HMM
Conditional Probability: $P(A∣B)=P(B)P(A∩B)$
Joint Probability: $P(A∩B)$
Marginal Probability: $P(A)=∑BP(A∩B)$ (for discrete variables)
$P(A)=∫BP(A∩B)dB$ (for continuous variables)
Bayes' Theorem: $P(A∣B)=P(B)P(B∣A)⋅P(A)$
Expectation (Mean): $E[X]=∑xx⋅P(X=x)$ (for discrete variables)
$E[X]=∫−∞∞x⋅f(x)dx$ (for continuous variables)
Variance: $Var(X)=E[(X−E[X])2]$

Marcov model
transition probabilities aij = P(Si|Sj) $a_{ij} = P(s_i \mid s_j)$
initial probabilities Pi(i) = P(Si) $\pi_i = P(s_i)$
P({‘Dry’,’Dry’,’Rain’,Rain’}) = P(‘Rain’|’Rain’) P(‘Rain’|’Dry’) P(‘Dry’|’Dry’) P(‘Dry’)=

hidden Markov model
matrix of transition probabilities A=(aij), aij= P(si | sj)
matrix of observation probabilities B=(bi (vm)), bi(vm) = P(vm | si)
a vector of initial probabilities pi=pi(i), pi(i) = P(si) . $\pi=(\pi_i)$, $\pi_i = P(s_i)$
P({‘Dry’,’Rain’}) = **P({‘Dry’,’Rain’} , {‘Low’,’Low’})** + P({‘Dry’,’Rain’} , {‘Low’,’High’}) + P({‘Dry’,’Rain’} , {‘High’,’Low’}) + P({‘Dry’,’Rain’} , {‘High’,’High’})

P({‘Dry’,’Rain’} , {‘Low’,’Low’})= P({‘Dry’,’Rain’} | {‘Low’,’Low’}) P({‘Low’,’Low’}) = P(‘Dry’|’Low’)P(‘Rain’|’Low’) P(‘Low’)P(‘Low’|’Low)

Using HMMs :
Evaluation problem.
HMM M=(A, B, pi) , O=o1 o2 ... oK , probability that model M has generated sequence O .
Decoding problem.
HMM M=(A, B, pi) ,O=o1 o2 ... oK , most likely sequence of hidden states si that produced O.
Learning problem.
O=o1 o2 ... oK , HMM (numbers of hidden and visible states), determine HMM parameters M=(A, B, pi) that best fit training data.

Evaluation Problem. Forward-Backward HMM algorithms
Forward recursion for HMM
forward variable alphak(i)
hidden state at time k $s_i : \alpha_k(i) = P(o_1 o_2 \ldots o_k, q_k = s_i)$

Initialization:

$$\alpha_1(i) = P(o_1, q_1 = s_i) = \pi_i \, b_i(o_1), \quad 1 \le i \le N.$$

Forward recursion:

$$\begin{aligned}\alpha_{k+1}(j) &= P(o_1 o_2 \ldots o_{k+1}, q_{k+1} = s_j) = \\ &\Sigma_i \, P(o_1 o_2 \ldots o_{k+1}, q_k = s_i, q_{k+1} = s_j) = \\ &\Sigma_i \, P(o_1 o_2 \ldots o_k, q_k = s_i) \, a_{ij} \, b_j(o_{k+1}) = \\ &[\Sigma_i \, \alpha_k(i) \, a_{ij}] \, b_j(o_{k+1}), \quad 1 \le j \le N, \; 1 \le k \le K-1.\end{aligned}$$

Termination:

$$P(o_1 o_2 \ldots o_K) = \Sigma_i \, P(o_1 o_2 \ldots o_K, q_K = s_i) = \Sigma_i \, \alpha_K(i)$$

Complexity : N^2K operations.

Backward recursion

Initialization:

$$\beta_K(i) = 1, \quad 1 \le i \le N.$$

$$P(X_t = i|O) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O)}$$

Backward recursion:

$$\begin{aligned}\beta_k(j) &= P(o_{k+1} o_{k+2} \ldots o_K \mid q_k = s_j) = \\ &\Sigma_i \, P(o_{k+1} o_{k+2} \ldots o_K, q_{k+1} = s_i \mid q_k = s_j) = \\ &\Sigma_i \, P(o_{k+2} o_{k+3} \ldots o_K \mid q_{k+1} = s_i) \, a_{ji} \, b_j(o_k) = \\ &\Sigma_i \, \beta_{k+1}(i) \, a_{ji} \, b_j(o_k), \quad 1 \le j \le N, \; 1 \le k \le K-1.\end{aligned}$$

Termination:

$$\begin{aligned}P(o_1 o_2 \ldots o_K) &= \Sigma_i \, P(o_1 o_2 \ldots o_K, q_1 = s_i) = \\ &\Sigma_i \, P(o_1 o_2 \ldots o_K \mid q_1 = s_i) \, P(q_1 = s_i) = \Sigma_i \, \beta_1(i) \, b_i(o_1) \, \pi_i\end{aligned}$$

Decoding problem

Brute force– exponential time.

Use efficient Viterbi algorithm instead.

Viterbi algorithm

if best path ending in qk= sj goes through qk-1= si then it should coincide with best path ending in qk-1= si .

