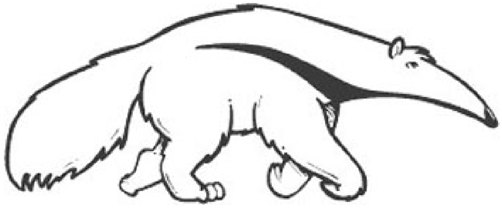
# Introduction to Artificial Intelligence

CS271P, Fall Quarter, 2019
Introduction to Artificial Intelligence
Prof. Richard Lathrop



**Read Beforehand:** **All assigned reading so far**

BREN:ICS INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

# Final Review

- First-Order Logic: R&N Chap 8.1-8.5, 9.1-9.5
- Probability: R&N Chap 13
- Bayesian Networks: R&N Chap 14.1-14.5
- Machine Learning: R&N Chap 18.1-18.4

# Review First-Order Logic
## Chapter 8.1-8.5, 9.1-9.5

- 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 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 $\forall$

- $\forall$ 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:

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

- **Note: $\forall$ x King(x) $\wedge$ Person(x) is not correct!**

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

  **$\forall$ x King(x) => Person(x) is the correct way to say this**

- **Note that => (or $\Leftrightarrow$) is the natural connective to use with $\forall$ .**

# 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:

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

- **Note: ∃ i  Integer(i) ⇒ Greater(i,0)  is not correct!**

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

  **∃ i  Integer(i) ∧ Greater(i,0) is the correct way to say this**

- **Note that ∧ is the natural connective to use with ∃ .**

# 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 ($\neg$ OR → AND $\neg$ ; $\neg$ AND → OR $\neg$).

**QUANTIFIER Rule is similar:** if you bring a negation inside a universal or existential, always switch between them ($\neg \exists \rightarrow \forall \neg$ ; $\neg \forall \rightarrow \exists \neg$).

# 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

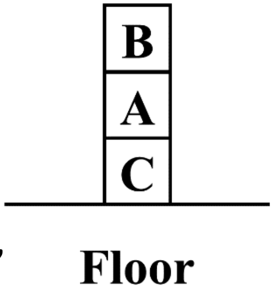
$\forall\, x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \land Parent(x, y)).$

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
  - 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
    - Some other interpretation might result in different truth values.



# 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 and all interpretations 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

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

2. Move $\neg$ inwards:

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

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

# Conversion to CNF contd.

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

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

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

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

$\forall x\, [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

5. Drop universal quantifiers:

$[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

6. Distribute $\lor$ over $\land$ :

$[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$

# Unification

- Recall: Subst($\theta$, p) = result of substituting $\theta$ into sentence p

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

  Unify(p,q) = $\theta$   where Subst($\theta$, p) = Subst($\theta$, q)

  where $\theta$ 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

| | | |
|---|---|---|
| 1) | UNIFY( Knows( John, x ), Knows( John, Jane ) ) | { x / Jane } |
| 2) | UNIFY( Knows( John, x ), Knows( y, Jane ) ) | { x / Jane, y / John } |
| 3) | UNIFY( Knows( y, x ), Knows( John, Jane ) ) | { x / Jane, y / John } |
| 4) | UNIFY( Knows( John, x ), Knows( y, Father (y) ) ) | { y / John, x / Father (John) } |
| 5) | UNIFY( Knows( John, F(x) ), Knows( y, F(F(z)) ) ) | { y / John, x / F (z) } |
| 6) | UNIFY( Knows( John, F(x) ), Knows( y, G(z) ) ) | None |
| 7) | 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)*

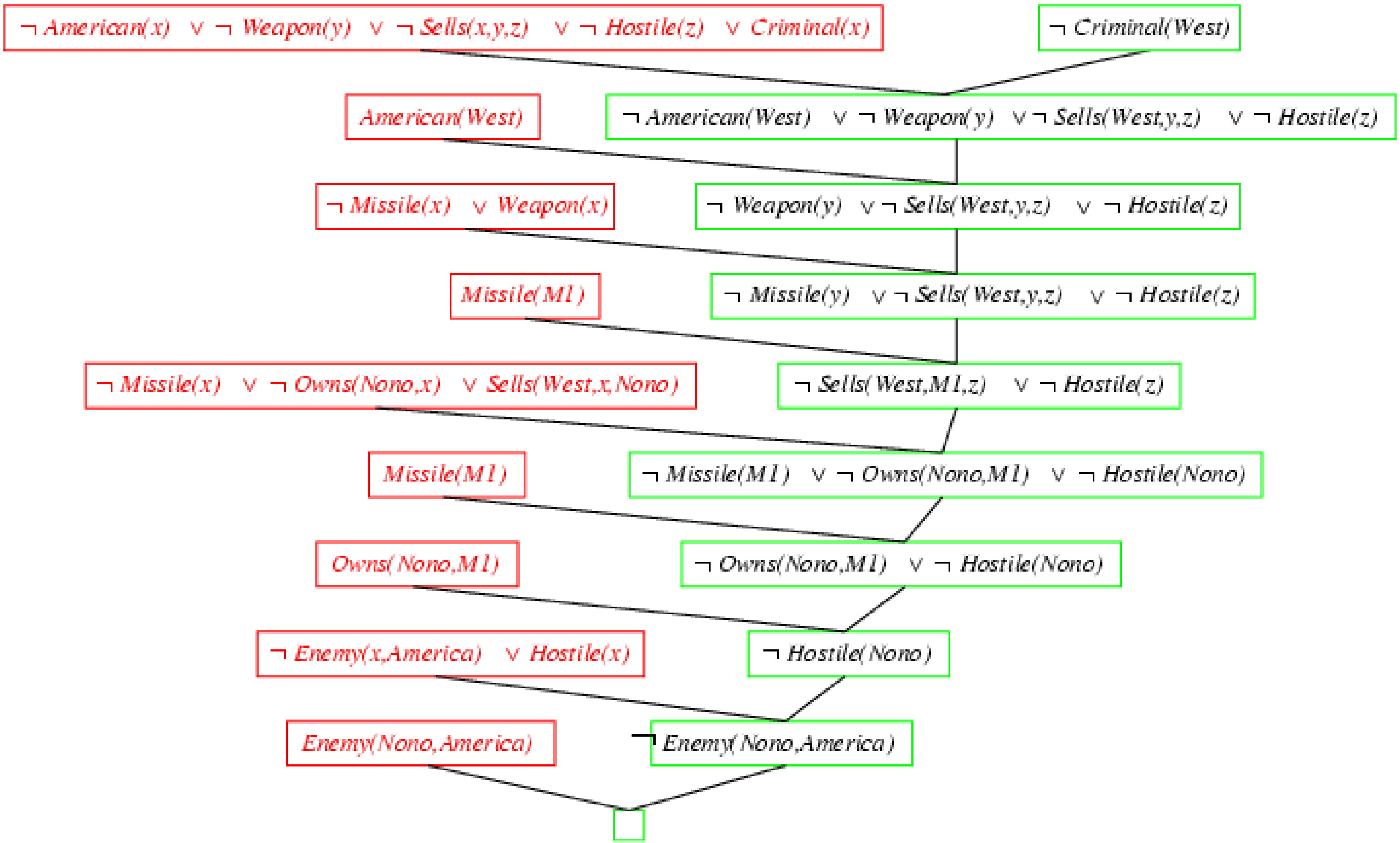
West, who is American …

*American(West)*

The country Nono, an enemy of America …

*Enemy(Nono,America)*

# Resolution proof:



# 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.

- Boolean random variables
  e.g., *Cavity (= do I have a cavity?)*
- Discrete random 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* $\vee$ *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 !

