

Learning Program Behavior Profiles for Intrusion Detection

->Michal Piekarczyk

Past Intrusion Detection Techniques

- network architecture
- The programs running
- network services
- Firewalls
- penetration audits
- personnel screening

- Are they enough?

Intrusion Detection Expert System (IDES)

- Tracked user statistics
- Deviations in file creation behavior
- Ways around IDES
- (from Stanford Research Inst)

Main Detection Variations

- - **misuse detection**: model the attacks
 - low false positives,
 - Deaf to new attacks
 - high miss rate
- **anomaly detection**: finds deviations to the normal
 - Many false positives => looks for *changes*
 - But can detect "novel attacks,"

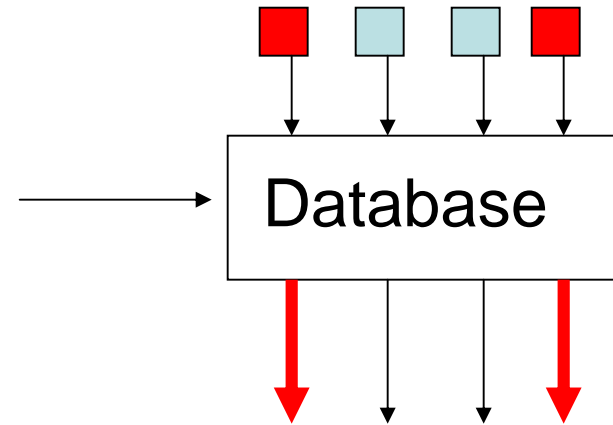
Maybe a potential flaw*

- Why not just distract?
- And attack

Analyzing Program Behavior for Anomaly Detection

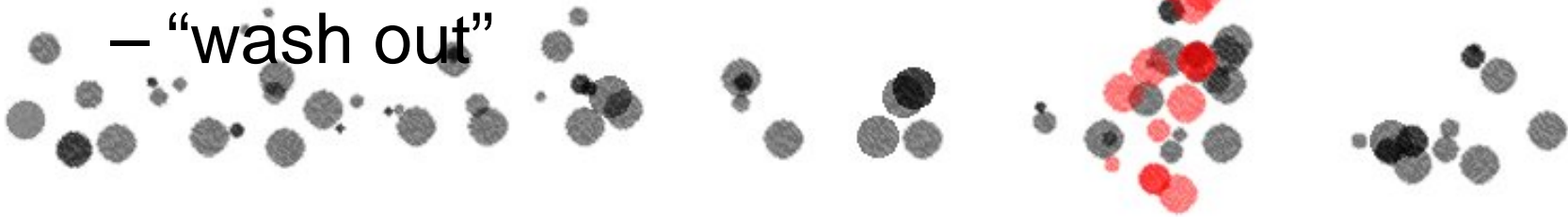
- Capture **System calls**
- Strings of system calls
 - Univ of New Mexico
- => **Normal Behavior**

- Do match ups
- Look for mismatch
- Many mismatches
- => **intrusion**



Temporal Clusters

- Mismatches happen close together
 - => should not average all
 - “wash out”



- Use "fixed-length frames"
- Columbia University

- * why not just scatter system calls?



an attack

Finite State Automata

- Designed to detect specific anomalies
 - But created by hand => not flexible
 - UNM
- Use deterministic FSAs to model
 - “acceptable program behavior“
 - Iowa State U

Authors' Goals

- Anomaly detection
- more self-learning approach
- wish to automatically build profiles
- Less false positives

Use of Neural Nets

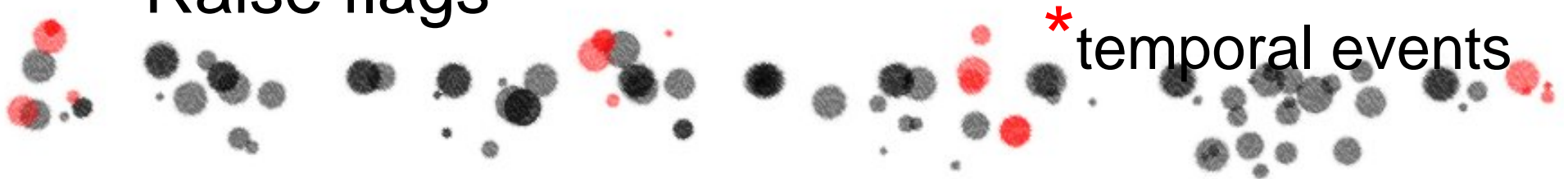
- Automatic behavior adaptation
- And better with errors