Learning problem

iterative expectation-maximization algorithm to find local maximum of P(O | M) - Baum-Welch algorithm.

aij = P(si | sj) = No of transitions from state sj to state si / no of transitions out of state sj

bi (vm) = P(vm | si) = no of times observation vm occurs in state si / no of times in state si

Baum-Welch algorithm

aij = P(si | sj) = expected No of transitions from state sj to state si / expected no of transitions out of state sj $\frac{\sum_k \xi_k(i,j)}{\sum_k \gamma_k(i)}$

bi (vm) = P(vm | si) = expected no of times observation vm occurs in state si / expected no of times in state si $\frac{\sum_{k, o_k = v_m} \gamma_k(i)}{\sum_k \gamma_k(i)}$

Pi i = P(si) = Expected frequency in state si at time k=1. $\gamma_1(i).$

Code Morphing

transposition, substitution, insertion, and deletion

“strong” metamorphic generator relies only on transposition

“Strong” metamorphic generator

Initialize with a seed virus (form assembly code)

Split into small blocks (6 lines of code)(subject to some conditions)

To generate a malware sample… Shuffle code blocks, add conditional jumps

Shuffling blocks – break signatures

include dead code insertion (“opaque predicates”)

Why insert dead code? – Makes statistical analysis more difficult

Experiments

Seed with NGVCK virus

Generate 200 morphed copies

- Assemble each morphed asm into exe
- Verify that seed virus detected by AV…
- morphed copies not detected by AV

Disassemble exes, extract opcodes

Train HMMs, – 5-fold cross validation

Score each model vs 40 benign samples

Why can we detect this morphed malware using HMMs... …but not using signatures?

Signatures – disrupted by transposition

HMM not affected by transposition, “sees” differences between viruses and benign

Transposition – highly effective anti-signature strategy, ineffective for machine learning

User behavior monitoring

observing, analyzing, and understanding the actions and activities

• login attempts • file accesses • application usage • system commands (login,file, app, command)

Significance in Cyber security (Early Detection, Insider Threats, Enhanced Response, Compliance)
Early Threat Detection:
unusual login times, access to unauthorized resources, abnormal data transfer volumes
Early detection allows security teams to investigate early, identify the root cause, and take action to mitigate the threat before it escalates

Prevention of Insider Threats:
come from employees, contractors, trusted entities with access to sensitive data/ systems.
User behavior monitoring ,flagging suspicious activities
unusual access patterns, data exfiltration attempts, privilege abuse,

Enhanced Incident Response:
analyzing user activity logs, access logs,
identify affected systems, determine extent of unauthorized access/data breaches, tracing the actions of threat actors.

Compliance and Governance:
User behavior monitoring
monitor and audit user activities to protect sensitive data, prevent unauthorized access

Key Components of User Behavior Monitoring (activity, auth, escalation, patterns, endpoint)
User Activity Logs:
recording actions - login attempts, file accesses, application usage, and system commands.
Logging login attempts
Recording file access events

User Authentication Patterns
Analyzing user authenticate, login times, locations, devices used.
Identifying frequent logins from unusual locations/devices.
Monitoring use of multi-factor authentication methods.

Privilege Escalation Monitoring
where users attempt to gain elevated privileges or access unauthorized resources.
Monitoring changes to user permissions or roles
Tracking the commands/scripts that may indicate privilege escalation attempts

Access Patterns and Permissions
Analyzing the frequency and nature of access requests
frequent access to confidential files
Monitoring changes to access permissions for critical resources.

Endpoint Behavior Analysis

Examining behavior of users on endpoints (devices), desktops, laptops, and mobile devices.
installation of unauthorized software or accessing restricted websites.
Analyzing user behavior for signs of malware infection or data exfiltration attempts.

Tools and Methods
Log Management and Analysis Tools: Splunk, ELK Stack
monitor user activities, detect anomalies, and investigate security incidents.

User Activity Monitoring Solutions : Varonis, SolarWinds User Device Tracker
real-time monitoring of user activities, file access, email communications, application usage,

Varonis -
Data Access Governance:
Behavior Analytics:
Data Protection:

SolarWinds User Device Tracker - network monitoring, track user activity, device connections
User Tracking: logins and logouts, activities across devices, user activity reports.
Device Mapping: Maps devices to users
Real-time Alerts: alerts for suspicious activities

User and Entity Behavior Analytics (UEBA) - Exabeam, Rapid7 InsightIDR
Users:
1.Devices 2.Applications 3.Accounts and Identities 4.Data and Resources 5.Network Traffic: use machine learning algorithms, analyze user behavior, detect anomalous activities

Exabeam
Behavioral Analytics:
Threat Hunting: search and investigate potential security threats
Incident Response Automation: Automates the investigation and response to security incidents

Rapid7 InsightIDR
cloud-based
User Behavior Analysis:
Automated Threat Detection:
Endpoint Detection and Response (EDR):

What is Cluster Analysis?
applications
stand-alone tool to get insight into data distribution
preprocessing step for other algorithm

Density-based Approaches
Discover clusters of arbitrary shape.

DBSCAN – first density based clustering
OPTICS – density based cluster-ordering
DENCLUE – density-based description of cluster and clustering

DBSCAN: Density Based Spatial Clustering of Applications with Noise
clusters of arbitrary shape in spatial databases with noise

Core, Border & Outlier Border
Core - it has more than a specified number of points (MinPts) within Eps, at the interior of a cluster.
border - has fewer than MinPts within Eps, in the neighborhood of a core point.
noise - not a core point nor a border point

Directly density-reachable - not symmetric
p is a core object and q is in p's e neighborhood.

Density-Connected - symmetric
pair of points p and q are density-connected if they are commonly density-reachable from a point o

DBSCAN: The Algorithm
select a p
Retrieve all points density-reachable from p wrt Eps and MinPts.
If p is a core point, a cluster is formed.
If p is a border point, no points are density-reachable from p and visits the next point

Density Based Clustering:
Advantages
Clusters can have arbitrary shape and size
Number of clusters is determined automatically
Can separate clusters from surrounding noise
Can be supported by spatial index structures
Disadvantages
Input parameters may be difficult to determine
In some situations very sensitive to input parameter setting