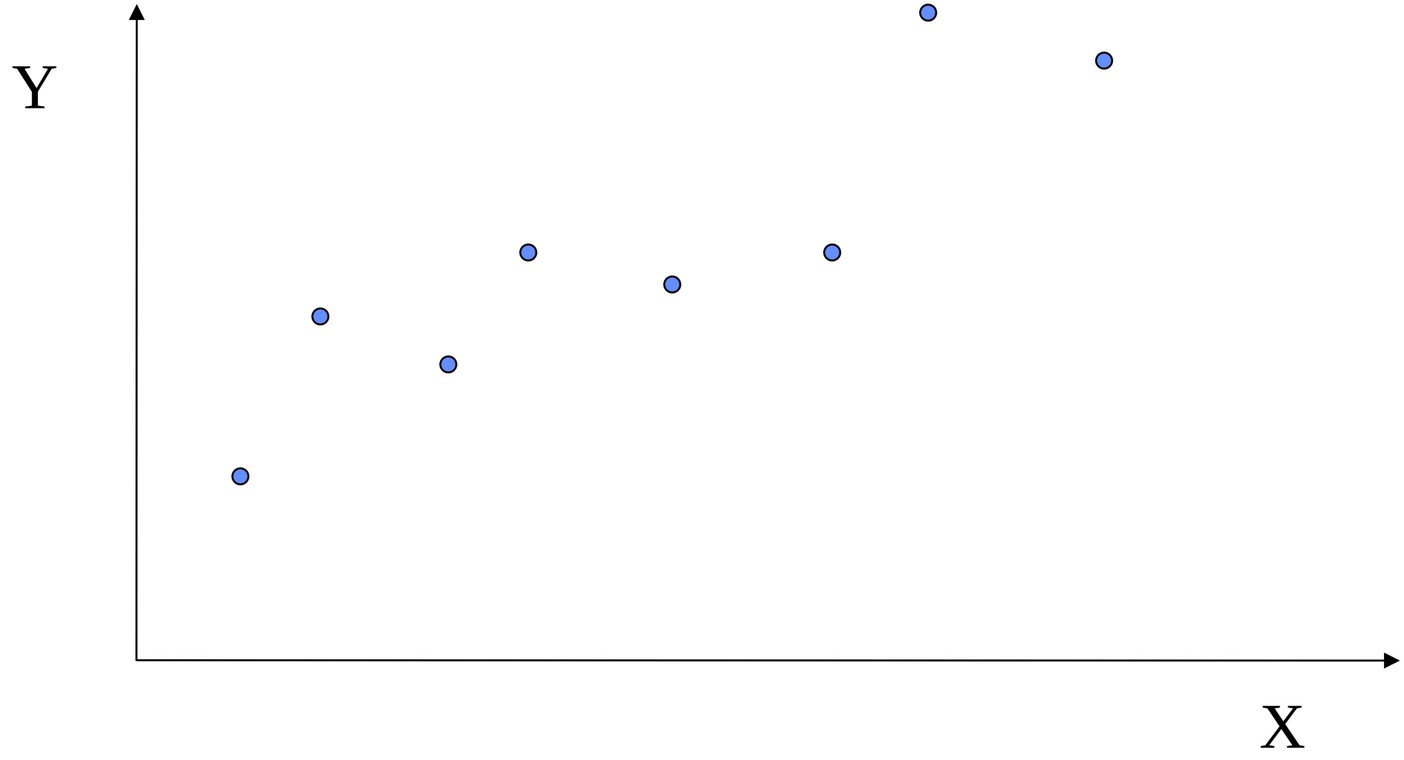
Example of Test Performance

Restaurant problem

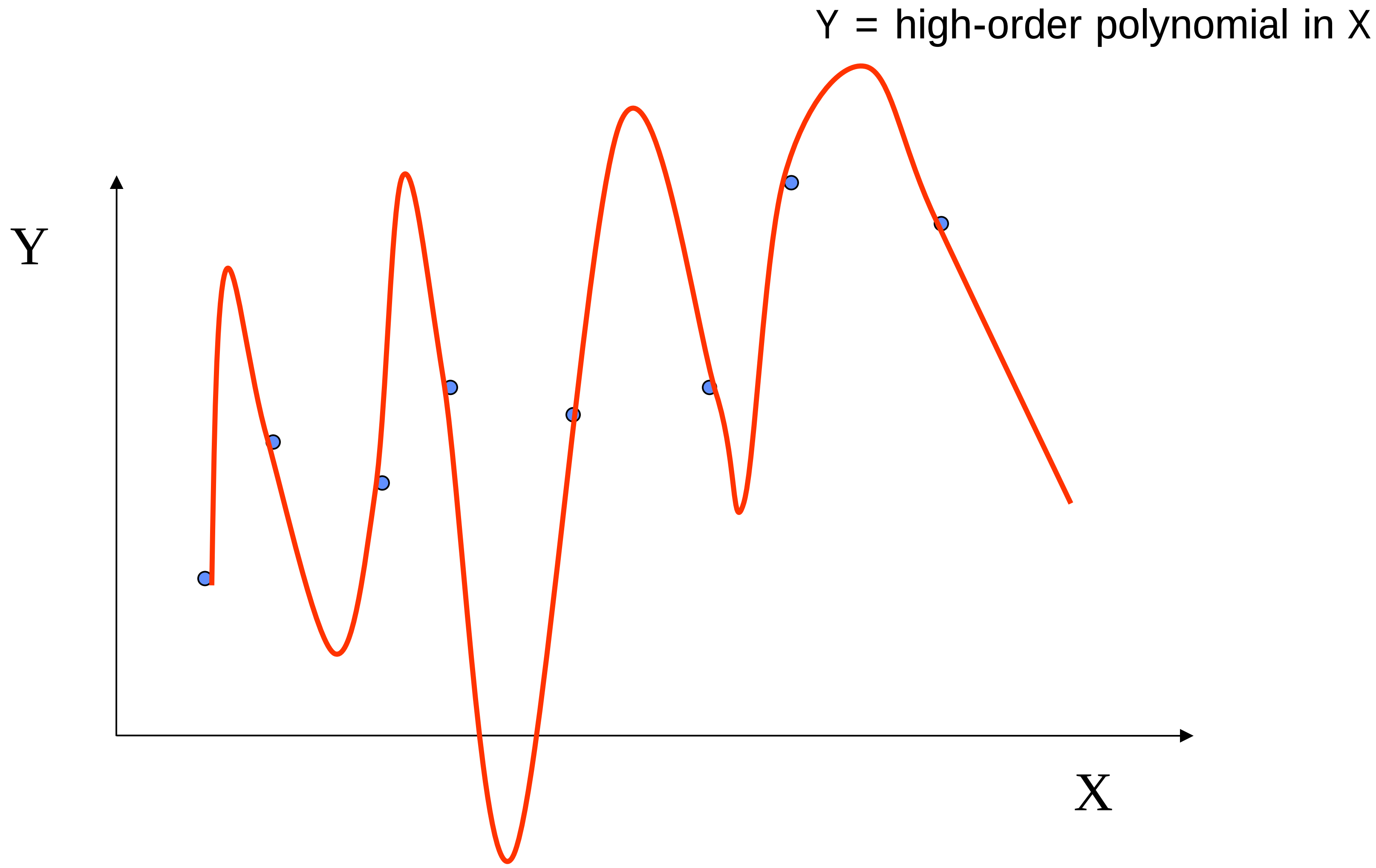
- simulate 100 data sets of different sizes
- train on this data, and assess performance on an independent test set
- learning curve = plotting accuracy as a function of training set size
- typical “diminishing returns” effect (some nice theory to explain this)



