EXPERT SYSTEMS

- if-then rules → production rules → production systems
- reduction: general knowledge → expert knowledge → expert systems
  ↓
  domain !

- production systems ⟺ Turing machine (all solvable problems)

- production systems = rule based systems = expert systems
  - advantages: more than one expert
    - can be multiplied and widely used
  - production rules: if condition/state/premise/antecedent
    then action/conclusion/consequent

- production system:
  1. set of production rules
     (1 PR = 1 chunk of information)
  2. working memory
     - contains facts (current state of the world)
       - current state in the procedure of solving the probl

  knowledge base

  3. match-act cycle
     - control structure of a production system
     - cycle: pattern matching - choosing the rule by resolving conflicts - applying the rule
     - if conditional part matches samples in working memory, rule is added to set of conflicted rules. If >1 rules in the set → choose one = conflict resolution and apply it - WM fire it

- chain - sequence of several conclusions that link problem & solution

Forward chaining

- starting from the known and advancing
- ideal for small amount of data & lots of possible solutions
- starts w/ known data - problem description
  - search through all rules → if new data can be derived → derive!
  - ⇒ BFS
- rule interpretation = forward inference = forward chaining
- inference engine: matching - left sides of production rules
  - conflict resolution - if >1 enabled rules, rule w/ higher priority is chosen
  - firing a rule - applying it → new fact to WM
    - → new rule to the rulebase (note that it is used..?)

Backward chaining

- choosing possible conclusion (hypothesis) and trying to prove it
- goal driven processing
- ideal for small amount of possible conclusions & large data
- starts w/ empty list of facts - list of goals is given for which the system tries to determine the values

1 Form a stack of goals that we want to prove
2 The top is to be proven. If empty → STOP
3 Find all rules which can derive current goal (have goal on the right side)
4 For each such rules:
  a) if all premises are satisfied → apply the rule → add conclusion to WM remove goal from stack to go back
  b) if not all premises are satisfied → do not apply the rule
  c) if a value is missing → find a rule whose right side derives it → → if ∃ → set that as intermediate goal on stack → back to 2
  d) if in c) ∄ → ask the user and add to WM → go to 4.a to next premise
5 If all rules are checked and ~~peen~~ goal can't be derived → underived → remove the goal & go back to 2

Bayes rule: $p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$

$p(x)$ ↓ $p(x|y)p(y) + p(x|\sim y)\,p(\sim y)$

if H then E with p ⇒ $p(E|H) = p$

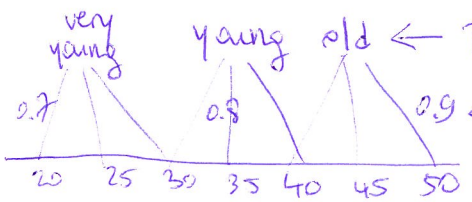
Assumption: facts E_i are mutually independent given any H

FUZZY LOGIC

- two laws don't apply: excluded middle & contradiction
- the extent to which an element belongs to a set

Classical set $\mu_A : X \to \{0, 1\}$

Fuzzy set $\mu_A : X \to [0, 1]$



concentration = "very" → $Con(A) = \mu_A(x)^2$

dilation = "more or less" → $Dil(A) = \mu_A(x)^{1/2}$

$\forall x \in X$, $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$ ("OR")

$\forall x \in X$, $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$ ("AND")

Negation: $\mu_{A^c}(x) = 1 - \mu_A(x)$

In fuzzy logic - generalized modus ponens

If A then B → fuzzy relation A×B on X × Y

Composition A∘R yields B. B = conclusion

$\mu_{A \times B}(x, y) = \min(\mu_R(x), \mu_R(y))$

Zadeh's rule of min-max comps.