# 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

- <u>Formal Definition</u>:
  - 2 random variables A and B are independent iff:

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

- <u>Informal Definition</u>:
  - 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 $X_1, X_2, \ldots X_k$, and C = disease, reducing to:

    **$P(X_1, X_2, \ldots X_K \mid C) = P(C) \prod P(X_i \mid C)$**

# 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

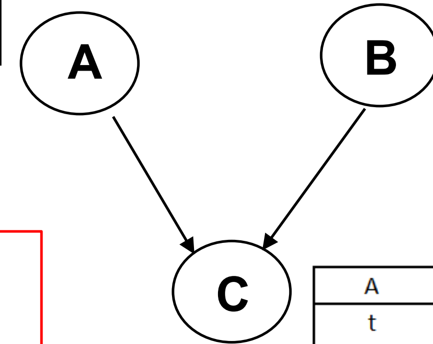
- A Bayesian network specifies a joint distribution in a structured form:

Full factorization

$$
\begin{aligned}
p(A,B,C) &= p(C|A,B)p(A|B)p(B) \\
&= p(C|A,B)p(A)p(B)
\end{aligned}
$$

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 |

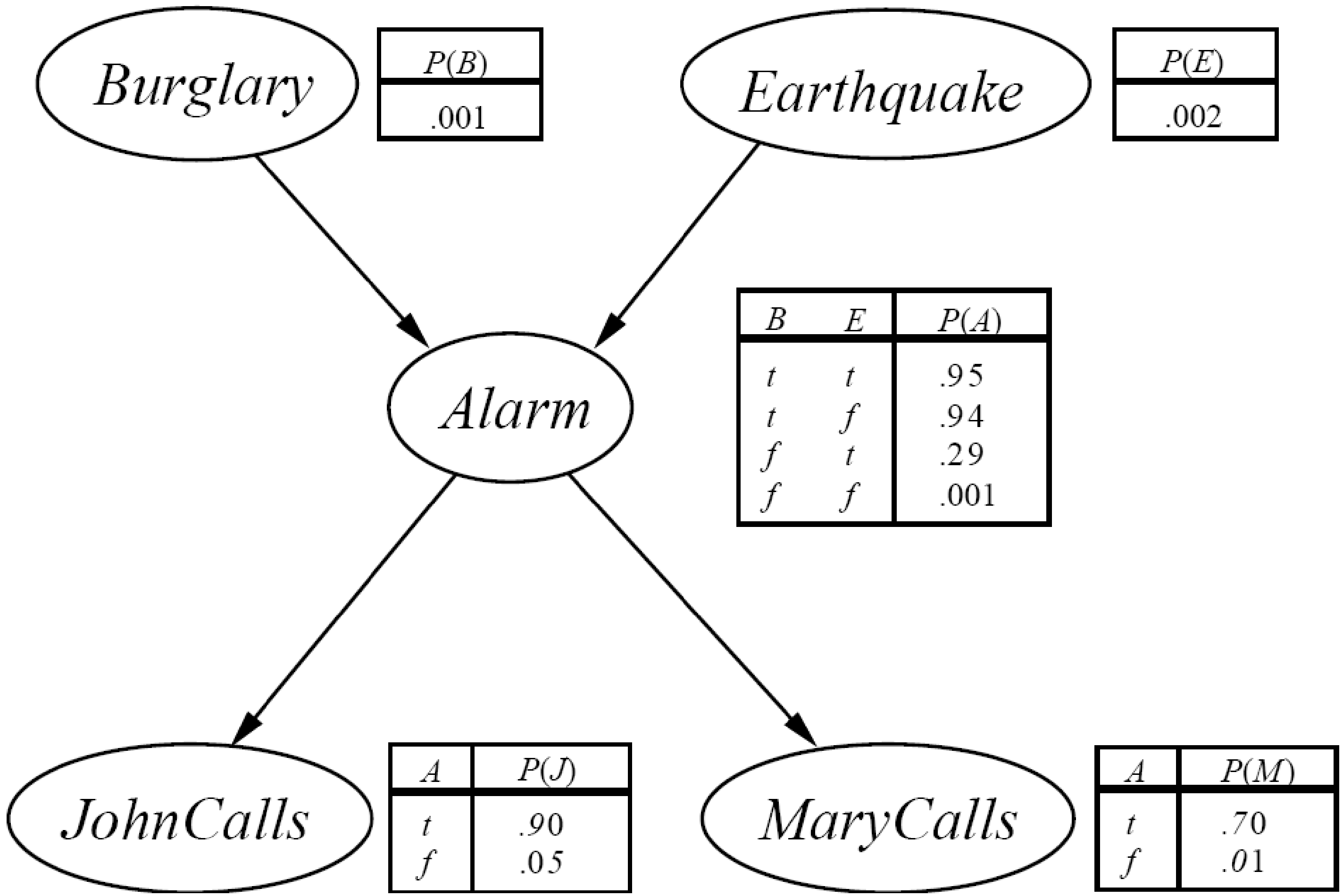
- Dependence/independence represented via a directed graph:
  - Node = random variable
  - Directed Edge = conditional dependence
  - Absence of Edge = conditional independence

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

# Burglar Alarm 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?

# The Resulting Bayesian Network




| $P(B)$ |
|---|
| .001 |

| $P(E)$ |
|---|
| .002 |

| $B$ | $E$ | $P(A)$ |
|---|---|---|
| $t$ | $t$ | .95 |
| $t$ | $f$ | .94 |
| $f$ | $t$ | .29 |
| $f$ | $f$ | .001 |

| $A$ | $P(J)$ |
|---|---|
| $t$ | .90 |
| $f$ | .05 |

| $A$ | $P(M)$ |
|---|---|
| $t$ | .70 |
| $f$ | .01 |

# Example of Answering a Simple Query

- What is P(¬j, m, a, ¬e, b) = P(J = false ∧ M=true ∧ A=true ∧ E=false ∧ B=true)

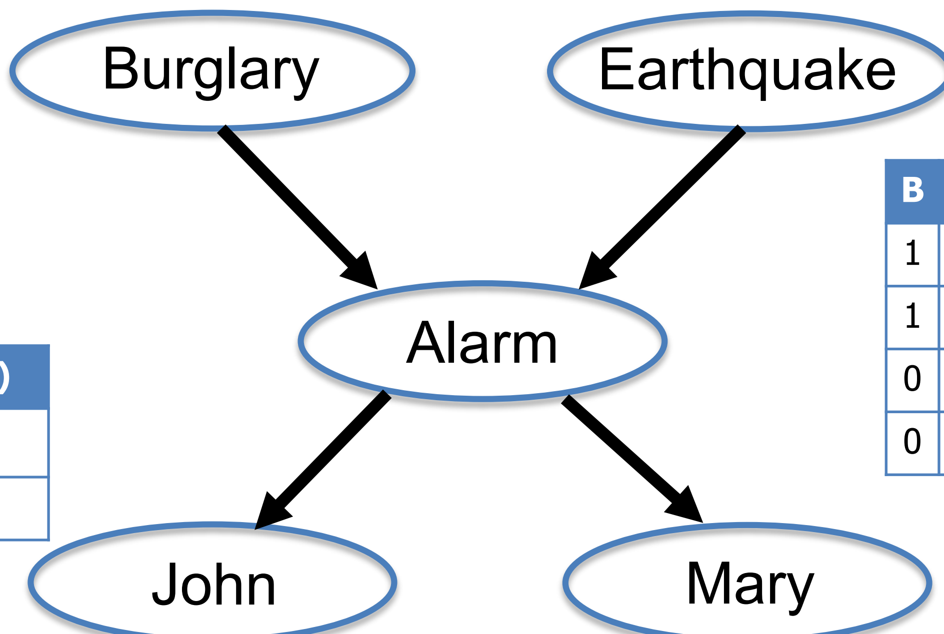
$P(J, M, A, E, B) \approx P(J \mid A)\, P(M \mid A)\, P(A \mid E, B)\, P(E)\, P(B)$ ; by conditional independence

