# MIT 6.034 Artificial Intelligence

Prof. Patrick Henry Winston

Fall 2010

# Contents

# 1 Introduction

## 1.1 Definition

*Algorithms enabled by constraints exposed by representations that support models targeted at thinking, perception, and action.*

Artificial Intelligence is applied through problem solving procedures, methods, techniques and algorithms.

## 1.2 Basic Idea

### 1.2.1 Generate-And-Test Algorithm

Generate-and-test search algorithm is a very simple algorithm that guarantees to find a solution if done systematically and there exists a solution.

**Algorithms:**

1. Generate a possible solution.
2. Test to see if this is the expected solution.
3. If the solution has been found quit, else go to step 1.

This approach involved building generators with certain properties: not redundant (should not give the same solution twice), they should also be informable (able to select a category and disregard other)

### 1.2.2 Rumpelstiltsin Principle

Being able to name what you're talking about gives you power over it, to understand and solve problems. Naming things grants power over concepts.

### 1.2.3 Simple $\neq$ Trival

Trivial ideas implies that they are worthless, useless. In AI, the most simple ideas are often the most powerful.

# 2 Reasoning: Goal Trees and Problem Solving

## 2.1 Question

$$\int \frac{-5x^4}{(1-x^2)^{\frac{5}{2}}} dx$$

Is the program, which can solve this integral problem, in any sense of the word **INTELLIGENT**?

## 2.2 Model

### 2.2.1 Problem Rediction or AND-OR Tree or Goal Tree

An and–or tree is a graphical representation of the reduction of problems (or goals) to conjunctions and disjunctions of subproblems (or subgoals).

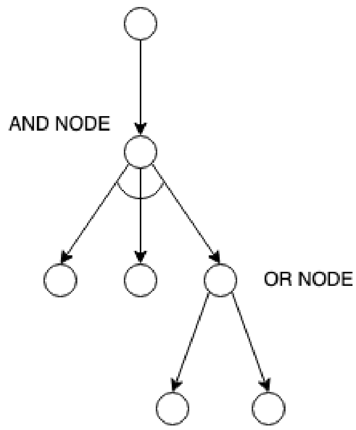



Figure 1: AND-OR Tree

### 2.2.2 Architecture

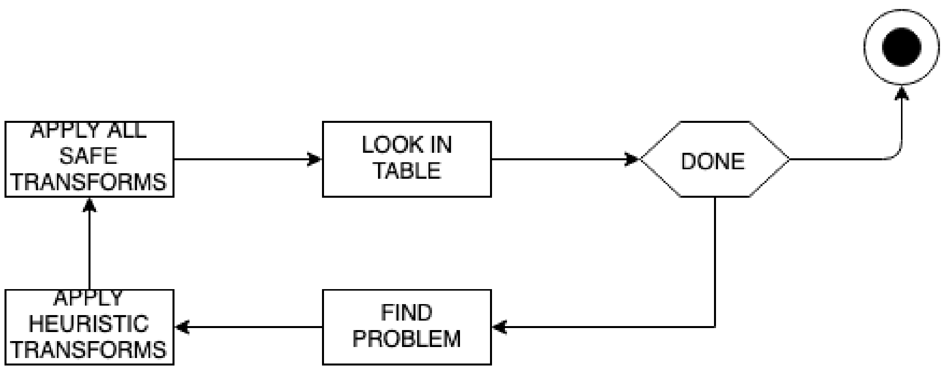



Figure 2: Problem Solving Architecture

## 2.3 Transformation

### 2.3.1 Safe Transformation

1. $\int -f(x)dx = -\int f(x)dx$
2. $\int cf(x)dx = c\int f(x)dx$
3. $\int \sum f_i(x)dx = \sum \int f_i(x)dx$
4. $\int \frac{P(x)}{Q(x)} \rightarrow DIVIDE$

### 2.3.2 Heuristic Transformation

A $f(\sin x, \cos x, \tan x, \cot x, \sec x, \csc x) \rightarrow g_1(\sin x, \cos x)$ *or* $g_2(\tan x, \cot x)$ *or* $g_3(\sec x, \csc x)$

B $\int f(\tan x)dx = \int \frac{f(y)}{1+y^2}dy$

C $\begin{cases} 1 - x^2 \rightarrow x = \sin y \\ 1 + x^2 \rightarrow x = \tan y \end{cases}$

### 2.3.3 Procedure

$$\begin{aligned}
\int \frac{-5x^4}{(1-x^2)^{\frac{5}{2}}}dx &\xrightarrow{1} \int \frac{5x^4}{(1-x^2)^{\frac{5}{2}}}dx \\
&\xrightarrow{2} \int \frac{x^4}{(1-x^2)^{\frac{5}{2}}}dx \\
&\xrightarrow{C} \int \frac{\sin^4 y}{\cos^4 y}dy \\
&\xrightarrow{A} \begin{cases} \int \frac{1}{\cot^4 s}dx \\ \int \tan^4 sdx \quad (Choose) \end{cases} \\
&\xrightarrow{B} \int \frac{y^4}{1+y^2}dy \\
&\xrightarrow{4} \int y^2 - 1 + \frac{1}{1+y^2}dy \\
&\xrightarrow{3} \int y^2 dy + \int -1dy + \int \frac{1}{1+y^2}dy
\end{aligned}$$

$$\int -1dy \xrightarrow{1} \int dy \;\; and \;\; \int \frac{1}{1+y^2}dy \xrightarrow{C} \int dz$$

## 2.4 Reflection

KNOWLEDGE ABOUT KNOWLEDGE IS POWER.
When we understand how something works, intelligence seems to vanish

## 2.5 Questions about the nature of knowledge

1. WHAT kind of knowledge is involved?
2. HOW is the knowledge represented?
3. HOW is it used?
4. HOW much knowledge is required?
   Table of integrals: 26 elements.
   Safe transformations: 12.
   Heuristic transformations: 12.
   Then, the integral program is done by **JAMES SLAGLE**
5. WHAT exactly?

# 3 Reasoning: Goal Trees and Rule-Based Expert Systems

## 3.1 Goal Centered Programming

### 3.1.1 Goal

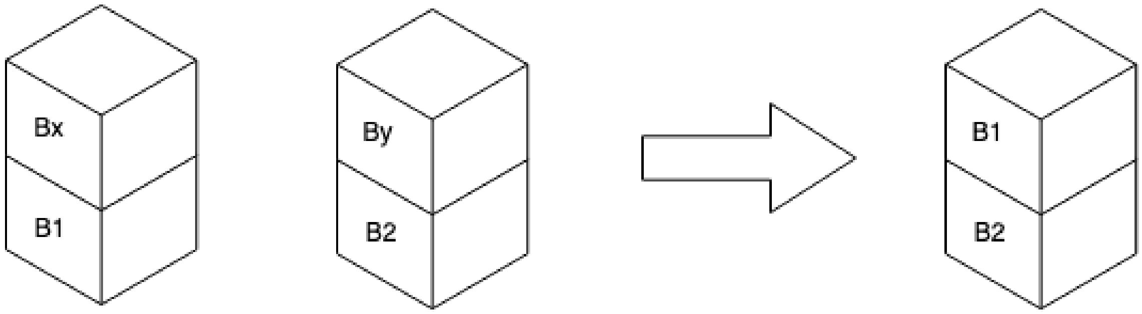



Figure 3: Put $B_1$ on $B_2$

### 3.1.2 Function

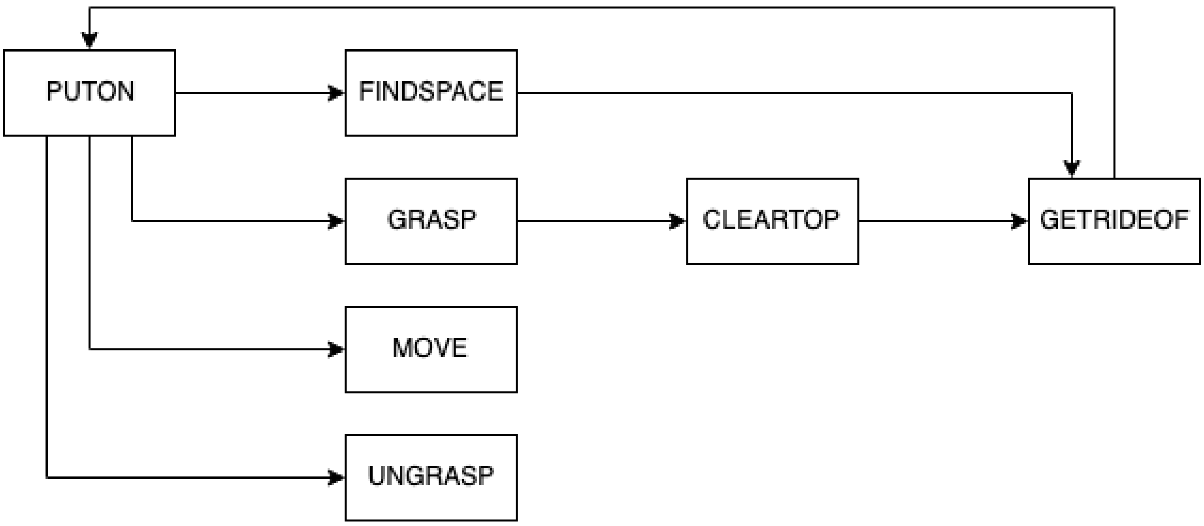



Figure 4: Related functions

### 3.1.3 Procedure

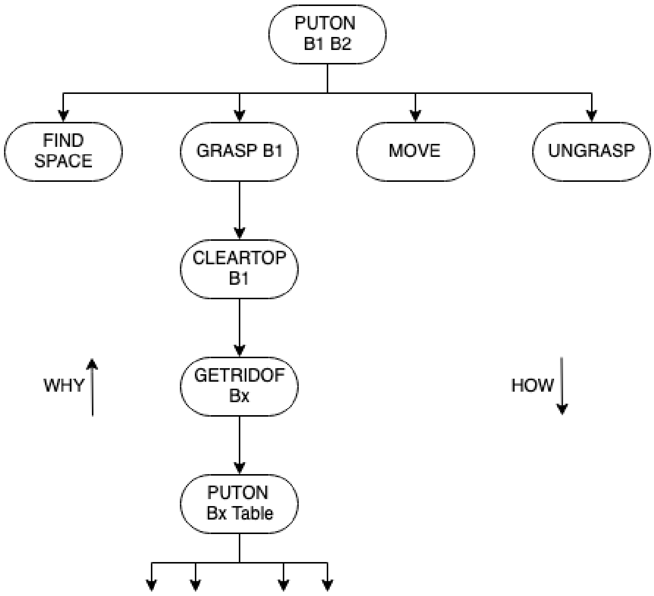



Figure 5: Related functions

For achieve their **GOALS**, the program leaves a **TRACE**, which is a **GOAL TREE**.

Then the program can answer questions about its own behavior as long as it builds an and/or tree.

Simon's ant: The complexity of the behavior is the max of the complexity of program and the complexity of environment.

## 3.2 Rule-Based Expert System

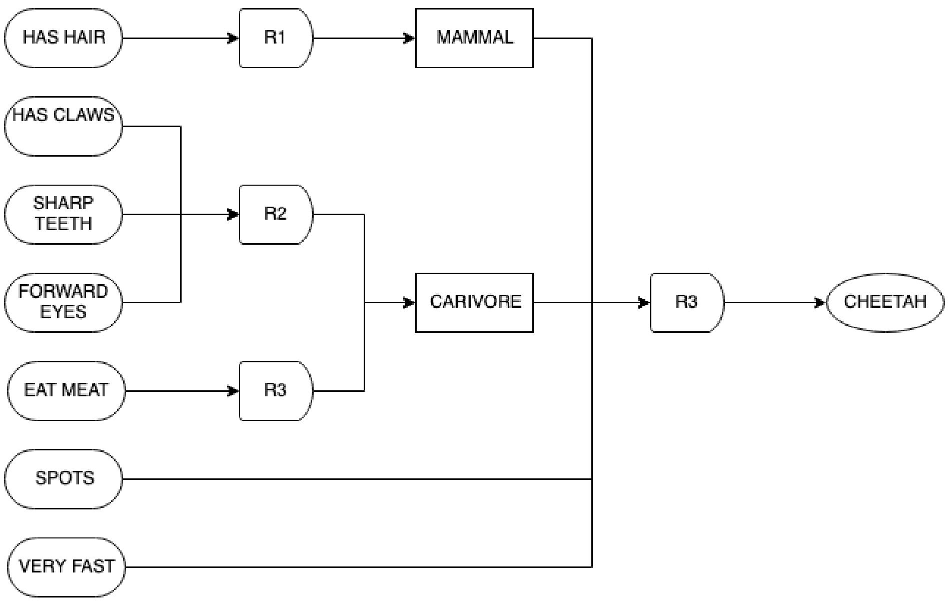



Figure 6: Forward Chaining Rule-Based Expert System

# 4 Search: Depth-First, Hill Climbing, Beam

## 4.1 Question

Find a path from $S$ to $G$.

