Strategii de cautare

Grafuri reprezentate

- Explicit –
- Implicit – nu se stie numarul de noduri si de arce, se descopera progresiv. graful asociat unei **probleme de cautare**

Reprezentarea solutiei problemei:

- Spatiul starilor
- Graf SI-SAU

Pentru a reprezenta si gasi o solutie:

• Structura simbolica

• Instrumente computationale

• Metoda de planificare

Caracteristicile mediului:

• Observabil / neobservabil – agentul care cauta stie sau nu in ce stare este

• Discret / continuu -

• Finit / infinit

• Determinist / nedeterminist – agentul stie sau nu stie care este starea urmatoarea dupa executarea unei actiuni (in determinist se poate ajunge in mai multe stari, fiecare cu o anumita probabilitate)

Criterii de caracterizare:

• Completitudine

• Optimalitate

• Complexitate: timp, spatiu

• Capacitatea de revenire

• Informare

Cautari neinformate:

- Grafuri **explicite**:
  - BFS, CFS
  - Dijkstra
  - Bellman-Ford
  - Floyd-Warshall
- Grafuri **implicite**:
  - Spatiul starilor
    1. BFS(coada), DFS(stiva), **IDS**
    2. Backtracking
    3. Cautare bidirectionala – trebuie sa avem definite si operatii inverse!
  - Grafuri SI-SAU
    1. BFS, DFS

Complexitatea strategiilor de căutare

Criteriu	BFS	DFS	AdMax	IDS	Bidirectionala
Timp	B^d	B^d	B^m	B^d	$B^{d/2}$
Spatiu	B^d	B*d	B*m	B^d	$B^{d/2}$
Optimalitate?	Da	Nu	Nu	Da	Da
Completa?	Da	Nu	Da daca m≥d	Da	Da

B – factor de ramificare, **d** – adancimea solutiei,
m – adancimea maxima de cautare (AdMax)

Cautari informate (cu euristici, creste eficienta cautarii):

- Best first:
  - Calitatea unui nod este estimata de func de evaluare euristica **w(n)** pentru nodul **n**
  - Exemple: Hill-climbing, best-first
- A*
  - E un fel de best first + Dijkstra
  - w(s) = **f(s)**
    1. **g(s)** estimeaza costul real **g*(s)** intre **S_i si S:** suma costurilor arcelor din **starea initiala** pana **in starea curenta S**
       - **g = g*** cand avem arbore, nu avem cicluri
       - defineste cat de buna e solutia
    2. **h(s)** estimeaza costul real **h*(s)** intre **S si S_f**
       - defineste cat de repede ajungem la solutie
       - trebuie sa fie **admisibila:** pt orice stare S **h(S) < h*(S) & h(S_f) = 0**
       - proprietatea de admisibilitate pentru A*: folosim h admisibil + costurile arcelor >= c
         - garantat gaseste calea de cost minim spre solutie

- daca h2(S) > h1(S) => algoritmul cu h2 e mai **informat**
- h este **consistenta** daca este **h(S) <= h(S') + cost(S, S') & $h(S_f) = 0$**
- h e **monotona** daca **$f(S_j) \geq f(S_i)$** pentru orice **j > i**
- h e **consistenta <=>** h e **monotona**
- h e **consistenta =>** h e **admisibila**
- h este **ε-admisibila** daca **h(S) h (S) * ≤ + ε**, ε > 0
  - se gaseste intotdeauna o solutie al carei cost depaseste costul solutiei optime cu cel mult **ε**
  - **algoritm A* ε -admisibil**
  - **solutie ε -optimala**

3. **f(s) = g(s) + h(s)**

- Se modifica algoritmul best first
- putem utiliza o functie euristica ne-admisibila, dar **ε -admisibil**
- A* ponderat: **f(S) = g(S) + W * h(S)**, W>1 – detour index

Cautari informate cu memorie limitata:

- **Beam search**
  - limiteaza dimensiunea lui OPEN si pastreaza cele mai bune noduri pentru f
  - incompleta
- **DFID -** Depth first iterative deepening cu cost
  - BFS cu o serie de DFS care opereaza pe o frontiera de cautare care creste succesiv
  - **Limita de cost** in loc de AdMax
  - utilizeaza doua limite **U** si **U'** pentru urmatoarea iteratie
  - apeleaza repetitiv functia **DFID** care cauta o cale de cost minim **p**
  - actualizeaza **U'** la valoarea minima a costului cailor generate
  - Daca spatiul de cautare nu contine solutia si este infinit, algoritmul nu se termina
  - functie **iterativa BuclaDFID**
  - functie **recursiva DFID**
  -
- **IDA* -** Iterative deepening A*
  - Bazat pe **DFID cu cost**
  - Garantat sa gasesasca solutia de cost minim
  - functia iterativa **BuclaIDA***
  - functia recursiva **IDA***
- **MA*** - Memory bound A*
- **SMA*** - Simplified memory bound A*
  - Cand ramane fara spatiu elimina nodurile cu costuri mari, dar se tine minte costul

Determinarea euristicii h:

- Manual in functie de problema

sau

- Generare automata de euristici
    - **Transformare abstracta** prin relaxarea unor restrictii
        1. Functie care duce fiecare stare din spatiul de stari S intr-un spatiu abstract S'
        2. d(F(u), F(v)) (**spatiul abstract)** ≤ d(u, v) => aceasta distanta poate fi utilizata ca o **euristica admisibila**
        3. Calculate pe parcursul cautarii sau stocate in **pattern databases**
    - **Pattern databeses**
        1. Memoreaza o colectie de solutii ale unor subprobleme care trebuie rezolvate pentru a rezolva problema
        2. Fiecare solutie de subproblema are o functie euristica precalculata (costul cautarii) si memorata
        3. **Pattern** – o specificare partiala a unei stari
        4. **Target pattern** – a specificare partiala a starii scop
- Exemple:
    - Problema comis-voiajorului
        1. h1(S) = cost(Si, S)
        2. h2(S) = costul arborelui de acoperire de cost minim al oraselor neparcurse pana in starea S
    - 8-puzzle
        1. h1 = nr de patrare care nu sunt la locul lor
        2. h2 = suma dist. Manhattan a fiecarui patrat in starea curenta fata de starea finala
        3. cu relaxare: h3(S) = h2(S) + 3 * T(S), T(S) – scorul pt fiecare patratica din S
        4. h2 > h1

Strategii de cautare

- **Strategii de cautare locala**
  - opereaza asupra starii curente generand succesorii
  - calea catre solutie nu are importanta
  - nu conteaza calitatea solutiei
  - Maxim global, maxim local, platou, umar
  - Folosesc **putina memorie**
  - Gasesc solutii destul de bune in spatii finite f mari si chiar in spatii infinite
  - Algoritmi:
    1. **Hill-climbing** (greedy local search)
    2. **Simulated annealing**
    3. **Local beam search**

- **Strategii de cautare online**
  - Cautare + actiune
  - pentru fiecare actiune, agentul primeste perceptia care ii spune in ce stare a ajuns
  - Metode:
    1. **Online DFS**
       - Agentul nu are un model al mediului, il descopera progresiv
       - Nodurile sunt expandate pe baza locatiei lor
    2. **Programare dinamica asincrona (ADP)**
       - Descompunerea problemei in subprobleme nedistincte
       - **Principiul optimalitatii** – o cale este optima ↔ orice segment (subcale) a acesteia este optima
       - Ordinea de actualizare este arbitrara
       - Spatiu de stari mare – nepractic
       - oloseste ca fundament pt A* in timp real

    3. **Learning Real-Time A* (LRTA*)**
       - Intrepatrunde calculul miscarii urmatoare cu executia miscarilor si alege miscarile intr-un timp constant
       - Construieste si actualizeaza o tabela ce contine estimari eursitice ale distantei fiecarei stari la starea scop
       - Actualizeaza estimarea de cost a starii pe care a parasit-o si alege miscarea (aparent) cea mai bunanitial, intrarile in aceasta tabela corespund unor evaluari euristice sau 0 si sunt subestimari
       - Prin explorarea repetata a spatiului, valorile sunt actualizate pana cand ajung, intr-un final, la valorile corecte
    4. **Cautare cu tinta mobila (MTS)**
       - Generalizare a LRTA*
       - Starea scop se schimba pe parcursul cautarii
       - PS nu are harta mediului dar stie pozitia T sis tie starile adiacente