$P(\neg j, m, a, \neg e, b) \approx P(\neg j \mid a)\, P(m \mid a)\, P(a \mid \neg e, b)\, P(\neg e)\, P(b)$

$= 0.10 \times 0.70 \times 0.94 \times 0.998 \times 0.001 \approx .0000657$

| 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 |

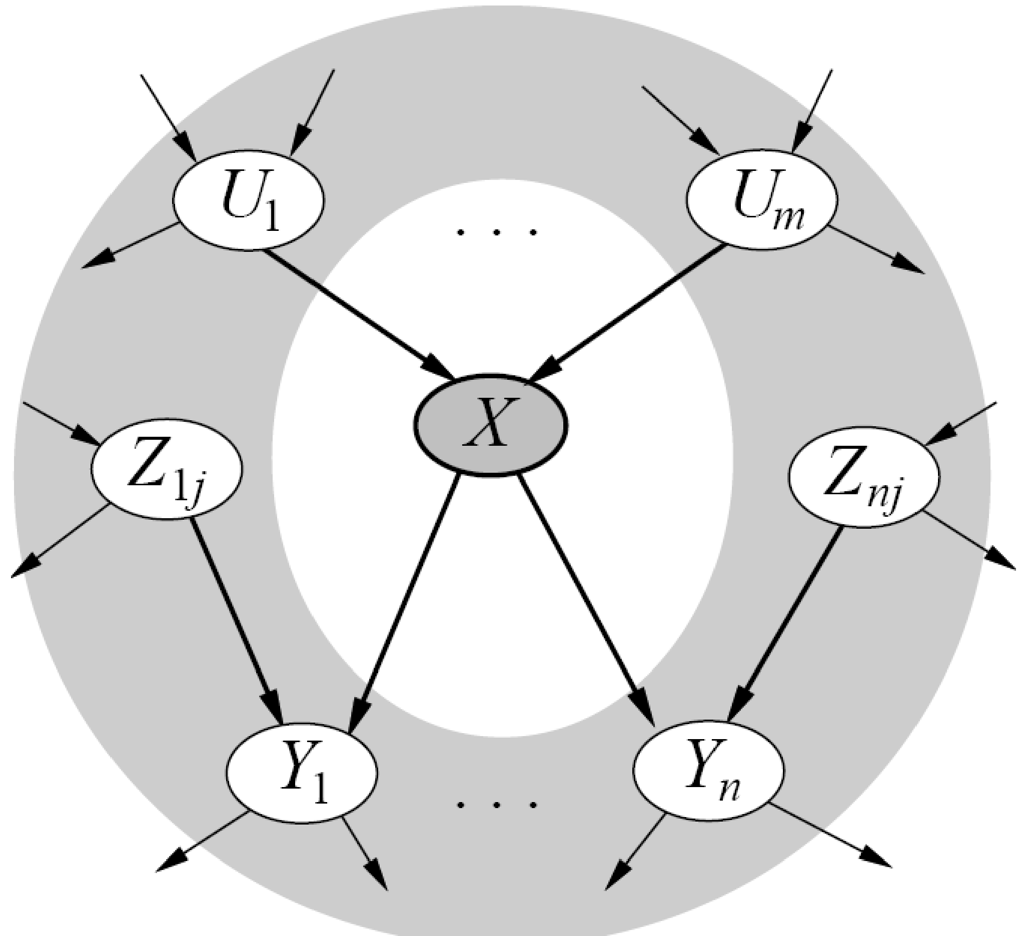
# Given a graph, can we "read off" conditional independencies?

**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.



# 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

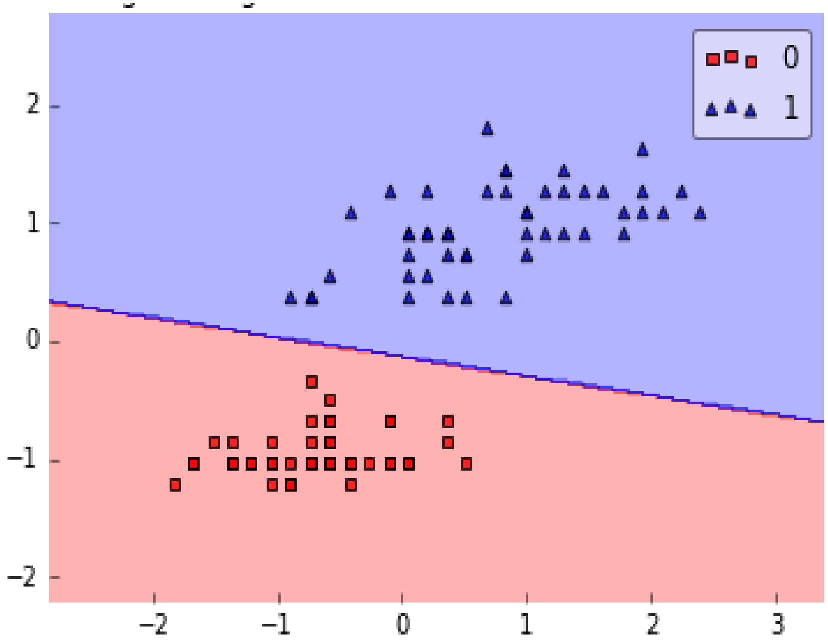
# Review Intro Machine Learning Chapter 18.1-18.4

- 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

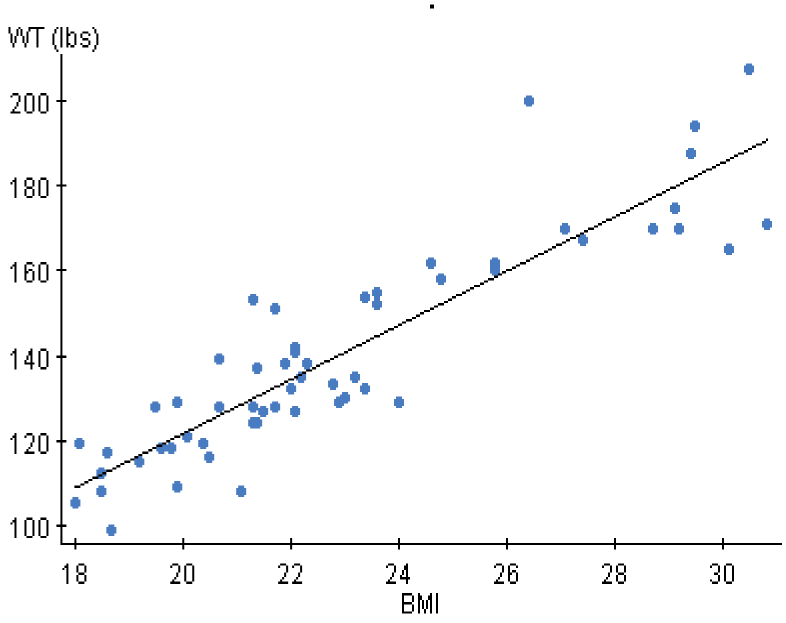
# Supervised Learning

- 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, $\theta$) ≈ f(x) for all training data points x
  - $\theta$ are the parameters of our predictor function h

- Examples:
  - h(x, $\theta$) = 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)

