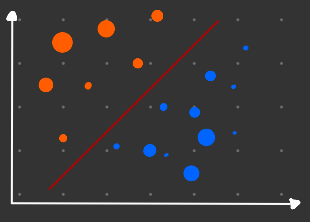
GRAPH KERNELS

GED

GED IS USUALLY IMPLEMENTED AS AN A^* SEARCHING ALGORITHM, CONSISTING, AT EACH STEP, TO EXPLORE ALL POSSIBLE STATES OBTAINABLE, UNTIL THE GOAL STATE IS REACHED, THE SOLUTION IS THE LENGTH OF 'P': SHORTEST PATH FROM THE STARTING STATE TO THE GOAL ONE; GIVEN 'b' THE MAXIMUM NUMBER OF SOLUTION OBTAINABLE FROM A GENERIC STATE, THE COMPLEXITY IS $O(b^p)$.
FOLLOWS THAT, DESPITE THE USAGE OF EFFICIENT HEURISTICS AND TAKEN INTO ACCOUNT THAT FOR EACH SOLUTION A 'SUBGRAPH ISOMORPHISM' MUST BE CHECKED, AND IS KNOWN TO BE AN NP-COMPLETE PROBLEM, SO, THIS IS NOT THE BEST APPROACH TO COMPARE BIG #GRAPHS; ALSO CHOOSING COSTS FOR DIFFERENT OPERATION, EVEN IF IT BRINGS FLEXIBILITY, IS A DIFFICULT OPERATION.
COMPARABLE WITH "GRAPH ISOMORPHISM", "IDEAL KERNEL", "RANDOM WALK KERNEL (BEST $O(n^3)$)"

KERNEL TRICK

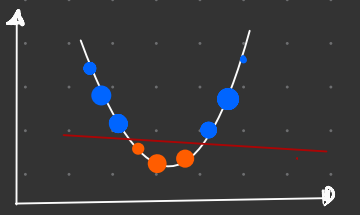
NOT ALWAYS DATA IS LINEARLY SEPARABLE BY SUPPORT VECTOR CLASSIFICATION, SO WE MUST APPLY A TRANSFORMATION $\phi(x)$ TO MAP IT IN AN HIGHER DIMENSIONAL SPACE WHERE IT BECAME SEPARABLE.



// SEPARABLE IN 2 DIMENSIONS

// NOT SEPARABLE IN 1 DIMENSION

$\Rightarrow$



// SEPARABLE IN 2 DIMENSION

IN REAL APPLICATION, WHERE DATA CAN HAVE MANY FEATURES THE COMPUTATIONAL COST IS TOO HIGH, THE KERNEL TRICK IS A SOLUTION TO THIS PROBLEM.
THE DATA X IS REPRESENTED IN AN $n \times n$ MATRIX M S.T. $M_{is} = K(x_i, x_s)$ WHERE $K(x_i, x_s)$ IS THE KERNEL FUNCTION. IT TAKES AS INPUT VECTORS IN ORIGINAL SPACE RETURNING THEIR DOT PRODUCT IN THE FEATURE SPACE: $K(x_i, x_s) = \phi(x_i) \cdot \phi(x_s)$
THE FUNCTION TO OPTIMIZE FOR THE SUPPORT VECTOR CLASSIFICATION CONTAINS THE DOT PRODUCT OF $\phi(x)$ AND $\phi(x)$, USING THE KERNEL FUNCTION WE CAN OBTAIN THAT VALUE WITHOUT KNOWING ANYTHING OF THE TRANSFORMATION $\phi(x)$.