- **Problema satisfacerii restrictiilor**
  - O serie de variabile
  - O serie de domenii de valori pt fiecare val (discret sau continuu)
  - O serie de restricitii – pot fi specificate **explicit** (R sub forma de val explicite) sau **implicit** (R sub forma de functie)
  - **CSP** (problema de satisfacere a restrictiilor)
    1. **Totala –** satisfac toate restriciile
    2. **Partiala –** nu satisfac toate restrictiile
    3. **Binara –**
  - Reducerea timpului
  - **Backtracking** metoda de baza
  - **Imbunatatiri BKT:**
    1. Algoritmi care modifica spatiul de cautare prin eliminarea unor portiuni care nu contin solutii
       - Inainte de inceperea cautarii -> **propagare locala a restrictiilor**
       - Cauta si elimina in acelasi timp -> **BKT**
    2. Utilizarea **euristicilor** in cautare
  - Cautarea cu **BKT**:
    1. **BKT cu MAC** (Maintaining arc consistency)
       - **Forward checking** – eliminam din domeniu valorile care nu sunt compatibile cu valorile precedente?
    2. **Backjumping**
       - determina nivelul in care apare eventual un conflict
  - **Eursitici**:
    1. Euristici generale
       - Ordonarea variabilelor
         - Aleator
         - Minimum remaining value (MRV) – incepe cu fail-first
         - Degree heuristic
       - Ordonarea valorilor
         - Least constrained value
  - **CSP Partiala**
    1. Memoreaza **cea mai buna solutie** gasita pana la un anumit moment (gen IDA*) – distanta d fata de solutia perfecta
    2. Abandoneaza calea de cautare curenta care nu poate duce la o solutie mai buna
    3. **NI** - numarul de inconsistente gasite in "cea mai buna solutie" depistata pana la un moment dat – limita necesara
    4. **S** - limita suficienta – cate restrictii poate viola
    5. **PBKT**

Strategii de cautare in jocuri

Jocuri cu 2 adversari

- **Minimax**
  - Jucatorul isi maximizeaza castigul si adversarul minimizeaza castigul jucatorului
  - Arbore de joc – AJ
  - Frunzele au scorul final al jucatorilor
  - De la Frunze la radacina, nodurile **MAX** iau valoarea maxima a succesorilor si nodurile **MIN** iau val minima
  - Se poate pana la o adancime
- **Afla-Beta**
  - Imbunatatiere la minimax
  - Se elimina o parte din nordurile din spatiul de cautare => pruning
  - Actualizeaza **α**(cea mai mare val MAX) si **β**(cea mai mica val MIN) si elimina subarborii
    1. Taierea **α**
    2. Taierea **β**
  - Imbunatatiri**:**
    1. **IDS**
    2. **Memoize -** tabela hash cu pozitiile de joc
    3. **Efectul de orizont** – cauta mai mult decat limita de cautare pentru anumite pozitii
- **MCTS** (Monte Carlo Tree Search)
  - algoritm **probabilistic**
  - un fel de **best-first search**
  - metoda buna pentru luarea deciziilor pentru spatiu de cautare mare
  - jucam aleator si dam un “merit” nodurilor parcurse pana cand tinde sa aleaga valorile corecte?
  - 2 ipoteze:
    1. Adevarata valoare a unei mutari poate fi aproximata cu **simulari aleatoare**
    2. Valorile obtinute pot fi utilizate pentru a ajusta **politica de selectie**
  - Baza metodei **– unda de joc** = un **joc rapid** dintr-o anumita stare pana la sfarsit, obtinandu-se **castig/pierdere sau un scor**
  - La fiecare iteratie:
    1. **Selectie** – aleg cel mai interesant nod neexpandat
    2. **Expandare** – il expandez si adaug copiii in arbore
    3. **Simulare** – Simulez de la nodurile noi pentru a obtine un rezultat
    4. **Backpropagation** – propag rezultatul inapoi catre nodurile parcurse si le actualizez valoarea
  - Converge spre valorile Minimax, dar lent
  - Mai eficient decat AlfaBeta
  - **Algoritm anytime**
  -