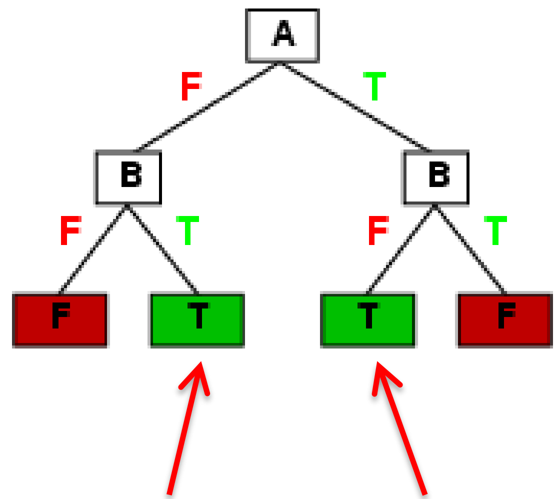
In learning, we get to choose

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

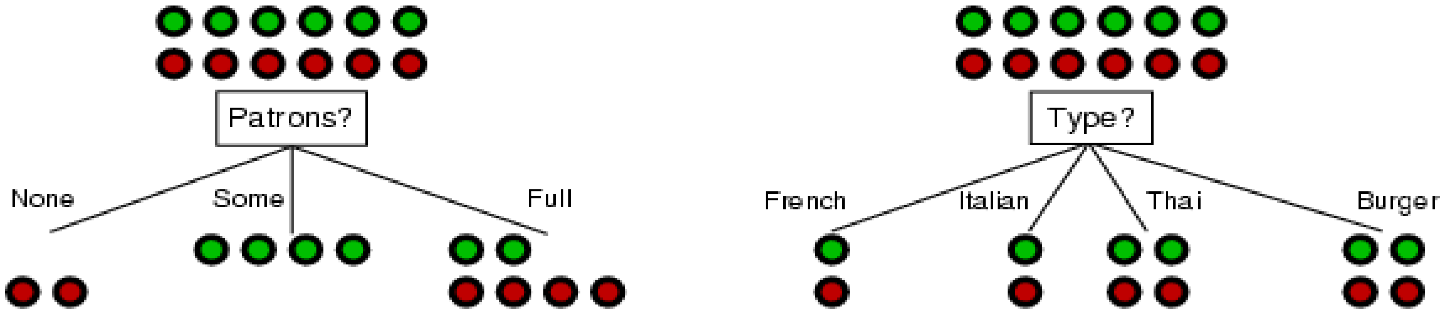
# Pseudocode for Decision tree learning

**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**
        $best \leftarrow$ CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        $tree \leftarrow$ a new decision tree with root test *best*
        **for each** value $v_i$ of *best* **do**
            $examples_i \leftarrow$ {elements of *examples* with $best = v_i$}
            $subtree \leftarrow \mathrm{DTL}(examples_i, attributes - best, \mathrm{MODE}(examples))$
            add a branch to *tree* with label $v_i$ and subtree *subtree*
        **return** *tree*

# Choosing an attribute

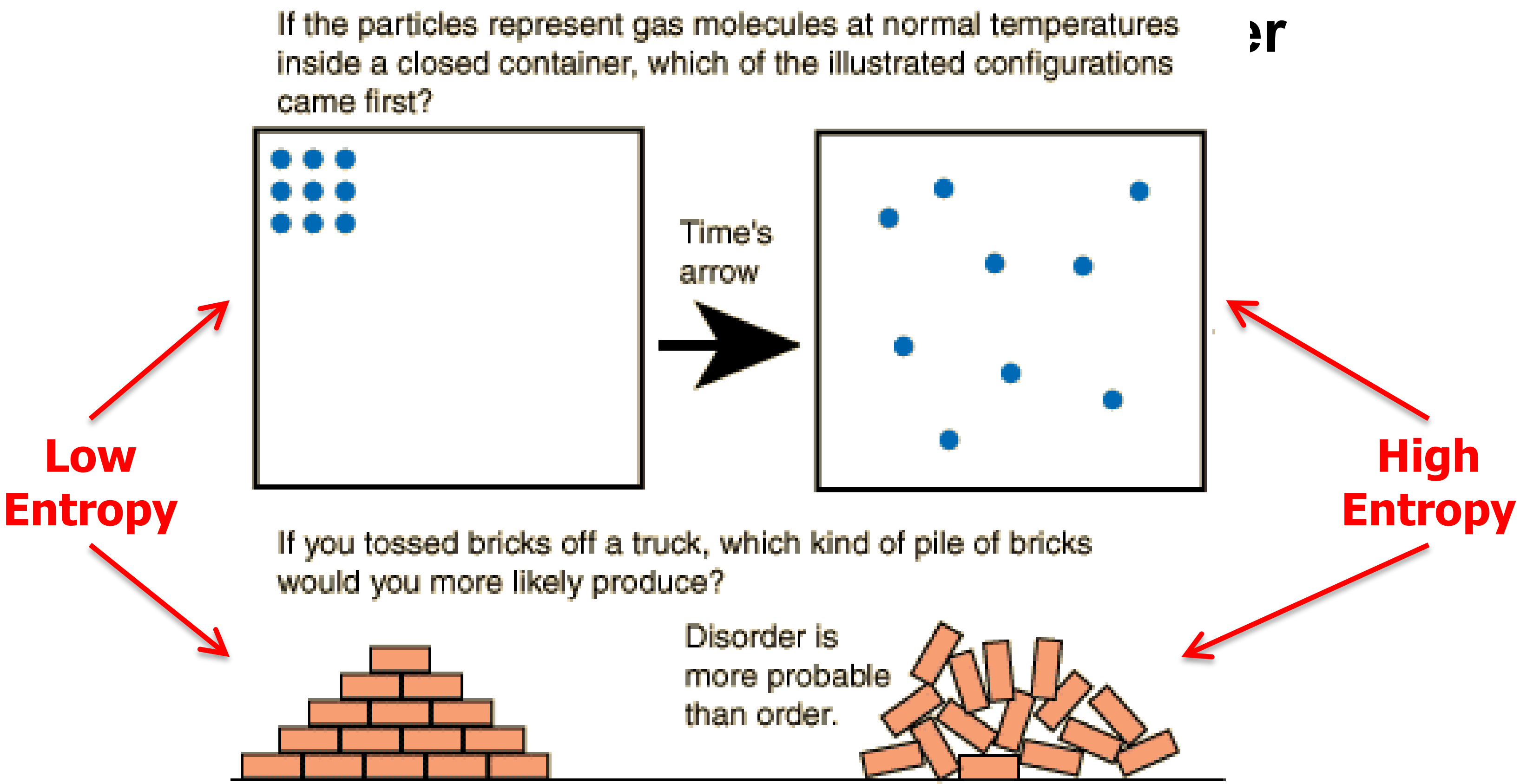
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"