Method1: Equality Matching

- Used Sun 's **Basic Security Module (BSM)**
 - Captures system call **sequences**
 - Classifies them
- 20 out of 200 were ***typical*** system calls
- Found begin / end patterns

Method1: Equality Matching

- Store sequences of BSM events
- Keep anomaly counter
- Fixed-size windows
- Raise flags

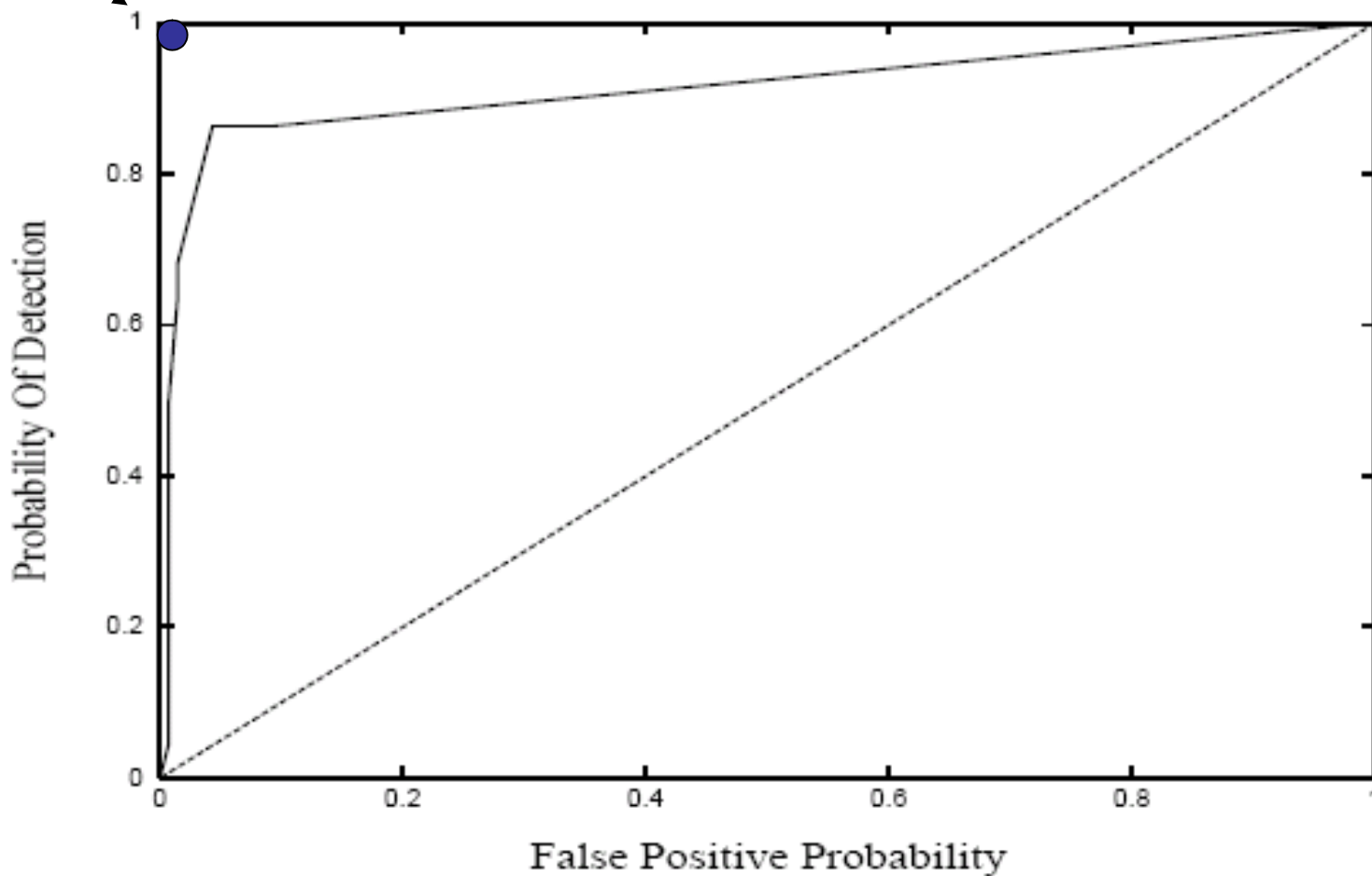


Attack Types Tested

- Denial of service
- Probe attacks
- User to root
- Remote to local

Attack Type	Instances	Detections	Percent Detected
u2r	22	19	86.4
r2l	3	2	66.7
Total	25	21	84%

Oracle point



Method2: Back propagation

- Others' research:
 - Per-user studies, also with NN's
- Ghosh: software behavior