Jocuri cu mai multi jucatori

- **Max^n** – **Minimax** generalizat
  - Alpha-beta in adancime nu poate fi aplicat
- **Paranoic** – reduce la joc de 2 jucatori, 1 vs all
  - Reduce la 2 jucatori si poate face **minimax**
  - Pe masura ce numarul jucatorilor creste beneficiul adus de taiere scade
  - Presupunerea Paranoic este **foarte pesimista**
  - Cu cat se cauta mai adanc, cu atat e mai proasta estimarea

Q-learning – metoda de invatare a functiei de evaluare

- Foloseste **ecuatia lui Bellman**
- **Deep Q-Learning -** Combina Q-Learning cu DNN

Teoria probabilitatilor

Probabilitate **neconditionata (apriori)** - inaintea obtinerii de probe pt o ipoteza / eveniment

Probabilitate **conditionata (aposteriori)** - dupa obtinerea de probe - **P(A|B)**

- $0 \leq P(A) \leq 1$
- P(S) = 1 (sau P(adev) = 1 si P(fals) = 0)
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- $P(A \vee \sim A) = P(A) + P(\sim A)$
- $-P(fals) = P(adev) \Rightarrow P(\sim A) = 1 - P(A)$

Evenimente:

- **Mutual exclusive** – nu se pot intampla in acelasi timp
- **Exhaustive** - $A \cup \sim A = S$

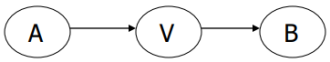
Reguli:

- A si B mutual **exclusive** ➔ $P(A \vee B) = P(A) + P(B)$
- **Regula produsului:** $P(A|B) = P(A, B) / P(B)$ ➔ $P(A, B) = P(A|B) * P(B)$
- **Bayes**:
  - $P(B|A) = P(A|B) * P(B) / \mathbf{P(A)}$ *sau*
  - $P(B|A) = P(A|B) * P(B) / \mathbf{[P(A|B)*P(B) + P(A|\sim B)*P(\sim B)]}$
- **inferenta prin enumerare**: $P(X|e) = \alpha\, P(X, e) = \alpha \sum_{Y=y} P(X, e, \mathbf{Y})$
- $P(X1,X2|Y) = P(X1|Y)*P(X2|Y)$

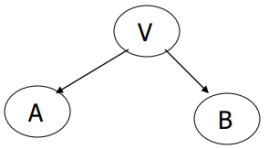
Modelul Bayesian naiv:

$$P(Cauza \mid Efect1, Efect2, \ldots) = \alpha\, P(Cauza) * \prod_i P(Efect_i \mid Cauza)$$

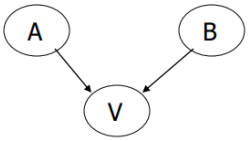
2.4 Inferente probabilistice



$$P(A \wedge V \wedge B) = P(A) * P(V|A) * P(B|V)$$



$$P(A \wedge V \wedge B) = P(V) * P(A|V) * P(B|V)$$



$$P(A \wedge V \wedge B) = P(A) * P(B) * P(V|A,B)$$

Eliminarea variabilelor – se calculeaza de la dreapta la stanga -> ????