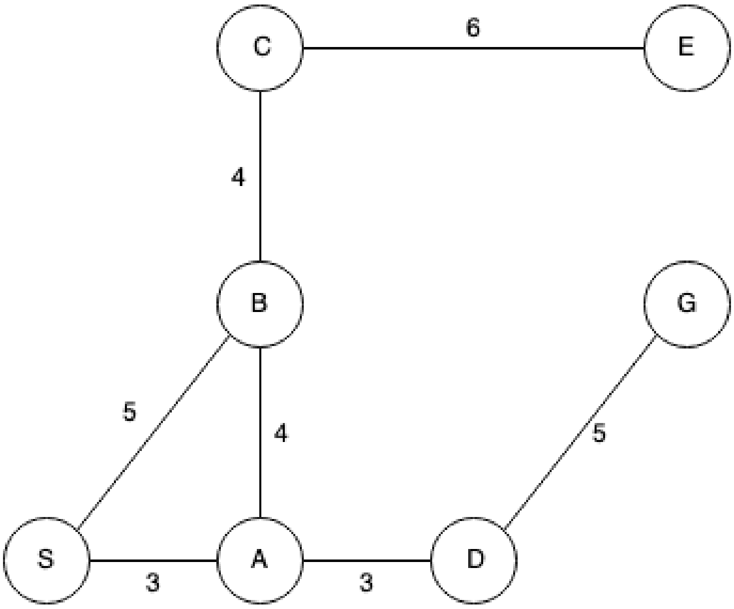



Figure 7: Map

## 4.2 British Museum Approach–find every possible path

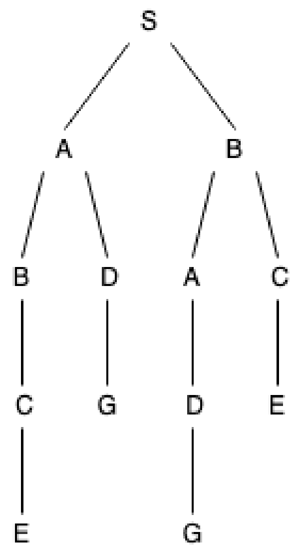



Figure 8: British Museum Algorithm

## 4.3 Depth First Search/Breadth First Search

**e,g: Depth First Search**

(S, A) (S, B)
(S, A, B) (S, A, D) (S, B)
(S, A, B, C) (S, A, D) (S, B)
(S, A, B, C, E) (S, A, D) (S, B)
(S, A, D, G) (S, B)

## 4.4 Hill Climbing

just like depth first search instead of using lexical order to break ties, we're going to break ties according to which node is closer to the goal.

**PROBLEM**

1. Local
2. Telephone Pole Problem
3. Ridge

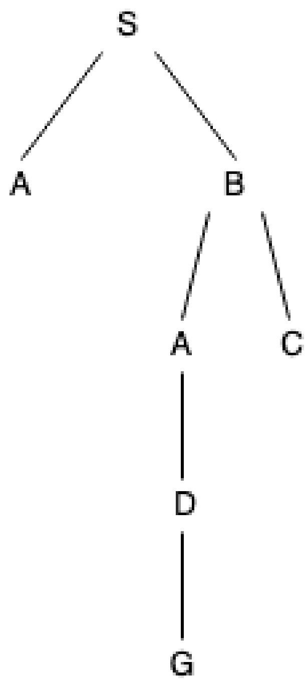



Figure 9: Hill Climbing

## 4.5 Beam Search

complement, addition of and informing heuristic to breadth first search. Limit the number of paths to be considered at any level to some small fixed number.

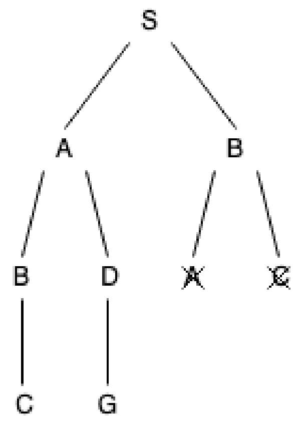



Figure 10: Beam Search (w=2)

## 4.6 Procedure

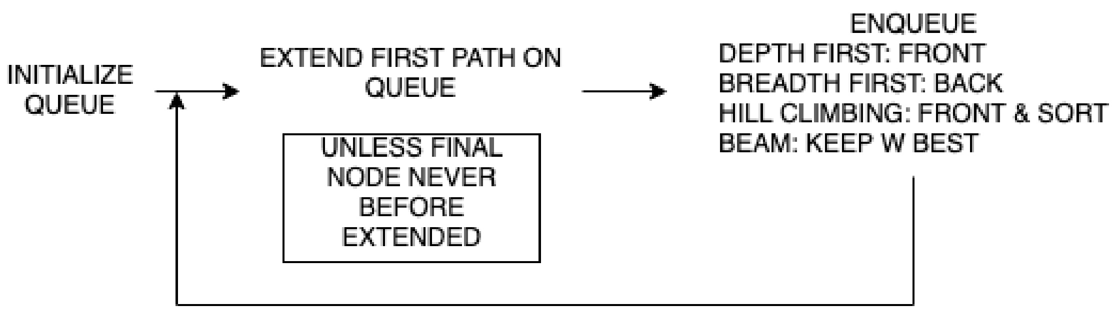



Figure 11: Procedure

## 4.7 Property

| Method | BACKTRACKING | USE ENQEUE LIST | INFORMED |
|---|---|---|---|
| BRITISH MUSEUM | X | X | X |
| DEPTH FIRST | V | V | X |
| BREADTH FIRST | X | V | X |
| HILL CLIMBING | V | V | V |
| BEAM | V | V | V |

# 5 Search: Optimal, Branch and Bound, A*

## 5.1 Question: find the best path

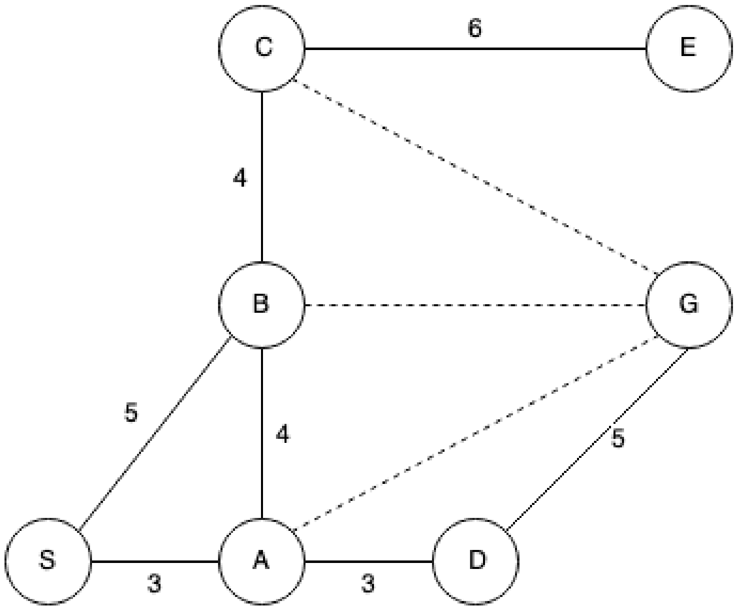



Figure 12: Map with actual distance & airline distance

## 5.2 Branch & Bound

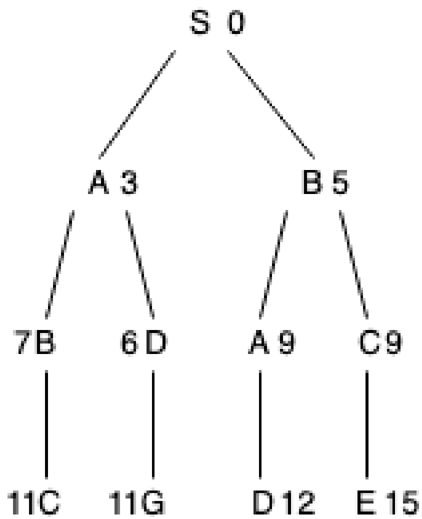



Figure 13: Branch & Bound

## 5.3 with Extended List

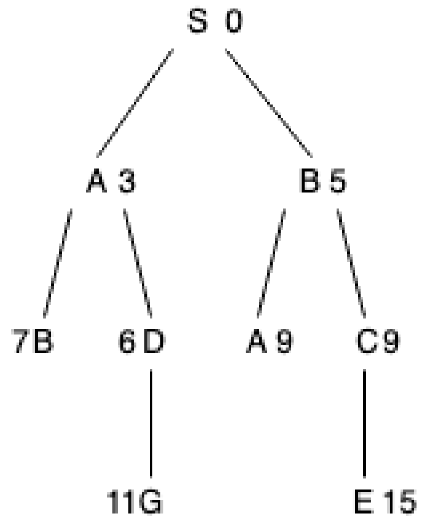



Figure 14: Branch & Bound + Admissible Heuristic

## 5.4 with Admissible Heuristic

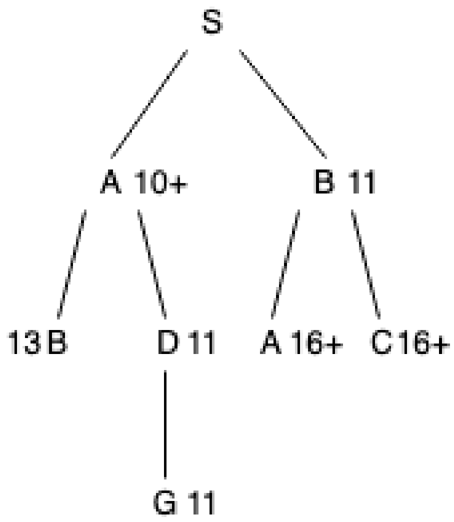



Figure 15: Branch & Bound + Admissible Heuristic

## 5.5 A*


Figure 16: Flow Chart

**ADMISSIBLE:** $H(X,G) \leq D(X,G)$
**CONSISTENCE:** $|H(X,G) - H(Y,G)) \leq D(X,Y)$
'Extended' in this part is the pure pass through. Not same as **Dijkstra's Algorithm**, so **A*** may fail at **non-Euclidean** arrangements

# 6 Search: Games, Minimax, and Alpha-Beta

## 6.1 Ways to Play

1. **Analysis + Strategy + Tactics → Move**
2. **If · Then** Rules
3. Look Ahead + Evaluate

$$\begin{aligned} S =& g(f_1, f_2, \cdots, f_n) \\ =& c_1 f_1 + c_2 f_2 + \cdots + c_n f_n \end{aligned}$$

4. **British Museum Algorithm**
5. *Look Ahead as far as possible

## 6.2 Minimax Algorithm

Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player.

In Minimax the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible.

Every board state has a value associated with it. In a given state if the maximizer has upper hand then, the score of the board will tend to be some positive value. If the minimizer has the upper hand in that board state then it will tend to be some negative value. The values of the board are calculated by some heuristics which are unique for every type of game.

