

Introduction to Network and Systems Security

Aprendizagem Aplicada à Segurança

**Mestrado em Cibersegurança
DETI-UA**



Attacks to Networks and Systems

- Objectives:

- Fun and/or hacking reputation
- Political purposes
- Military purposes
- Economical purposes
- Other?

- Technical objectives:

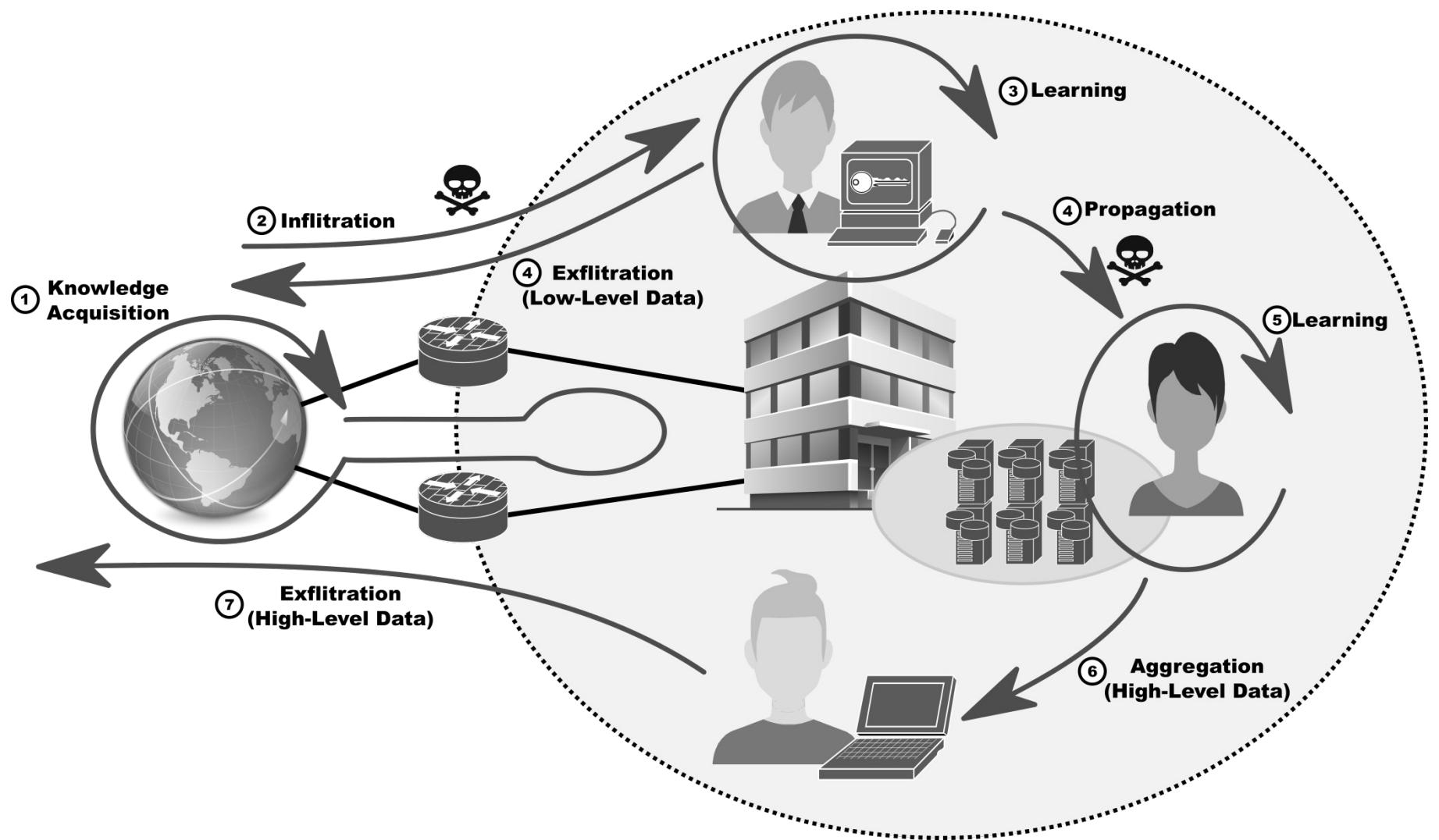
- Operation disruption
- For data interception
- Both
 - Disruption to intercept!
 - Intercept to disrupt!

- Most attacks never seen before

- Zero-day attacks

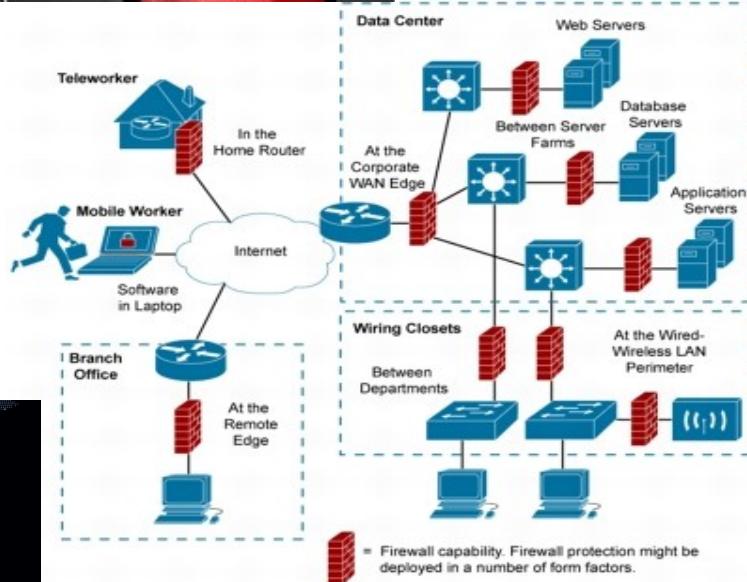


Attack Phases



Traditional Defenses

- Vulnerability patching.
 - Firewalls
 - ◆ Centralized.
 - ◆ Distributed.
 - Intrusion Prevention and Detection Systems (IDS/IPS).
 - Antivirus.



- All rely on previous knowledge of the threat and/or problem!

“Intelligent” Defenses

- Detection of unknown threats and/or problems.
 - ◆ In time to deploy counter-measures.
- Application of Big Data and Data Science techniques to network and systems monitoring data.
- Some traditional solutions start to incorporate AI into their equipment
 - ◆ E.g., Palo Alto Network Firewalls, Cisco Appliances, ...
- Still limited to manufacturer based solutions and localized data.
- Still limited in scope.
 - ◆ Obvious threats vs. Stealth threats.
- Optimal deployment requires an overall network and systems knowledge.
 - ◆ Network and Systems Awareness.



Awareness

- **Direct Awareness**
 - ◆ By direct observation.
- **Indirect Awareness**
 - ◆ By analysis of reactions to events.
- **Awareness by Correlation**
 - ◆ Joint analysis of multiple sources of data to detect hidden patterns and relations.
 - ◆ Big Data Problem.
- **Awareness by Prediction**
 - ◆ Detection of patterns over time.
 - ◆ Black Swan Problem!
- Its all an **Inference, Validation, Correction** loop.



Cyber Situational Awareness (1)

- Ability to effectively **Acquire Data** by **Monitoring** networks and systems to:
 - ◆ Optimize services,
 - ◆ Detect and counter-act anomalous activity/events.
- **Analyze/Process** data to know and characterize
 - ◆ Network entities,
 - An entity should be understood as a person, a group, a terminal, a server, an application, etc...
 - ◆ Data flows,
 - ◆ Services and users perception of service.



Cyber Situational Awareness (2)

- All data sources are acceptable.
 - ◆ Never assume data irrelevance!
- Data may be:
 - ◆ Quantitative.
 - ◆ Allows for statistical analysis and may serve as machine learning training input.
 - ◆ e.g., number of packets, number of flows, number of contacted machines, etc...
 - ◆ Qualitative.
 - ◆ Can be transformed to quantitative data by counting techniques and statistical characterization
 - ◆ e.g., error message X, address Y contacted, packet of type Z, etc...



Cyber Situational Awareness (3)

- Time is relevant.
 - ◆ History is relevant.
 - ◆ Relative and absolute.
 - ◆ An event occurs in a specific time instant, and it is part of a sequence of events.
- Timescale(s) of analysis must:
 - ◆ Include the target characteristics,
 - ◆ Allow the perception of the event in time for a response.
- Data may be re-scaled for multiple analysis purposes.



Cyber Awareness Steps

- Data acquisition.
- Data processing.
 - ◆ Creation of time sequences with different counting intervals (minimum timescales).
 - ◆ Creation of time sequences with different statistical metrics (larger timescales).
- Creation of entities' behavior profiles and objects' data patterns.
 - ◆ Usually time dependent.
- Classification of entities' behaviors and/or objects' data patterns.
 - ◆ Identification/classification.
 - ◆ Anomaly detection.
 - ◆ Rogue agent or manipulated/forged data object.



Cyber Situational Awareness

Aprendizagem Aplicada à Segurança

**Mestrado em Cibersegurança
DETI-UA**



Awareness

- **Direct Awareness**
 - ◆ By direct observation.
- **Indirect Awareness**
 - ◆ By analysis of reactions to events.
- **Awareness by Correlation**
 - ◆ Joint analysis of multiple sources of data to detect hidden patterns and relations.
 - ◆ Big Data Problem.
- **Awareness by Prediction**
 - ◆ Detection of patterns over time.
 - ◆ Black Swan Problem!
- Its all an **Inference, Validation, Correction** loop.



Cyber Situational Awareness (1)

- Ability to effectively **Acquire Data** by **Monitoring** networks and systems to:
 - ◆ Optimize services,
 - ◆ Detect and counter-act anomalous activity/events.
- **Analyze/Process** data to know and characterize
 - ◆ Network entities,
 - An entity should be understood as a person, a group, a terminal, a server, an application, etc...
 - ◆ Data flows,
 - ◆ Services and users perception of service.



Cyber Situational Awareness (1)

- All data sources are acceptable.
 - ◆ Never assume data irrelevance!
- Data may be:
 - ◆ Quantitative.
 - ◆ Allows for statistical analysis and may serve as machine learning training input.
 - ◆ e.g., number of packets, number of flows, number of contacted machines, etc...
 - ◆ Qualitative.
 - ◆ Can be transformed to quantitative data by counting techniques and statistical characterization
 - ◆ e.g., error message X, address Y contacted, packet of type Z, etc...



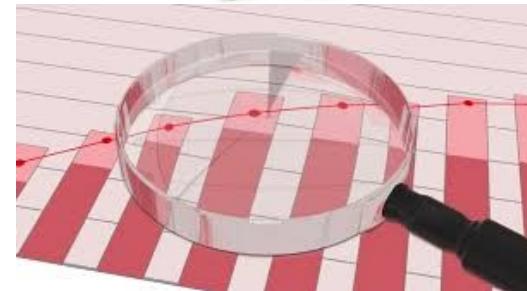
Cyber Situational Awareness (1)

- Time is relevant.
 - ◆ Relative and absolute.
 - ◆ An event occurs in a specific time instant, and it is part of a sequence of events.
- Timescale(s) of analysis must:
 - ◆ Include the target characteristics,
 - ◆ Allow the perception of the event in time for a response.
- Data may be re-scaled for multiple analysis purposes.



Situational Awareness Steps

- Data acquisition.
- Data processing.
 - ◆ Creation of time sequences with different counting intervals (minimum timescales).
 - ◆ Creation of time sequences with different statistical metrics (larger timescales).
- Creation of entities' behavior profiles.
 - ◆ Usually time dependent.
- Classification of entities' behaviors.
 - ◆ Identification/classification.
 - ◆ Anomaly detection.



Network Attack Vectors



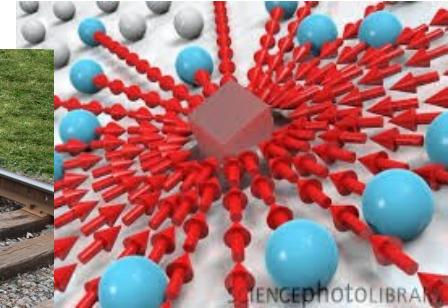
Type of Attacks (1)

- Objectives:
 - ◆ Fun and/or hacking reputation
 - ◆ Political purposes
 - ◆ Military purposes
 - ◆ Economical purposes
 - ◆ Other?
- Technical objectives:
 - ◆ Operation disruption
 - ◆ For data interception
 - ◆ Both
 - ◆ Disruption to intercept!
 - ◆ Intercept to disrupt!



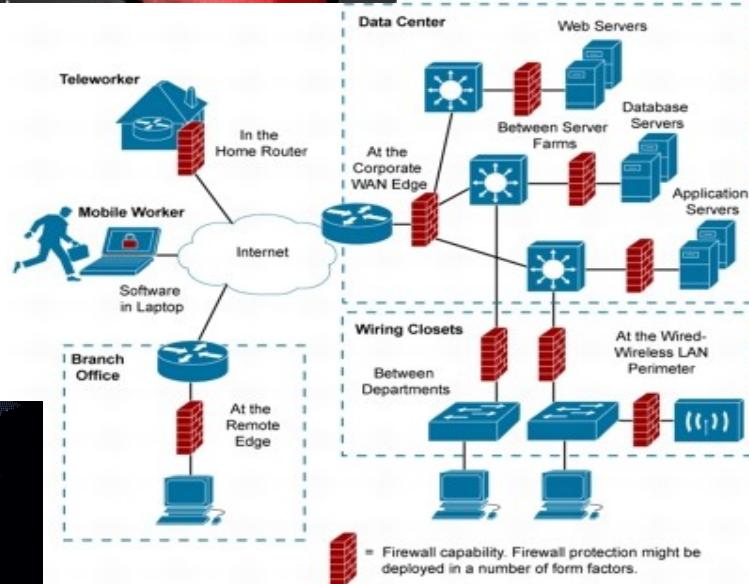
Type of Attacks (2)

- Technical objectives:
 - ◆ Operation disruption.
 - (Distributed) Denial-of-Service.
 - ◆ Resources hijack.
 - Spam,
 - Crypt-currency mining/masternodes,
 - Platform to other attacks!
 - ◆ Data interception/stealing.
 - Personal data
 - As final goal,
 - Or as tool to achieve more value information!
 - Technical data,
 - Usually used to achieve more value information!
 - Commercial data
 - Digital objects, financial and/or engineering plans, ...
- Disruption may be used to achieve interception!
- Interception may be used to achieve disruption (operational or commercial)!



Traditional Defenses

- Vulnerability patching.
- Firewalls
 - ◆ Centralized.
 - ◆ Distributed.
- Intrusion Prevention and Detection Systems (IDS/IPS).
- Antivirus.



- All rely on previous knowledge of the threat and/or problem!



“Intelligent” Defenses

- Detection of unknown threats and/or problems.
 - ◆ In time to deploy counter-measures.
- Application of Big Data and Data Science techniques to network and systems monitoring data.
- Some traditional solutions start to incorporate AI into their equipment
 - ◆ E.g., Palo Alto Network Firewalls, Cisco Appliances, ...
- Still limited to manufacturer based solutions and localized data.
- Still limited in scope.
 - ◆ Obvious threats vs. Stealth threats.
- Optimal deployment requires an overall network and systems knowledge.
 - ◆ Network and Systems (Cyber) Situational Awareness.