Evenimentele X si Y sunt **independente conditional** fiind dat un eveniment Z daca stiind ca Z apare, aparitia lui X nu influenteaza aparitia lui Y si invers

D-separabilitatea este un concept care surprinde astfel de relatii de independenta conditionala derivate

- Z **d-separa** pe X si Y intr-un DAG daca X si Y sunt independente conditional fiind dat Z dpv al axiomelor grafului.
- **(X ⊥ Y | Z)** – X este independent conditional de Y fiind dat Z

- Fie cazul a 3 noduri

Caz a: Efect cauzal

Lantul cauzal $X \rightarrow Z \rightarrow Y$

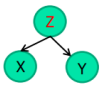
- X nu influenteaza Y via Z daca Z observat

Caz b: Efect evidenta

- $Y \rightarrow Z \rightarrow X$
- Identic cu cazul a:
- X nu influenteaza Y via Z daca Z observat
- **daca (X ⊥ Y | Z) nu este indeplinit atunci nici (Y ⊥ X | Z) nu este indeplinit**

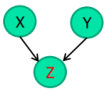
Caz c: Cauza comuna

- $X \leftarrow Z \rightarrow Y$
- X nu influenteaza pe Y via Z daca Z observat



Case d: Efect comun

- $X \rightarrow Z \leftarrow Y$
- Influenta nu poate merge de-a lungul lantului $X \rightarrow Z \leftarrow Y$ daca Z nu este observat



Deci X nu influenteaza pe Y via Z daca Z NU este observat

Daca influenta poate curge de la X la Y via Z spunem ca lantul $X \leftrightarrow Z \leftrightarrow Y$ este <u>activ</u>

- Efect cauzal $X \rightarrow Z \rightarrow Y$ – activ $\leftrightarrow$ Z nu este observat
- Efect evidenta $Y \rightarrow Z \rightarrow X$ – activ $\leftrightarrow$ Z nu este observat
- Cauza comuna $X \leftarrow Z \rightarrow Y$ – activ $\leftrightarrow$ Z nu este observat
- Efect comun $X \rightarrow Z \leftarrow Y$ - activ $\leftrightarrow$ fie Z fie unul din descendentii lui Z este observat

Inferenta prin aproximare -> esantionare:

- Esantionare directa -> Esantionare cu lanturi Markov
  - Se genereaza esantioane dintr-o distributie de probabilitate cunoscuta
  - esantionam fiecare variabila pe rand in ordine topologica
- Eliminarea esantioanelor nepotrivite
  - avem de estimat o probabilitate conditionata P(X|**e**)
  - se elimina acele esantioane care nu se potrivesc cu probele **e**
  - elimina prea multe esantioane
- Esantionare ponderata
  - Elimina ineficienta eliminarii esantioanelor prin generarea numai a esantioanelor care sunt consistente cu probele e
  - Inainte de a le numara, fiecare eveniment este ponderat cu o **plauzibilitate** (likelihood) pe care esantionul o acorda probei
  - **Plauzibilitatea** (likelihood) este masurata ca produsul probabilitatilor conditionate pentru fiecare variabila proba, fiind dati parintii ei

Invatare automata. Arbori de decizie

Învățare supervizată – determinarea ipotezei de invatare pe baza unor date **etichetate**

- Simbolica sau non-simbolica

Învățare nesupervizată - determinarea ipotezei de invatare/a unei structuri pe baza unor date **neetichetate**

- Simbolica sau non-simbolica

Învățarea prin recompense

Metode de invatare supervizata

- **Arbori de decizie**
- **Clasificare probabilistica (Naïve Bayes)**
- K-Nearest neighbours (k-NN pt clasificare)
- SVM
- Retele neurale
- Invățare mulțime (ensemble learning): **păduri aleatoare, ADA boost**

Regresie

- Regresie liniara
- SVR
- Arbori de regresie
- K-Nearest neighbours (k-NN pt regresie)

Clasificare vs regresie

- Clasificare: iesirea algoritmului e o **eticheta** (clasa): arbori, retele neurale, etc.
- Regresie: reprezentam ipoteza de invatare ca o **functie liniara**

Metode de invatare nesupervizata

- Clustering
  - **k-means clustering** (grupare)
  - k-means ierarhic (grupare ierarhica)
- Retele neurale
  - Retele neurale cu auto-organizare
  - Autoencoders
  - Deep belief networks
- Expectation maximization (EM)
  - HMM – algoritmul Baum-Welch
  - **Retele Bayesiene** – estimarea parametrilor in cazul datelor incomplete
  - Soft clustering – clasele se pot suprapune
- Principal Component Analysis (PCA)