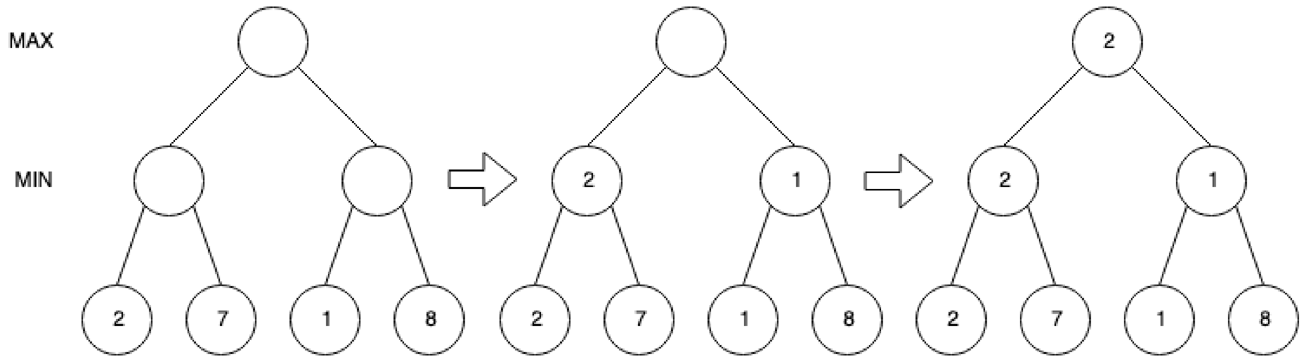



Figure 17: Minimax Algorithm

## 6.3 Alpha-Beta Pruning

Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithm. It cuts off branches in the game tree which need not be searched because there already exists a better move available.

**Alpha** is the best value that the maximizer currently can guarantee at that level or above.

**Beta** is the best value that the minimizer currently can guarantee at that level or above.

Example 1

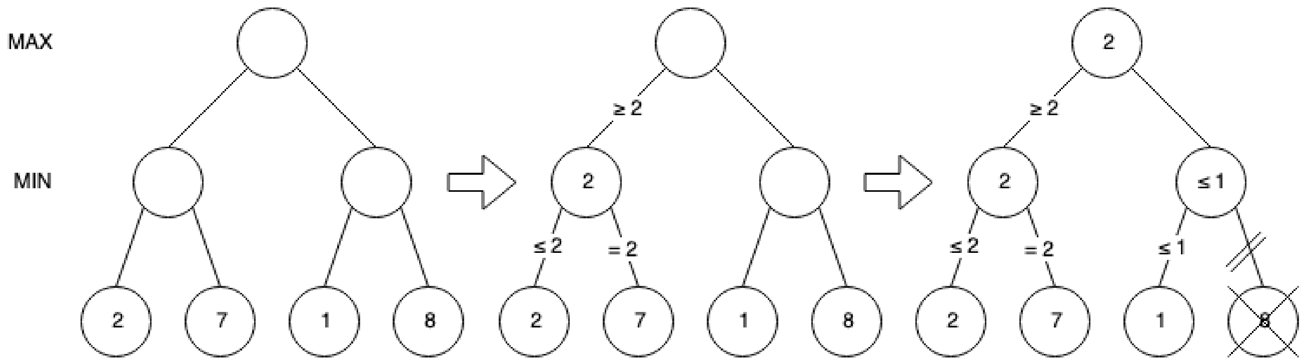



Figure 18: Alpha-Beta Pruning Algorithm

Example 2

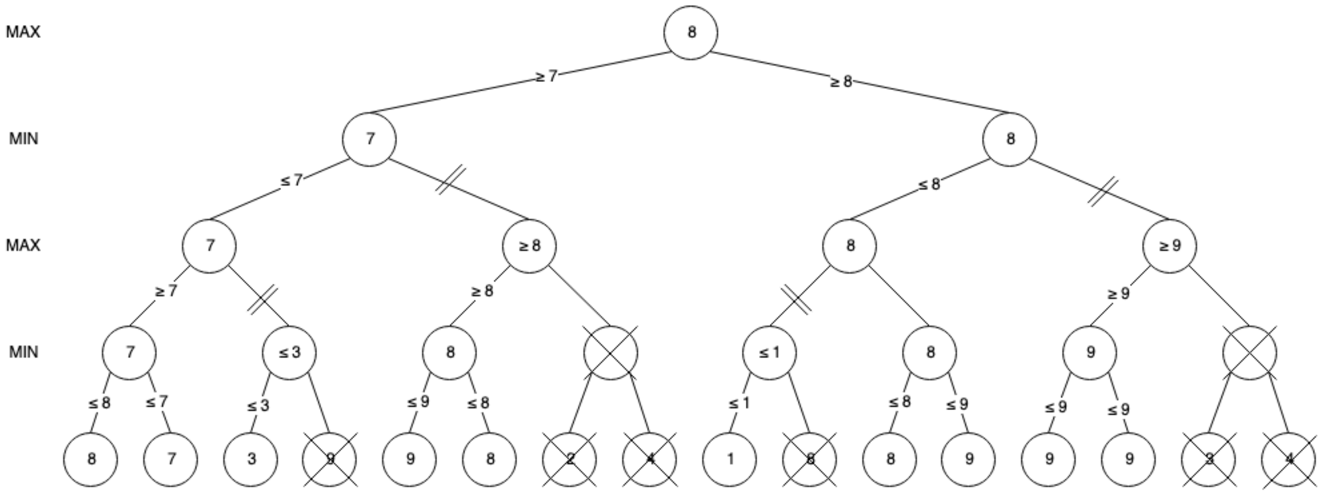



Figure 19: Alpha-Beta Pruning Algorithm

## 6.4 Progressive Deepening–Anytime Algorithm

Start from the very first level,give an insurance policy for every level we try to calculate

$$C = 1 + b + \cdots + b^{d-1} = \frac{1-b^d}{1-b} = \frac{b^d-1}{b-1} \approx b^{d-1}$$

## 6.5 Deep Blue

DEEP BLUE = Minimax + $\alpha - \beta$ + Progressive Deepening + Parallel Computing + Opening Book + Special-purpose Stuff for the End Game + **Uneven Tree Development**

# 7 Constraints: Interpreting Line Drawings

**Object**: Determine how many objects are in a line drawing?

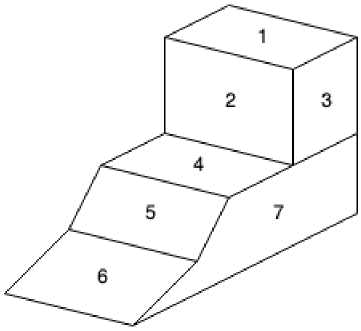



Figure 20: Line Drawing

## 7.1 Guzman's Solution

**Two perspectives**

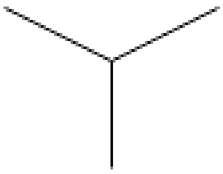


Figure 21: Fork type junction

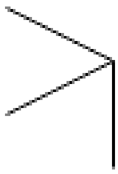


Figure 22: Arrow type junction

1. Fork type junctions: three pairs of faces seem to belong to the same object.
2. Arrow type junctions: faces on either side of she shaft belong to the same object.

**Transform the line drawing to a GRAPH**

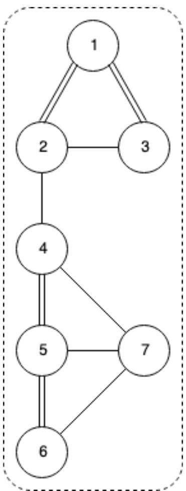



Figure 23: 1 LINK

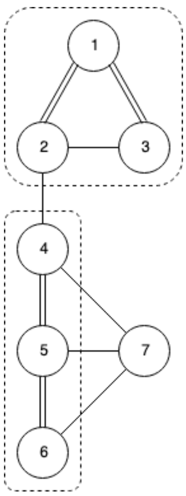



Figure 24: 2 LINK

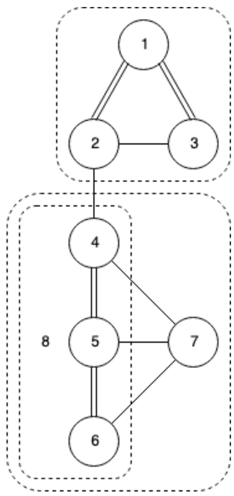



Figure 25: 2 LINK*

## 7.2 Dave Huffman's Solution

**Construct a mathematical model (Three Assumptions)**

1. General Position

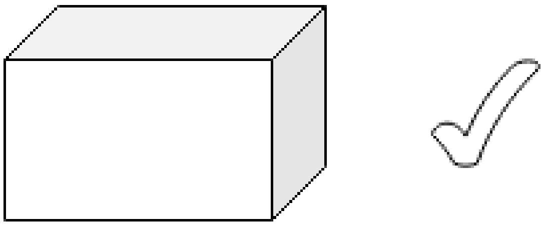


Figure 26: Nice case

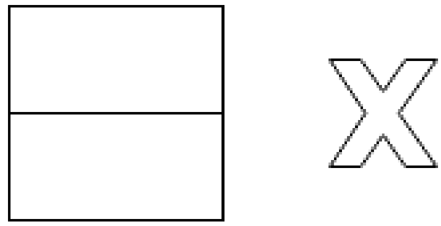


Figure 27: Weird case

2. Trihedral: all vertexes out there are going to be formed from three planes.
3. Four kinds of symbols

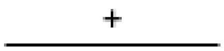



Figure 28: Convex

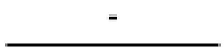



Figure 29: Concave

Figure 30: Boundary

Figure 31: Boundary

***Boundary**: which side you would see the object if you walk along the direction of this arrow

**All possible ways that junctions can have line labels arranged around them**

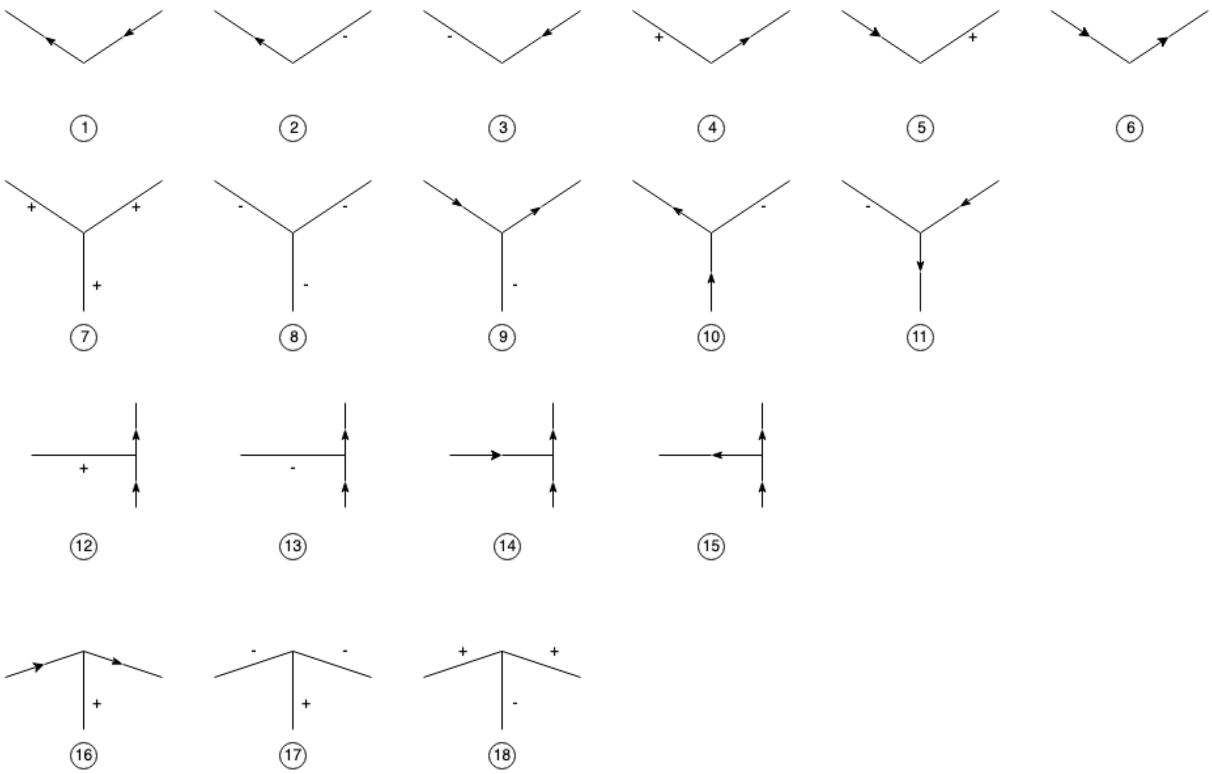


Figure 32: All kinds of junctions

## 7.3 David Waltz's Solution

\+ **CRACKS**
\+ **SHADOWS**
\+ **NON-TRIHEDRAL VERTEXES**
\+ **LIGHT**
Then, 4 labels $\Longrightarrow 50^{+}$ labels, 18 junctions $\Longrightarrow 1000's$ junctions.