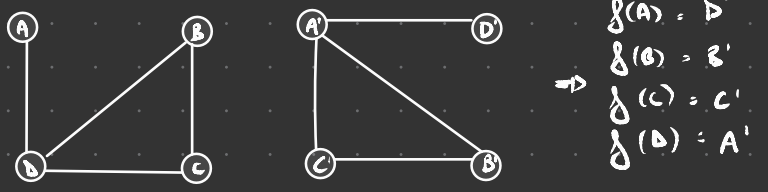
SIGNATURE / ANOMALY - BASED DETECTIONS

- SIGNATURE BASED DETECTION IS USED FOR ALREADY KNOWN THREATHS, IT RELIES ON LIST ON INDICATOR OF COMPROMISE (IOCs), INCLUDING MALICIOUS EXTENSION, HASHES, BYTE SEQUENCES, DOMAINS ETC... SO THEIR MAIN DISADVANTAGE IS THAT AN UNEXPECTED BEHAVIOR WOULD NOT BE RECOGNIZED AS A THREATH, SO SYSTEMS RELYING ON IT MUST ALWAYS BE UNDER MANTAINANCE TO DON'T BECAME OBSOLETE OR VULNERABLE TO NEW KINDS OF ATTACK (ZERO DAYS EXPLOIT). THE ADVANTAGE IS INSTEAD ON THE DETECTION SPEED FOR KNOWN ATTACKS (RCEs, DDOS, FILE INTRUSIONS, ...)

- ANOMALY BASED DETECTION, ON THE OTHER HAND, IS USED TO NOTICE CHANGES IN BEHAVIOR. IT INVOLVES FIRST A TRAINING OF THE SYSTEM WITH A NORMALIZED BASELINE, AND THE ON THE COMPARISON BETWEEN THE REAL BEHAVIOR WITH THE KNOWN BASELINE TO CAPTURE ANOMALIES AND GENERATE ALERTS THE DISADVANTAGE IS THE HIGH RATE OF FALSE POSITIVES, SO CONSUMING ADDITIONAL RESOURCES A TIME BUT IS MORE RESILIENT TO ZERO DAYS EXPLOITS.

ISOMORPHISM

GIVEN G_1 AND G_2 AN ISOMORPHISM IS A BISECTION BETWEEN THE THEIR VERTEX SETS $f: V(G_1) \to V(G_2)$, IF IT EXISTS THEN G_1 AND G_2 ARE IDENTICAL: $(x,y) \in E(G_1) \Leftrightarrow (f(x), f(y)) \in E(G_2)$
BY NOW THERE IS NOT A POLYNOMIAL KNOWN ALGORITHM TO SOLVE IT, BUT IS NOT EVEN PROVED TO BE NP-COMPLETE.
THIS PROBLEM IS A SPECIAL CASE OF THE SUBGRAPH ISOMORPHISM PROBLEM ('DOES G_1 CONTAIN A SUBGRAPH $G_1' \subseteq G_1$ ISOMORPHIC TO G_2?'), KNOWN TO BE NP-COMPLETE.

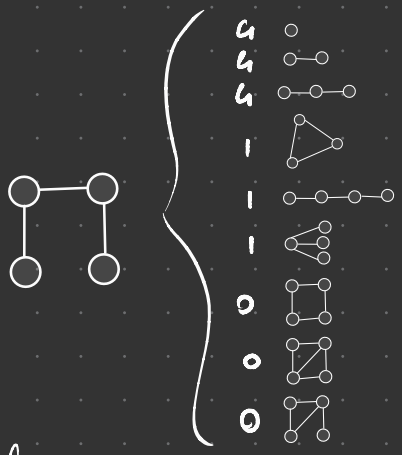


ANOTHER APPROACH FOR GRAPHS COMPARISON IS THE 'GRAPH EDIT DISTANCE', AND ALSO THE 'IDEAL GRAPH KERNE'. BUT THE BEST METHOD SO FAR IS THE 'RANDOM WALK KERNEL'.

IDEAL GRAPH KERNEL

WAY OF COMPARING FEATURE GRAPHS OF DIFFERENT SIZE. THE PROBLEM IS NP-COMPLETE WE EXTRACT FEATURES TO BRING THE PROBLEM IN AN HIGER DIM SPACE

- ADVANTAGES: allow graphs with specific substructures to be identified (like hamiltonian cycles)
- DISADVANTAGES: to expressive (NP-COMPLETE)



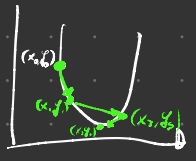
GRADIENT DESCENT

Allows to find the minimum of a fuction.
Can be used in the LINEAR REGRESSION.