$\mu_{A \times B} : X \times Y \to [0, 1]$

$\mu_B(w) = \max(\min(\mu_A(v), \mu_R(v, w))$

ML - derive new knowledge from examples/data using generalization (induction)

- program = algo + data + knowledge

- learning: supervised - (x, y) → we learn $f(x) = y$
  reinforced
  unsupervised

- supervised l.: pattern recognition - to classify
  regression - learning functions

Naive Bayes Classifier

Bayes rule $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

$P(H|E)$

$P(h_1)$ - a prior of h_1

$P(h_1|A)$ - a posterior of h_1

$P(A|h_1)$ - likelihood of h_1

- "naive" because of the assumption that all atributes are independent given the class

$\Rightarrow P(x|h) = P(a_1, \dots, a_T|h) = \prod_{t} P(a_t|h)$

MAP-hypothesis maximum a posterior hypothesis = h_{MAP}

$h_{MAP} = \underset{h_i \in H}{\operatorname{argmax}} P(h_i|D)$

$h_{MAP} = \underset{h_i \in H}{\operatorname{argmax}} \frac{P(D|h_i) \cdot P(h_i)}{P(D)}$, $P(D)$ is constant

$h_{MAP} = \underset{h_i \in H}{\operatorname{argmax}} P(D|h_i) \cdot P(h_i)$, IF all $P(h_i)$ at equal

$(h_{MAP} = \underset{h_i \in H}{\operatorname{argmax}} P(D|h_i))$

Artificial Neural Network - interconnected simple processing elements (units, nodes) which serve for distributed parallel data processing

ANN - great for nonlinear input-output function
- robust to errors and missing data

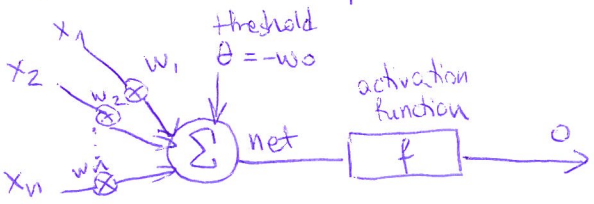
Two types of weight adjustments:
1. on-line learning - for each example
2. batch learning - for each epoch

- the info of input-output mapping is stored implicitly in the neuron weights

Two types of learning:
1. supervised - (input, output)
2. unsupervised - only inputs

- training / validation (tune network parameters) / test set

Artificial neuron

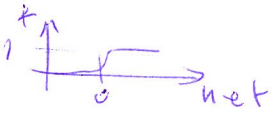


$$net = w_1x_1 + w_2x_2 + \dots + x_nw_n - \theta = \underbrace{w_0}_{-\theta}\underbrace{x_0}_{1} + w_1x_1 + \dots + w_nx_n = \sum_{i=0}^{n} w_i x_i$$

$$o = f(net)$$

Perceptron rule
↳ if classified ✓ → no change
× → apply correction: $w_i(k+1) = w_i(k) + \eta(t-o)\cdot x(k)$

BACKPROPAGATION
sigmoid activation f: $f(net) = \frac{1}{1+e^{-a\cdot net}}$



↳ universal approximator - can app. arbitrary f w/ arb. precision
↳ nonlinear function
↳ must be diff.

$$f'(net) = f(net)\cdot[1 - f(net)]$$

BACKPROPAGATION algorithm

↳ uses gradient descent to minimize output error $E(w)$

D-training set

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

In multilayer networks - output layer can have many neurons:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

Algo:

Initialize weights to random

While termination criterion is not met do

For each example (x, t) from set D do

Compute output o_u for each unit u

For each output unit k compute error δ_k

$$\delta_k \leftarrow o_k (1 - o_k)(t_k - o_k)$$

For each hidden unit

$$\delta_h \leftarrow o_h (1 - o_h) \sum_{s \in Downstream(h)} w_{hs} \delta_s$$

Adjust weights w_{ij}

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

$$\Delta w_{ij} = \eta \delta_j o_i$$

$\eta(t-o)$ | $w_i(k+1) = w_i(k) + \eta(t-o)x(k)$

End for

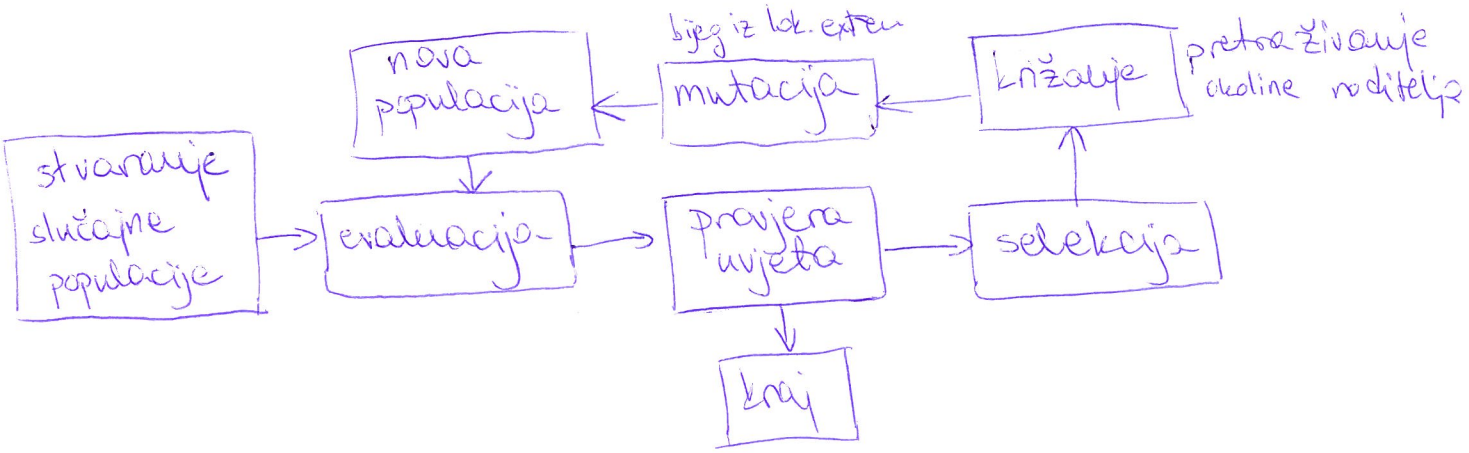
End while

učiti = mijenjati jakosti veza

Prirodom inspirirani optimizacijski algoritmi

GENETSKI ALGO

- svaki kromosom - jedno rješenje
- svako rješenje - svoja dobrota (fitness) ili kazna ima ju



Vrste križanja : s 1 točkom prijeloma
s n točaka prijeloma
uniformno križanje

Mutacija : zadana je vjer. mutacije bita koji okreće bit
može biti velika promjena

Izbor roditelja : proporcionalna selekcija (Rulette-wheel selection)
što je dobja to ima veću šansu biti izabrana

$$probsel(i) = \frac{fit(i)}{\sum_{j=1}^{n} fit(j)}$$

Alg.

```
P = stvori_početnu_populaciju(VEL_POP)
evaluiraj(P)
ponavljaj_dok_nije_kraj
    nova_populacija P' = ∅
    ponavljaj_dok_je_ veličina(P') < VEL_POP
        odaberi R1, R2 iz P
        (D1, D2) = križaj(R1, R2)
        mutiraj D1, mutiraj D2
        dodaj D1 i D2 u P'
    kraj
    P = P'
    evaluiraj(P)
kraj
```

Algoritam kolonije mrava

```
ponavljaj dok nije kraj
    ∀ mravac
    | stvori rješenje
    | vrednuj rješenje
    kraj ∀
    ispari feromonske tragove
    ponovi za_sve_ili_neke mrave
    | ažuriraj feromonske tragove
    kraj ponovi
kraj ponavljanja
```

stvori rješenje: nasumično pri. putove $p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{il}^\alpha \eta_{il}^\beta} & j \in N_i^k \\ 0, & j \notin N_i^k \end{cases}$

vrednuj rješenje: računa ukupnu cijenu puta

ispari tragove: $\tau_{ij} \leftarrow \tau_{ij}(1-\delta)$ skupo ako ima puno bridova!

ažuriraj tragove: $\Delta\tau_{ij}^k = \begin{cases} 1/c^k, & \text{ako je brid } ij \text{ na stazi mravak} \\ 0, & \text{inače} \end{cases}$

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

---

slaba UI - strojevi izgledaju da se ponašaju inteligentno ponašajući pristup

jaka UI - strojevi imaju svjesne umove

↳ funkcionalizam - nisu bitni neuroni, samo procesi-funkcije

kineska soba - nema pravih intencionalnih stanja

računala nemogu: znanje o objektima koji ∃ + semantika

mogu: poznavanje riječi + sintaksa

Intelligence is embodied. It's the result of interaction (behavior) w/ surroundings.