- *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




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

# Entropy, H(p), with only 2 outcomes

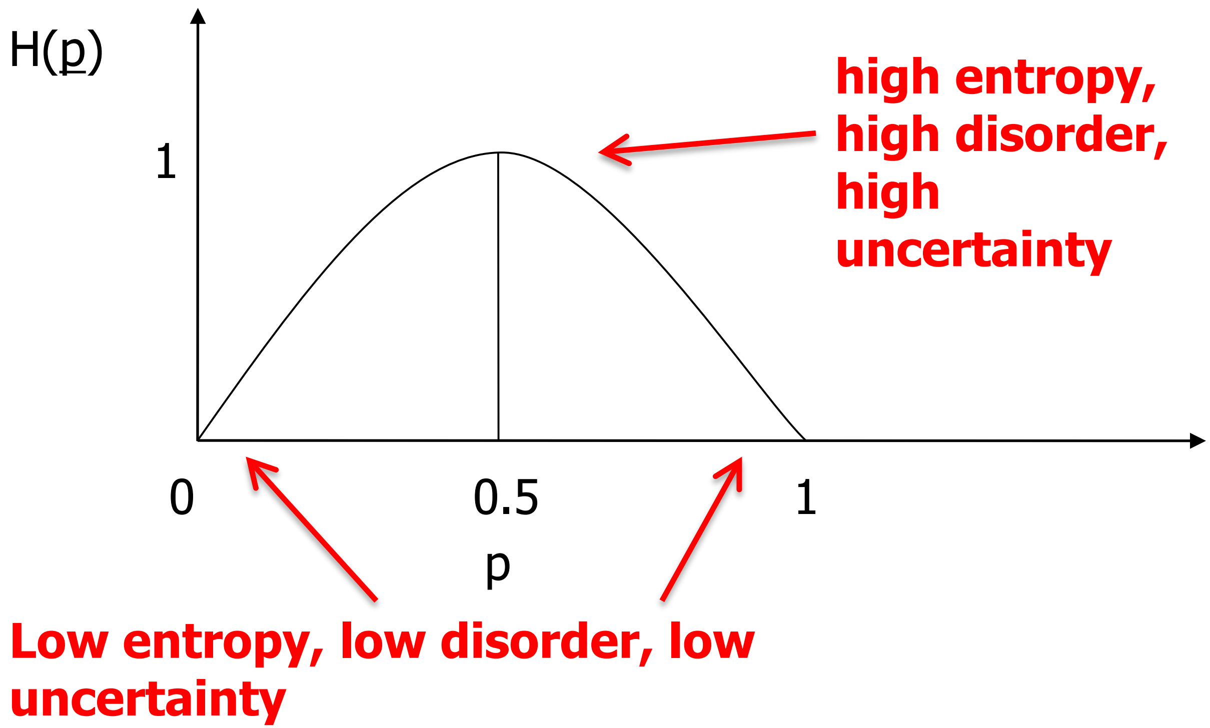
Consider 2 class problem:

$p$ = probability of class #1,

$1 - p$ = probability of class #2

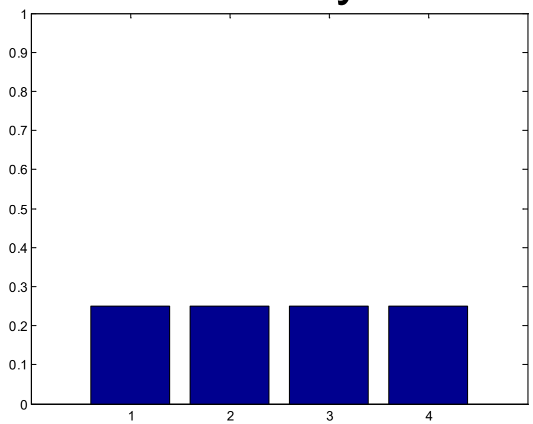
In binary case:

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

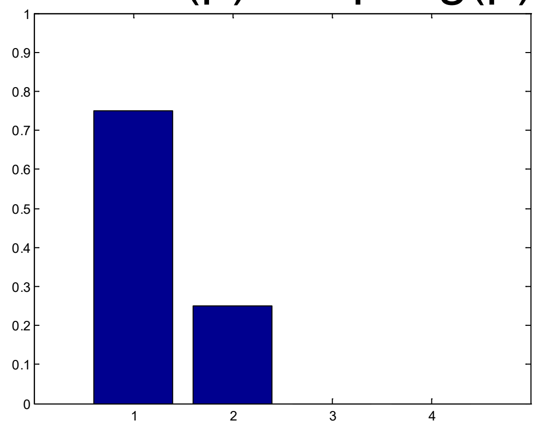


# Entropy and Information

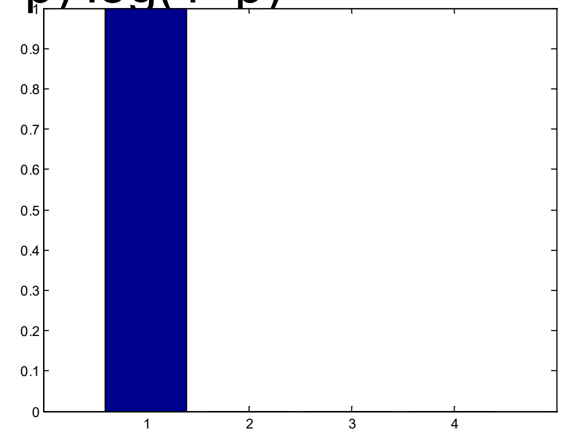
- **Entropy H(X) = E[ log 1/P(X) ] = $\sum_{x\in X}$ P(x) log 1/P(x)**
  **= $-\sum_{x\in X}$ P(x) log P(x)**
  – Log base two, units of entropy are "bits"
  – If only two outcomes: H(p) = – p log(p) – (1–p) log(1–p)




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




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




**H(x) = 1 log 1**
**= 0 bits**

**Max entropy for 4 outcomes**

**Min entropy**

# Information Gain

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

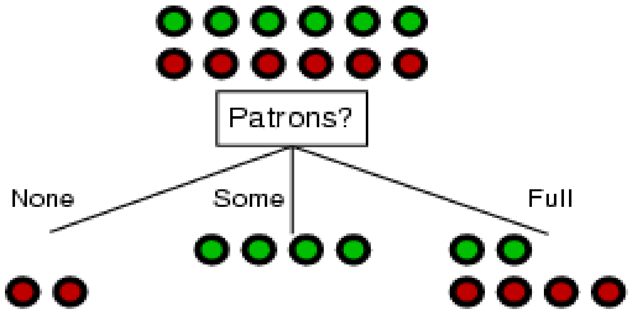
  <u>before further partitioning the data</u>

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

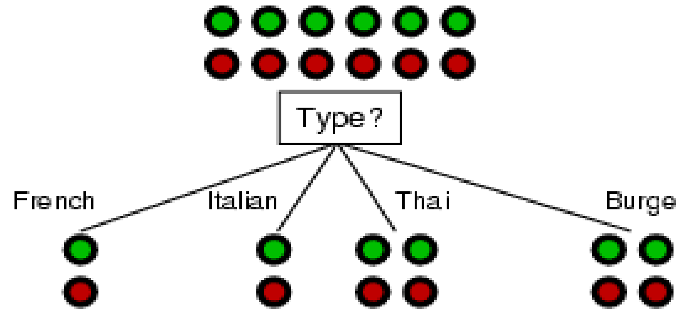
  = weighted average entropy of conditional class distribution,