**Waltz's Algorithms** (use Huffman's set to demonstrate Waltz's algorithm)

# 8 Constraints: Search, Domain Reduction

**Object: Graph Coloring**

## 8.1 Term

1. **Variable** $v$: something that can have assignment
2. **Value** $x$: something can be an assignment
3. **Domain** $D$: bag of value
4. **Constrain** C: limit on variable values

## 8.2 Procedure–Domain Reduction Algorithm

**for** each DFS assignment **do**
    **for** each variable $v_i$ considered **do**
        **for** each $x_i$ in $D_i$ **do**
            **for** each constraint $C(x_i, x_j)$ where $x_j \in D_j$ **do**
                **if** $\nexists\ x_j \ni C(x_i, x_j)$ satisfied **then**
                    remove $x_i$ from $D_i$
                **end if**
                **if** $D_i$ is empty **then**
                    backup
                **end if**
            **end for**
        **end for**
    **end for**
**end for**

## 8.3 Consider

1. Nothing
2. Assignment
3. Check Neighbors
4. **Propagate** Checking $v$ with $D$ Reduced to 1 Value
5. **Propagate** Checking through $v$ with Reduced $D$
6. Everything

# 9 Constraints: Visual Object Recognition

## 9.1 David Marr's idea–JUST A IDEA

**IDEA**: Based on the idea that you start off by looking at edges and you end up in several steps of transformation, producing something that you can look up in a library of descriptions.

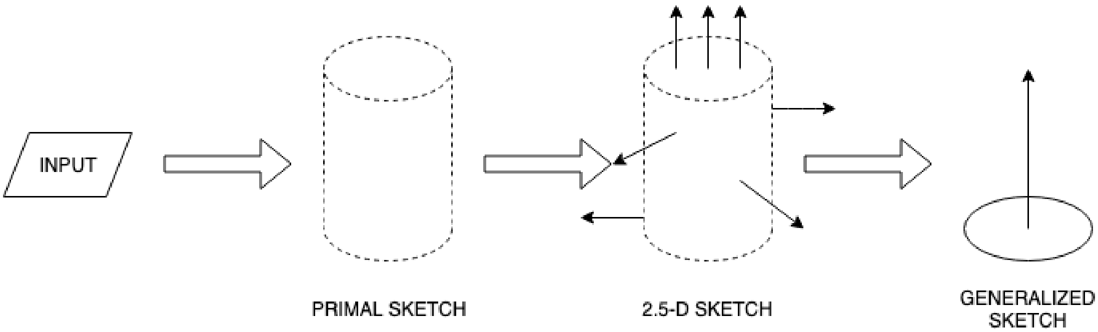


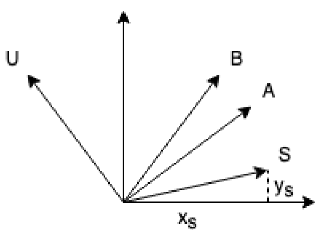

Figure 33: Marr's idea

1. Input from camera
2. Form this edge based description of what's out there in the world
3. Decorate the primal sketch with some vectors, some surface normals, showing where the faces on the object are oriented
4. Transform this 2.5 dimensional sketch to generalized sketch (function+direction)
5. Match against the library of such descriptions, and results in recognition

## 9.2 Shimon Ullman–Alignment Theory

**IDEA**: Assume we have a transparent object where all the vertexes are visible, and neglect the effect of perspectives, then we can reconstruct any view of that object.

$\Theta_A$, $\Theta_B$ and $\Theta_U$ are corresponding angles between $\vec{S}$ and $\vec{A}$, $\vec{B}$, $\vec{U}$. If we only consider 2D rotation, we need two picture as follows.




$$
\begin{aligned}
x_A &= x_S \cos\Theta_A - y_S \sin\Theta_A \\
x_B &= x_S \cos\Theta_B - y_S \sin\Theta_B \\
x_U &= x_S \cos\Theta_U - y_S \sin\Theta_U \\
\Longrightarrow x_U &= \alpha x_A + \beta x_B
\end{aligned}
$$

Figure 34: Example

If include translation element, we need add a constant item $\tau$.

## 9.3 Correlation Theory

$$\max_{X,Y} \int f(x,y)g(x-X, y-Y)$$

To some extends, it's like convolution. However, this course was opened in 2010, but CNN was presented in 2012, so there is not much explanation here.

# 10 Introduction to Learning, Nearest Neighbors

## 10.1 Types of Learning

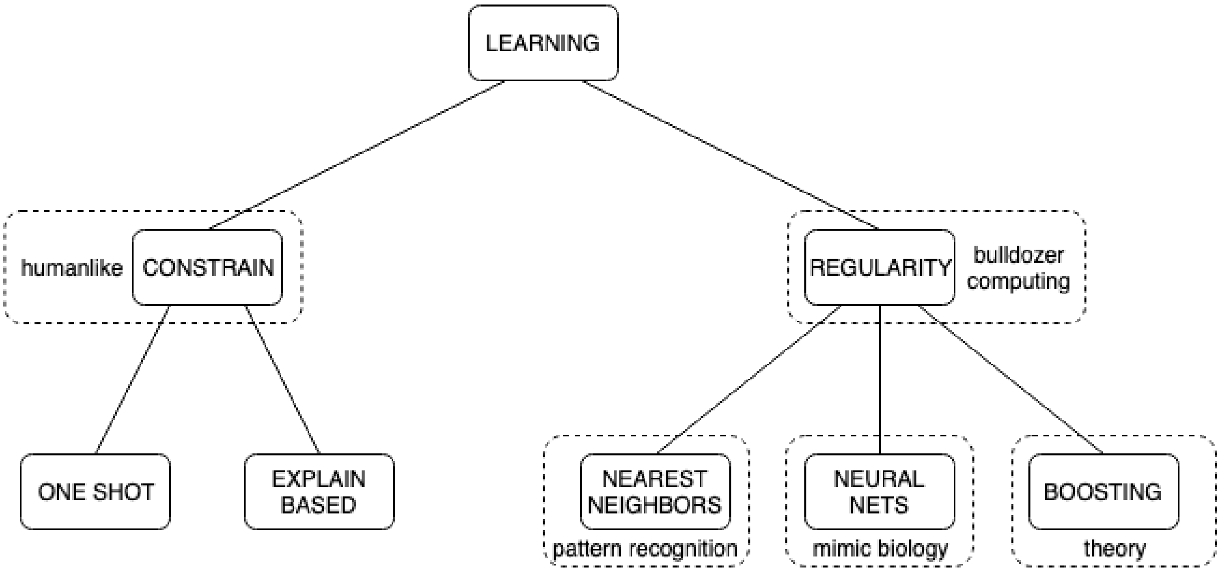



Figure 35: Learning types

## 10.2 Mechanism of Pattern Recognition

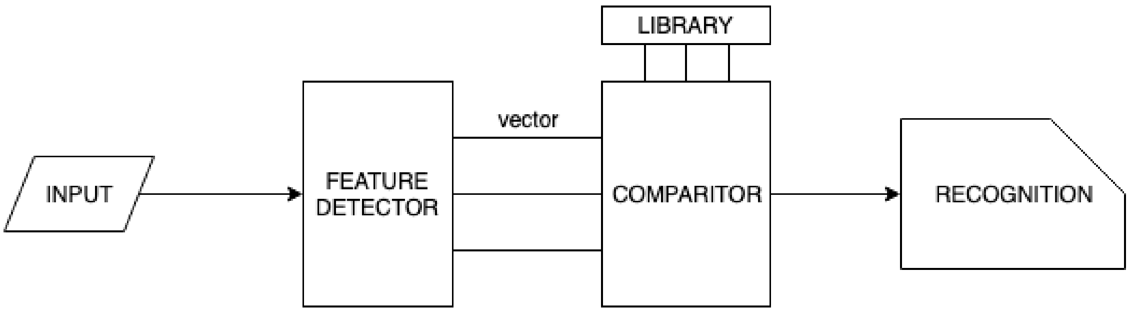



Figure 36: Mechanism of Pattern Recognition

Use Different Distance Metric to Generate Decision Boundary

## 10.3 Learning

Why LEARNING?

# 11 Learning: Identification Trees, Disorder

## 11.1 Target

Use data to build a recognition mechanism

| Vampire | Shadow | Garlic | Complexion | Accent |
|---|---|---|---|---|
| **No** | ? | Yes | PALE | None |
| **No** | Yes | Yes | Ruddy | None |
| **Yes** | ? | No | Ruddy | None |
| **Yes** | No | No | Average | Heavy |
| **Yes** | ? | No | Average | Odd |
| **No** | Yes | No | Pale | Heavy |
| **No** | Yes | No | Average | Heavy |
| **No** | ? | Yes | Ruddy | Odd |

## 11.2 Difference between the Dataset of Identification Tree and Nearest Neighbor

1. Some features are non-numeric
2. Some features don't matter
3. Some features only some of time
4. Some tests Cost
5. *Identification tree is talking in terms of **tests**, not a **vector of real values**

## 11.3 Procedure——Occam's Razor

### 11.3.1 Naive Strategy

Based on the number of sample individuals are put into a homogeneous set.

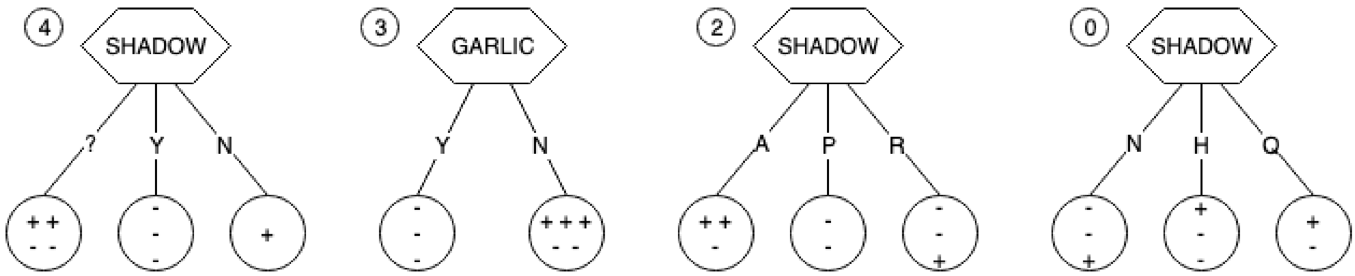



Figure 37: First test

| Vampire | Shadow | Garlic | Complexion | Accent |
|---|---|---|---|---|
| **No** | ? | Yes | PALE | None |
| [redacted] | | | | |
| **Yes** | ? | No | Ruddy | None |
| [redacted] | | | | |
| **Yes** | ? | No | Average | Odd |
| [redacted] | | | | |
| **No** | ? | Yes | Ruddy | Odd |