Caracteristici importante

- **Regimuri de invatare:**
  - Batch
  - Incremental
- **Zgomot:**
  - zgomot intrari (de ex valorile atributelor)
  - zgomot iesiri (alterare iesiri)
- **Ipoteza favorita a inductiei** (Inductive bias)
  - O multime de presupuneri (explicite sau implicite) pe care algoritmul de invatare se bazeaza pentru a obtine un model (a generaliza) din setul de invatare

Inductive bias

Spatiu de ipoteze restrictionat (Restricted Bias)

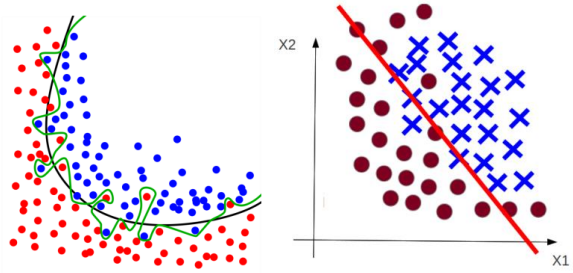
- Functii liniare sau polinomiale de ordin mic

Ipoteza preferata (Preference Bias)

- Occam's razor
- Nerest neighbours
- Independenta conditionala maxima
- Margine maxima
- CNN, Deep RL agents

Overfitting = modelul include zgomot sau sabloane de date nerelevante

Underfitting = modelul nu se potriveste nici datelor de învățare și nici nu poate generaliza la date noi

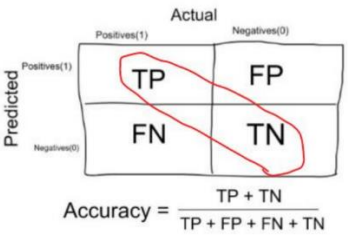


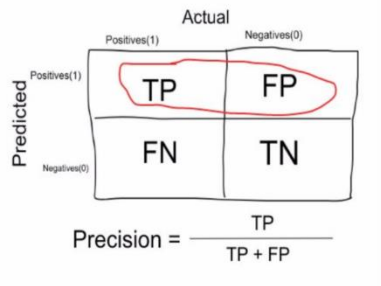
Regularizare – evita overfitting, simplifica modelele cu complexitate prea mare

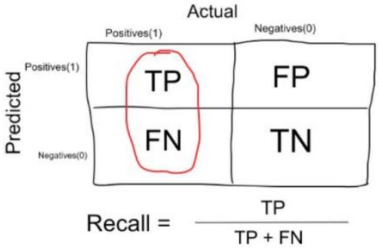
Cross-validation - Antrenarea se face pe subseturi de date de învățare și evaluare pe restul

Masura performantei

- **Regresie**
  - **Mean Absolute Error** (MAE)
  - **Root Mean Square Error** (RMSE)
- **Clasificare**
  - **Confusion matrix**
  - **Accuracy**
  - **Precision** (hit)
  - **Recall sau Sensitivity** (misses)

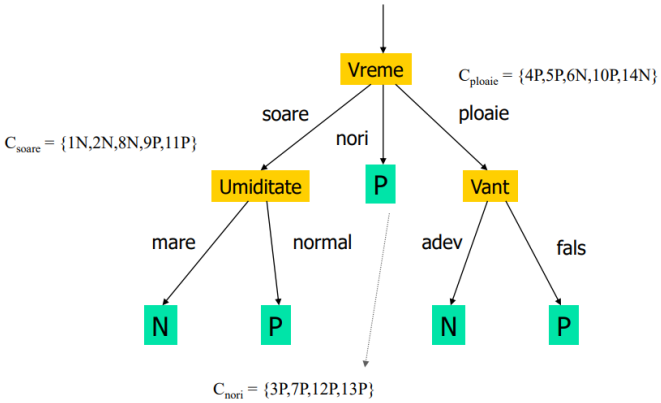






Arbori de decizie

- Cunostintele reprez prin **arbori de decizie multicai** (**AD**)
- Problema de invatare = clasificare
- Invatare suprevizata
- Strategie = invatare batch (ne-incrementala)
- Construit de la radacina la Frunze
- Univers de obiecte **U** cu atribute
- Obiectele apatin unei ***clase***