Disruption Attacks

• Distributed DoS

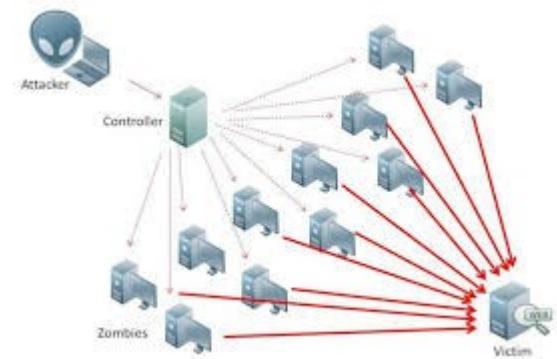
- ◆ Multiple slow/small devices generating traffic to a target
 - ➡ TCP vs. UDP
- ◆ Purpose of disruption
 - ➡ By political/economical/"reputation"
 - ➡ Redirection to other service/location?
- ◆ Solution at target
 - ➡ Load-balancers
 - ➡ For TCP, maybe its possible to survive making active (with licit client validation) session resets (server/firewalls)
 - White list solution, for completed session negotiation
 - ➡ For UDP/DNS, block requests for known external relay/redirection DNS servers (blocks attack amplification, IP target spoofing)
 - Doesn't work with large botnets and direct requests to target

◆ Solution at source

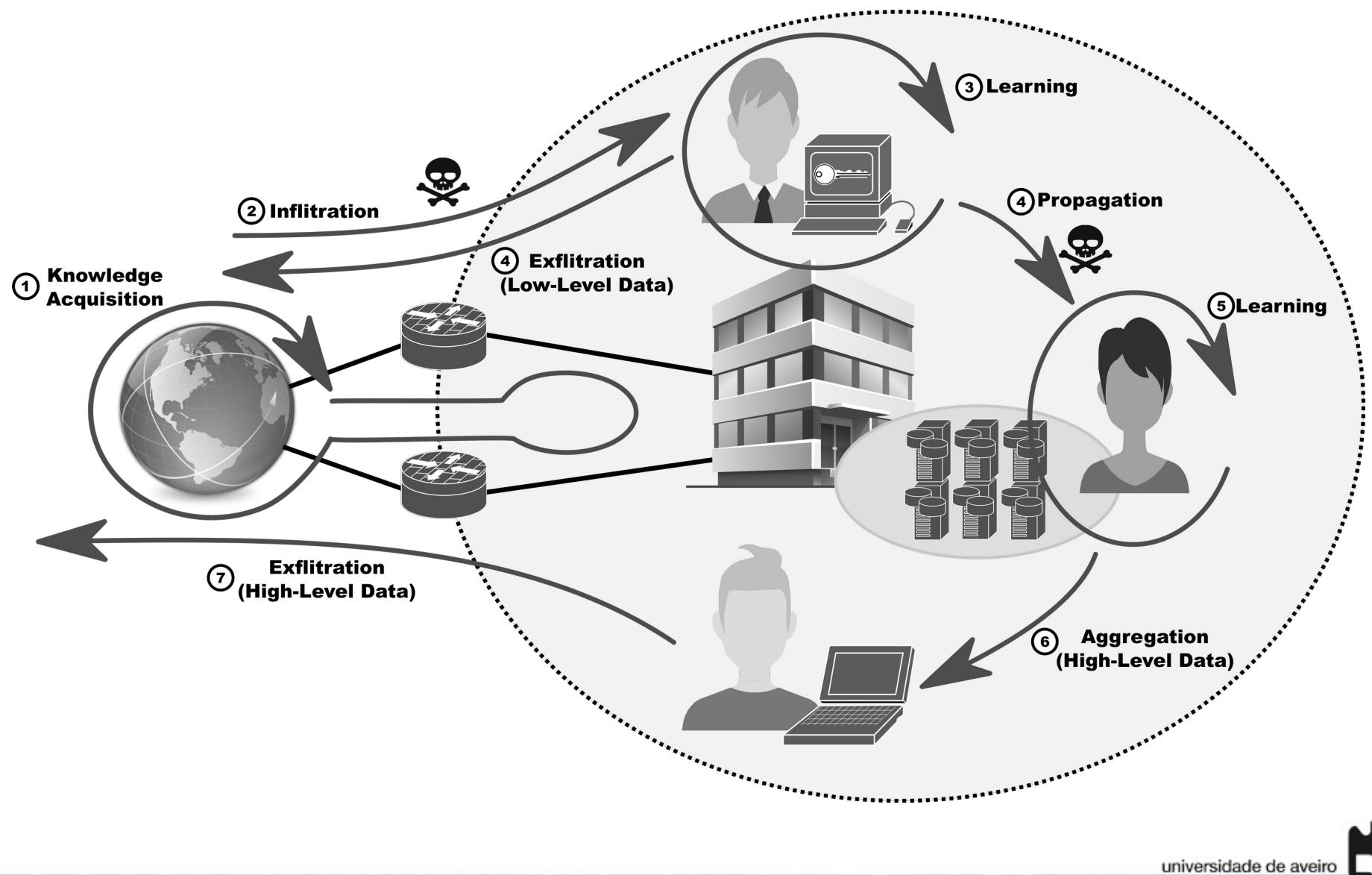
- ➡ Anomalous behaviors detection
 - Low traffic variations hard to detect
 - Time and periodicity changes are easier to detect
 - Destinations of traffic changes
 - With "really low" data rates is impossible to detect

• Denial o service by physical signal jamming

- ◆ Pure disruption, or
- ◆ Disruption to activate secondary channels (more easily compromised).
- ◆ Solution
 - ➡ Detect, localized source and physically neutralize.



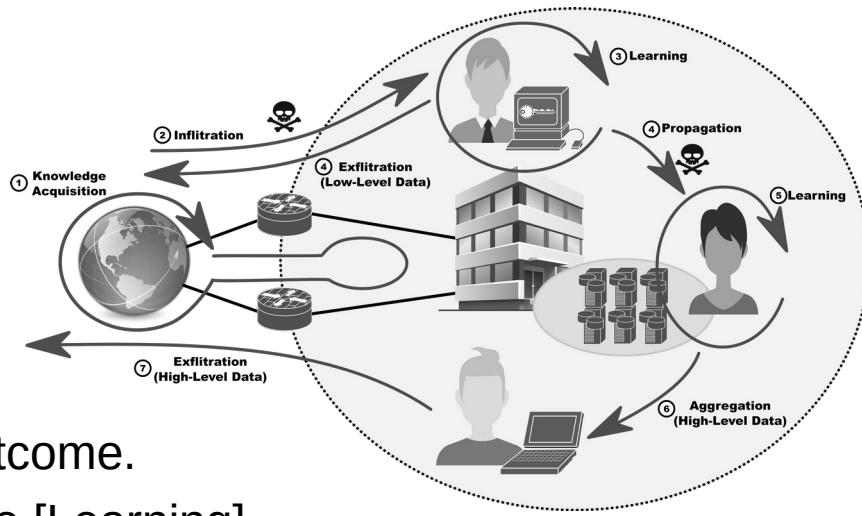
More Advanced Attacks Phases



Attacks are Done Incrementally

- Escalation of goals and privileges.

- ◆ Public knowledge opens doors to private information and access to protected domains [Infiltration].
- ◆ The first illicit access to a protect domain may not provide a relevant outcome.
- ◆ Attacker must acquire more knowledge [Learning].
- ◆ The additional knowledge allows to access other secure domain zones/devices/data with increasing relevance [Propagation].
 - ◆ At any phase the attacker may require additional knowledge [Learning].
- ◆ When a relevant outcome is acquired it must be transferred to outside of the protected domain [Exfiltration].
- ◆ Direct exfiltration may denounce the relevant points inside of the secure domain.
 - ◆ The relevant outcome must be first transferred inside the protected domain to a less important point [Aggregation].
 - ◆ Attacker chooses a point that may be detected and lost without harm.



Infiltration Phase

- Licit machines must be compromised to implement the different attacks phases.
 - ◆ Ideally in a privileged “zone” of the network, and/or
 - ◆ With access credentials, and/or
 - ◆ User credentials, address(es), hardware key, etc...
 - ◆ With “special” software, and/or
 - ◆ Target data.
- May include the installation of software or usage of licit vulnerable software.
- May be remotely controlled (constantly or not).
 - ◆ Command and control (C&C).
- May have autonomous (AI) bots installed to perform illicit actions.
 - ◆ When remote C&C is not possible or subject to easy detection.



Remotely by Exploiting Licit Users

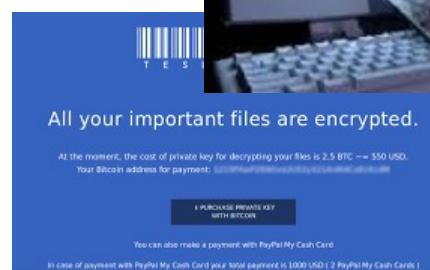
- Objectives:

- Objectives:
 - ◆ Credentials acquisition.
 - ◆ Software insertion.
 - ◆ Ramsomware.



- Vectors:

- Vectors:
 - ◆ E-mail and social networking
 - ◆ Phishing for credentials.
 - ◆ Office macros.
 - ◆ Binaries execution.
 - ◆ Downloadable software
 - ◆ Cracks.
 - ◆ Non-certified software stores.
 - ◆ ...



Remotely by Attacker Actions

- Possible when network/systems have unpatched (unresolved) vulnerabilities.
 - ◆ Limited in time.
- Possible when network/systems are poorly configured/designed
 - ◆ Less limited in time.
 - ◆ Hard to perform discovery without detection by traditional defense systems.
 - ➡ Sometimes poorly configured/designed systems are not protected by adequate systems (if any).
- Usually not done first.
- Done after acquiring some credentials/privileges from licit users.
 - ◆ Using direct connections/services.
 - ◆ Easier to hide (stealth attacks) by having reduce activity or mimicking licit usage.



Locally by Physical Interaction

- Objectives:

- ◆ Traffic interception.
- ◆ Local network access to exploit vulnerabilities.
- ◆ Direct access to machine.

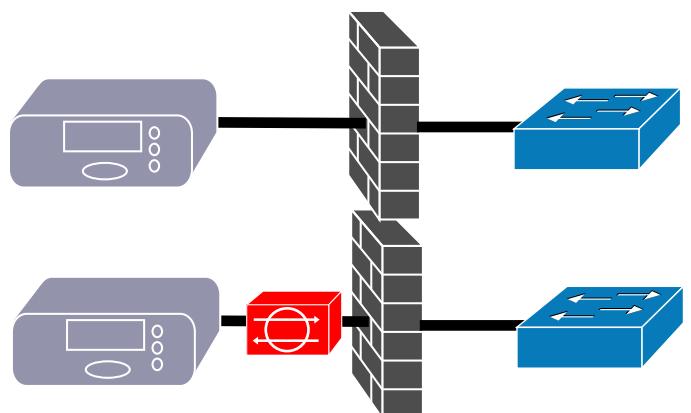
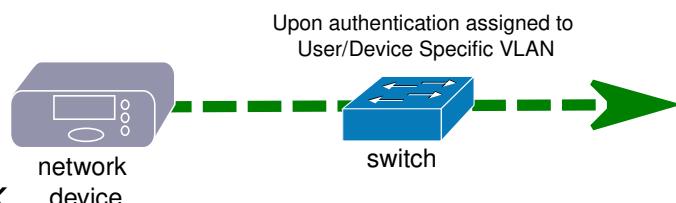
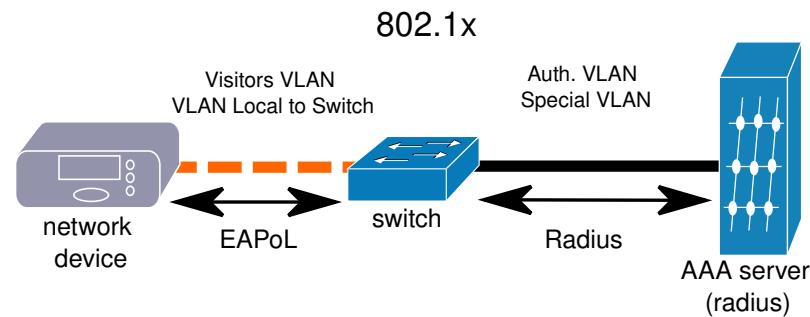
- Vectors:

- ◆ Ethernet ports at public/unprotected locations
 - With VLAN separation
 - Without VLAN separation
 - Protected by 802.1X
- ◆ Network taps at public/unprotected locations
- ◆ Fake access points.
 - Rogue access points
- ◆ Network devices access
 - Unprotected serial/console ports, USB ports, etc...
- ◆ USB ports (short time access)
 - Long time objectives
 - Trojan/root kits injection.
 - Short time objectives
 - Device data acquisition (contacts, messages, sms, etc...)
- ◆ Sitting down at a terminal or with a device!
- ◆ Other?



Illicit usage of Ethernet ports

- Common protection:
 - ◆ VLAN separation/isolation.
 - ◆ 802.1X.
- Unused ports
 - ◆ VLAN separation/isolation and/or 802.1X may be enough to mitigate more dangerous attacks (L2 or L3 access to internal machines).
 - ◆ Switches MAC flooding attacks and Network overload (Local DoS) are possible.
- In use ports
 - ◆ Using an inline device it is possible to break 802.1X using terminal/user authentication.
 - ◆ Traffic pass-through.
 - ◆ After 802.1X authentication performs inline MAC spoofing.
 - ◆ Allows for traffic snooping, injection, and MITM attacks.



Network Tapping

- Switch rogue mirror ports.
 - ◆ Allows for traffic snooping and injection, no MITM attacks.
 - ◆ Solution: Constant monitoring of configuration changes on network devices.
- Ethernet cable tap
 - ◆ Allows for traffic snooping and injection, no MITM attacks.
 - ◆ Solution: Electrical variations. Maybe...?
- Optical cable tap
 - ◆ Allows for traffic snooping and injection, no MITM attacks.
 - ◆ Solution: Quantum cryptography



Wireless Attack Vectors

- Rogue APs

- ◆ WPA PSK and WPA2 PSK are not compromised.
 - Unless device associates to networks with (fake) SSID of known networks with different credentials and/or secure protocols.
 - Decision to connect based only on stored SSID and not other parameters.
- ◆ WPA Enterprise and WPA2 Enterprise security may be compromised on 2nd phase authentication.
 - Credentials not recoverable (maybe with MSCHAPv2).
 - Permits “accept everyone” strategy for MITM attacks.
- ◆ Open+Web-based authentication are very vulnerable.
 - Fake entry portals.
- ◆ Allows DoS.
 - Force user to search other networks. Make user choose insecure/fake network.

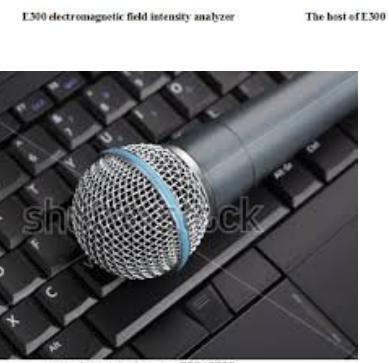
- Wireless Interception (possible injection).

- Electromagnetic effects

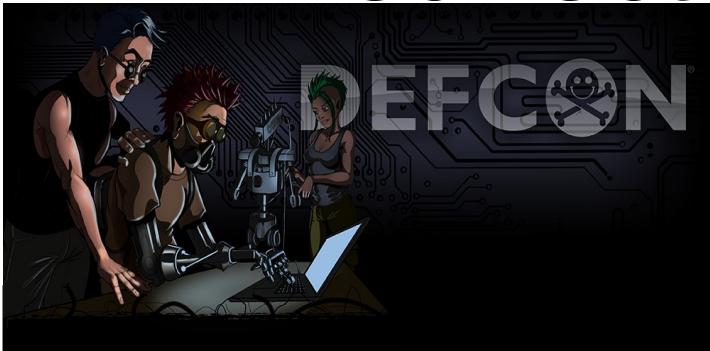
- ◆ Wireless mouses, keyboards, ...
 - Solution: additional information to scramble data.

- By Sound

- ◆ Keystrokes sounds.



BGP & Internet-Scale Traffic Redirection Attack (2008)



Stealing The Internet

An Internet-Scale
Man In The Middle Attack

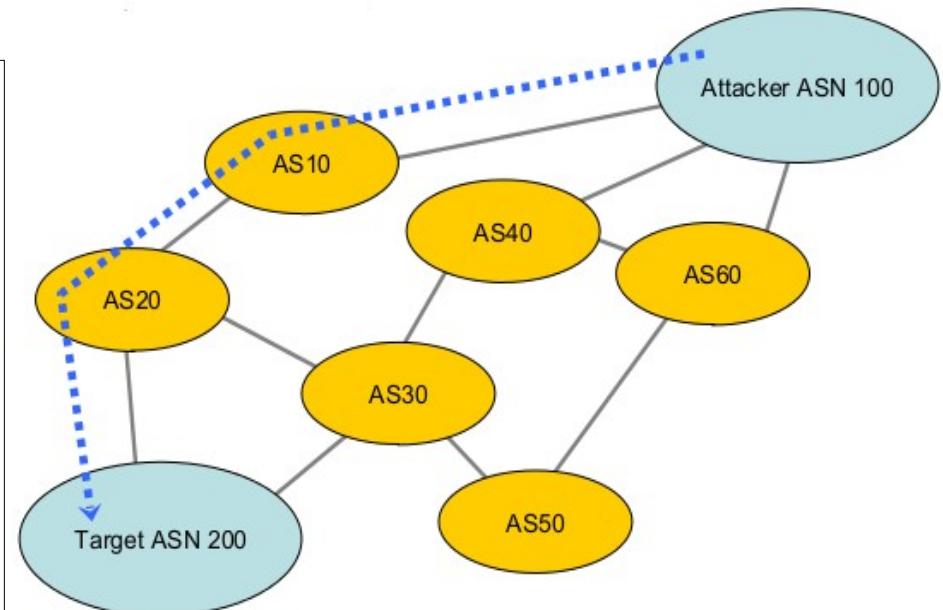
Defcon 16, Las Vegas, NV - August 10th,
2008

Alex Pilosov – Pure Science
Chairman of IP Hijacking BOF
ex-moderator of NANOG mailing list
alex@pilosoft.com