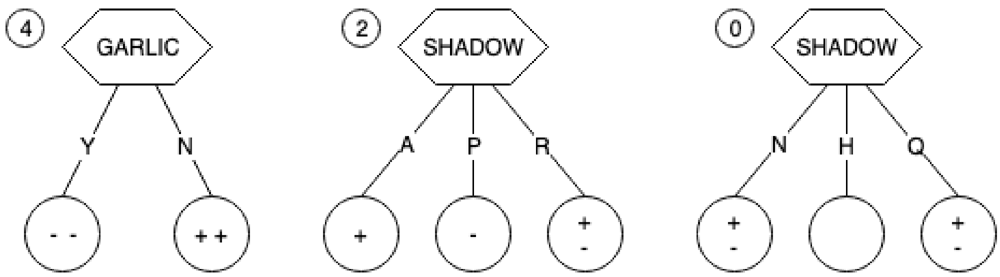



Figure 38: Second test

Then, we can construct a identification tree

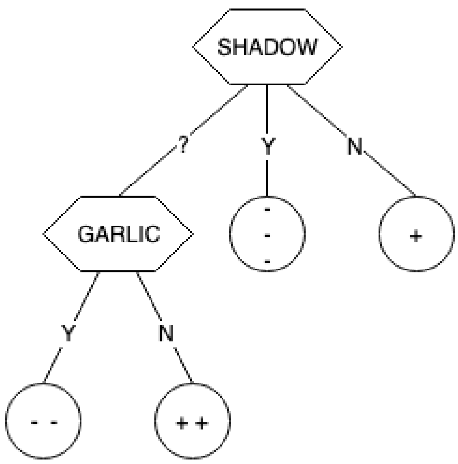



Figure 39: Identification tree

### 11.3.2 Large Dataset

Evaluate the disorder in sets.

$$D(set) = -\frac{P}{T}\log_2\frac{P}{T} - \frac{N}{T}\log_2\frac{N}{T}$$

$$Q(test) = \sum_{setsproduced} D(set)\frac{\#of\ samples\ in\ set}{\#of\ samples\ handled\ in\ test}$$

Then we will get the same answer as shown in Figure 39

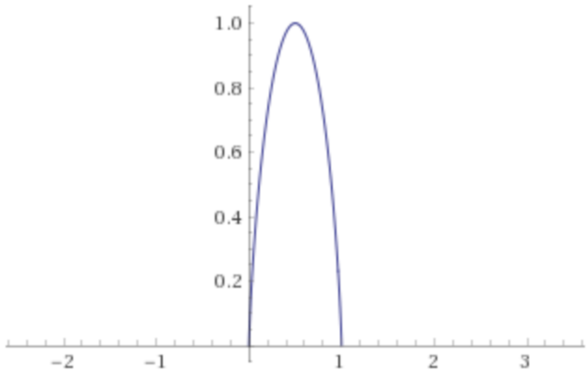



Figure 40: The relationship between $D(set)$ and $\frac{P}{T}$

## 11.4 Decision Boundary

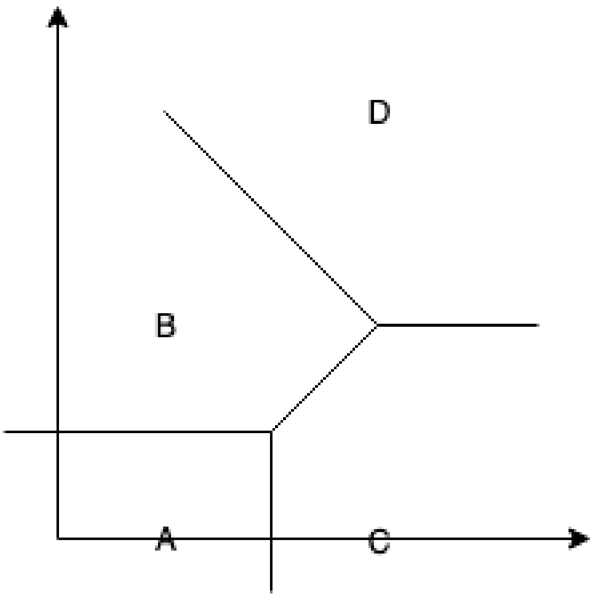



Figure 41: Nearest neighbors

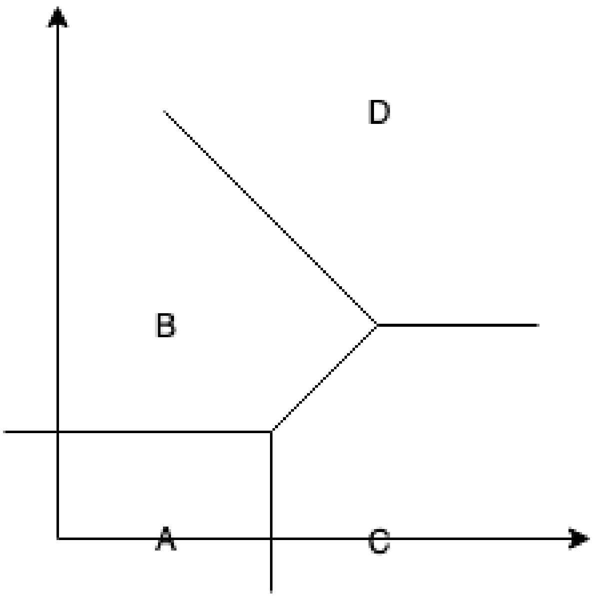



Figure 42: Identification tree

## 11.5 Tree to Rule

Go down each branch to the leaf. Sometimes, we can simplify the description.

# 12 Neural Nets

## 12.1 Model Real Neuron

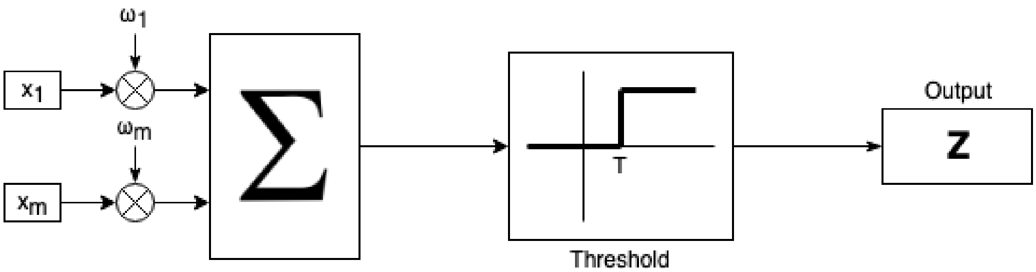



Figure 43: Neural Model

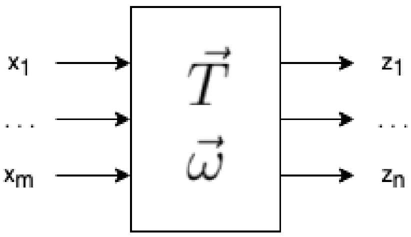
**INCLUDE**

1. All or none
2. Cumulative influence
3. Synaptic weight

**EXCLUDE**

1. Refractory period
2. Axonal bifurcation
3. Time patterns

## 12.2 Training




$$\vec{Z} = f(\vec{x},\ \vec{\omega},\ \vec{T})$$

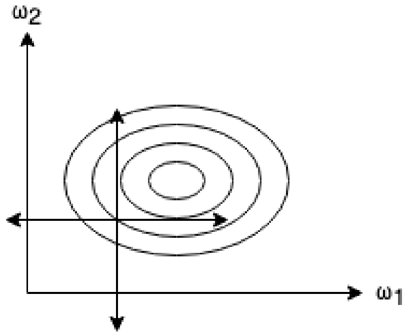
Figure 44: Neural Model

Training a neural net is adjusting these weights and thresholds.

$$Neural\ Net \implies Function\ Approximator$$

**desired**: $\vec{d} = g(\vec{X})$
**performance**: $P = -\frac{1}{2}||\vec{d} - \vec{a}||^2$ (define for mathematical convenient)

## 12.3 Gradient Descent




$$\Delta\vec{\omega} = r(\frac{\partial P}{\partial \omega_1} i + \frac{\partial P}{\partial \omega_2} j)$$

Figure 45: Gradient Descent

**But we cannot use gradient descent/ascent–Discontinuous for our function.**

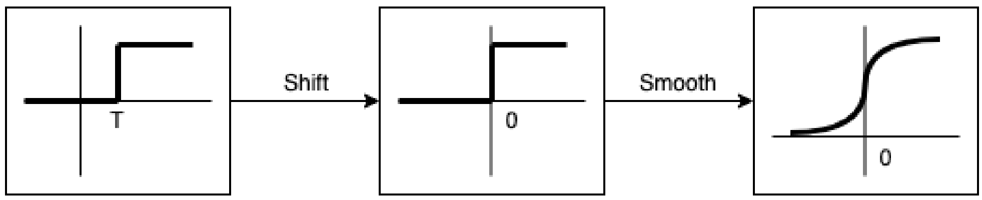



Figure 46: Change threshold function

## 12.4

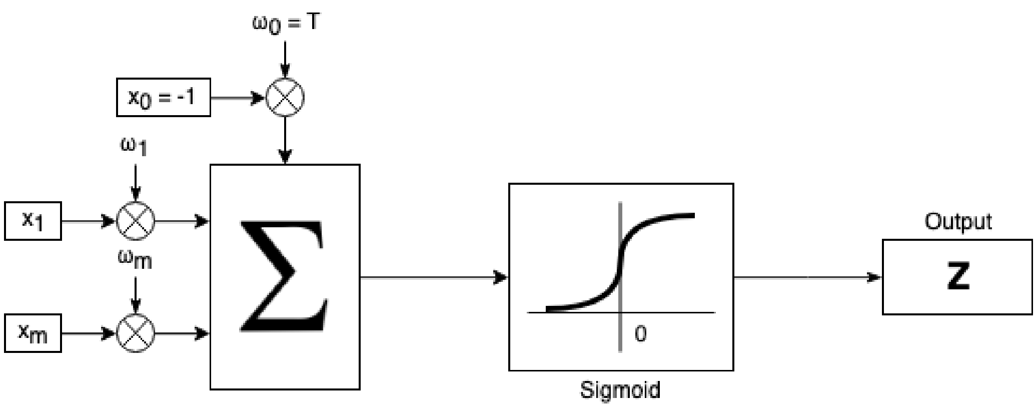



Figure 47: New Neural Model

For sigmoid function $\beta = \frac{1}{1+e^{-\alpha}}$:

$$\begin{aligned}
\frac{d\beta}{d\alpha} =& \frac{d}{d\alpha}(1+e^{-\alpha})^{-1} \\
=& e^{-\alpha}(1+e^{-\alpha})^{-2} \\
=& \frac{e^{-\alpha}}{1+e^{-\alpha}} \cdot \frac{1}{1+e^{-\alpha}} \\
=& \beta(1-\beta)
\end{aligned}$$

## 12.5 Backpropagation Algorithm

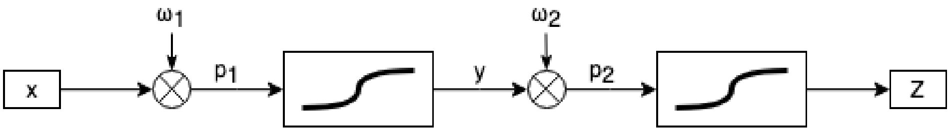



Figure 48: Sample Neural Net

$$\frac{\partial P}{\partial \omega_2} = \frac{\partial P}{\partial z} \cdot \frac{\partial z}{\partial p_2} \cdot \frac{\partial p_2}{\partial \omega_2}$$
$$\frac{\partial P}{\partial \omega_1} = \frac{\partial P}{\partial z} \cdot \frac{\partial z}{\partial p_2} \cdot \frac{\partial p_2}{\partial y} \cdot \frac{\partial y}{\partial p_1} \cdot \frac{\partial p_1}{\partial \omega_1}$$
$$\frac{\partial P}{\partial z} \cdot \frac{\partial z}{\partial p_2} \cdot \frac{\partial p_2}{\partial \omega_2} = y \cdot z(1-z) \cdot (d-z)$$
$$\frac{\partial P}{\partial z} \cdot \frac{\partial z}{\partial p_2} \cdot \frac{\partial p_2}{\partial y} \cdot \frac{\partial y}{\partial p_1} \cdot \frac{\partial p_1}{\partial \omega_1} = x \cdot y(1-y) \cdot \omega_2 \cdot z(1-z) \cdot (d-z)$$