Continutul de informatie:

$$I(AD_{p,n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Cantitatea de informatie necesara pentru a termina constructia arborelui

$$E(A) = \sum_{i=1}^{v}\frac{p_i + n_i}{p+n} I(AD_{pi,ni})$$

Castigul informational

$$G(A) = I(AD_{p,n}) - E(A)$$

Calcul G(A) pt Ex

- 14 exemple, 9∈P, 5∈N
- **$I(AD_{p,n})$** = 0.940 bits

$$-\frac{9}{14}\log_2\frac{9}{14}-\frac{5}{14}\log_2\frac{5}{14}$$

- vreme
  - soare - 2∈P, 3∈N ⇒ **$I(AD_{p1,n1})$ = 0.971**
  - nori - 4∈P, 0∈N ⇒ **$I(AD_{p2,n2})$ = ?**
  - ploaie - 3∈P, 2∈N ⇒ **$I(AD_{p3,n3})$ = ?**
- **E(vreme) = 0.694 bits** ← $\frac{5}{14}I(AD_{p1,n1})+\frac{4}{14}I(AD_{p2,n2})+\frac{5}{14}I(AD_{p3,n3})$
- **G(vreme) = 0.940-0.694= 0.246 bits**
- G(temperatura) = 0.029 bits
- G(umiditate) = 0.151
- G(vant) = 0.048 bits

C_{soare} = {1N,2N,8N,9P,11P}
C_{nori} = {3P,7P,12P,13P}
C_{ploaie} = {4P,5P,6N,10P,14N}

4

Cazuri speciale???

Random forests

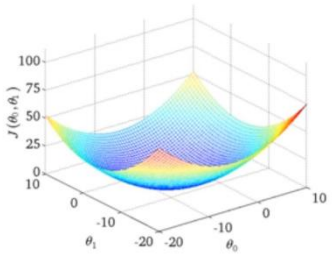
- Combina mai multe metode de invatare pentru a creste acuratetea invatarii
- **Bagging** = obtine o medie a unor modele de invatare
- Idee - Selectam aleator din multimea de invatare o submultime de exemple si tot aleator selectam o submultime de atribute, apoi construim, pentru fiecare selectie cate un arbore (random selection of attributes and random seletion of examples)
- => **padure de arbori**
- **Avantaje:**
  - Acuratete mare de invatare
  - Reduce influenta unui model favorizant (biased)
  - Mai putin sensibil la zgomote
  - Totusi are tendinta la overfitting pentru anumite seturi de date

Regresia liniara

- **invatare supervizata**
- Invata valori continue pe baza exemplelor de invatare
- **$h_\theta(x) = \theta_0 + \theta_1 x$**
- ca sa gasim θ_0 si θ_1 trebuie sa minimizam o **functie de cost => eroarea patratica**

$$J(\theta_0,\theta_1) = \frac{1}{2n}\sum_{i=1}^{n}(h(x^i)-y^i)^2$$

- din planul asta trebuie sa alegem minimul =>
- => **Scaderea gradientului**
  - Metoda iterativa pentru gasirea minimului unei functii
  - se fac pasi proportionali cu **negativul derivatei**/gradientului curent => minimul functiei



Arbori de regresie

- Arborii de regresie sunt **arbori de decizie** cu **valori continue**
- Modelele predictive de regresie liniare sunt modele globale in care o singura formula predictive ar trebui sa fie valabila in tot spatiul
- atribute care interactioneaza in moduri **complicate** => **regresie neliniara** => **partitionarea** spatiului in regiuni mici cu interactiuni **mai simple** =>
  => **partitionare recursiva** & **model simplu pentru fiecare partitie,**
- Folosesc **arbori de decizie** pentru a reprezenta partitionarea
- **Frunzele** corespund unor regiuni din spatiul de intrare care sunt **similare ca iesiri si cu intrari alaturate**
- **...?**