  <u>after partitioning the data according to the values in A</u>

## Choosing an attribute




IG(Patrons) = 0.541 bits

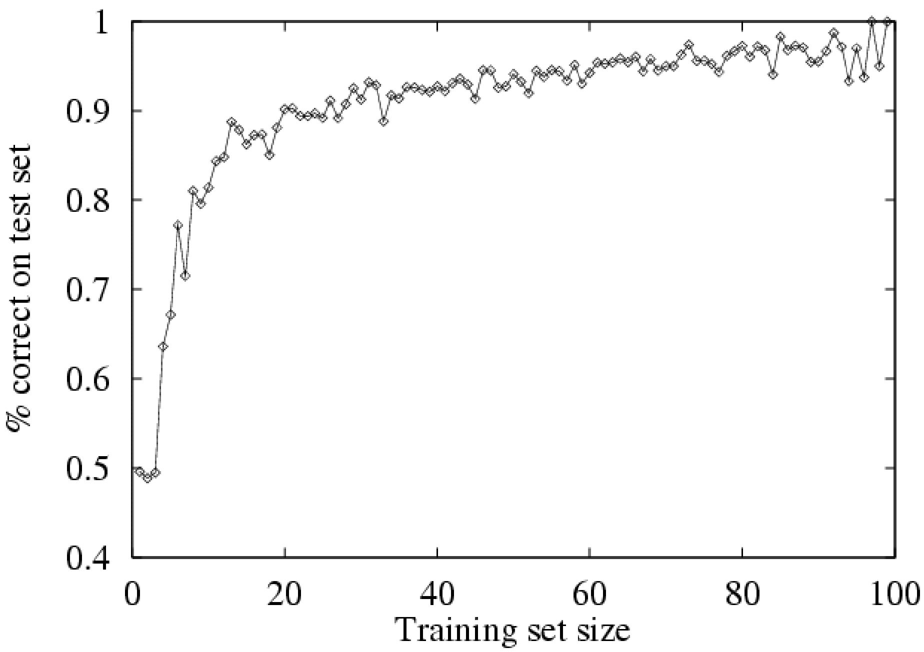



IG(Type) = 0 bits

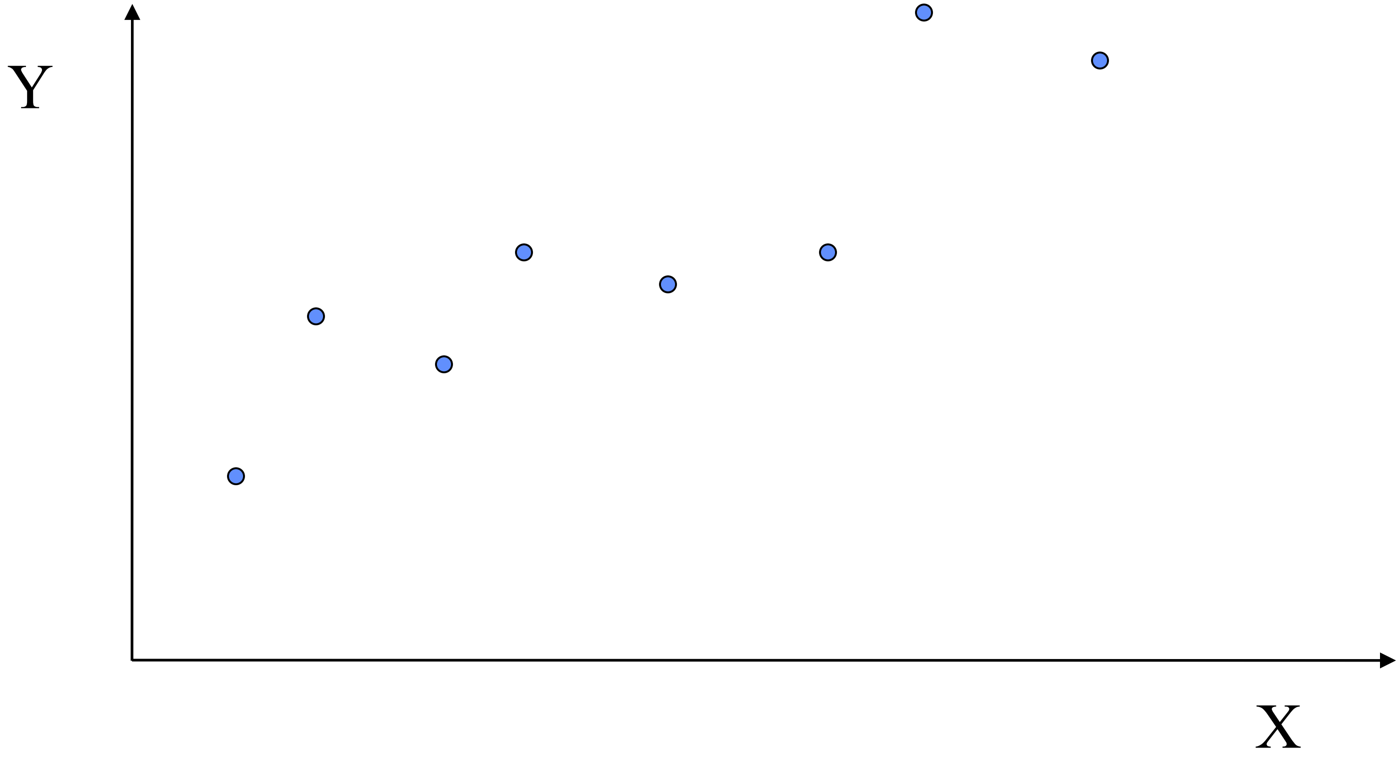
# Example of Test Performance

Restaurant problem

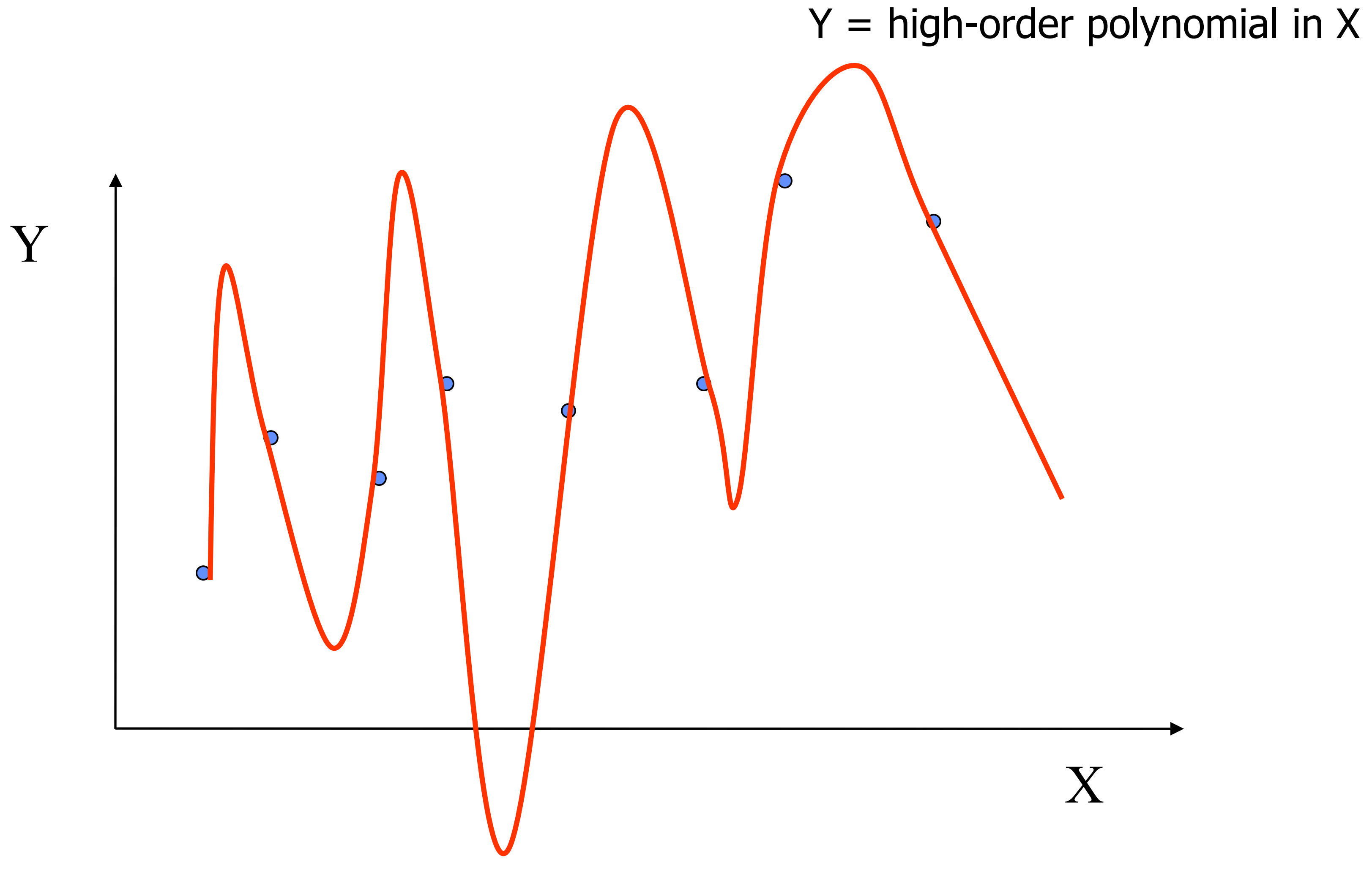
- simulate 100 data sets of different sizes
- train on this data, and assess performance on an independent test set
- learning curve = plotting accuracy as a function of training set size
- typical “diminishing returns” effect (some nice theory to explain this)