Neural Net

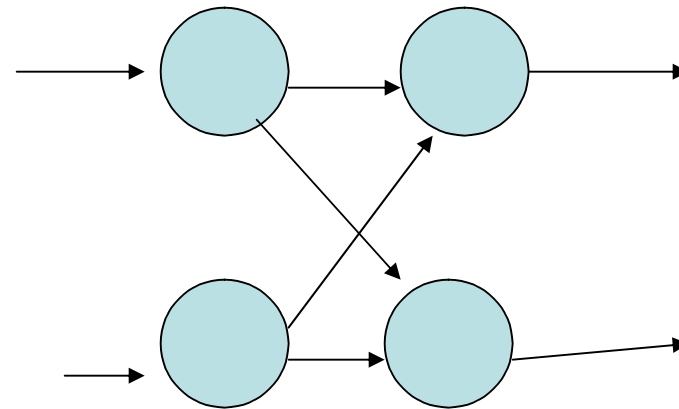
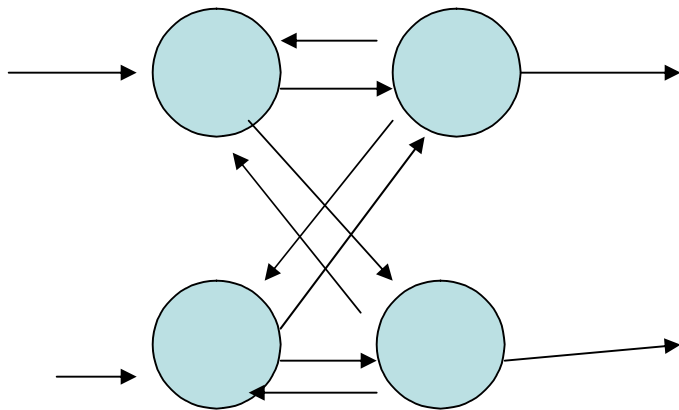
- Uses a **Perceptron** network
- With a large training set
- Train until error stabilizes
 - Or ran out of data