Tony Kapela – Public Speaking Skills
CIO of IP Hijacking BOF
tk@5ninesdata.com



BGP MITM – Plan reply path

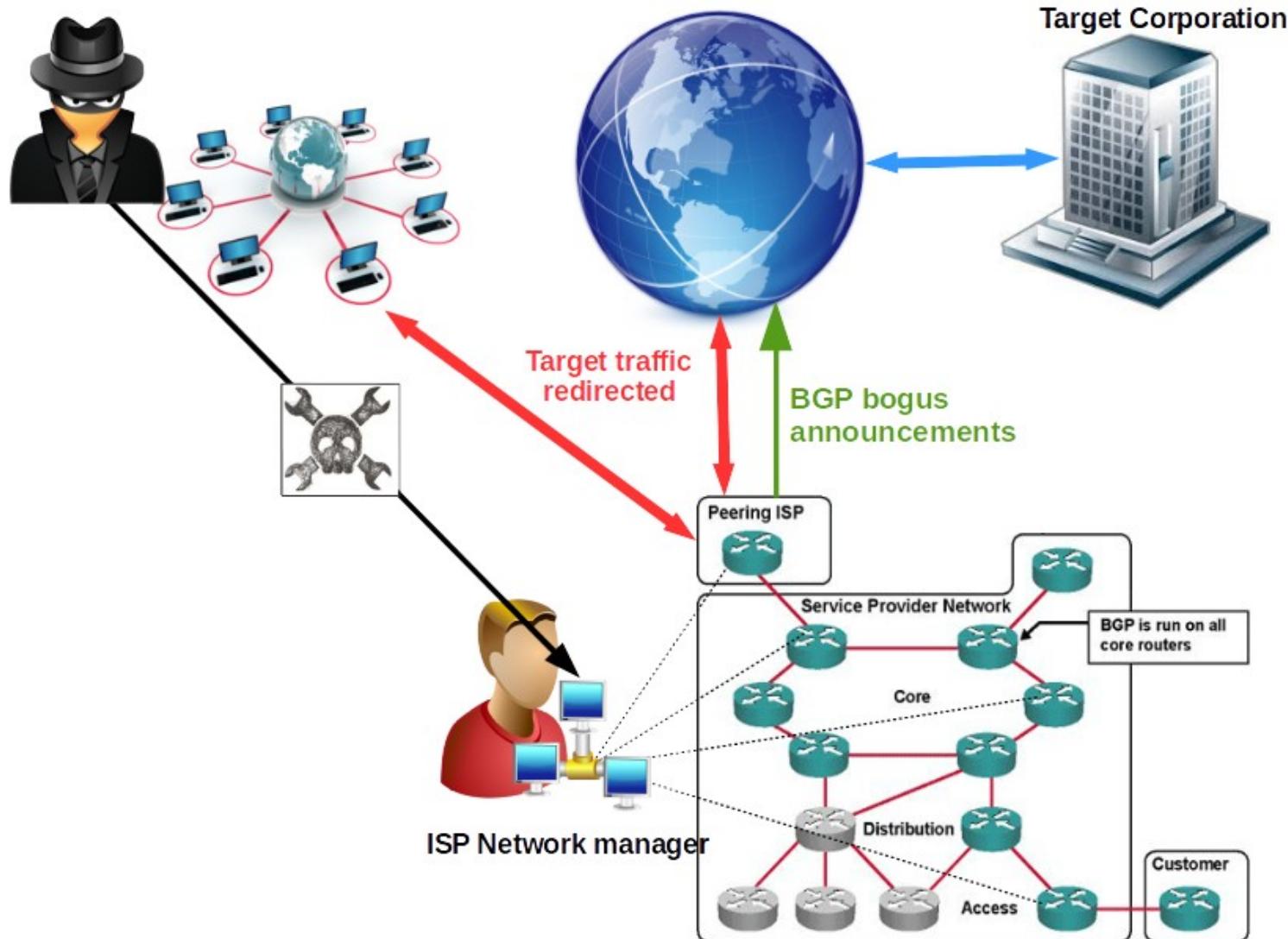


<http://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf>



universidade de aveiro

MP-BGP Attack Vector



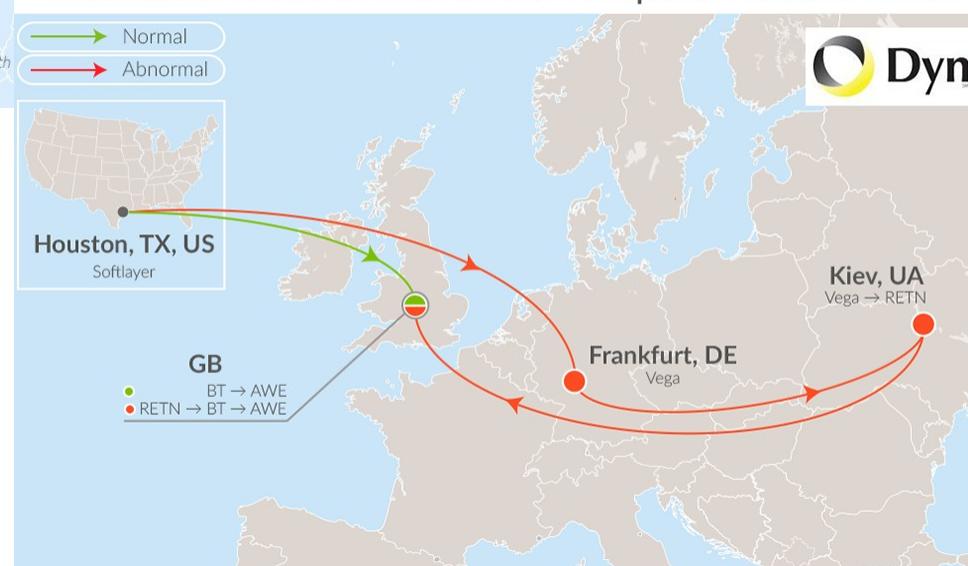
Latest (known/public) Reports

Traceroute Path 1: from Guadalajara, Mexico to Washington, D.C. via Belarus



Dyn Research
THE NEW HOME OF •renesys®

Redirected traffic to UK Atomic Weapons Establishment



<http://dyn.com/blog/uk-traffic-diverted-ukraine/>

Propagation Phase

- Done using a mixture of methodologies:
 - ◆ Credentials exploitation.
 - Direct usage or by using allowed applications.
 - ◆ Impersonating users and systems.
 - Similar to credential exploitation but more advanced based on acquired knowledge (licit behavior).
 - Requires time to learn and mimic licit behavior.
 - Time patterns, traffic patterns, application patterns, etc...
 - ◆ Vulnerability exploitation.
 - Inside a protected domain systems are many times considered in a secure zone.
 - Less maintained and legacy OS/applications may be required to run (no patching).
 - Broader range of vulnerabilities



Aggregation and Exfiltration Phase

- Data transferred from machine to machine.
- Internally [Aggregation] it can be done using existing channels.
- Externally [Exfiltration]
 - ◆ It can be done directly using existing channels.
 - ◆ File copy, email, file sharing, etc...
 - ◆ Can be detected.
 - ◆ It can be done hiding information within existing/allowed channels and illicit communications.
 - ◆ Slower data transfer, harder (impossible?) to detect.
 - ◆ Examples:
 - Usage of steganography in photos (via social networking).
 - Usage of embed data in text and voice messages.
 - ...



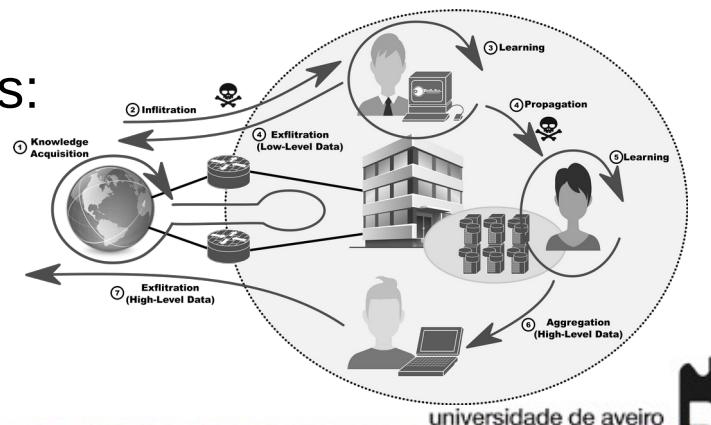
Data Manipulation/Forgery

- Attacks may not exfiltrate data, but manipulate it at source.
- If attacker has administration rights my also compromise common validation tools.
 - ◆ Hash functions, version control, modification flags, logging,etc...
- External data analysis tools must be used to detect drastic/characteristic alterations of data patterns.
 - ◆ Image and video manipulation (fakes).
 - ◆ Database entries/tables/blobs.
 - ◆ Source code repositories (supply chain attacks).
 - ◆ Service/device configurations.
 - ◆ Etc...
- Methodologies may be applied in pseudo-real time or “*a posteriori*” for forensics purposes.



Challenges

- Traditional network/systems defenses cannot guarantee total security of machines.
- Cannot prevent infiltration.
 - ◆ User Liberty vs. Data Confidentiality vs. Security equilibrium.
 - ◆ New threats/methodologies are almost impossible to prevent.
- A network manager must assume that any machine may be compromised at any time.
- Solution:
 - ◆ Monitor network and systems to prevent more damaging actions from/in compromised machines.
 - ◆ Detect attack in more important phases:
 - Network Propagation,
 - Network Aggregation,
 - Data Exfiltration (Most Important!).



Security News and Events

- www.bleepingcomputer.com
- www.securityboulevard.com
- www.threatpost.com
- www.reddit.com/r/security/
- www.reddit.com/r/cybersecurity/



Data Acquisition



Core and End-to-End Monitoring

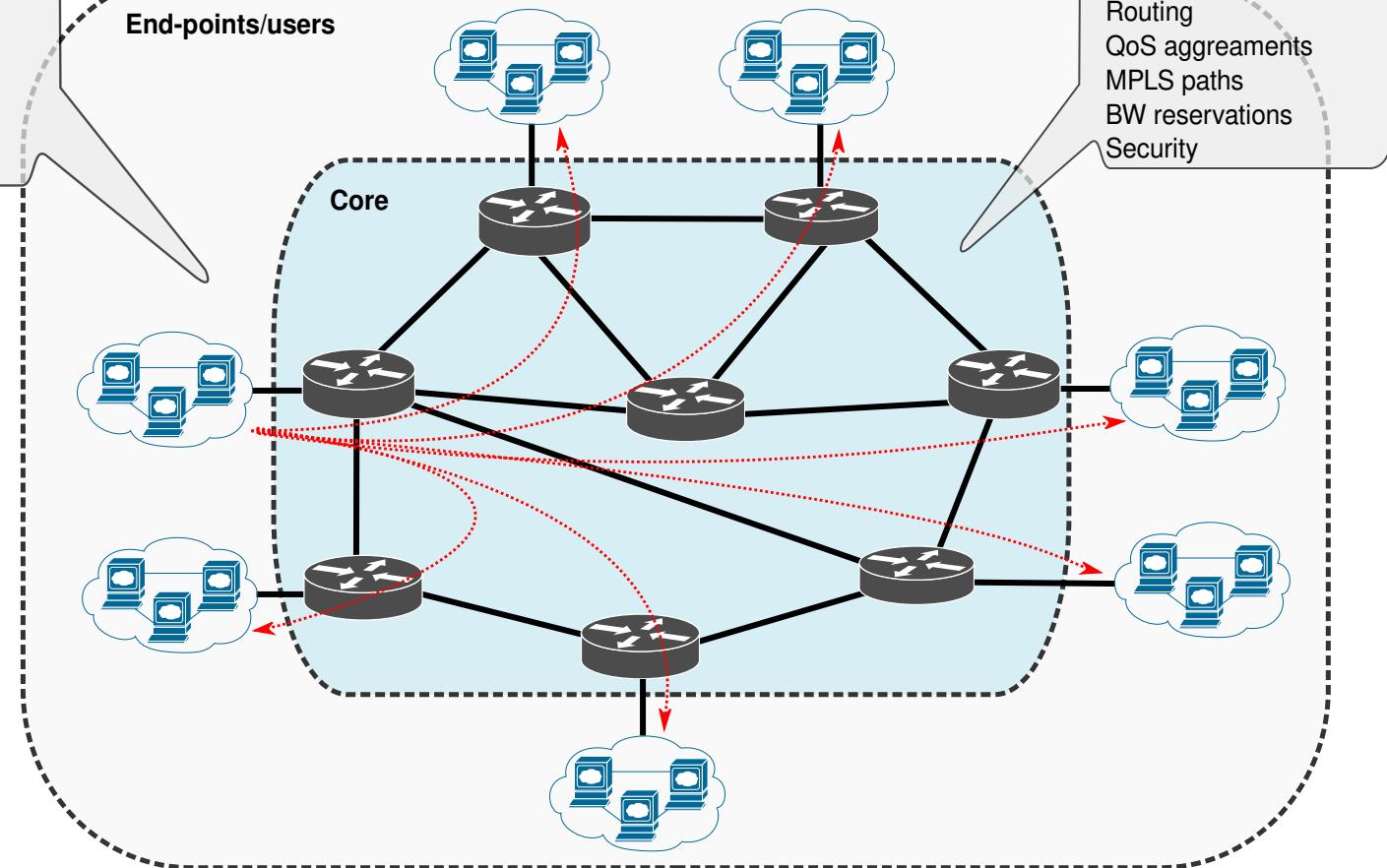
End-to-end measurements

- delay
- jitter
- throughput
- losses
- BW reservations
- reserved paths validation
- Demands per destination
- global
- per service/app
- per QoS usage

End-points/users

Core configurations

- Node awareness
- Service awareness
- Nodes performance
- Links performance
- Routing
- QoS agreements
- MPLS paths
- BW reservations
- Security



Core and End-to-End Monitoring

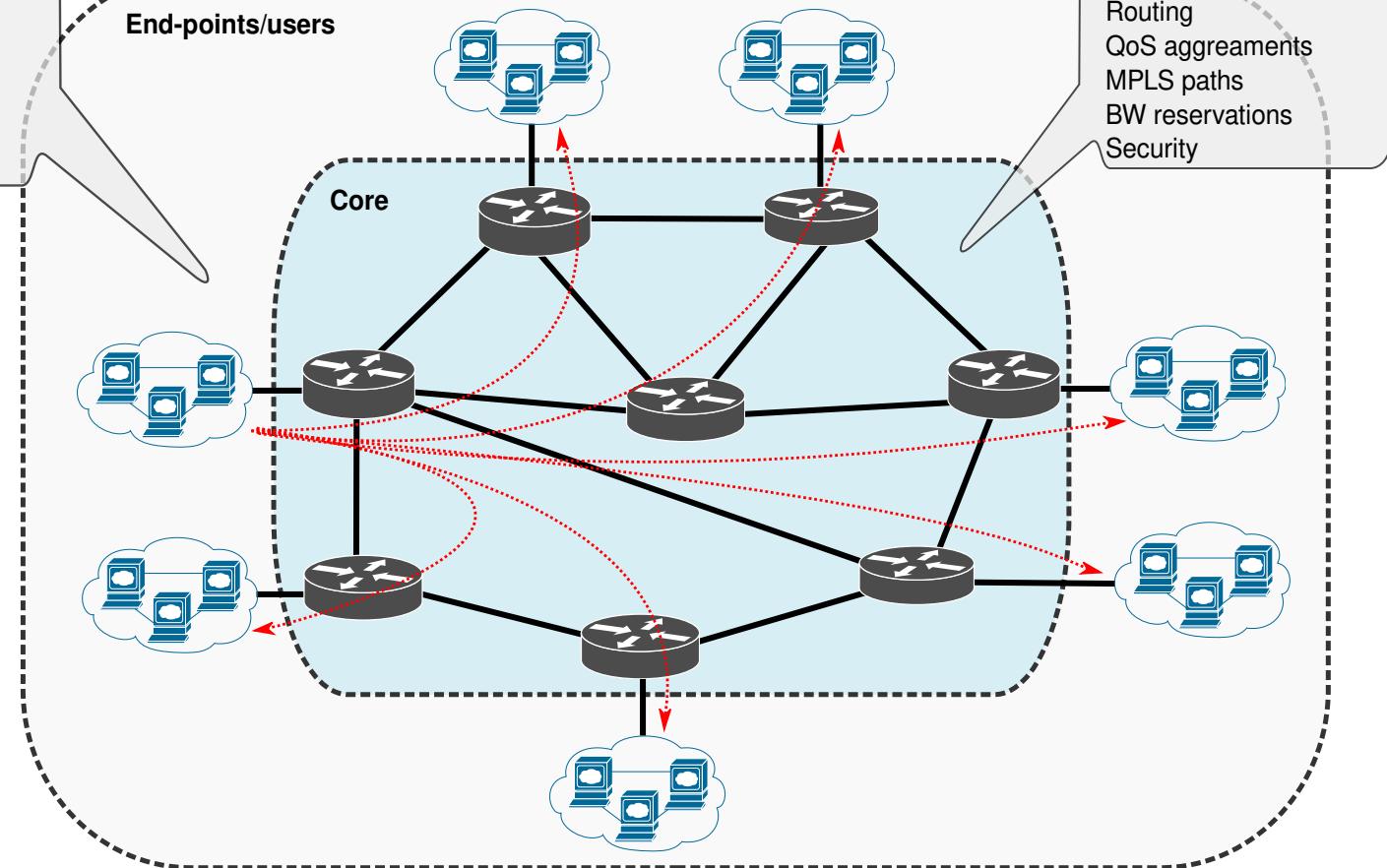
End-to-end measurements

- delay
- jitter
- throughput
- losses
- BW reservations
- reserved paths validation
- Demands per destination
- global
- per service/app
- per QoS usage