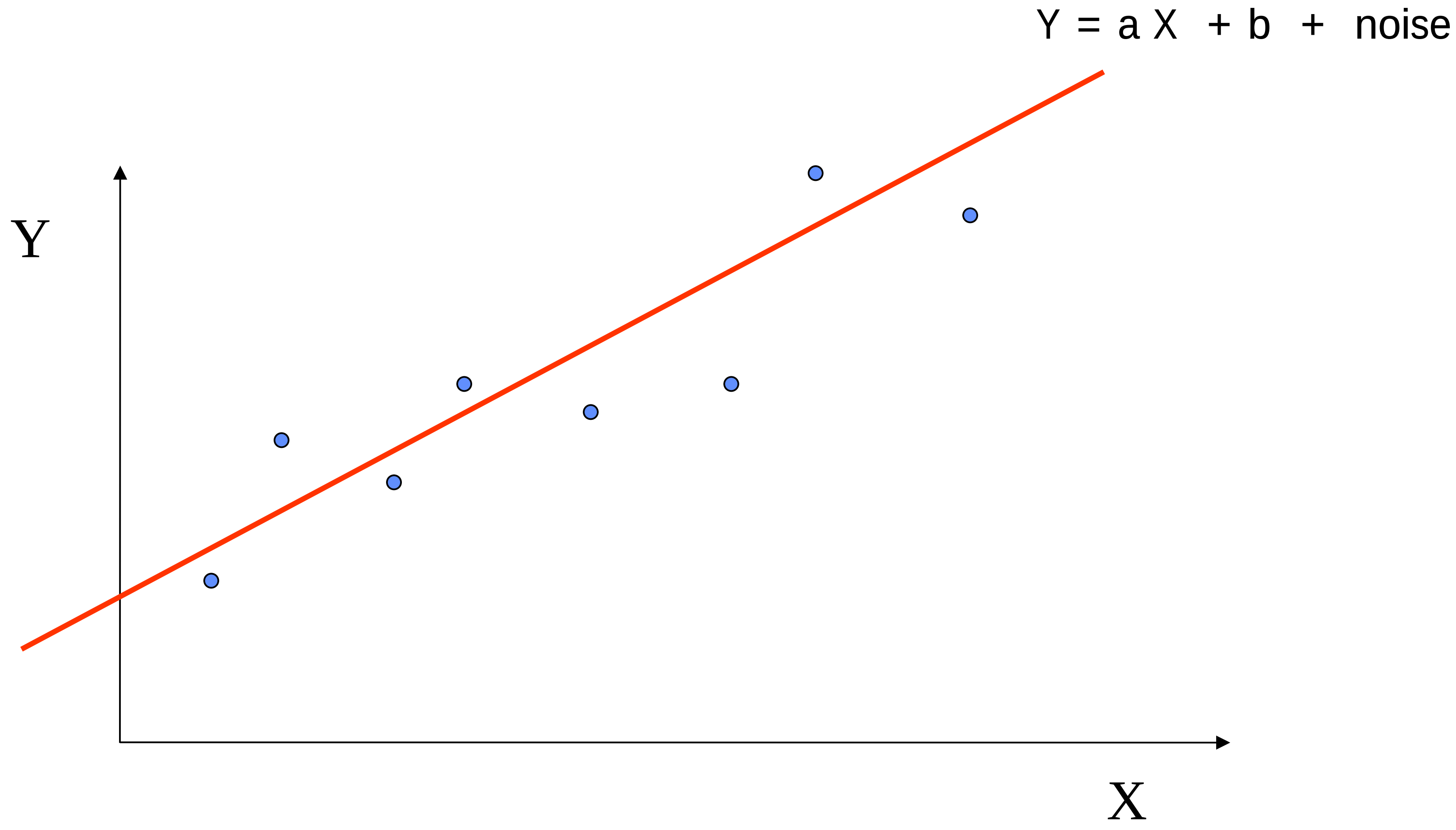
Overfitting and Underfitting



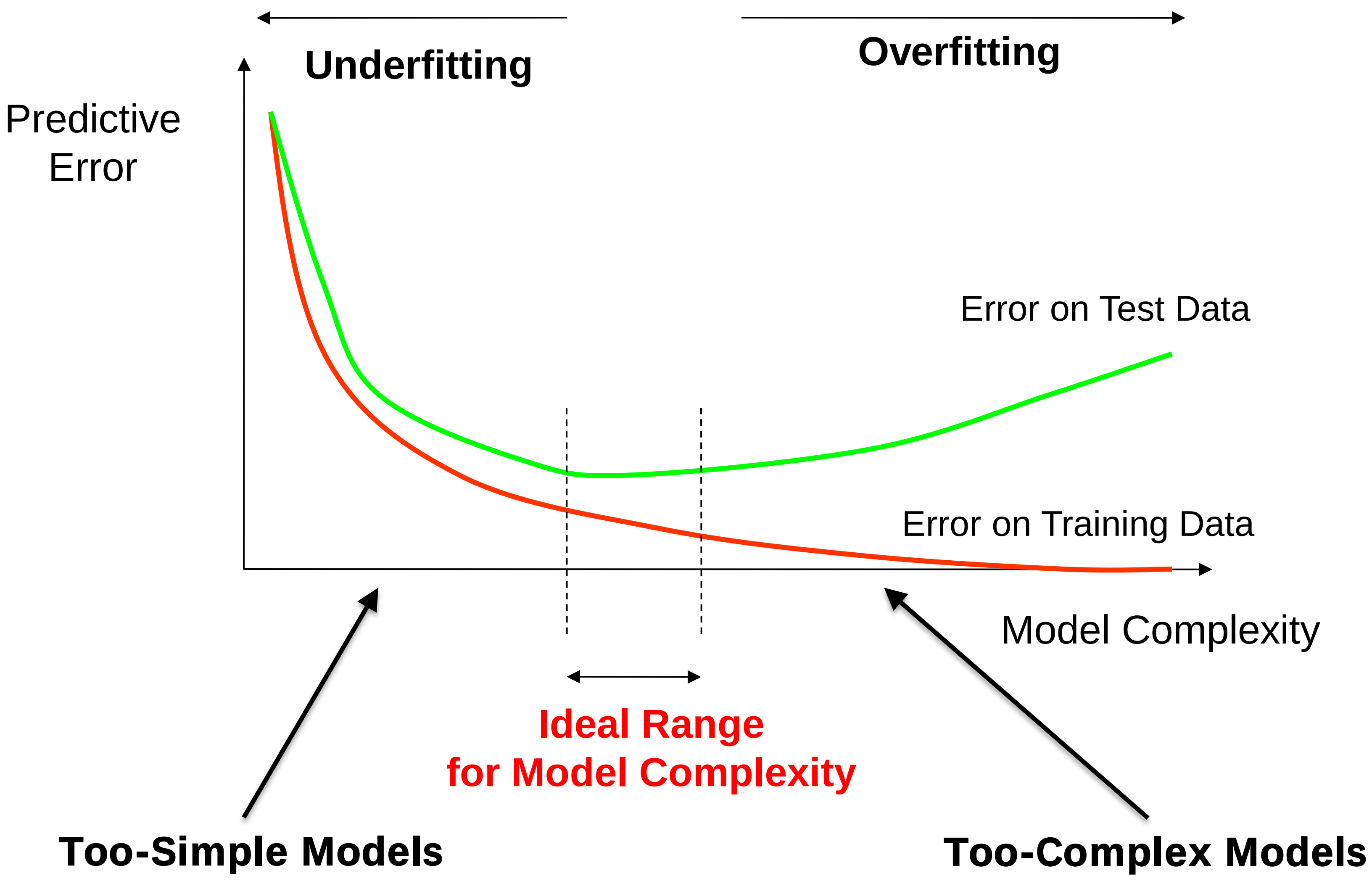
A Complex Model



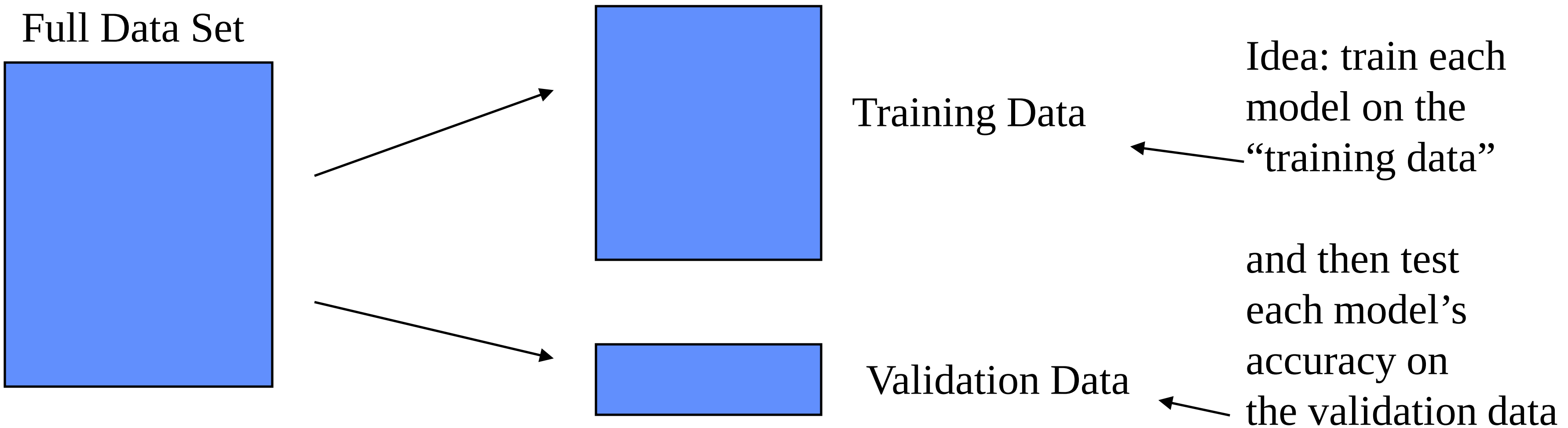
A Much Simpler Model



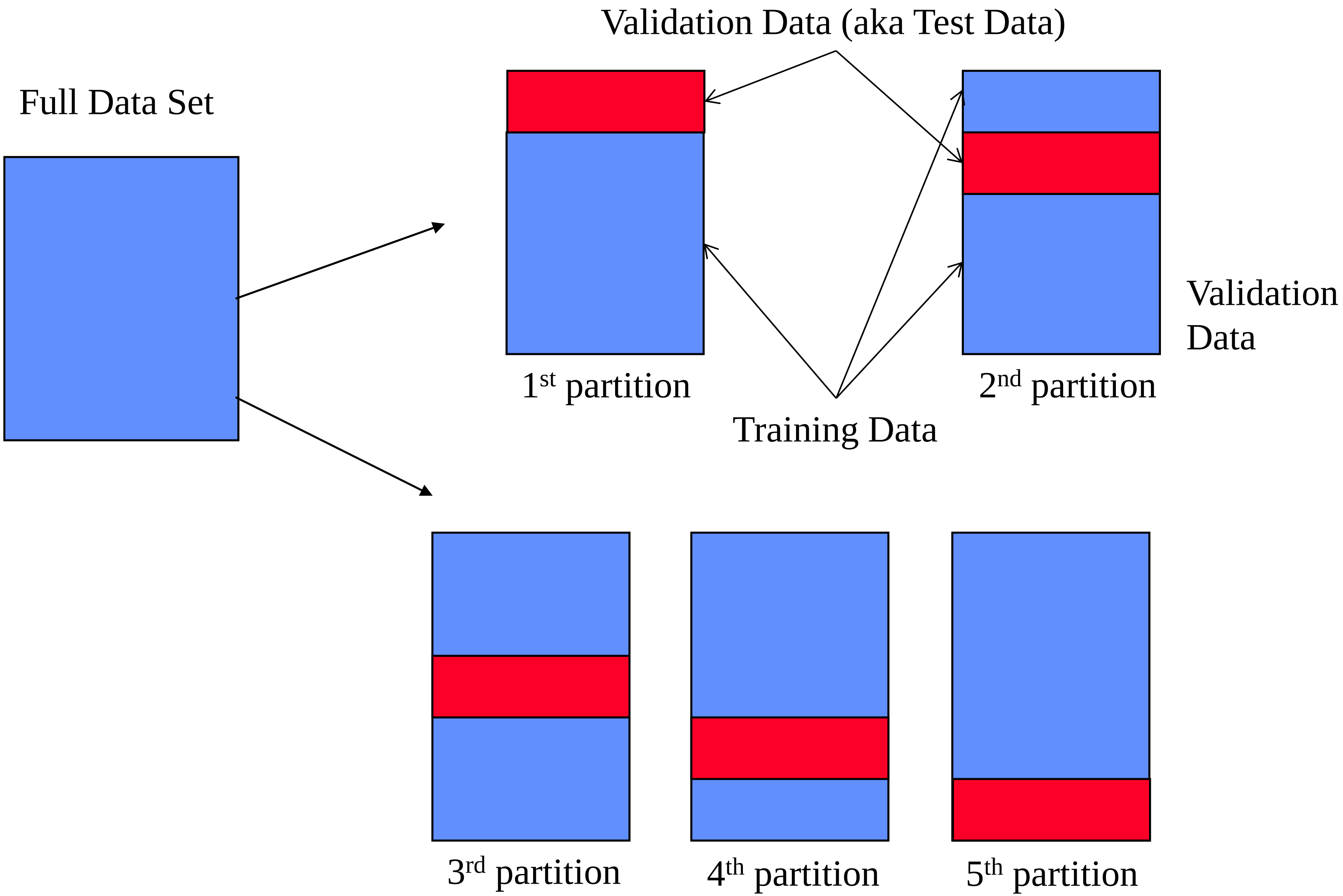
How Overfitting affects Prediction



Training and Validation Data



Disjoint Validation Data Sets



The k-fold Cross-Validation Method

- Why just choose one particular 90/10 "split" of the data?
  - In principle we could do this multiple times

- "k-fold Cross-Validation" (e.g., k=10)
  - randomly partition our full data set into k disjoint subsets (each roughly of size n/k, n = total number of training data points)
    - for i = 1:10 (here k = 10)
      - train on 90% of data,
      - Acc(i) = accuracy on other 10%
    - end
    - Cross-Validation-Accuracy = $1/k \sum_i Acc(i)$
  - choose the method with the highest cross-validation accuracy
  - common values for k are 5 and 10
  - Can also do "leave-one-out" where k = n

You will be expected to know

- Understand Attributes, Error function, Classification, Regression, Hypothesis (Predictor function)
- What is Supervised Learning?
- Decision Tree Algorithm
- Entropy
- Information Gain
- Tradeoff between train and test with model complexity
- Cross validation

Final Exam Review

- Propositional Logic B: R&N Chap 7.1-7.5
- Predicate Logic, Knowledge Representation: R&N Chap 8.1-8.5, 9.1-9.2
- Probability: R&N Chap 13
- Bayesian Networks: R&N Chap 14.1-14.5
- Intro Machine Learning: R&N Chap 18.1-18.4