# 13 Learning: Genetic Algorithms

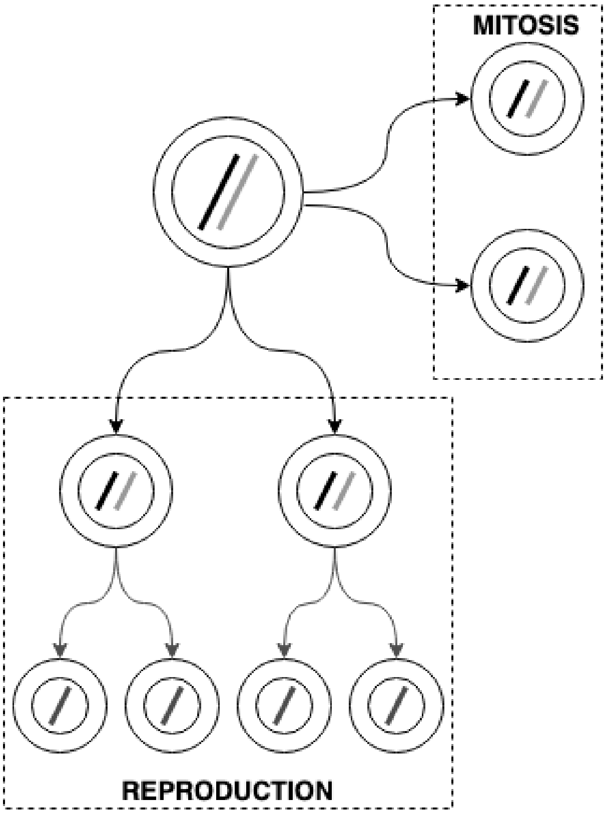



Figure 49: Genetic Biology

## 13.1 Imitation (ATCG $\longrightarrow$ 01)

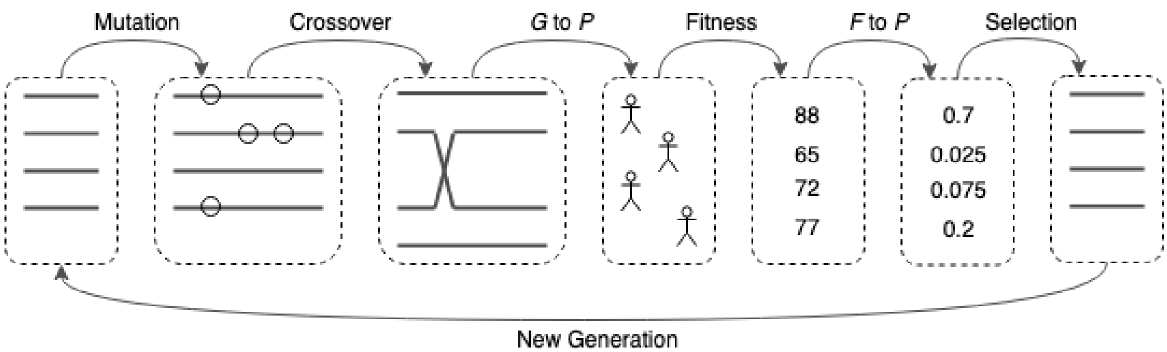



Figure 50: Genetic Algorithm

**G**: Genotype
**P**: Phenotype
**F**: Fitness
**P**: Probability

## 13.2 Mechanism

### 13.2.1 Fitness

$$P_i = \frac{f_i}{\sum_j f_j}$$

$$f(x, y) = (\sin \omega x)^2 (\sin \omega y)^2 e^{-\frac{x+y}{\sigma}}$$

### 13.2.2 Rank Space

$$\begin{aligned} P_1 =& P_c \\ P_2 =& (1 - P_c) P_c \\ & \dots \\ P_{n-1} =& (1 - P_c)^{n-2} P_c \\ P_n =& (1 - P_c)^{n-1} \end{aligned}$$

### 13.2.3 Diversity

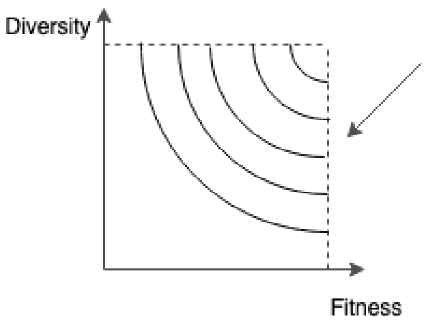



Figure 51: Genetic Biology

# 14 Learning: Sparse Spaces, Phonology

## 14.1 Point

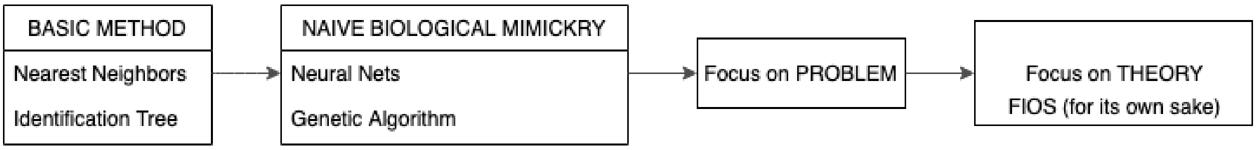



Figure 52: Learning Process

## 14.2 Phonological Rules

### 14.2.1 Distinctive features Theory

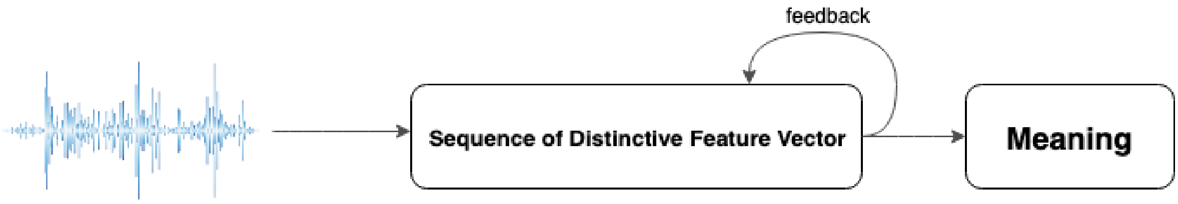



Figure 53: Distinctive features

### 14.2.2 YIP-SUSSMAN Machine/Learner

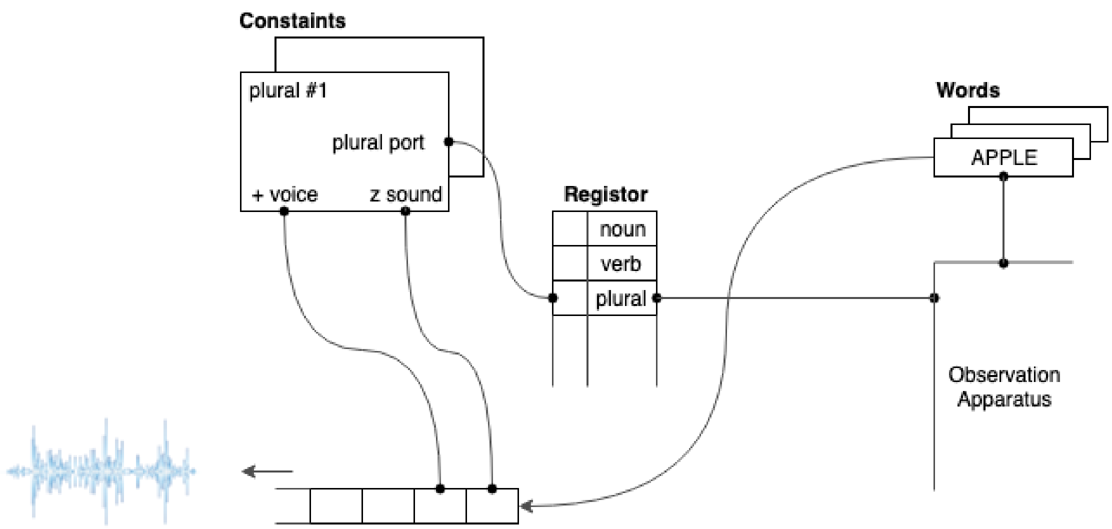



Figure 54: YIP-SUSSMAN Machine

| /æ/ | /p/ | /l/ | /z/ | |
|---|---|---|---|---|
| + | − | − | − | syllabic |
| ... | ... | ... | ... | ... |
| + | − | + | + | voiced |
| + | − | + | − | continuous |
| ... | ... | ... | ... | ... |
| − | − | − | + | strided |

Figure 55: Four distinctive features of "APPLES"

If we present this machine with a pair of apples.

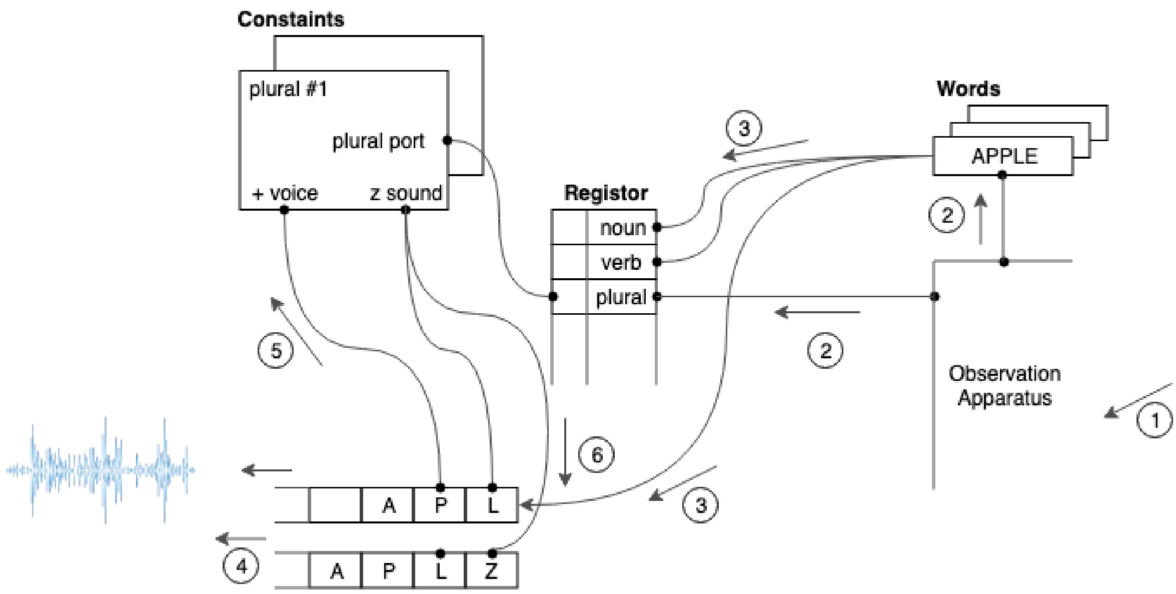



Figure 56: Work flow

1. The vision apparatus comes in and produces the notation, the concept of 2 apples.
2. Information flows from that meaning register to the "APPLE" word; information also flows to the register and mark it as plus plural
3. Information flows from word to register so as to indicate that it's a noun but not a verb; also writes "a", "p", and "l" into the buffer
4. Left flow of the word

## 14.3 Sparse Spaces(!More information need)

1. COLLECT positive(+) and negative(-) examples
2. PICK positive(+) seed
3. GENERALIZE $+ \longrightarrow *$("*" means "Don't care")
4. UNTIL we admit or match a negative example

## 14.4 Marr's Catechism

1. Specify the problem
2. Devise a representation
3. Determine an approach/thought/method
4. Pick a mechanism/Devise a algorithm
5. Experiment