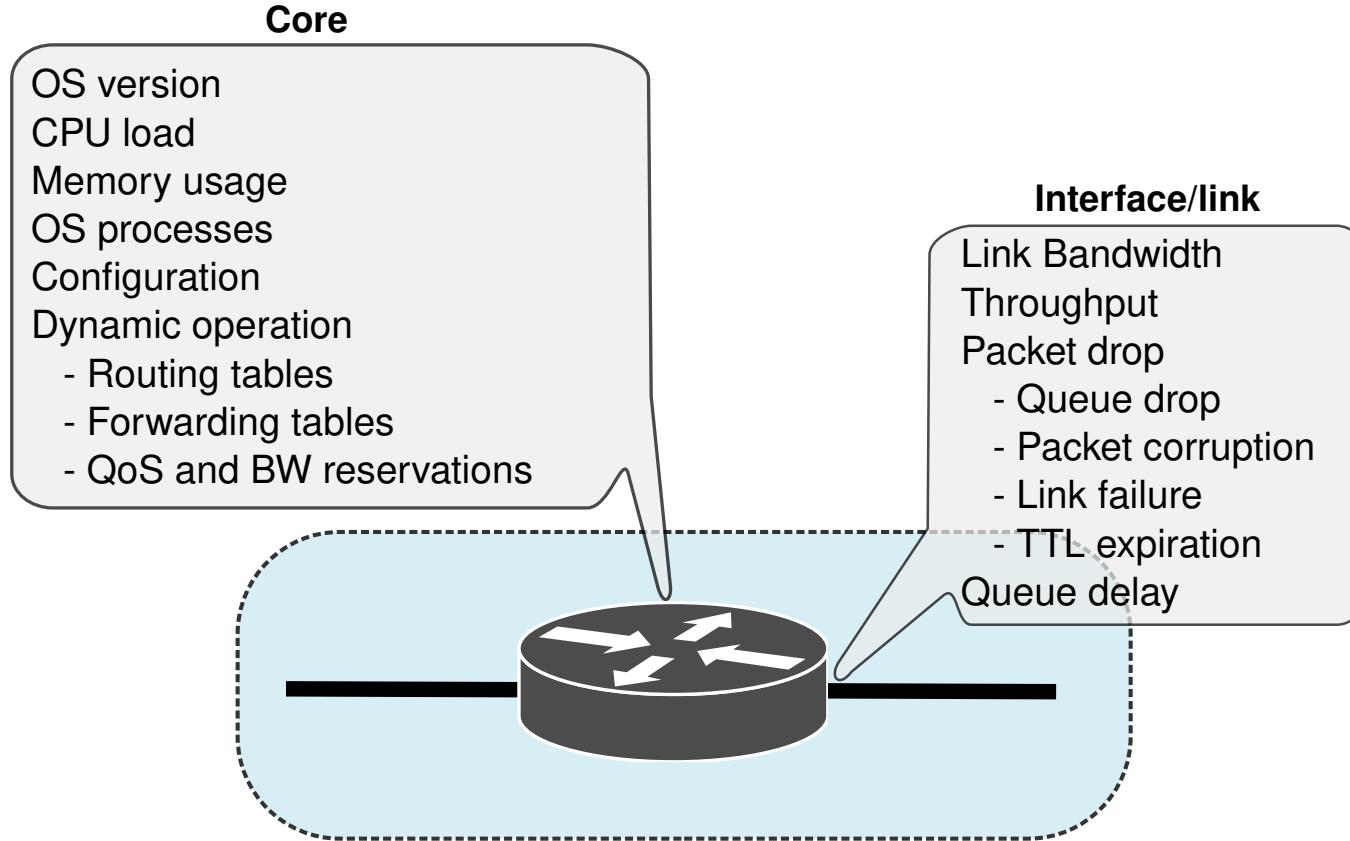
End-points/users

Core configurations

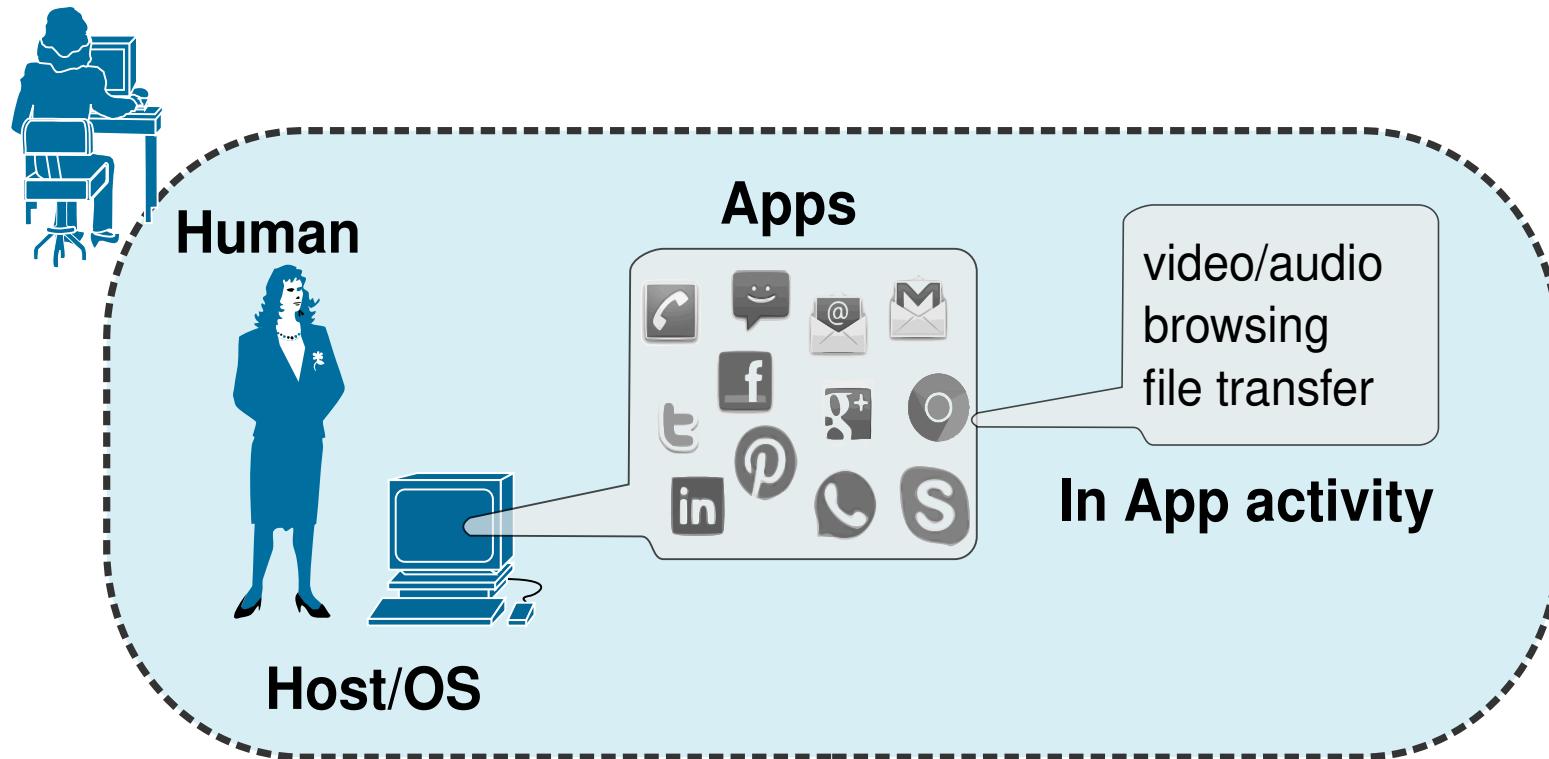
- Node awareness
- Service awareness
- Nodes performance
- Links performance
- Routing
- QoS agreements
- MPLS paths
- BW reservations
- Security



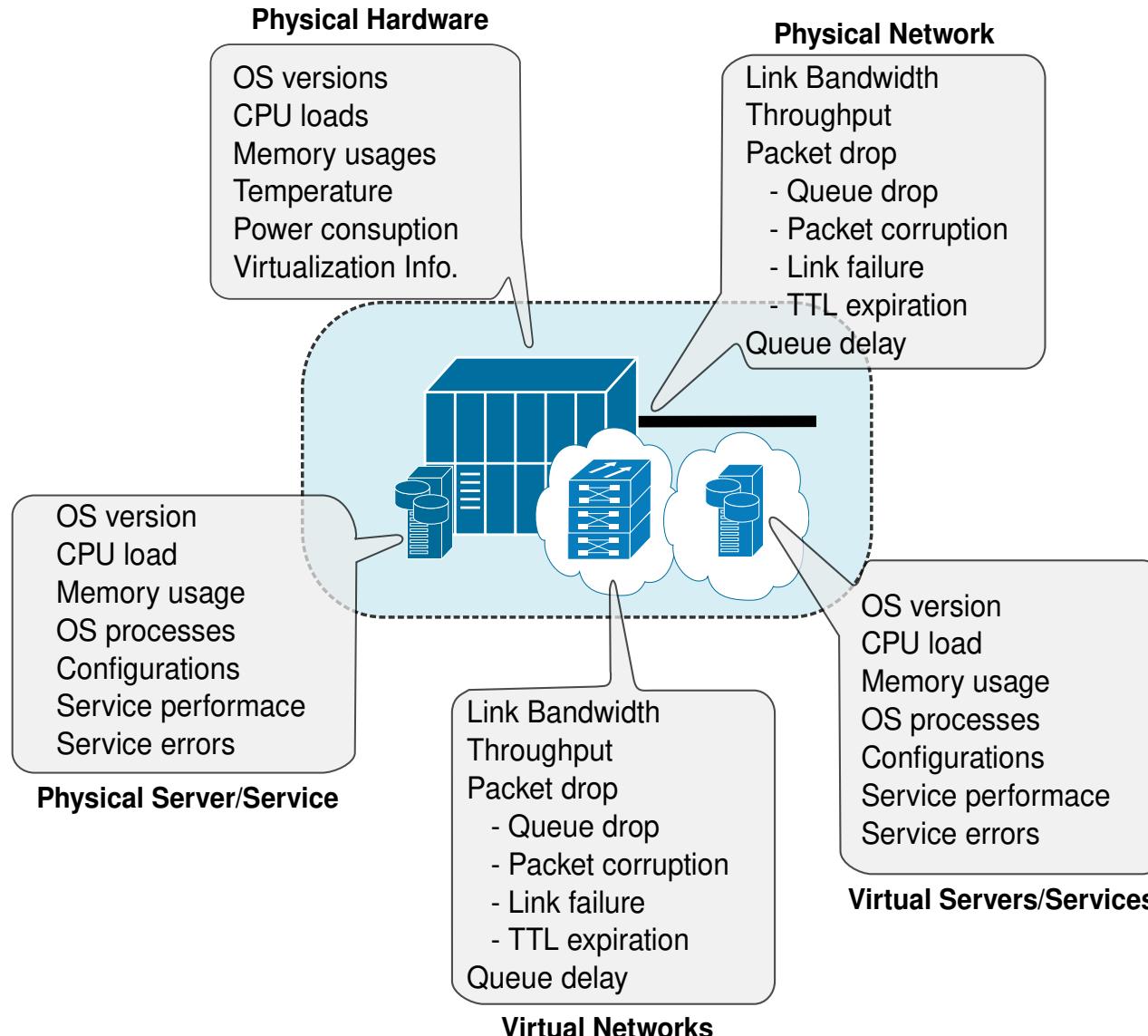
Node Monitoring



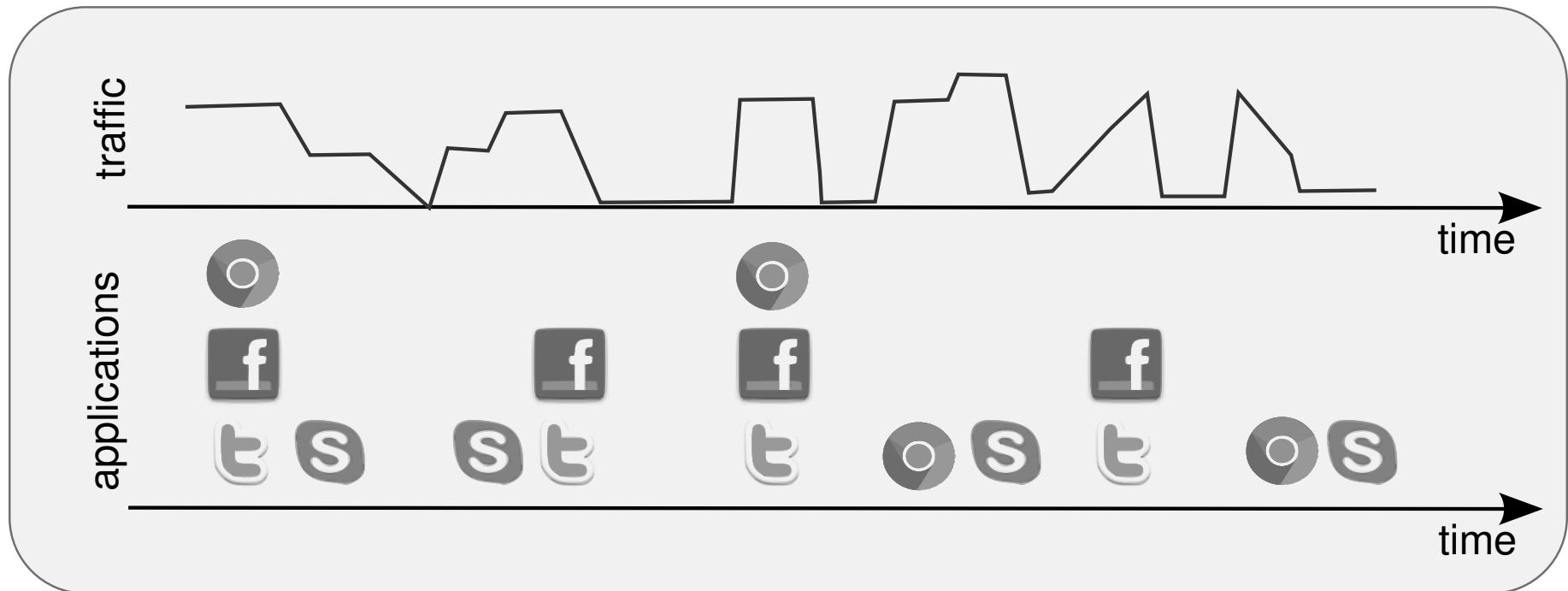
End-User/Host/App Monitoring



Server/Service/Cloud Monitoring



Overtime Monitoring



Data Sources

- SNMP
 - ◆ Used to acquire knowledge about current states of nodes/links/servers.
 - ◆ Local information. May be used to extrapolate to global information.
 - ◆ (Often) Requires the usage of vendor specific MIBs.
- Flow exporting
 - ◆ Used to characterize users/services in terms of amount of traffic and traffic destinations.
 - ◆ Medium and large time-scale information.
 - ◆ Protocols: Cisco NetFlow, IPFIX – Standard, Juniper jFlow, and sFlow
- Packet Captures / RAW statistics / DPI vs. SPI
 - ◆ Used to characterize users/services in small time-scales.
 - ◆ Requires distributed dedicated probes.
- Access Server/Device logs and/or CLI access.
 - ◆ Used to acquire knowledge about past and current state.
- Active measurements
 - ◆ Introduces entropy on network and requires (for many measurements) precise clock synchronization
 - ◆ E.g., one-way delay/jitter, round-trip delay/jitter.



SNMP

- Used for acquiring the status and usage of nodes links and services over time.

- ◆ Requires periodic pulling to obtain information over time

- Used for obtain:

- ◆ Network elements and interconnections,
 - ◆ Network deployed services.

- Used for estimating, characterizing, and predicting

- #### ◆ Data flow performance.

- Packet losses and (by indirect inference) delay/jitter at nodes.

- Allows to obtain information about current and future service performance

- ## ◆ Nodes performance,

- Memory/CPU usage, number of processes, etc...

- Allows to detect points of failure, service degradation nodes, unstable nodes

- ## ◆ Network link usage,

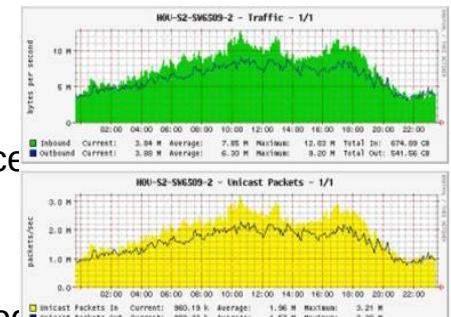
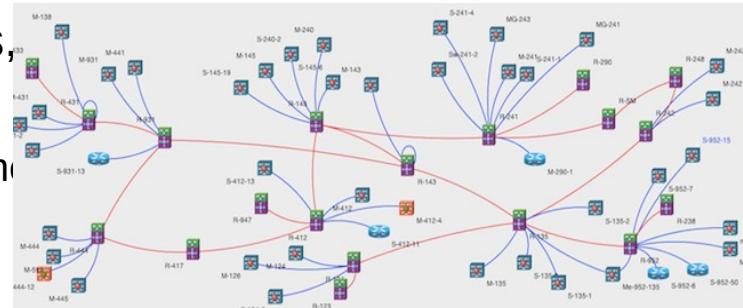
- Ingress/egress bytes and packet counts.

- Allows to perform optimizations in terms of routing (load balancing), link upgrade, and introduction of redundancy.

- ## ◆ Data/flow routing,

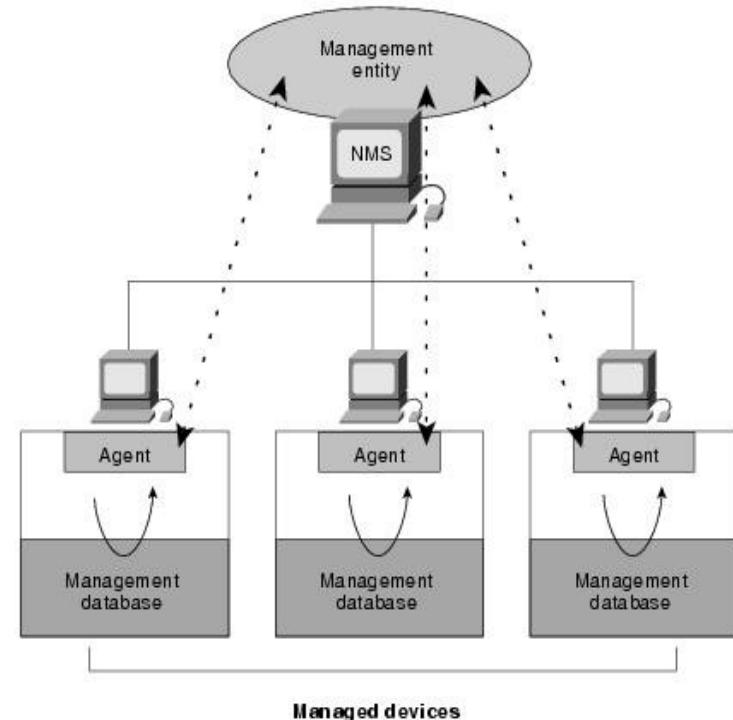
- At Layer 2, Layer 3 and MPLS levels.

- Allows to understand how data flows and how may react to disruptive events.



SNMP Basic Components

- An SNMP-managed network consists of three key components:
 - Managed devices
 - ◆ Network node that contains an SNMP agent.
 - ◆ Collect and store management information and make this information available using SNMP.
 - ◆ Can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.
- Agents
 - ◆ Network-management software module that resides in a managed device.
- Network-management systems (NMSs)
 - ◆ Executes applications that monitor and control managed devices.
 - ◆ Provide the bulk of the processing and memory resources required for network management.
 - ◆ One or more NMSs must exist on any managed network.