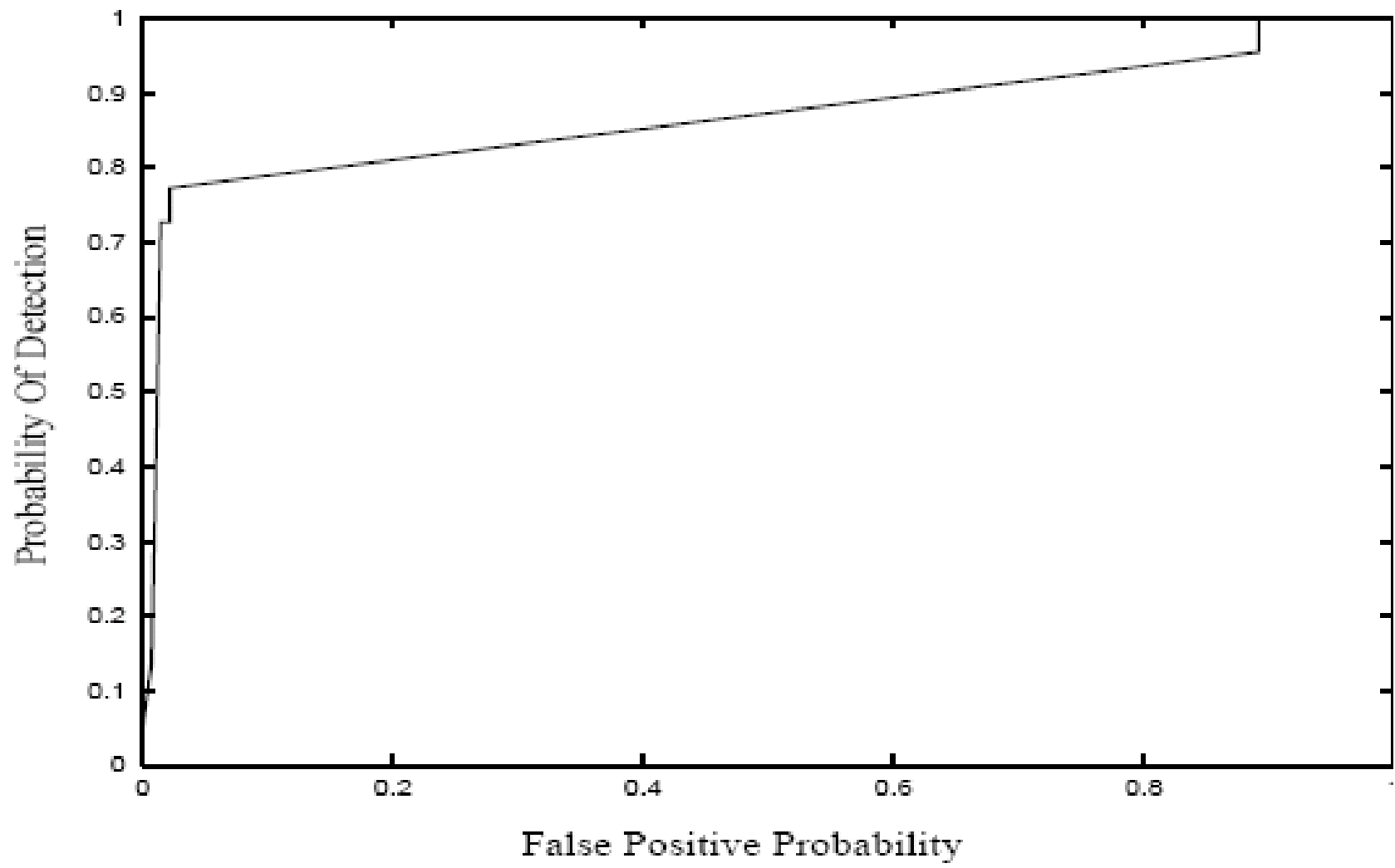
Leaky Bucket Algorithm

- Emphasizes
 - closely temporally co-located anomalies
- Leaks
 - Sparsely located ones
- Occasional anomalies allowed, a **+**
—=> real life

Back Propagating v Feed Forward

- Output is not independent of prior inputs



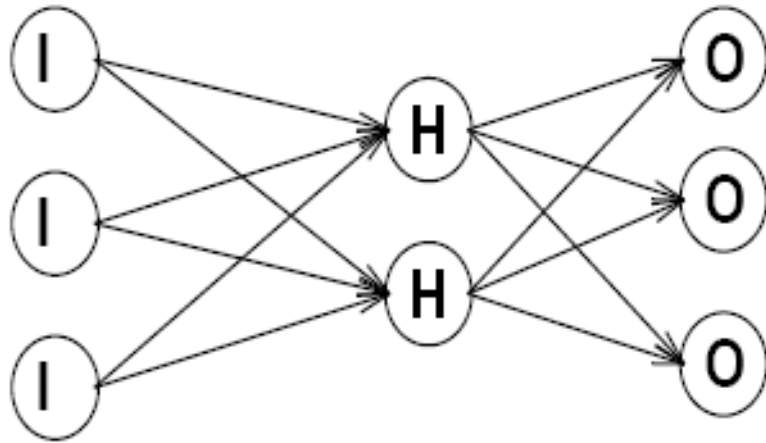


Method3: Elman Nets

- Recognize recurrent features
- Programs generate similar
 - BSM sequences
- More anomalous => more impact on Net
- Distinction: have cycles in topology

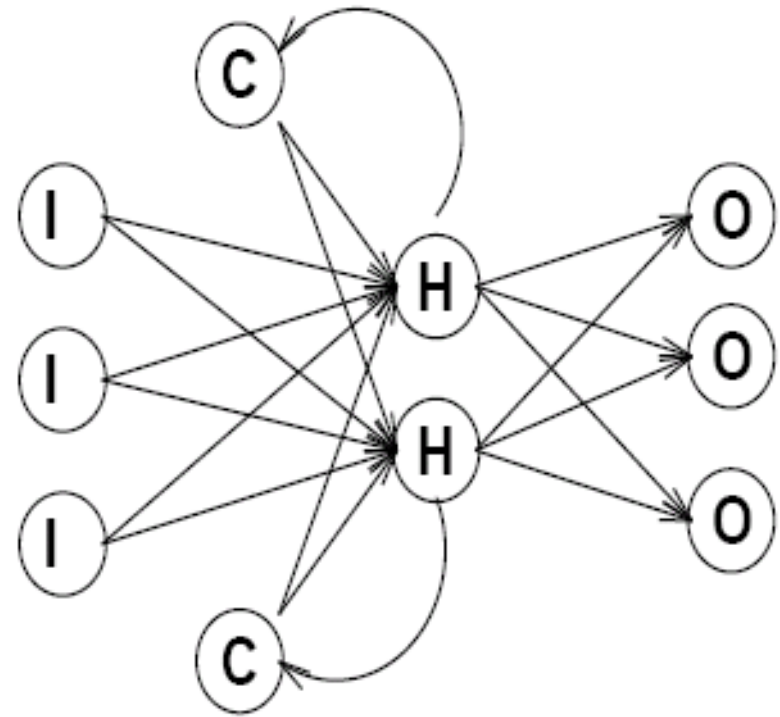
- Cycles: **Delay** loops
- Output based on
 - Current state
 - Current input
- Anomaly detection:
 - Compare current **output** to **predicted output**
 - I_{n+1} and O_n