- given $\min_{x,y} f(x,f)$ to find repeat the following:

$$\begin{cases} x = x - \alpha \cdot f_x'(x,f) \\ y = y - \alpha \cdot f_y'(x,y) \end{cases}$$

  to you can update x,y simoultaneously or not



DRAWBACKS | can be stucked in LOCAL MIN;
to high learning rate will find a too fast approximation
to low " " doesn't converge

RANDOM WALK KERNELS

WALK seq. of nodes that allow repetition

FEATURE SPACE #WALK of length $k = 1, \dots, n$ from v to u

- G and G' are equal if same feature space

TRICK 1 MATRIX MOLTIPLICATION $A_2 = A_1 \cdot A_1$, $A_3 = A_2 \cdot A_1, \dots$ where A_1 is ADJ(G)

- Potentially we should compute all features vector but there are efficient tricks to convert the sequence $\{A_1, A_2, \dots\}$ in a numerical series of which computing the limit

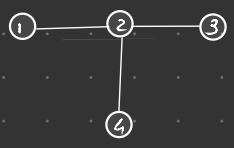
SIMILARITY COMPUTATION Similarity using kernel function $k(x, y) = x \cdot y$ | x: #walks $\forall u, v \in V$ | y: #walks $\forall u, v \in V'$

↳ **TRICK** DIRECT PRODUCT $G \times G'$ | $V_{GG'} = V_G \times V_{G'}$ and

$$E_{GG'} = \{((u,u'),(v,v')) \mid \exists (u,v) \in G \wedge \exists (u',v') \in G'\}$$

⇒ Total runtime $O(n^6)$ that can become $O(n^3)$

- $O(n^2)$ builds nodes of $G \times G'$
- $O(n^4)$ builds edges of $G \times G'$



$$A_1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \end{array}$$

$$A_2 = A_1 \cdot A_1 = \begin{bmatrix} 0&1&0&0 \\ 1&0&1&1 \\ 0&1&0&0 \\ 0&1&0&0 \end{bmatrix} \cdot \begin{bmatrix} 0&1&0&0 \\ 1&0&1&1 \\ 0&1&0&0 \\ 0&1&0&0 \end{bmatrix} = \begin{bmatrix} 1&0&1&1 \\ 0&3&0&0 \\ 1&0&1&1 \\ 1&0&1&1 \end{bmatrix}$$

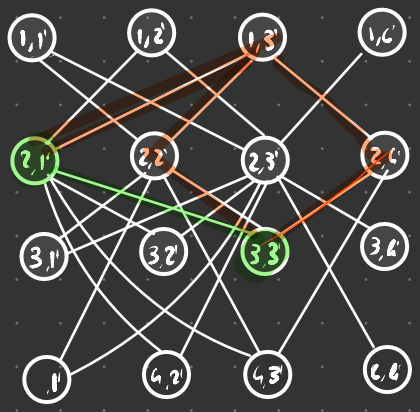
$$A_3 = A_2 \cdot A_1 = \begin{bmatrix} 0&1&0&0 \\ 1&0&1&1 \\ 0&1&0&0 \\ 0&1&0&0 \end{bmatrix} \cdot \begin{bmatrix} 1&0&1&1 \\ 0&3&0&0 \\ 1&0&1&1 \\ 1&0&1&1 \end{bmatrix} = \begin{bmatrix} 0&3&0&0 \\ 3&0&3&3 \\ 0&3&0&0 \\ 0&3&0&0 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0&1&0&0 \\ 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&0 \end{bmatrix}$$

$$A_2 = A_1 \cdot A_1 = \begin{bmatrix} 0&1&0&0 \\ 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} 0&1&0&0 \\ 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&0 \end{bmatrix} = \begin{bmatrix} 1&0&1&0 \\ 0&2&0&1 \\ 1&0&2&0 \\ 0&1&0&1 \end{bmatrix}$$

$$A_3 = A_2 \cdot A_1 = \begin{bmatrix} 0&1&0&0 \\ 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} 1&0&1&0 \\ 0&2&0&1 \\ 1&0&2&0 \\ 0&1&0&1 \end{bmatrix} = \begin{bmatrix} 0&2&0&1 \\ 2&0&3&0 \\ 0&3&0&2 \\ 1&0&2&0 \end{bmatrix}$$