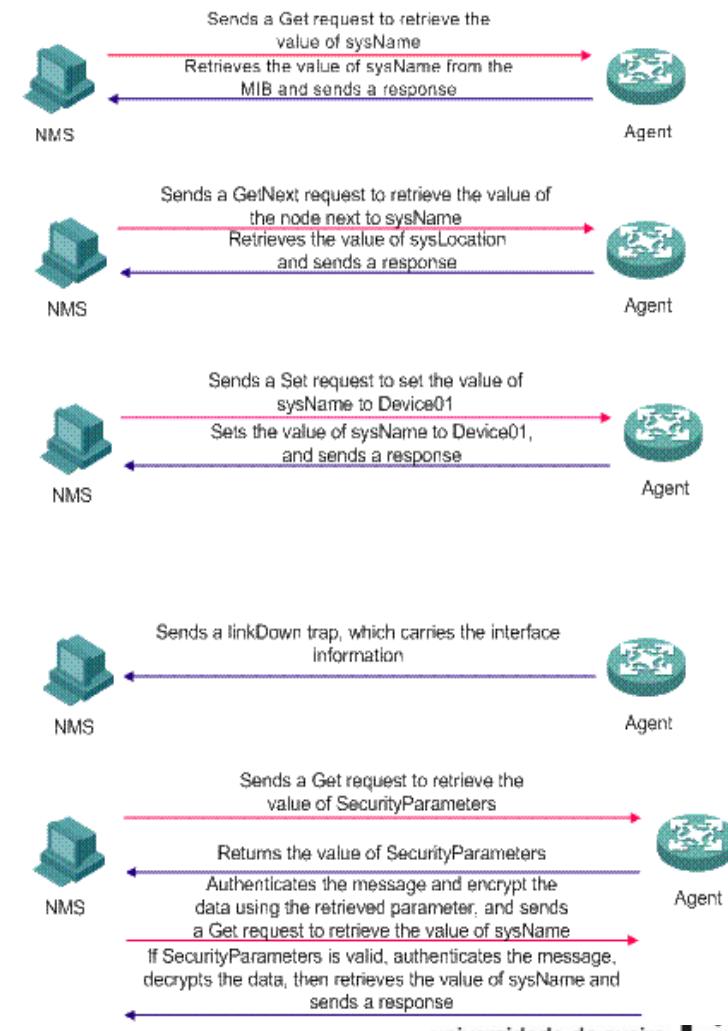
SNMP Versions

Model	Level	Authentication	Encryption	What Happens
v1	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for authentication.
v3	authNoPriv	MD5 or SHA	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithm.
v3	authPriv	MD5 or SHA	DES or AES	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides DES 56-bit or CFB128-AES-128 encryption in addition to authentication based on the CBC-DES (DES-56) standard.



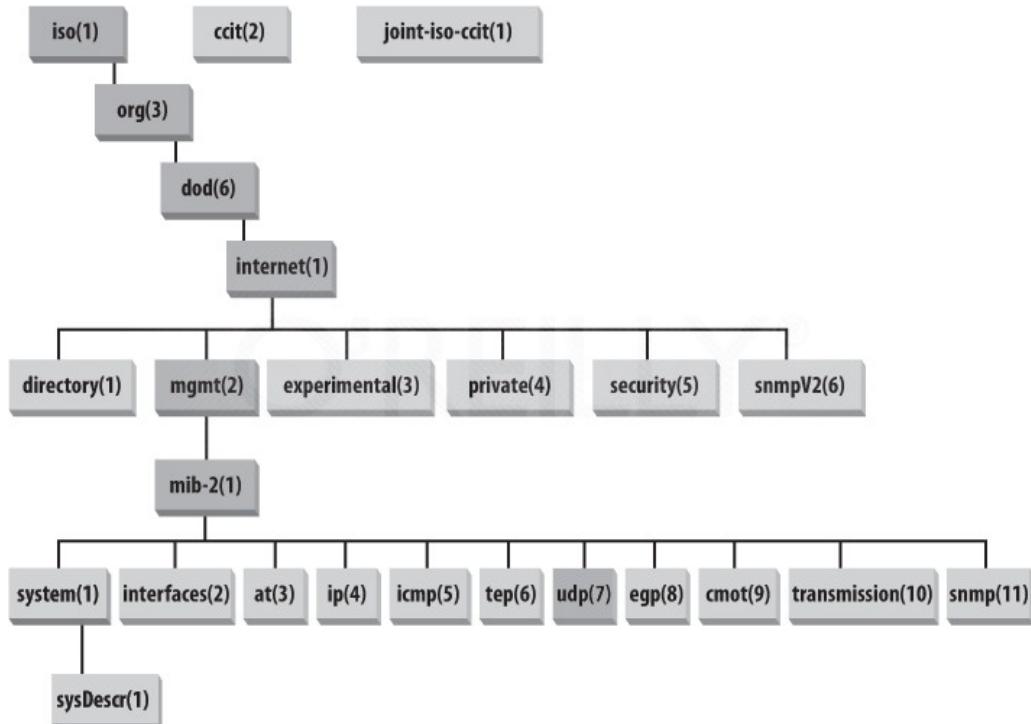
SNMP Operations

- SNMP provides the following five basic operations:
 - ◆ Get operation
 - ➡ Request sent by the NMS to the agent to retrieve one or more values from the agent.
 - ◆ GetNext operation
 - ➡ Request sent by the NMS to retrieve the value of the next OID in the tree.
 - ◆ Set operation
 - ➡ Request sent by the NMS to the agent to set one or more values of the agent.
 - ◆ Response operation
 - ➡ Response sent by the agent to the NMS.
 - ◆ Trap operation
 - ➡ Unsolicited response sent by the agent to notify the NMS of the events occurred.
- In SNMPv3 get operations are performed using authentication and encryption.



MIB Modules and Object Identifiers

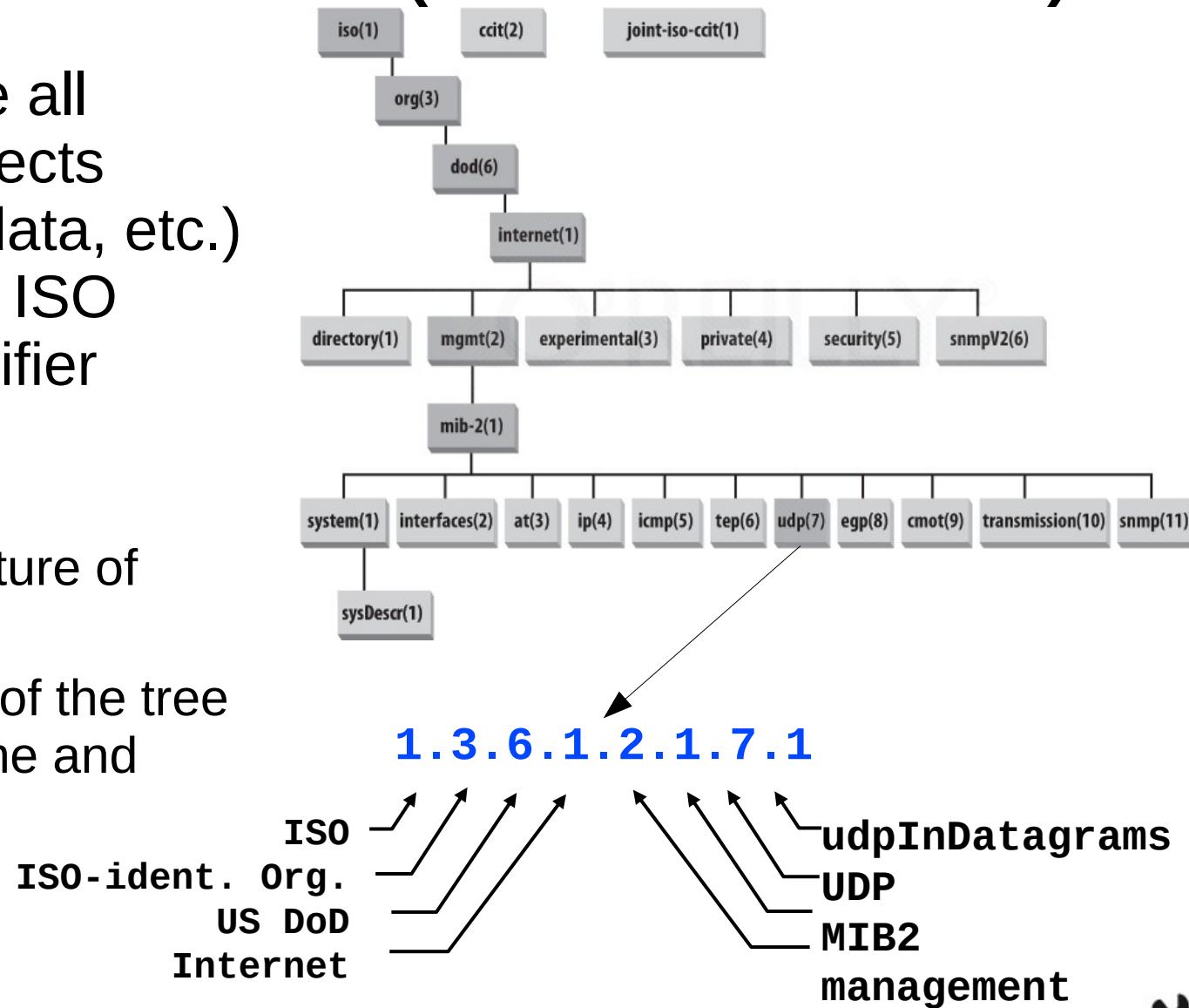
- An SNMP MIB module is a specification of management information on a device
- The SMI represents the MIB database structure in a tree form with conceptual tables, where each managed resource is represented by an object
- Object Identifiers (OIDs) uniquely identify or name MIB variables in the tree
 - Ordered sequence of nonnegative integers written left to right, containing at least two elements
 - For easier human interaction, string-valued names also identify the OIDs
 - MIB-II (object ID 1.3.6.1.2.1)
 - Cisco private MIB (object ID 1.3.6.1.4.1.9)
- The MIB tree is extensible with new standard MIB modules or by experimental and private branches
 - Vendors can define their own private branches to include instances of their own products



SNMP Names (numbers/OID)

- To nominate all possible objects (protocols, data, etc.) it is used an ISO Object Identifier (OID) tree:

- Hierarchic nomenclature of objects
- Each leaf of the tree has a name and number



SNMP MIBs

- Management Information Base (MIB): set of managed objects, used to define information from equipments, and created by the manufacturer
- Example: UDP module

<u>Object ID</u>	<u>Name</u>	<u>Type</u>	<u>Comments</u>
1.3.6.1.2.1.7.1	UDPInDatagrams	Counter32	Number of UDP datagrams delivered to users.
1.3.6.1.2.1.7.2	UDPNoPorts	Counter32	Number of received UDP datagrams for which there was no application at the destination port.
1.3.6.1.2.1.7.3	UDPIInErrors	Counter32	The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.
1.3.6.1.2.1.7.4	UDPOutDatagrams	Counter32	The total number of UDP datagrams sent from this entity.



Relevant MIBs

- Interface characteristics, configurations, status, ans stats:
 - ◆ IF-MIB and IP-MIB.
 - ◆ Cisco extra information: CISCO-QUEUE-MIB, CISCO-IF-EXTENSION-MIB
- Nodes management information (description, general information, CPU/memory status, etc...):
 - ◆ SNMPv2-SMI and ENTITY-MIB.
 - ◆ Vendor specific: CISCO-SMI, JUNIPER-SMI, etc...
 - ◆ Cisco extra: CISCO-PROCESS-MIB, CISCO-FLASH-MIB, CISCO-ENVMON-MIB, CISCO-IMAGE-MIB, etc...
- Node routing and traffic-engineering:
 - ◆ IP-MIB, IP-FORWARD-MIB
 - ◆ Cisco extra information: CISCO-CEF-MIB, CISCO-PIM-MIB
 - ◆ MPLS-TE-MIB, MPLS-LSR-MIB, MPLS-VPN-MIB
- Node services:
 - ◆ Vendor specific: CISCO-AAA-SESSION-MIB, CISCO-SIP-UA-MIB, etc...
- Node monitoring mechanisms:
 - ◆ RMON-MIB, RMON2-MIB, CISCO-SYSLOG-MIB, CISCO-RTTMON-MIB, CISCO-NETFLOW-MIB, CISCO-IPSEC-FLOW-MONITOR-MIB, etc...



NetFlow

- Cisco NetFlow services provide network administrators IP flow information from their data networks.
 - ◆ Network elements (routers and switches) gather flow data and export it to collectors.
 - ◆ Captures data from ingress (incoming) and/or egress (outgoing) packets.
 - ◆ Collects statistics for IP-to-IP and IP-to-MPLS packets.
- A flow is defined as a unidirectional sequence of packets with some common properties that pass through a network device.
 - ◆ A flow is identified as the combination of the following key fields:
 - Source IP address, Destination IP address, Source port number, Destination port number, Layer 3 protocol type, Type of service (ToS), and Input logical interface.
- These collected flows are exported to an external device, the NetFlow collector.
- Network flows are highly granular
 - ◆ For example, flow records include details such as IP addresses, packet and byte counts, timestamps, Type of Service (ToS), application ports, input and output interfaces, autonomous system numbers, etc.
- NetFlow has three major versions: v1, v5 and v9.
 - ◆ v1 is only recommended for legacy devices without support to v5 or v9.
 - ◆ V1 and v5, do not support IPv6 flows.

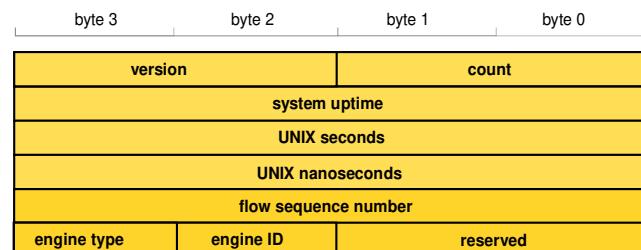
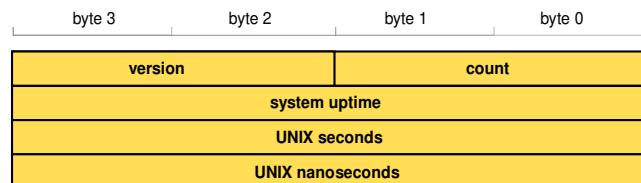


NetFlow versions 1 and 5

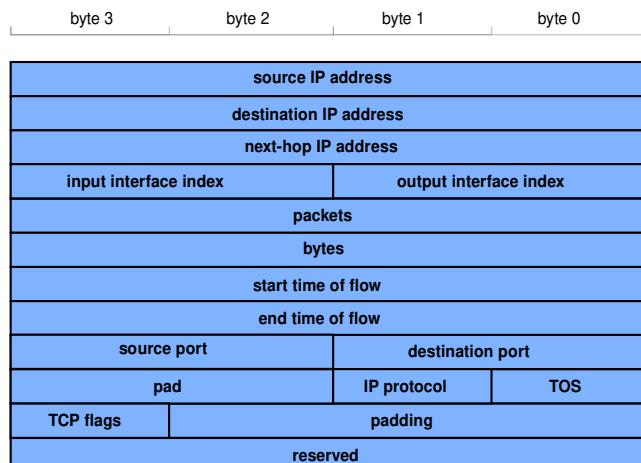
- NetFlow v1/v5 packets are UDP/IP packets with a NetFlow header and one or more NetFlow data Records