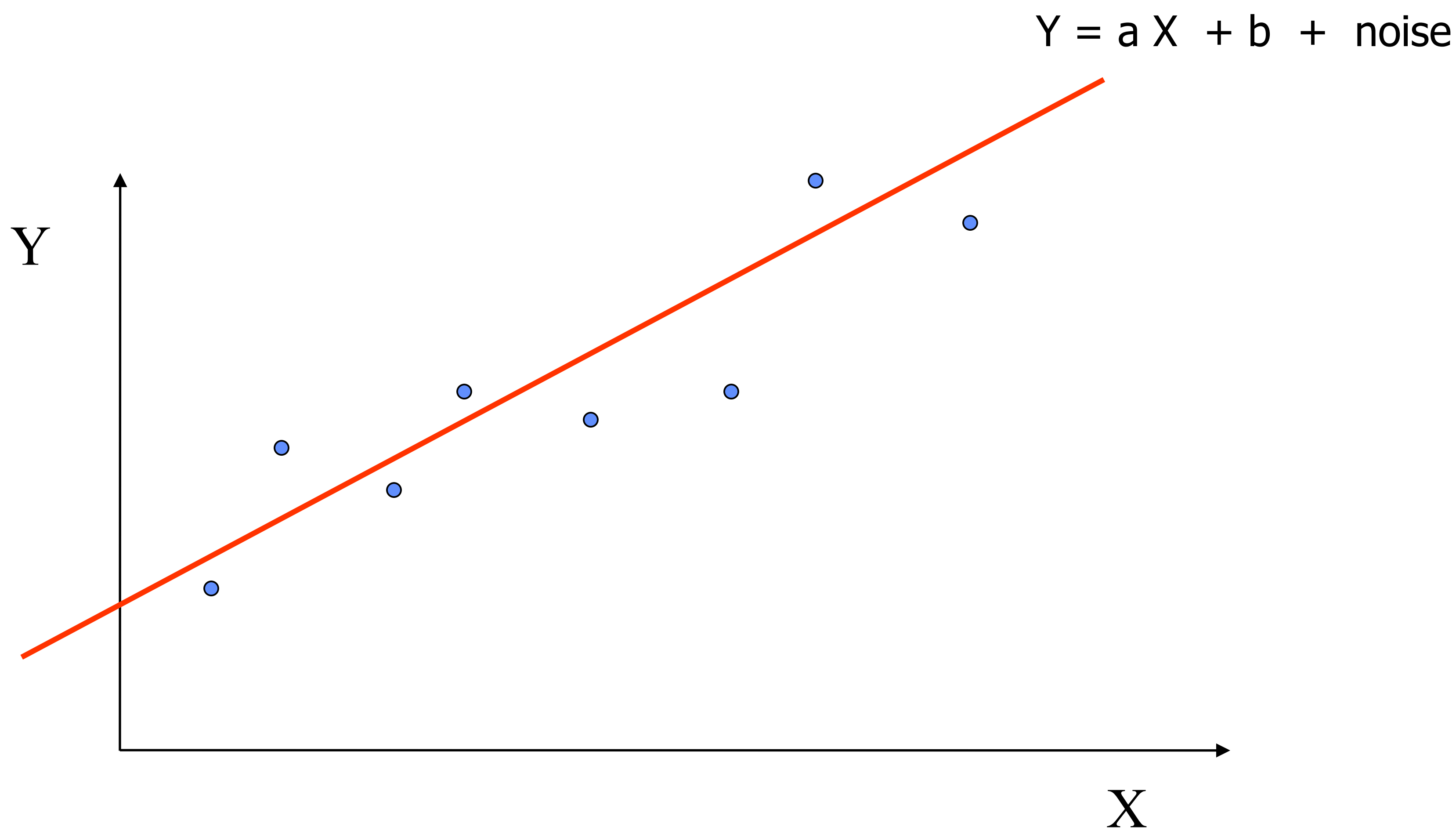
# Overfitting and Underfitting



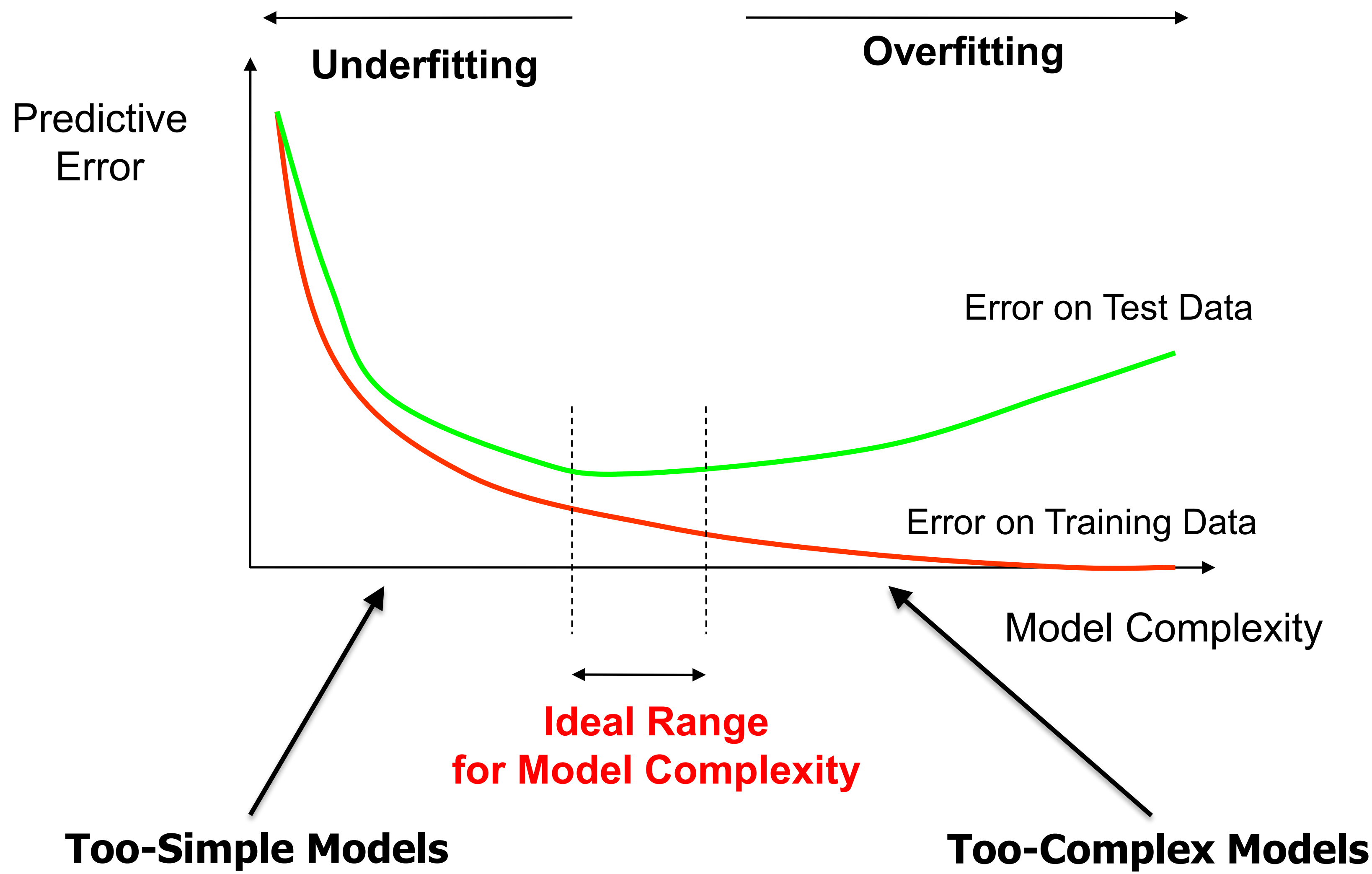
# A Complex Model



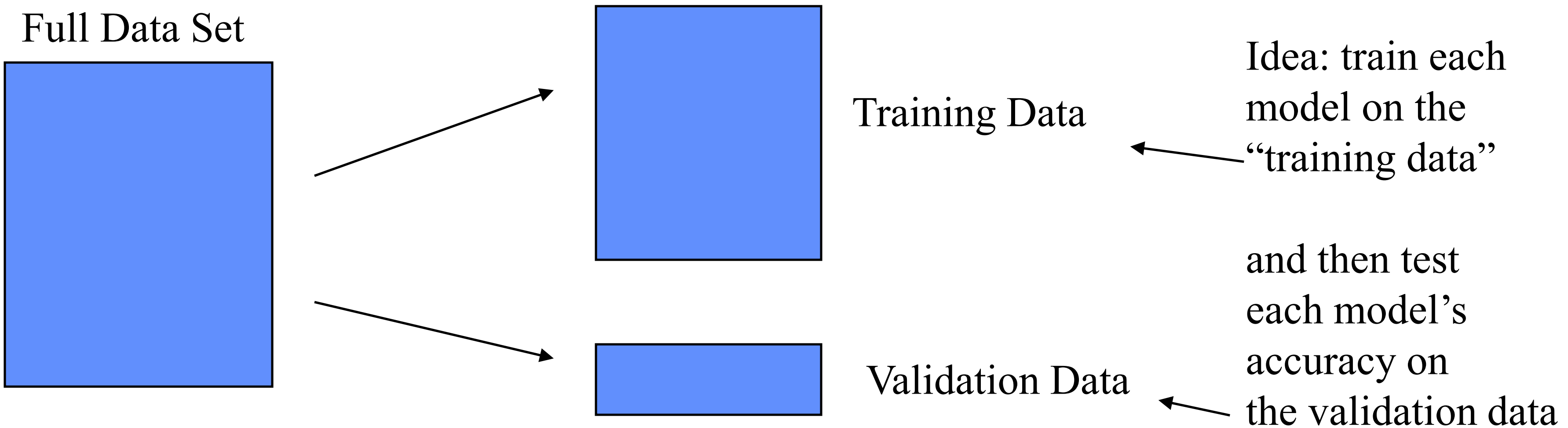
## A Much Simpler Model



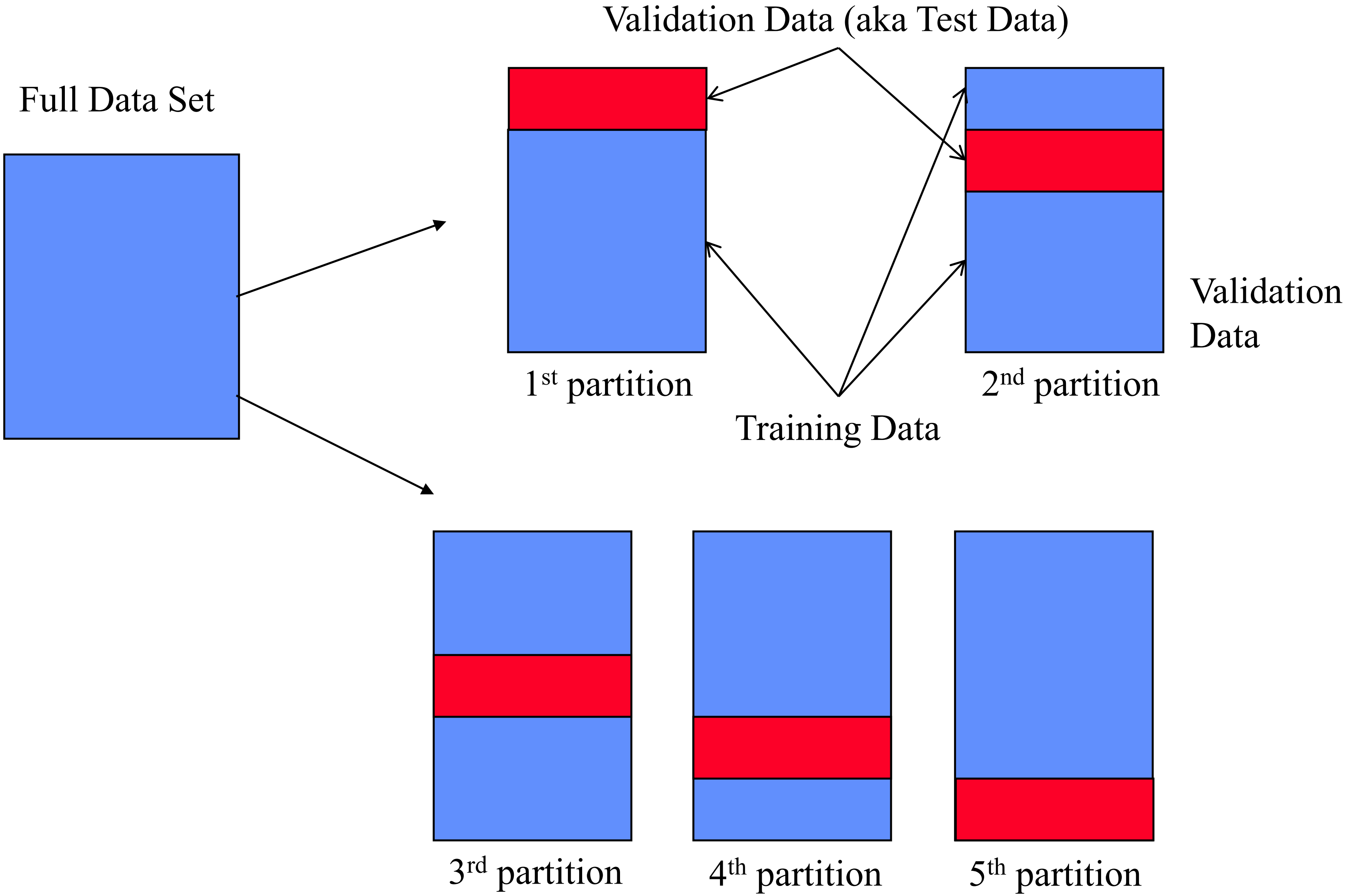
## How Overfitting affects Prediction



# 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 $\Sigma_i$ 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 Review

- First-Order Logic: R&N Chap 8.1-8.5, 9.1-9.5
- Probability: R&N Chap 13
- Bayesian Networks: R&N Chap 14.1-14.5
- Machine Learning: R&N Chap 18.1-18.4