DIRECT PRODUCT GRAPH

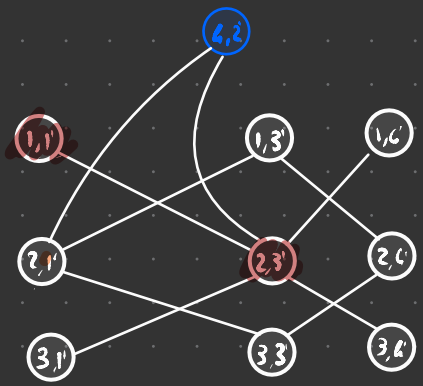


- How many walks from $(2,1')$ and $(3,3')$ of lenght $1,2,3,4,5$?

Length 1: $1 \cdot 1 = 1$
Length 2: $0 \cdot 1 = 0$
Length 3: $2 \cdot 2 = 4$
Length 4: 0
Length 5: ...

} $\#\text{WALK}_{(2,1') \text{ in } G} \cdot \#\text{WALK}_{(3,3') \text{ in } G'}$

- Do same with LABELS



- How many walks from $(2,3')$ and $(1,1')$ of lenght $1,2,3,4,5$?

Length 1: 1
Length 2: 0
Length 3: 5
Length 4: 0
Length 5: 20

} MARIA METHOD :)

Q-LEARNING

$$R = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 10 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q(s,A) = Q(s_t, A) + \alpha \left(R_t + \gamma \max_{a \in A} \{ Q(s_{t+1}, a) - Q(s_t, A) \} \right)$$

EPISODE 1

$Q(1, UP) = Q(1, UP) + 0.5(1 + 0.7 \cdot \max_{a \in A} \{Q(4,a)\} - Q(1, UP)) = 0.5$

$Q(4, RIGHT) = Q(4, RIGHT) + 0.5(1 + 0.7 \cdot \max_{a \in A} \{Q(5,a)\} - Q(4, RIGHT) = 0.5$

$Q(5, RIGHT) = Q(5, RIGHT) + 0.5(10 + 0.7 \cdot \max_{a \in A} \{Q(6,a)\} - Q(5, RIGHT) = 5$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \end{bmatrix}$$

...

$$Q = \begin{bmatrix} 0 & 3 & 0 & 4 & 0 & 0 \\ 3 & 0 & 8 & 0 & 6 & 0 \\ 0 & 7 & 0 & 0 & 0 & 9 \\ 2 & 0 & 0 & 0 & 5 & 0 \\ 0 & 6 & 0 & 0 & 5 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

* RANDOM ACTION (1, UP) WAS BETTER

$Q(1, RIGHT) = 3 + 1/2 (1 + 0.7 \cdot \max_{a \in A} \{Q(2,a)\} - 3) \approx 8 + 7/10 \cdot 8 - 3 = 4.8$

$Q(2, RIGHT) = 8 + 1/2 (1 + 0.7 \cdot 9 - 8) = 7.65$

$Q(3, UP) = 9 + 1/2 (10 + 0.7 \cdot 0 - 9) = 9.5$

$$Q = \begin{bmatrix} 0 & 3 & 0 & 4 & 0 & 0 \\ 3 & 0 & 8 & 0 & 6 & 0 \\ 0 & 7 & 0 & 0 & 0 & 9.5 \\ 2 & 0 & 0 & 0 & 5 & 0 \\ 0 & 6 & 0 & 0 & 5 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

⇒ Now the best strategy is the same made in the last step

REINFORCEMENT LEARNING

An agent act in an environment to reach a GOAL from a starting state learning from rewards for taking actions $a \in A$ to change state $s \in S$ under a policy π