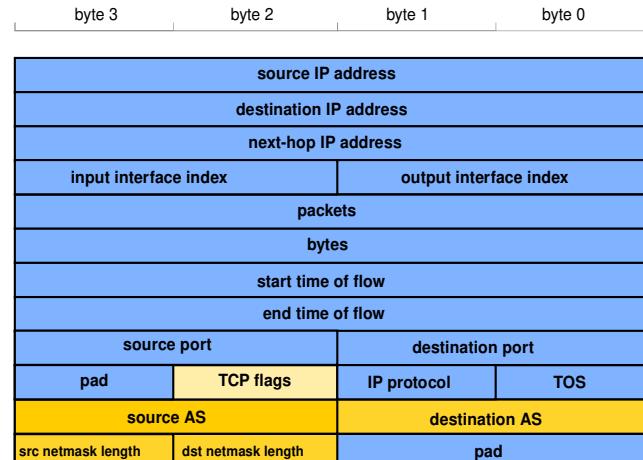
Header format



Record format



Version 1

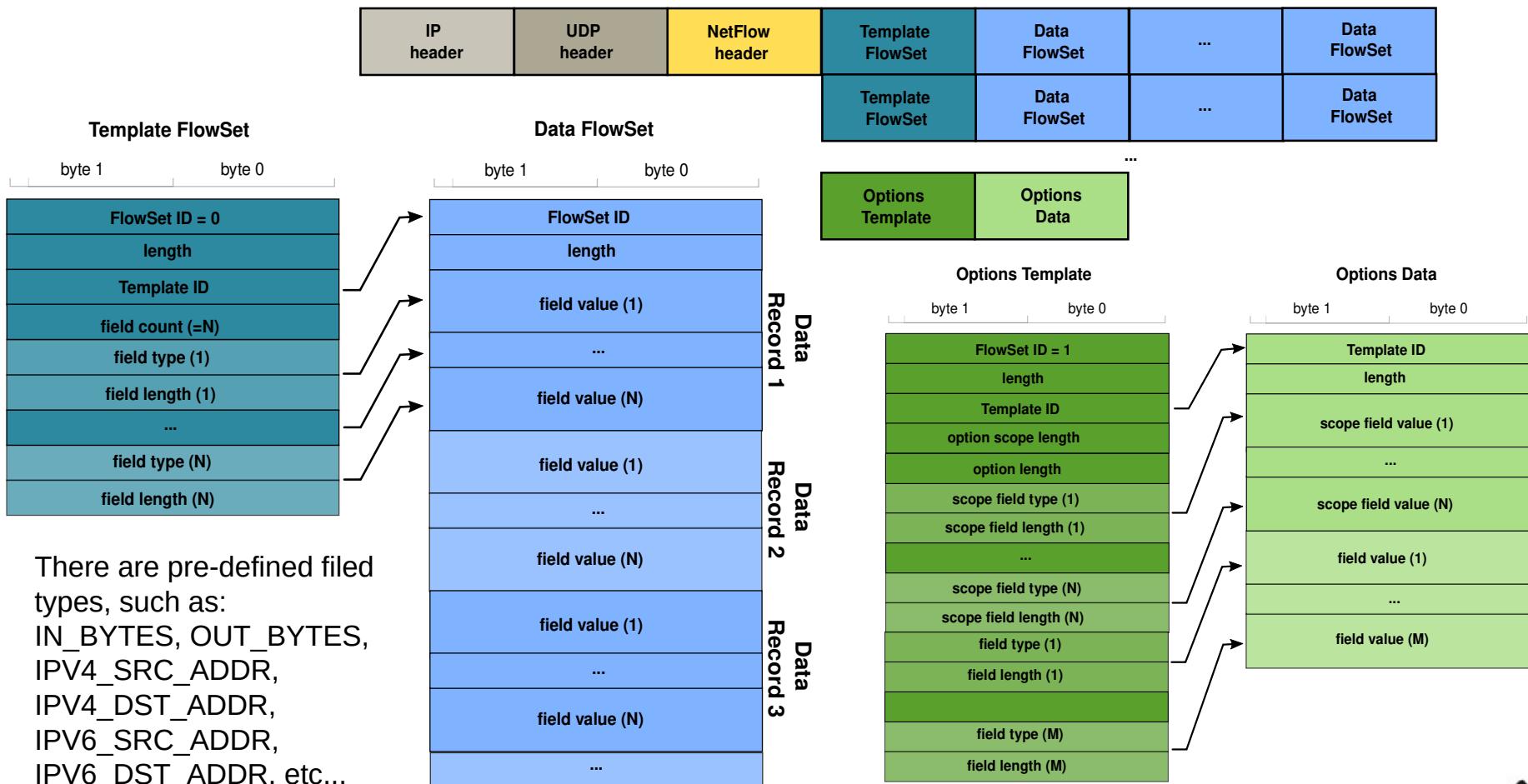


Version 5



NetFlow version 9

- NetFlow v9 packets are UDP/IP packets with a NetFlow header, one or more Template FlowSets (may be suppressed, if sent previously), one or more Data FlowSets, and, optionally, an Options Template and Data Record.



NetFlow Usage

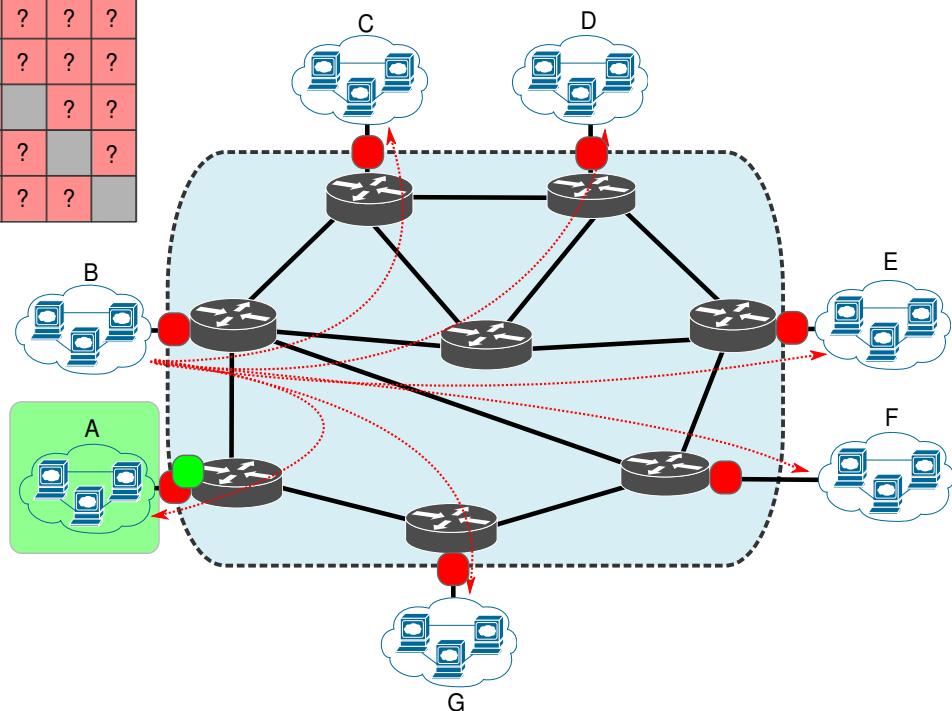
- Used to characterize users/services in terms of amount of traffic.
 - ◆ Users/Groups (overall or per-app) → Applied in (V)LAN interfaces.
 - ◆ Services → Applied to data-center interfaces
- Used to characterize traffic destinations (to egress points) from a specific ingress point in a network: traffic matrices.
 - ◆ Ingress/Egress points may be:
 - ✚ Network access links (distribution layer L3SW, Internet access routers, user VPN server links),
 - ✚ Network core border links (core border routers),
 - ✚ BGP peering links (AS Border routers).
- Used to characterize “in network” routing.
 - ◆ Complex to implement and process.



NetFlow Deployment

- Interfaces to monitor depend on objective:
 - ◆ Traffic matrix inference – all core border interfaces.
 - ◆ User/group flow generation inference - access interface from user/group.
- Egress vs. Ingress monitoring:
 - ◆ Traffic matrix inference – ingress OR egress.
 - ◆ User/group flow generation inference – both directions.

	A	B	C	D	E	F	G
A	?	?	?	?	?	?	?
B	?	?	?	?	?	?	?
C	?	?	?	?	?	?	?
D	?	?	?	?	?	?	?
E	?	?	?	?	?	?	?
F	?	?	?	?	?	?	?
G	?	?	?	?	?	?	?



IPFIX (v10) and Flexible NetFlow

- IPFIX is very similar to NetFlow v9
 - ◆ Uses version 10 in a similar header.
 - ◆ Also has Templates and Data Records.
 - ◆ Also has Options Templates and Options Data Records.
- IPFIX made provisions for NetFlow v9 and added support for it.
 - ◆ IPFIX lists an overview of the “Information Element identifiers” that are compatible with the “field types” used by NetFlow v9.
- IPFIX has more field types than the ones defined for NetFlow v9.
 - ◆ Also allows a vendor ID to be specified which a vendor can use to export proprietary/generic information.
- IPFIX allows for variable length fields.
 - ◆ Useful to export variable size strings (e.g., URLs).
- NetFlow v9 extension “Flexible NetFlow” aims to be equally flexible as IPFIX.



sFlow and jFlow

- sFlow
 - ◆ Uses sampling techniques designed for providing continuous site-wide (and enterprise-wide) traffic monitoring of high speed switched and routed networks.
 - ◆ Allow monitoring network traffic at Gigabit speeds and higher.
 - ◆ Allow to scale the monitoring of tens of thousands of agents from a single sFlow collector.
 - ◆ Supported by multiple vendors.
 - ➡ Including Cisco in
- jFlow is used in Juniper equipments.
 - ◆ Similar to NetFlow, however version 9 it also allows the usage of flow sampling techniques



Network Passive Probing

Packet Capturing

- User for:
 - ◆ Specific and detailed data inference,
 - ◆ Infer small and medium timescale dynamics.

- Probe types

- ◆ Switch mirror port,
- ◆ In-line,
- ◆ Network tap.

- Filtering/sampled by

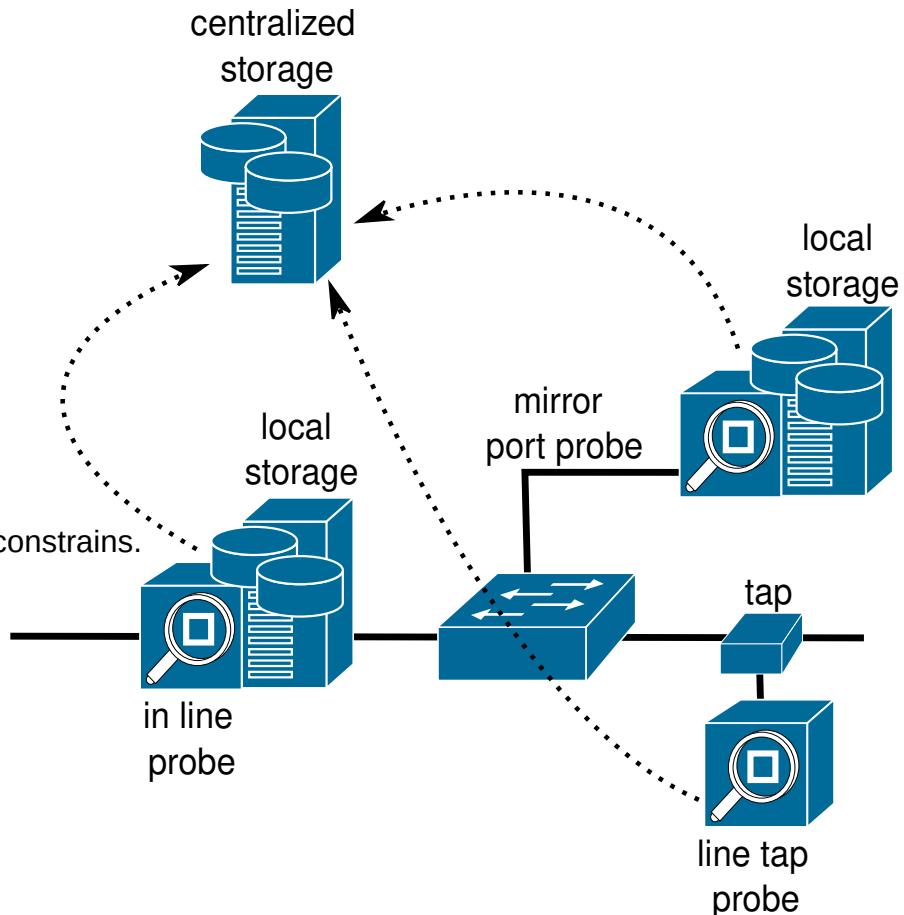
- ◆ User/terminal address/VLAN/access port,
- ◆ Group address/VLAN/access port,
- ◆ Protocols (UDP/TCP),
- ◆ Upper layer protocols,
 - ▶ Hard to identify due to encryption and legal/privacy constraints.
- ◆ UDP/TCP port number/range.

- Data processing

- ◆ Packet/byte count,
- ◆ Flow count,
- ◆ IP addresses and port distribution,
- ◆ App/service statistics and distribution.

- Local vs. Centralized storage and processing.

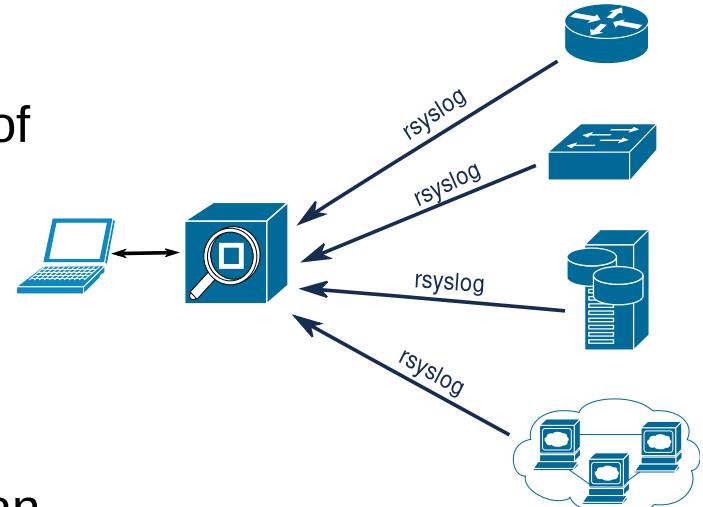
- ◆ Data upload to centralized point should not have impact on measurements.
- ◆ Local storage/processing requires probes with more resources.



Log Files Access

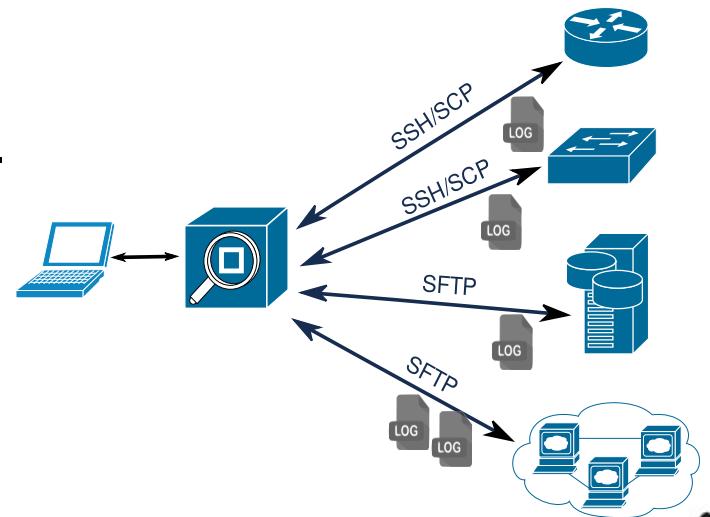
- **rsyslog**

- ◆ Able to accept inputs from a wide variety of services, transform them, and output the results to diverse network destinations.
 - ◆ Over TCP and/or SSL/TLS.
- ◆ Timing controlled by monitored node/device.
- ◆ Many post- and cross-processing tasks can be made on the monitored node/device.



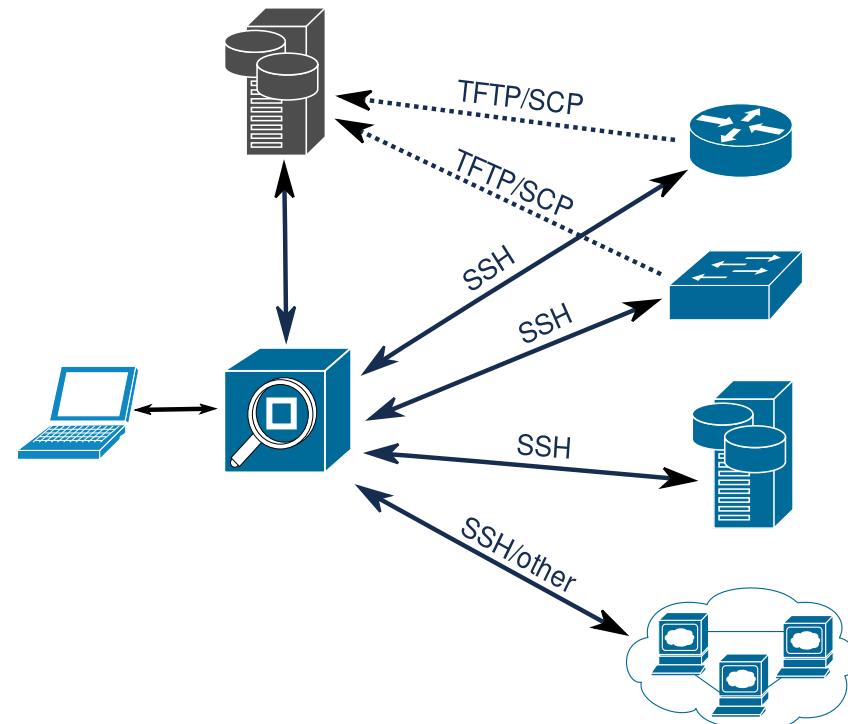
- Direct access to log files

- ◆ Using any remote access to remote files.
 - ◆ Requires special permissions.
- ◆ SSH/SCP, SFTP, etc...
- ◆ Timing controlled by central point.
- ◆ Requires all heavy post- and cross-processing in a central point.



Remote CLI Access

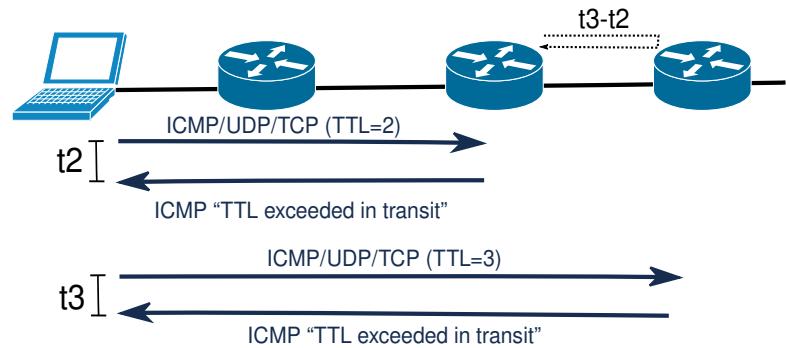
- Using a remote console to devices,
 - Using SSH, telnet (insecure), or proprietary protocols,
 - Retrieve configurations and device's processes status.
 - Devices can also upload configurations to a central point.
 - Using TFTP (insecure) or SFTP/SCP (many devices do not support it).
- Send “show” like CLI commands, retrieve output, parse information.