# 15 Learning: Near Misses, Felicity Conditions

## 15.1 Learning Process

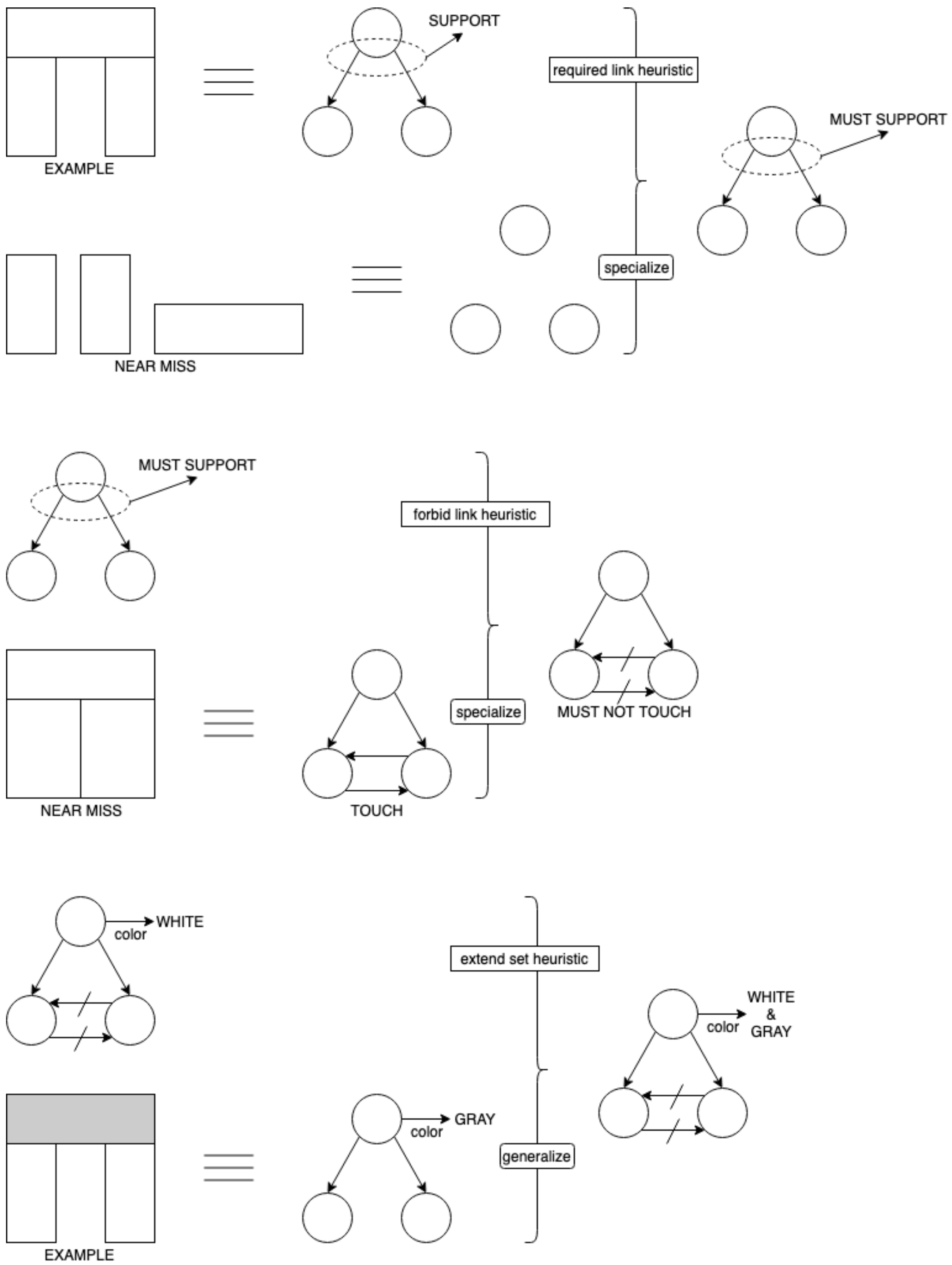



Figure 57: Specialization

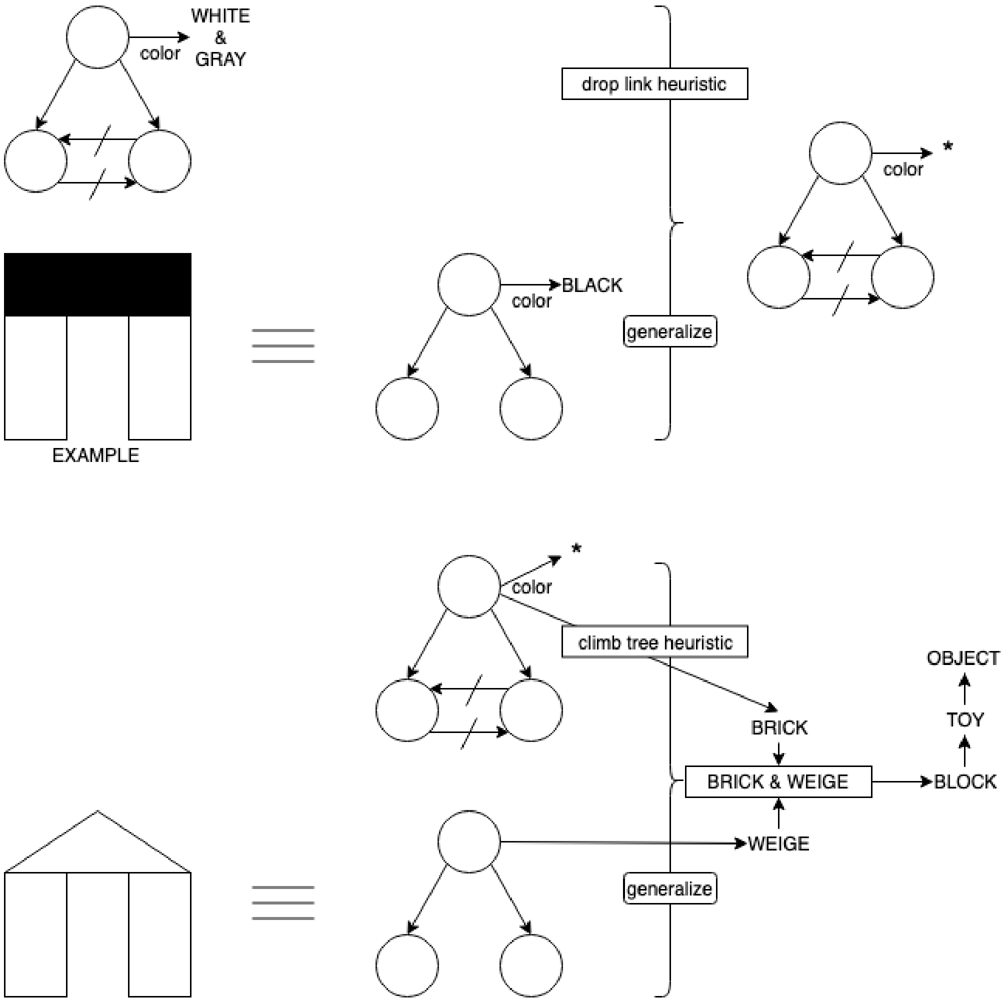



Figure 58: Generalization

## 15.2 Five Qualities

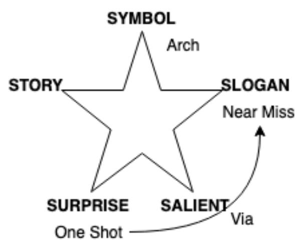



Figure 59: Five "S"

# 16 Learning: Support Vector Machines

## 16.1 Decision Boundary

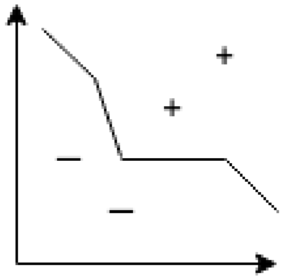



Figure 60: Nearest Neighbors

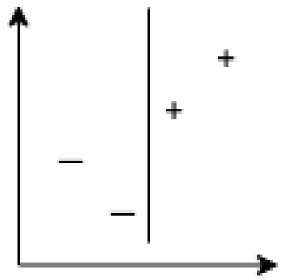



Figure 61: Identification Tree

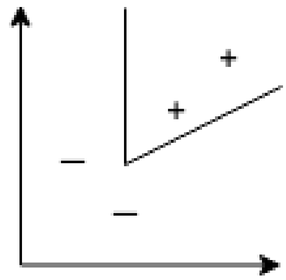



Figure 62: Neural Nets

## 16.2 Vladimir Vapnik's Idea–Support Vector Machine

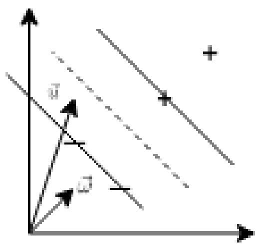



Figure 63: Sample Space

$$\boxed{\vec{w} \cdot \vec{u} \geq c}$$

**Decision Rule: if $\vec{w} \cdot \vec{u} + b \geq 0$, then $u$ is a positive point** $(c = -b)$
Then, we are going to lay on some additional constraints

$$\vec{w} \cdot \vec{x_+} + b \geq 1$$
$$\vec{w} \cdot \vec{x_-} + b \leq 1$$

Define $y_i$ such that $y_i = +1$ for positive samples and $y_i = -1$ for negative samples

$$y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1$$
$$y_i(\vec{w} \cdot \vec{x_i} + b) - 1 \geq 0$$

For $x_i$ located in the gutter

$$\boxed{y_i(\vec{w} \cdot \vec{x_i} + b) - 1 = 0}$$

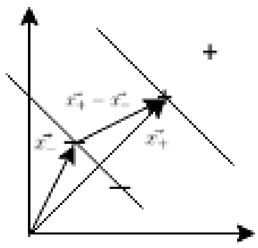


Figure 64: Sample Space

Let $\vec{w}$ be a normal vector

$$\begin{aligned}WIDTH =&(\vec{x_+} - \vec{x_-}) \cdot \frac{\vec{w}}{||\vec{w}||}\\ =&\frac{(\vec{x_+} \cdot \vec{w} - \vec{x_-} \cdot \vec{w})}{||\vec{w}||}\\ =&\frac{(1+b)-(b-1)}{||\vec{w}||}\\ =&\frac{2}{||\vec{w}||}\end{aligned}$$

**MAX** $WIDTH \Longrightarrow$ **MAX** $\frac{2}{||\vec{w}||} \Longrightarrow$ **MAX** $\frac{1}{||\vec{w}||} \Longrightarrow$ **MIN** $||\vec{w}|| \Longrightarrow$ **MIN** $\frac{1}{2}||\vec{w}||^2$
For finding the extremum of a function with constraints, we have to use ***Lagrange Multiplier***

$$\begin{aligned}L =&\frac{1}{2}||\vec{w}||^2 - \sum_i \alpha_i[y_i(\vec{w} \cdot \vec{x_i} + b) - 1]\\ \frac{\partial L}{\partial \vec{w}} =&\vec{w} - \sum_i \alpha_i y_i \vec{x_i} = 0 \Longrightarrow \boxed{\vec{w} = \sum_i \alpha_i y_i \vec{x_i}}\\ \frac{\partial L}{\partial b} =& -\sum_i \alpha_i y_i = 0 \Longrightarrow \boxed{\sum_i a_i y_i = 0}\end{aligned}$$

Plug the expression for $\vec{w}$ back in the expression for $L$

$$\begin{aligned}L =&\frac{1}{2}(\sum_i \alpha_i y_i \vec{x_i})(\sum_j \alpha_j y_j \vec{x_j}) - (\sum_i \alpha_i y_i \vec{x_i})(\sum_j \alpha_j y_j \vec{x_j}) - \sum_i \alpha_i y_i b + \sum_i \alpha_i\\ =&\sum_i \alpha_i - \frac{1}{2}(\sum_i \alpha_i y_i \vec{x_i})(\sum_j \alpha_j y_j \vec{x_j}) - b\sum_i \alpha_i y_i\\ =&\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i \alpha_j y_i y_j \boxed{\vec{x_i} \cdot \vec{x_j}}\end{aligned}$$