- DETERMINISTIC $\pi \Rightarrow a = \pi(s)$ // Always same a from s
- STOCASTIC $\pi \Rightarrow \pi(a|s) = P_\pi[A=a \mid S=s]$ // distribution over actions given states

The reward is assigned by

- VALUE FUNCTION $V_\pi(s)$
- ACTION-VALUE FUNCTION $Q_\pi(s,a)$

The environment is represented in the MODEL

- TRANSITION MODEL: predicts next state
- REWARDS MODEL: predicts next reward

MARKOV DECISION PROCESS

MARKOV CHAIN if $P[S_{t+1}|S_t] = P[S_{t+1}|S_t, S_{t-1}, \dots, S_0]$

MARKOV DECISION PROCESS tuple $M = \langle S, A, P, R, \gamma \rangle$ (STATES, ACTIONS, STATE TRANSITION PROBABILITY MATRIX, REWARD FUNCTION, DISCOUNT FACTOR [0,1])

RETURN Total sum of discounted rewards from t $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

- VALUE FUNCTION $V_\pi(s) = E_\pi[G_t \mid S_t = s] \Rightarrow V_*(s) = \max_\pi V_\pi(s)$
- STATE-VALUE FUNCTION $Q_\pi(s,a) \; E_\pi[G_t \mid S_t = s, A_t = a] \Rightarrow Q_*(s) = \max_\pi Q_\pi(s)$

BELLMAN EQUATIONS decompose VALUE FUNCTION in IMMEDIATE REWARDS + DISCOUNTED FUTURE REWARDS

can be represented as a matrix $V_\pi = R^\pi + \gamma P^\pi V_\pi$ and computed in $O(n^3)$

TH There's always an optimal policy π^*

1) DYNAMIC PROGRAMMING if all the environment is known
   - solve all subproblems
   - GPI (Generalize policy iteration) iterative procedure to evaluate and improve policies

$$\pi_0 \xrightarrow{\text{EVAL}} V_{\pi_0} \xrightarrow{\text{IMPROVE}} \pi_1 \rightarrow \dots$$

↳ **MONTE CARLO LEARNING** if no knowledge of MDP transition and rewards.

it uses the mean return value to choose policies but can return an approximation of the return G_t

↳ **TEMPORAL DIFFERENCE LEARNING** learn from incomplete episodes

can be 'VALUE BASED' using BELLMAN EQ, or 'OFF-POLICY LEARNER' learn optimal policy independently from actions

↳ **Q-LEARNING**

- Q-TABLE : knowledge of agent about maximum expected future rewards $Q_{STATE \times ACTION} = [\dots]$
- Q-FUNCTION : Bellman eq to update $Q(s,a) = Q(s,a) + \alpha\left(R(s,a) + \gamma \max_{a' \in A}\{Q(s',a')\} - Q(s,a)\right)$

2 APPROACHES

↳ GREEDY : at each step take best action in Q

↳ E-GREEDY : take the best with $P = \varepsilon$

EXPLORATION VS EXPLOITATION

- ↑ ALWAYS TAKE BEST ACTION KNOWN
- ↑ RISK TO COLLECT INFORMATIONS

MAB (Multy armed bandit)

- MDP simply fixed, there's no state
- There are k slots you have to choose one each step, receiving a reward
- You play the machine with highest expected rewards

2 APPROACHES
- BOUNDED : play each once, then start to choos the best
- ACTION-VALUE BASED ON PAST EXPERIENCE : choos a based on the average reward got on it so far $Q(a) = Q(a) + \frac{1}{N(a)}(R_a - Q(a))$

2 ACTION SELECTION METHODS
- GREEDY
- E-GREEDY

if $(|V| \neq |V'|)$ return -1

For $deg = 1 \dots \max\{ max\_deg(G), max\_deg(G') \}$:

For $P = P_1, \dots, P_m$:

ISO = TRUE

FOR $v \in P$:

if $(deg(v) \neq deg(v') \ \forall v' \in V')$:

ISO = FALSE

BREAK