Active Measurements

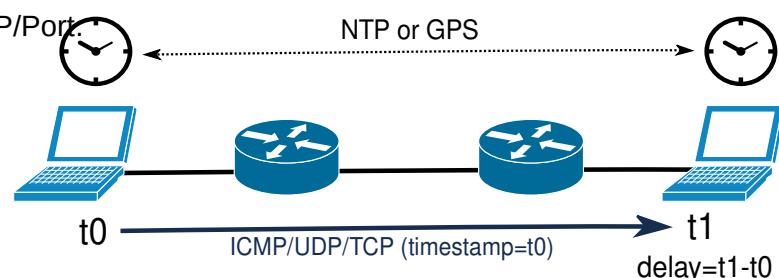
- Two-way delay/jitter

- ◆ End-to-end and middle hop.
- ◆ Requires the control of only one end.
- ◆ Ping and traceroute like solutions.
 - ◆ Requires that middle nodes respond to probes.
 - ICMP “TTL exceeded in transit” message.
 - ◆ ICMP, UDP or TCP.
 - UDP/TCP allows to test QoS (DiffServ) by IP/Port.



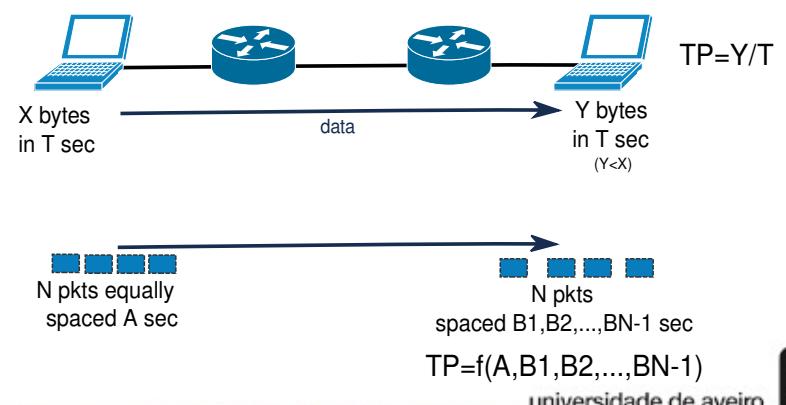
- One-way delay/jitter

- ◆ End-to-end.
- ◆ Requires control of both ends and clock synchronization.
 - ◆ May be complex/impossible for close nodes (low delay).



- End-to-end throughput

- ◆ Requires control of both ends.
- ◆ Directly sending/receiving large amounts of data.
- ◆ Indirectly using packet train techniques.
 - ◆ Prone to errors.



Open Source/Commercial Tools

- Cacti and Cricket
 - ◆ SNMP + RRDtool graphing
- Nagios
 - ◆ SNMP + HTTP + SSH + DB + other
 - ◆ Plugins
- Zenoss
- Zabbix
- Cisco Network Assistant
- Etc...



Data Processing

Aprendizagem Aplicada à Segurança

**Mestrado em Cibersegurança
DETI-UA**



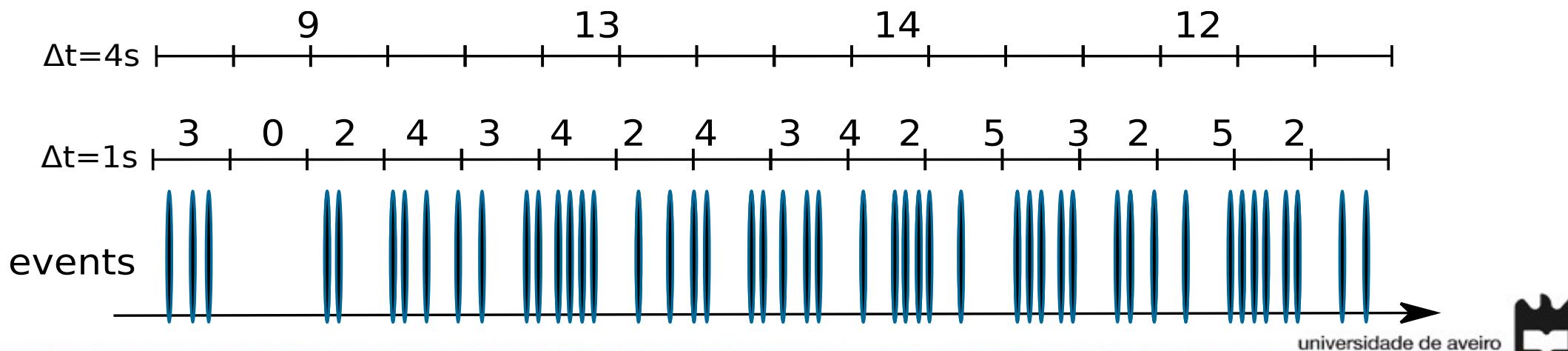
Qualitative Data

- Most monitored data is qualitative.
 - ◆ An event (with description) at a specific time (with a time-stamp).
 - ✚ 00:01:23.4566 – IP Packet [from A to B with 64 bytes]
 - ✚ 21:04:23.4566 – Error [id 404]
 - ✚ ...
- Must be converted to quantitative data.
- Some is pre-processed and it is already presented as quantitative.
 - ◆ Packets sent: 5467.
 - ◆ Bytes seen in the last 10 minutes: 18471947.
 - ◆ May require some additional processing.
 - ✚ Packets sent at 1s: 300pkts, Packets sent at 2s: 350pkts → Packets sent between 1s-2s: $350 - 300 = 50$ pkts.



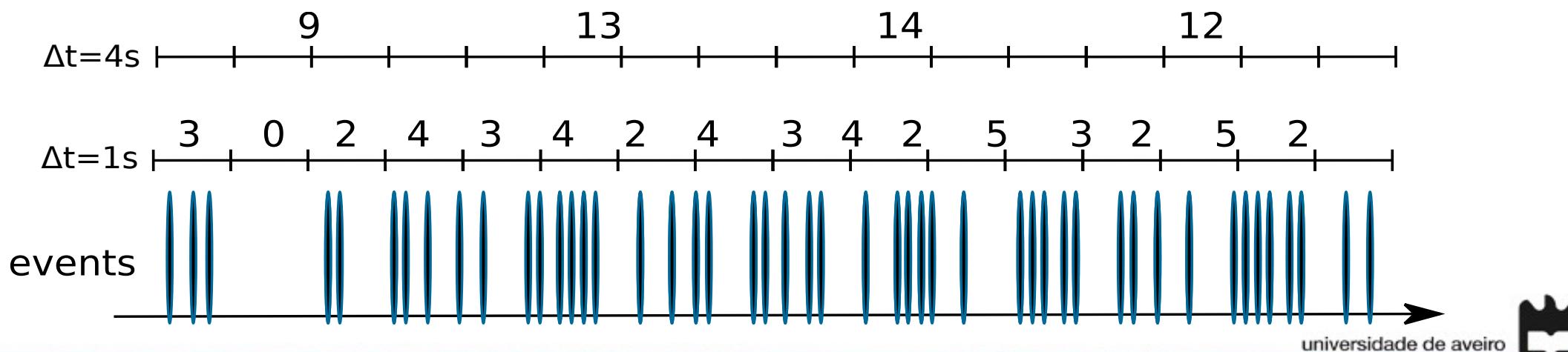
Qualitative → Quantitative Data (1)

- Events must be defined, identified and grouped:
 - All packets from IP 10.0.0.1,
 - All 400 errors accessing site X, etc...
- Sampling/Counting Interval
 - Time window in which the number of a specific event is counted, associated with a time index, and stored.
 - Minimum timescale.
- Events are counted in each sampling interval Δt .



Qualitative → Quantitative Data (2)

- Results in discrete time sequences for event:
 - ◆ For $\Delta t=1$: $X_k=\{3,0,2,4,3,4,2,4,3,4,2,5,3,2,5,2\}$
 - ◆ $X_0=3, X_1=0, \dots, X_{12}=2$
 - ◆ For $\Delta t=4$: $Y_k=\{9,13,14,12\}$
- Time sequences may be multi-dimensional:
 - ◆ Time sequences of n-tuples.
 - ◆ e.g., Number of packets, upload e download.
 - ◆ $Z_k=\{(3,9),(0,45),\dots(67,90)\}$



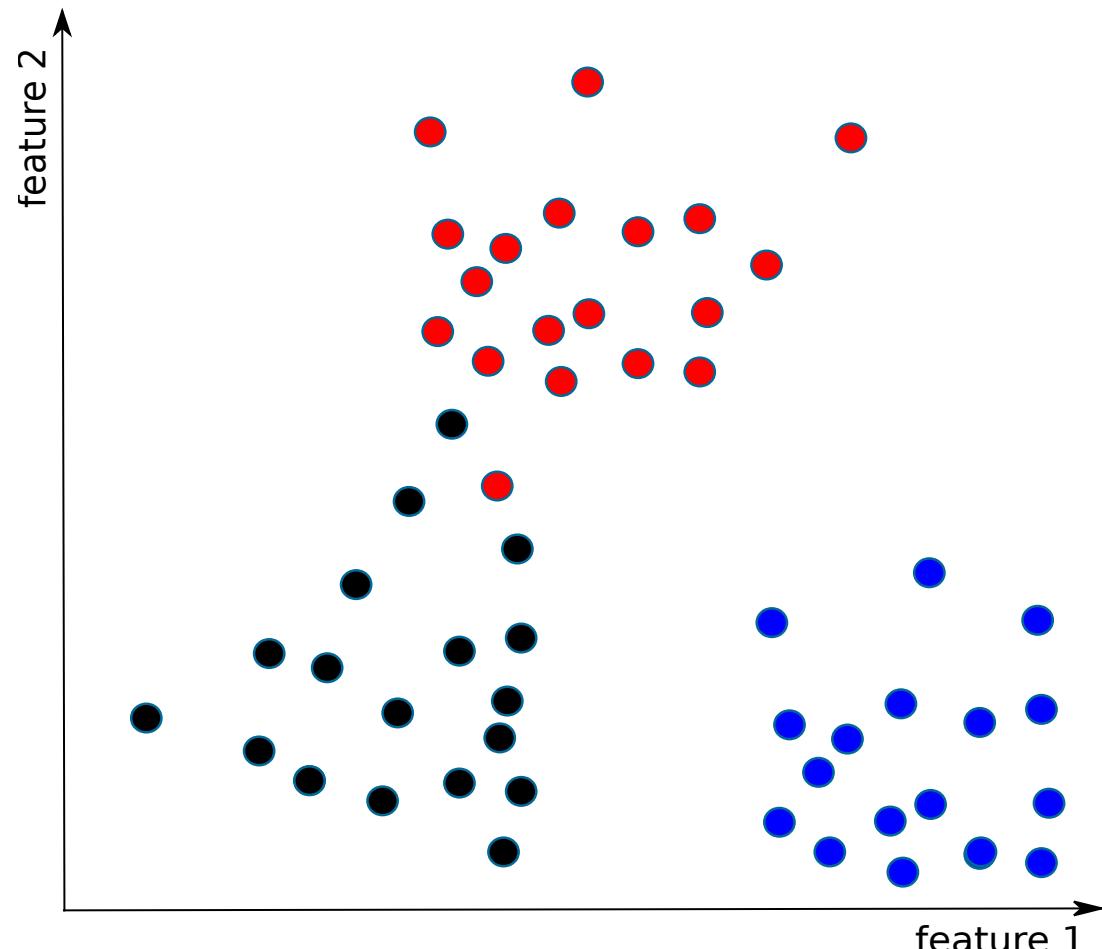
Time Windows and Entity Profile

- Sampling/Counting Window.
 - ◆ Provides time series of multiple metrics.
 - ◆ e.g., number of packets received by a terminal each second.
- Observation Window.
 - ◆ Features/Characteristics extraction Window.
 - ◆ Uses multiple Sampling/Counting Windows,
 - ◆ Statistics of respective time series.
 - ◆ Provides a n-tuple characterizing an entity behavior at a specif time.
 - ◆ e.g., 2-tuple with mean and variance of the number of packets received by a terminal in 30 seconds (30 counting 1s windows).
- Entity Profile
 - ◆ Pattern from multiple Observation Windows.
 - ◆ Provides a model to classify entities and detect anomalies.
 - ◆ May include time dynamics over time.



N-Dimensional Features Space

- A features' n-tuple defines a point in a N-Dimensional space that describes an entity behavior at a specific time.
- Allows to detect and define repetitive events and evolution over time.
- Allows to classify and discriminate behaviors.
- Allows to detect anomalies.



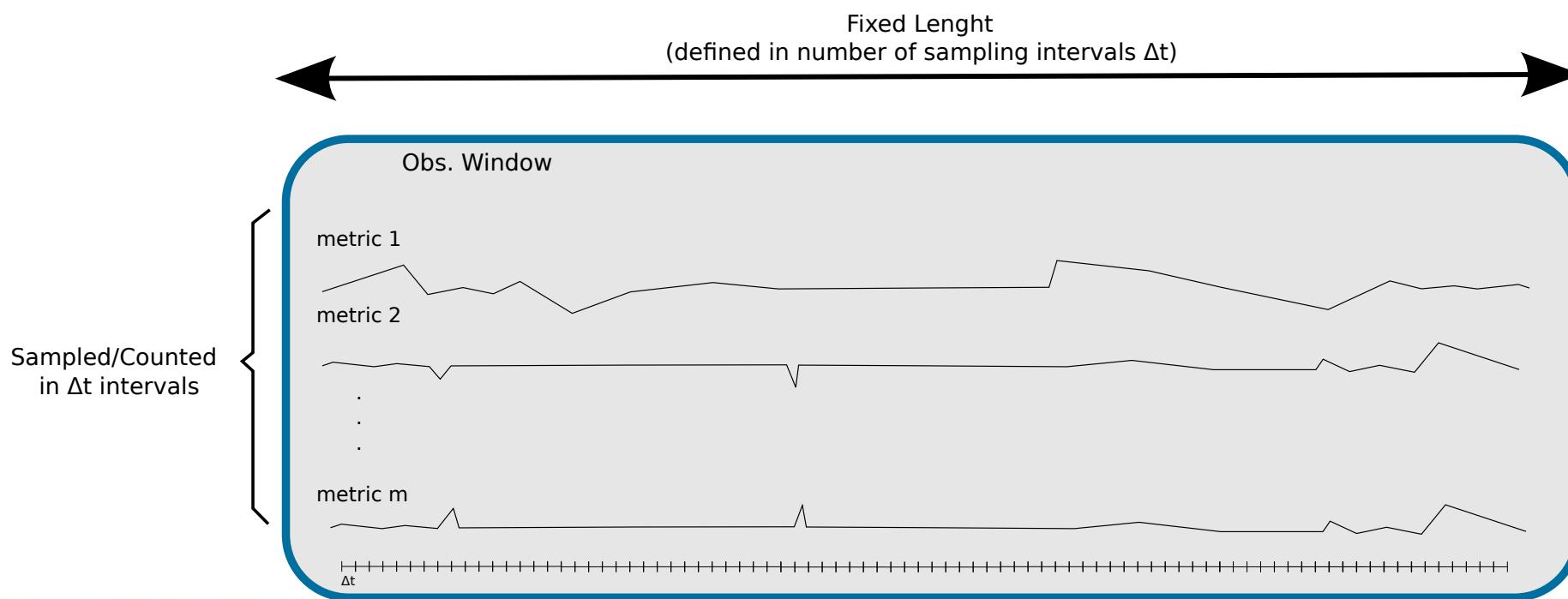
Data Formats

- The ideal data format is a n-tuple per time interval.
 - ◆ n metrics measured over time (n per observation).
$$(x_1, x_2, x_3, x_4, \dots, x_n)_k$$
 - ◆ Bi-dimensional data structure (time x metrics).
 - ◆ Optimal storing digital format:
 - Binary storage (array/matrix).
 - Sparse matrices could be advantageous.
 - Usage of fixed formats with integer indexes.
 - Avoid complex data structures with complex indexing of data, e.g.: python dictionaries.
 - Text formats are acceptable only in test scenarios.
 - Non-relational databases could also be an option.



Observation Window (1)

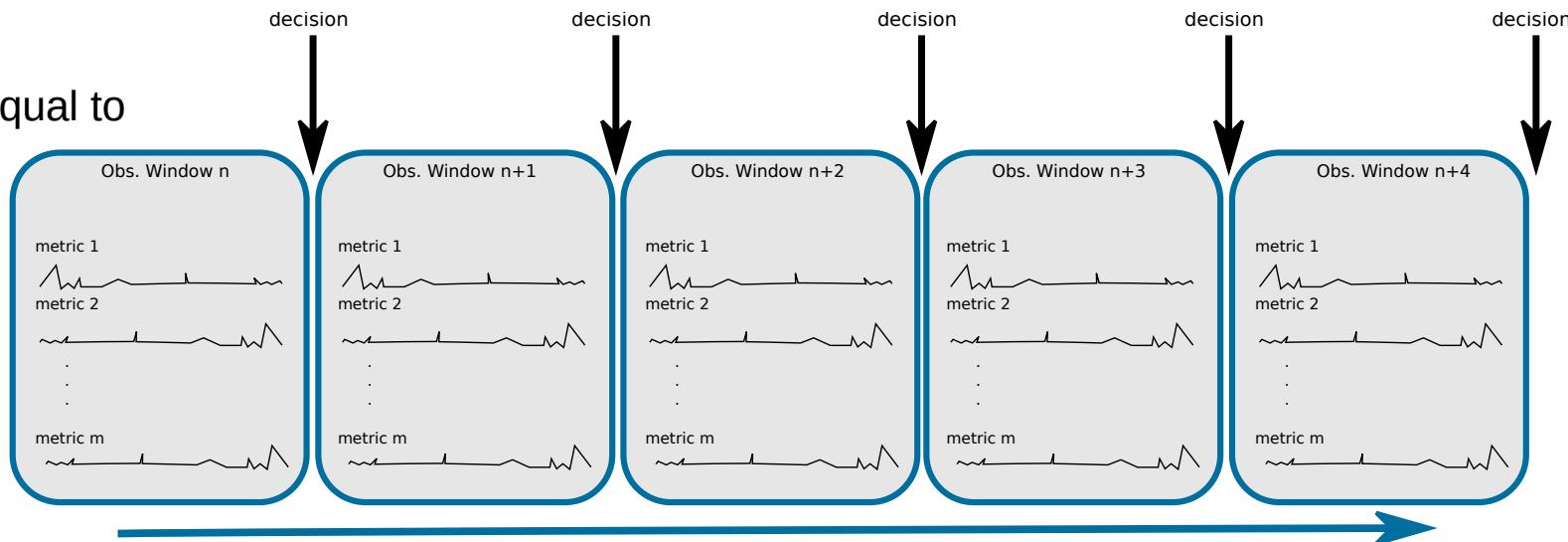
- An observation is constructed based on multiple sampling/counting metrics.
- Sampling/counting metrics should quantify activity events:
 - ◆ Start/End of activity.
 - ◆ Traffic Flows, Calls, Service usage, etc...
 - ◆ Amount of activity.
 - ◆ Traffic per sampling interval, activity duration, actions per sampling interval, etc...
 - ◆ Activity targets
 - ◆ IP addresses contacted, UCP/TCP ports used, services user IDs, points of access, etc...