Regresie logistica

- Binomiala
- Multinomiala

Diferente

Regresie liniara

– Distributia este Gausiana

- Valorile prezise sunt continue

Regresie logistica

– Distributia este Bernoulli/Binomiala

- Valorile prezise sunt probabilitati pentru apartenenta (sau non-apartenenta) la o clasa

(Distributie Bernoulli – 1 eveniment cu iesire 1/0
Distributie categoariala – 1 eveniment cu k iesiri
Distributie binomiala – n evenimente cu iesire 1/0
Distributie multinomiala – n evenimente cu k iesiri posibile)

Invatare "Ensemble"

- Mai multe ipoteze din acelasi set de date
- Combina mai multe ipoteze de invatare
- Metode:

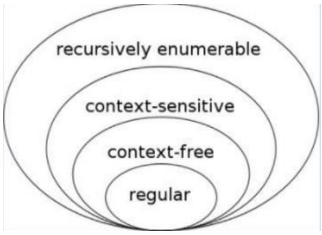
- **Naïve Bayes**
- **Bagging** – Random forests
- **Boosting** – construieste incremental o multime de ipoteze prin antrenarea fiecarei noi instante de model a.i. sa ia in considerare exemplele gresit clasificate de modelul anterior - **Adaboost**

Prelucrare LN

Categorii ilocutionale

- Asertive
- Directive
- Permisive
- Prohibitive
- Declarative
- Expresive

Gramatica / Clasa limbaj	Automat	Reguli de productie
Recursiv numarabile	Masina Turing	$\gamma \rightarrow \alpha$
Dependente de context	Masina Turing linear marginita	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Independente de context	Automate Push down	$A \rightarrow \alpha$
Regulate	Automate cu stari finite	$A \rightarrow a$ $A \rightarrow aB$



Analiza lexicala

Gramatici regulate

G=(N,Σ,R,S)

N este multimea de neterminale

Σ este multimea de terminale

R este multimea de reguli de rescriere

R:{ X→a *sau* X →aY | X,Y∈N si a ∈ Σ)

S – simbolul de start

Analiza semantica

Gramatici independente de context

$G=(N,\Sigma,R,S)$

N este multimea de neterminale

Σ este multimea de terminale

R este multimea de reguli de rescriere

$$R:N \rightarrow (N \cup \Sigma)^*$$

S – simbolul de start

Analiza:

- Top-down - cauta arborele de derivare de la **S** catre Frunze; se expandeaza neterminalul cel mai din stanga
- Bottom-up -de la cuvintele din fraza si cauta sa obtina **S**

Ambiguitati:

- **Ambiguitate de asociere** -- *I hit a man with an umbrella*
- **Ambiguitate de coordonare** -- *Old men and women*

Gramatici independente de context probabiliste (GICP)

VP →	Verb	[0.70]
	| Verb NP	[0.30]

- Fie se calculeaza probabilitatea pt fiecare arbore si se allege arborele cu probabilitatea maxima
- Sau analizez subsiruri si memorez analizele partiale
    - **Analiza CKY**
        - Bottom up
        - Chart parser
        - **Transformat in CNF!**
            - X → a cu a∈Σ (reguli lexicale)
            - X → Y Z cu Y, Z ∈ N (reguli sintactice)

G initiala

S → NP VP	0.9
S → VP	0.1
VP → V NP	0.5
VP → V	0.1
NP → NP NP	0.2
NP → N	0.7
N → people	0.5
N → fish	0.2
N → tanks	0.2
V → people	0.1
V → fish	0.6

G transformata in CNF

S → NP VP	0.9
S → V NP	0.05
S → people	0.001
S → fish	0.006
VP → V NP	0.5
VP → people	0.01
VP → fish	0.06
NP → NP NP	0.2
NP → people	0.35
NP → fish	0.14
NP → tanks	0.14
N → people	0.5
N → fish	0.2
N → tanks	0.2
V → people	0.1
V → fish	0.6

Definite Clause Grammar (DCG)

- Utilizeaza **Logica Predicatelor** pentru reprez gramaticilor
- Gramatici cu clauze definite
- Gramatica semantica