Pure feed-forward



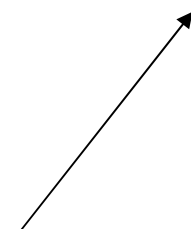
A

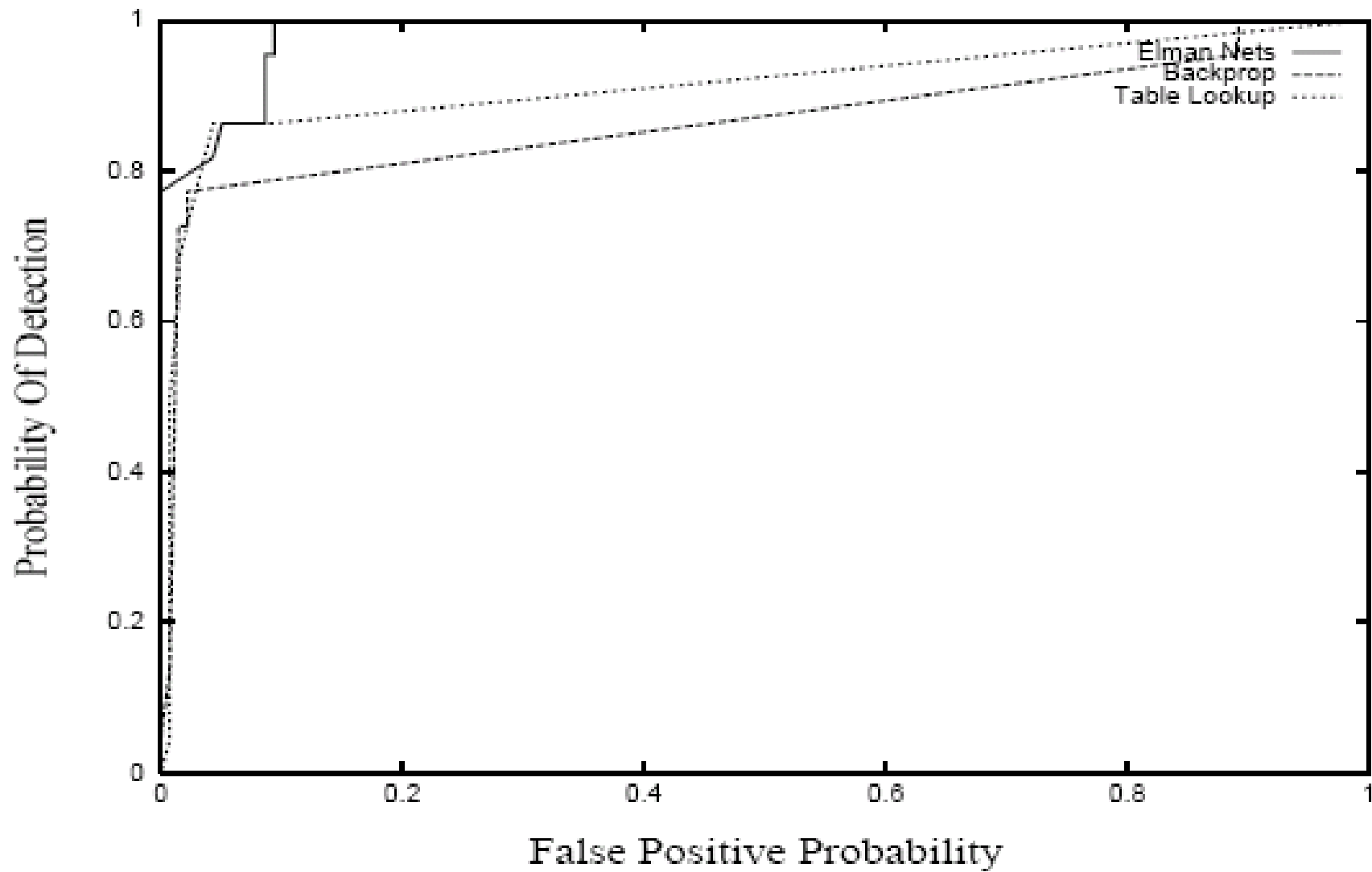
feed-forward with cycles



B

Context node





Stagnant Issues

- Corrupted training period
- The temporal clustering assumption

- Method 1: Equality Matching
- Method 2: Back Propagating N Nets
- Method 3: Cyclical ELman N Nets