**This maximization depends only on the dot product of pairs of samples**
Plug the expression for $\vec{w}$ back in the decision rule

$$\sum_i \alpha_i y_i \vec{x_i} \cdot \vec{u} + b \geq 0$$

**the decision rule also depends only on the dot product of those samples vectors and the unknown point**.

If the samples are not linearly separable

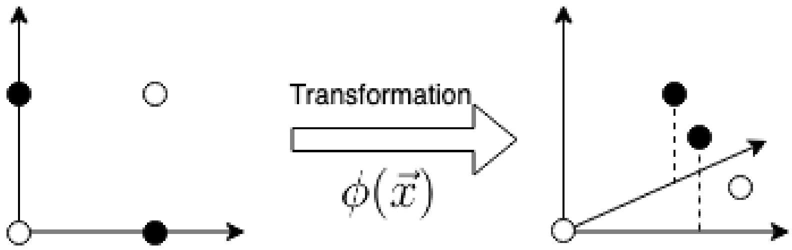



Figure 65: Space transformation

As the optimization only depends on the dot product, we just need to maximize

$$\phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

As the recognition also only depends on the dot product, we just need to maximize

$$\phi(\vec{x_i}) \cdot \phi(\vec{u})$$

What the matter is the dot product, even though we do not know the specific transformation. All we need is a ***Kernel Function***

$$K(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

## 16.3 Kernel Function

1. Linear kernel
$$(\vec{x_i} \cdot \vec{x_j} + 1)^n$$

2. Radial basis kernel
$$e^{-\frac{||\vec{x_i} - \vec{x_j}||}{\sigma}}$$

# 17 Learning: Boosting

## 17.1 Question

Consider **Binary Classification** problem. Assume there is a set of classifier

$$h \to [-1, +1]$$

The error rate of this classifier will range from 0 to 1 in terms of the fraction of the cases got wrong on a sample set

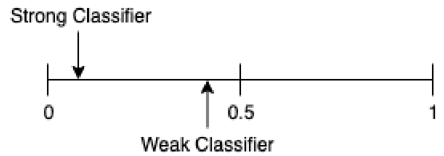



Figure 66: Error rate

**If it possible to make a strong classifier by combining several weak classifiers, and letting them "VOTE"**

$$H(x) = sign[h^1(x) + h^2(x) + h^3(x)]$$

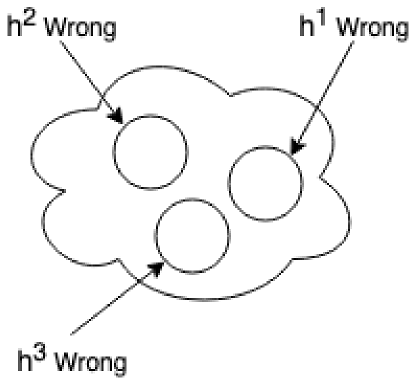



Figure 67: Good case

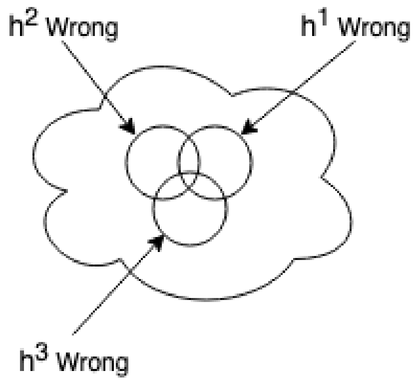



Figure 68: Bad case

**Strategy**

$$\begin{aligned} &\textbf{DATA} && \Longrightarrow h^1 \\ &\textbf{DATA} \text{ with an exaggeration of } h^1 \text{ errors} && \Longrightarrow h^2 \\ &\textbf{DATA} \text{ with an exaggeration of } h^1 \neq h^2 && \Longrightarrow h^3 \end{aligned}$$

## 17.2 Get Out The Vote

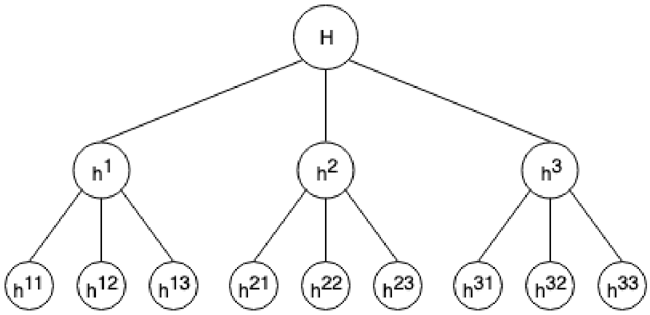



Figure 69: Get out the vote

## 17.3 Decision Tree Stumps

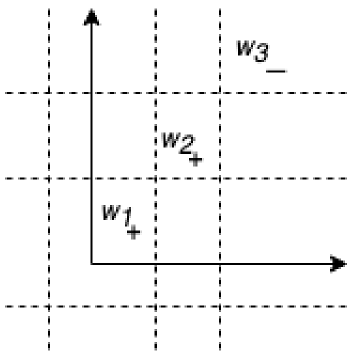



Figure 70: Decision tree stumps

$$\text{ERROR RATE: } \epsilon = \sum_{wrong} w_i \ (\sum_i w_i = 1)$$

In the beginning there's no reason to suppose that any one of these is more or less important than any of other, so $w_i = \frac{1}{N}$, the boosting idea is to find a way to change the weights going downstream

$$H(x) = sign[\alpha^1 h^1(x) + \alpha^2 h^2(x) + \alpha^3 h^3(x) + \cdots]$$

**WISDOM OF WEIGHTED CROWD OF EXPERTS**

## 17.4 Algorithm

1. Let $w_i^1 = \frac{1}{N(\#\text{ samples})}$
2. **REPEAT** Pick $h^t$ that minimizes $\epsilon^t$, then get $\alpha^t$
3. Calculate $w^{t+1}$
4. **UNTIL** This classifier produces a perfect set of conclusions on all the sample data

## 17.5 Mathematic Derivation

Suppose

$$w_i^{t+1} = \frac{w_i^t}{Z} e^{-\alpha^t h^t(x) y(x)}$$

$$\begin{aligned} w_i^{t+1} &\longrightarrow \text{the weight on the } ith \text{ sample at time } t+1 \\ w_i^t &\longrightarrow \text{the weight on the } ith \text{ sample at time } t \\ Z &\longrightarrow \text{normalizer} \\ `y(x) &\longrightarrow \text{function depends on } x \end{aligned}$$

Minimum error bound for whole thing if

$$\alpha^t = \frac{1}{2} \ln \frac{1-\epsilon^t}{\epsilon^t}$$

Plug the expression of $\alpha^t$ back to the above equation

$$w_i^{t+1} = \frac{w_i^t}{Z} \times \begin{cases} \sqrt{\dfrac{\epsilon^t}{1-\epsilon}}, \text{CORRECT} \\ \sqrt{\dfrac{1-\epsilon^t}{\epsilon}}, \text{WRONG} \end{cases}$$

For normalize

$$\begin{aligned} Z =& \sqrt{\frac{\epsilon^t}{1-\epsilon}} \sum_{CORRECT} w_i^t + \sqrt{\frac{1-\epsilon^t}{\epsilon}} \sum_{WRONG} w_i^t \\ =& \sqrt{\frac{\epsilon^t}{1-\epsilon}} \cdot (1-\epsilon) + \sqrt{\frac{1-\epsilon^t}{\epsilon}} \cdot \epsilon \\ =& 2\sqrt{\epsilon^t(1-\epsilon^t)} \end{aligned}$$

Plug the expression of $Z$ back to the above equation

$$w_i^{t+1} = \frac{w_i^t}{2} \times \begin{cases} \dfrac{1}{1-\epsilon}, \text{CORRECT} \\ \dfrac{1}{\epsilon}, \text{WRONG} \end{cases}$$

Then,

$$\begin{aligned} \sum_{CORRECT} w_i^{t+1} &= \frac{1}{2} \frac{1}{1-\epsilon} \sum_{CORRECT} w_i^t = \frac{1}{2} \\ \sum_{WRONG} w_i^{t+1} &= \frac{1}{2} \frac{1}{\epsilon} \sum_{WRONG} w_i^t = \frac{1}{2} \end{aligned}$$

## 17.6 Thank God Hole

1. Test Reduction

2. May not overfit, one reasonable explanation is that these decision tree stumps tend to wrap themselves so tightly around the error points so there is no room for overfitting because nothing else will fit in that same volume.

# 18 Representations: Classes, Trajectories, Transitions

# 19 Architectures: GPS, SOAR, Subsumption, Society of Mind

# 20 Probabilistic Inference I

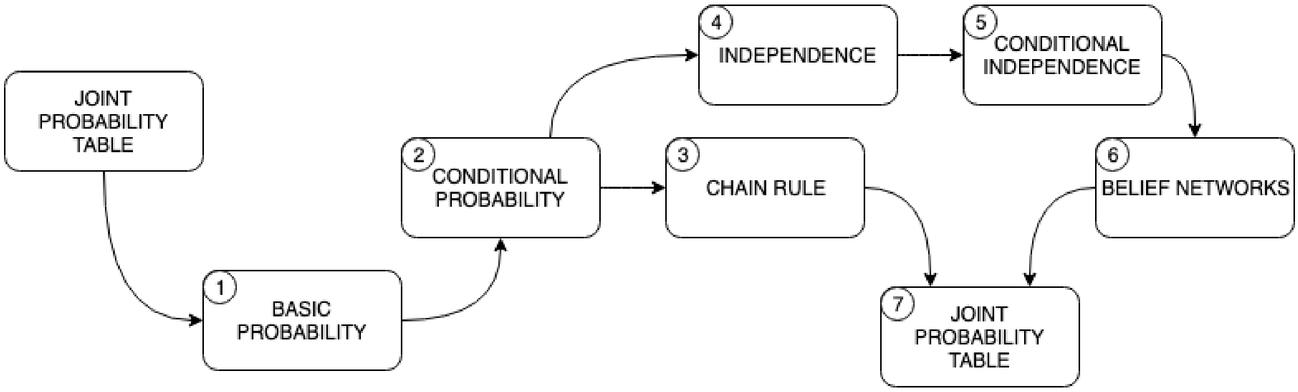



Figure 71: Procedure

## 20.1 Basic Probability (AXIOM)

1. $0 \leq P(a) \leq 1$
2. $P(TRUE) = 1$
   $P(FLASE) = 0$
3. $P(a) + P(b) - P(a,b) = P(a \vee b)$

## 20.2 Conditional Probability (DEFINITION)

$$P(a|b) = \frac{P(a,b)}{P(b)}$$
$$P(a,b) = P(a|b)P(b)$$
$$P(a,b,c) = P(a|b,c)P(b,c) = P(a|b,c)P(b|c)P(c)$$

## 20.3 Chain Rule

$$\boxed{P(x_1, x_2, \cdots, x_n) = \prod_{i=n}^{1} P(x_i|x_{i-1}, \cdots x_1)}$$

## 20.4 Independence

If $a$ independent of $b$

$$P(a|b) = P(a)$$

## 20.5 Conditional Independence

$$P(a|b,z) = P(a|z)$$
$$P(a,b|z) = P(a|z)P(b|z)$$

## 20.6 Belief Networks

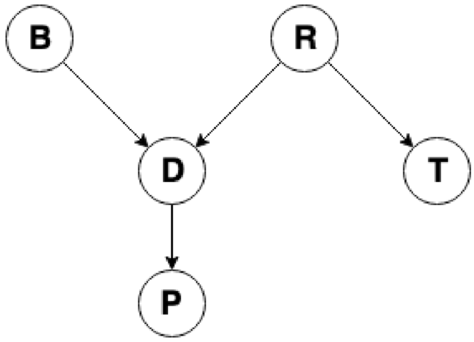



Figure 72: Procedure

Given its parents, every node is independent of all other non-descendants
1. Chew away those variables from the bottom and build a linear list of all these variables

$$(P, D, B, T, R)$$

the way to construct this list ensures that list arranges the elements so that for any particular element, none of its descendants appear to its left.
2. Use chain rule

$$\begin{aligned} P(p,d,b,t,r) =& P(p|d,b,t,r)P(d|b,t,r)P(b|t,r)P(t|r)P(r) \\ =& P(p|d)P(d|b)P(b)P(t|r)P(r) \end{aligned}$$

## 20.7 Naive Bayes' Inference

$$P(a|b) \equiv \frac{P(a,b)}{P(b)}$$
$$P(a|b)P(b) = P(a,b) = P(b|a)P(a)$$
$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

where $a$ is the class, $b$ is the evidences

$$P(C_i|E) = \frac{P(E|C_i)P(C_i)}{P(E)} = \frac{P(e_1, \cdots, e_n|C_i)P(C_i)}{d}$$

# 21 Model Merging, Cross-Modal Coupling, Course Summary