Observation Window (2)

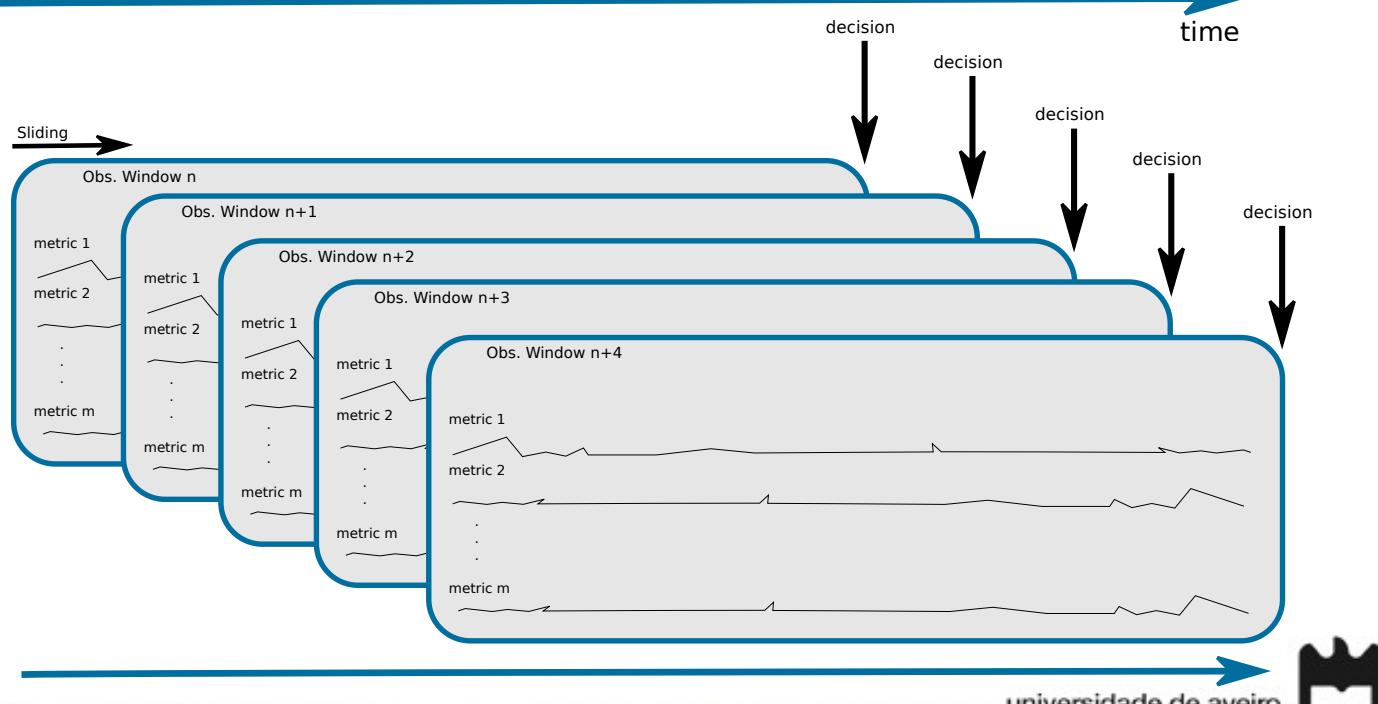
- Sequential

- ◆ Decision interval is equal to window size.



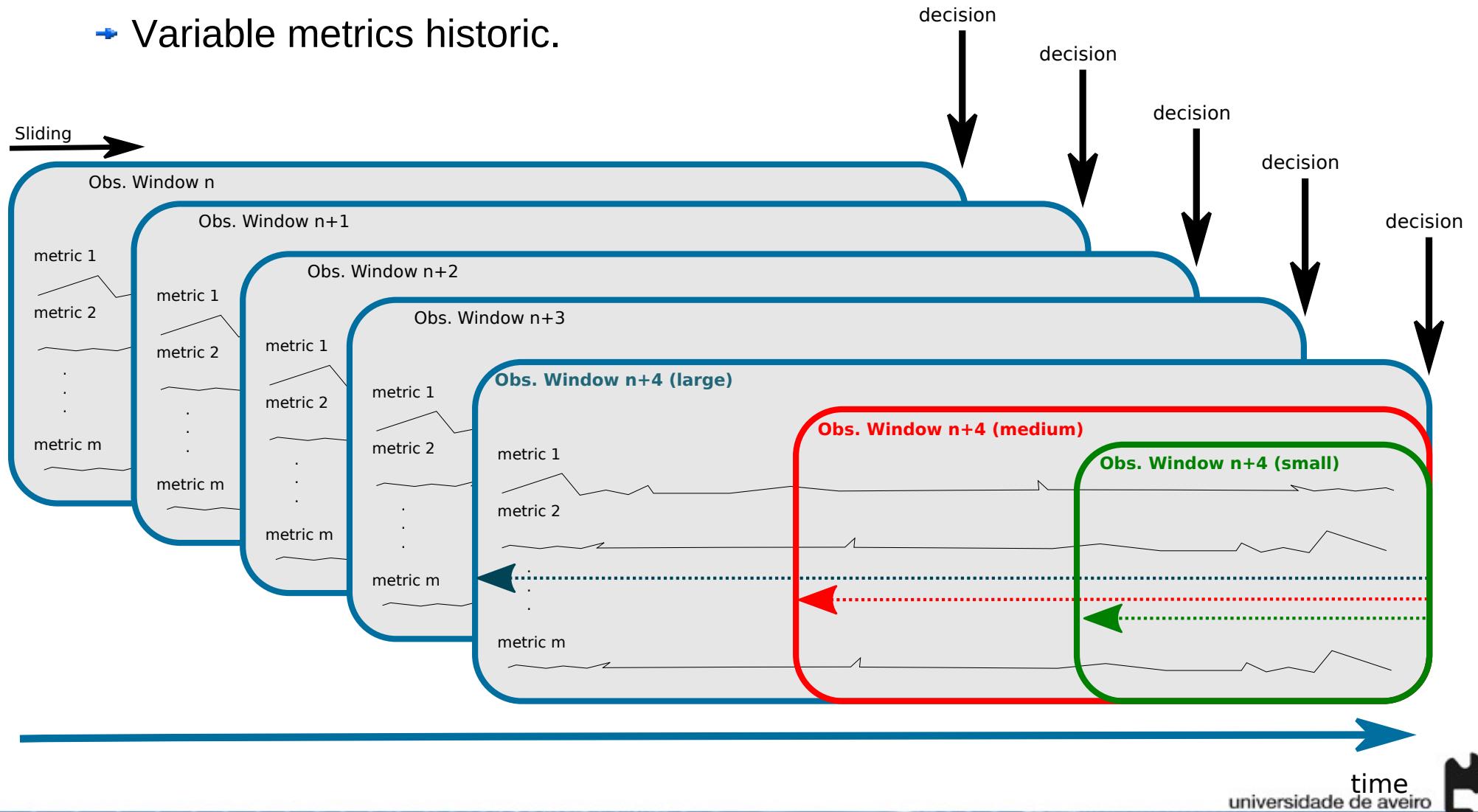
- Sliding

- ◆ Allows for longer periods of observation, while maintaining a short period of decision.



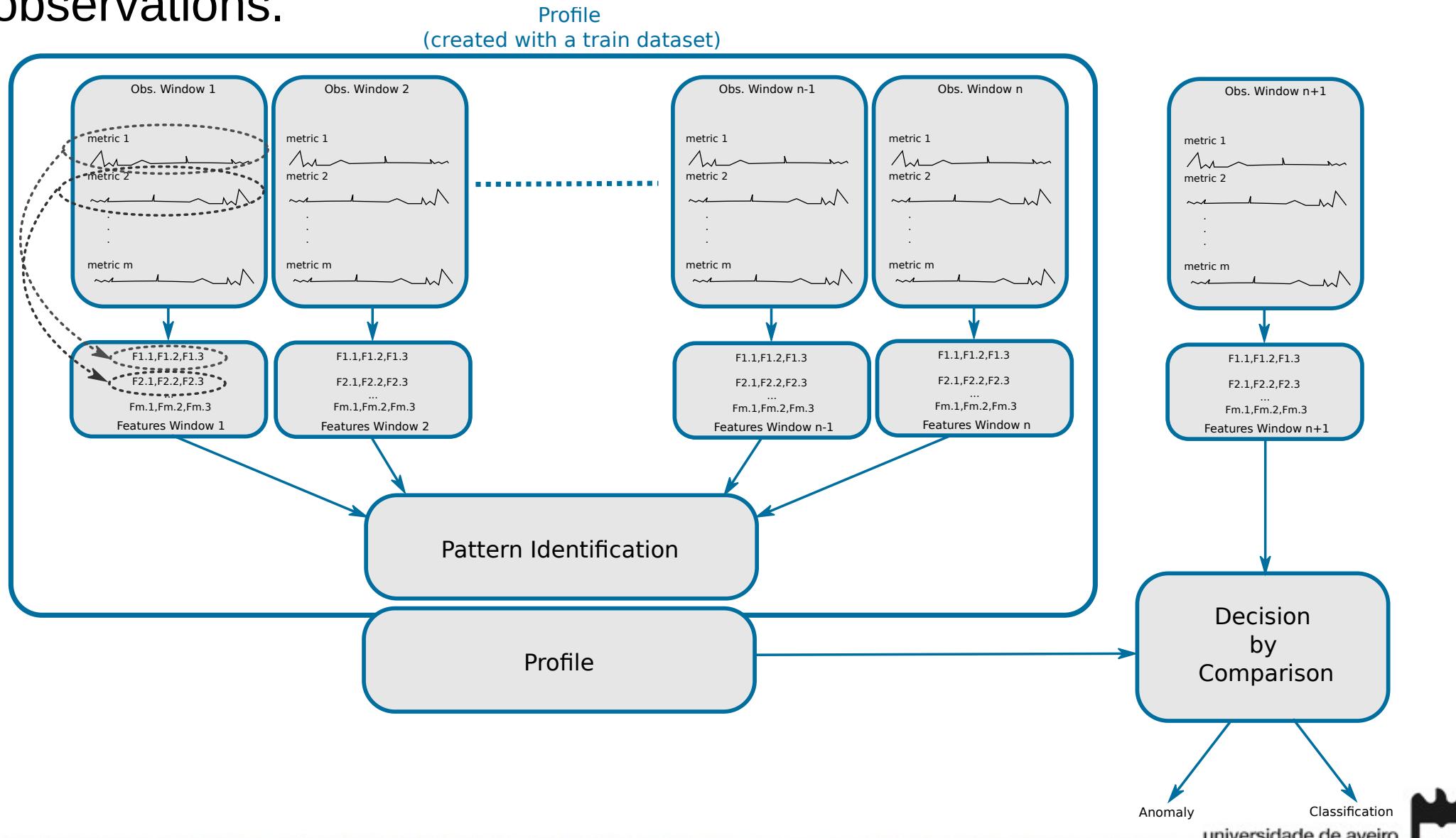
Multiple Observation Windows

- At each decision time point.
 - ◆ Construct observation windows with different lengths.
 - ◆ Variable metrics historic.



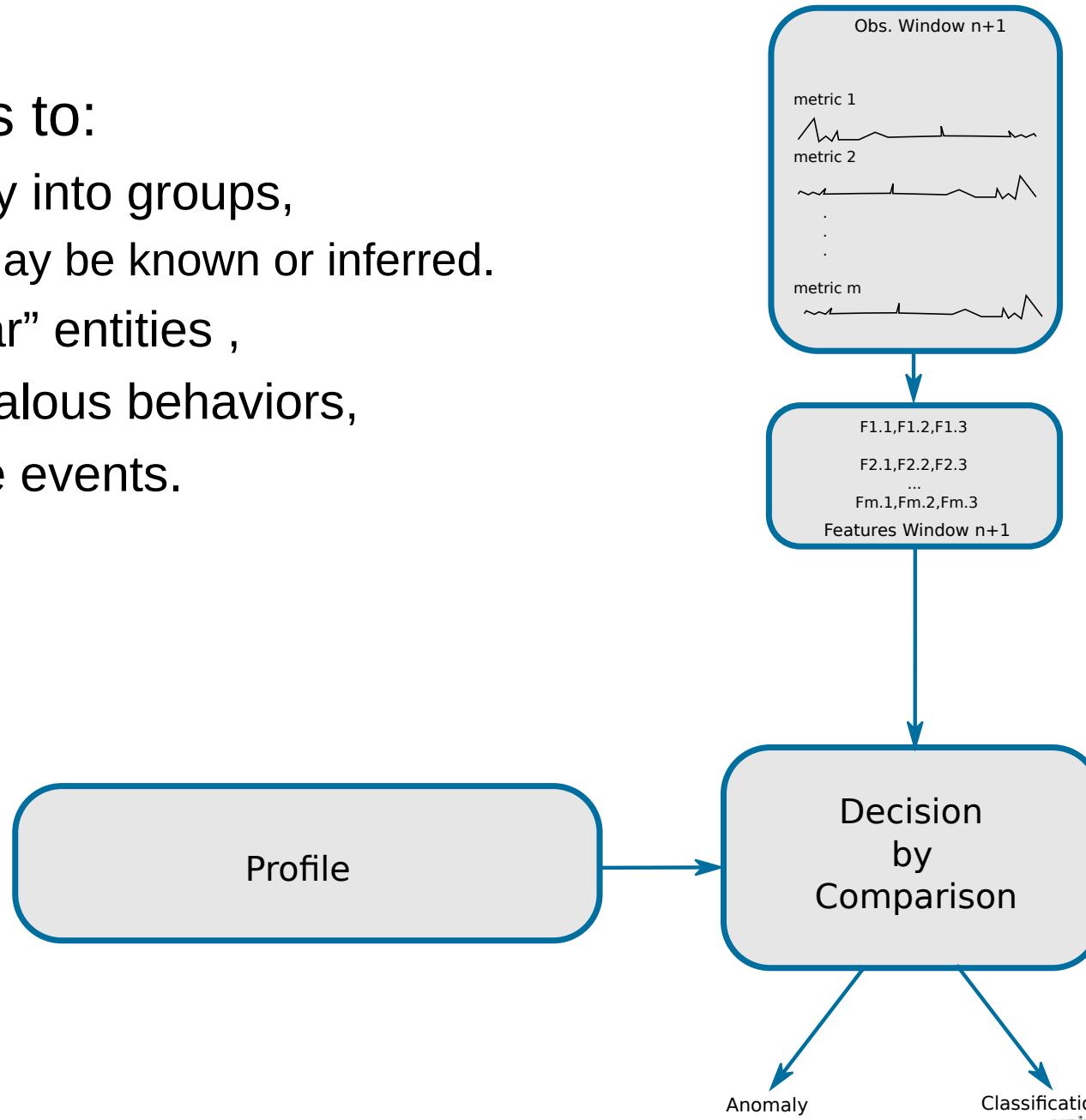
Entity Profiling

- Characterization of the observation windows after multiple observations.



Profile Comparison

- A profile allows to:
 - ◆ Classify entity into groups,
 - Groups may be known or inferred.
 - ◆ Group “similar” entities ,
 - ◆ Detect anomalous behaviors,
 - ◆ Predict future events.



Observation Features

- Time-independent descriptive statistics.
 - ◆ Mean, variance, quantiles, etc...
- Time-dependent descriptive statistics.
 - ◆ Time-relations between metrics over time
 - E.g., length of silences [number of sampling slots with metric equal to zero], length of activity [number of sampling slots with metric greater than zero], etc...
 - ◆ (Pseudo-)Periodicity components.
 - Time dependent.
 - Time multi-fractality (repetition of “similar events” in multiple time-scale).
 - Auto-correlation, FFT, CWT, DWT, and other spectral/frequency analysis.
- (Parameters of) Probabilistic functions/models.
 - ◆ Base function/model may be time independent or time dependent.



Descriptive Statistics (1)

- For a (equally) sampled-continuous time process:

$$X = \{x'_t = x_k, T_0 + k\Delta t \leq t < T_0 + (k+1)\Delta t, k = 1, 2, \dots, N\}$$

- Mean:** $\mu = \frac{1}{N} \sum_{i=1}^N x_i$

- Median:** $m_d = F^{-1}(0.5)$

- Variance:** $Var(X) = \sigma^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2$

- nth Central Moment:** $m_n = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^n$

- Quantiles/Percentiles** $Y = \{y_j\}_{1 \leq j \leq N} = \text{sorted}(\{x_k\}_{1 \leq k \leq N})$

- 64th percentile (64%)=0.64 quantile
- Quartiles: 25%, 50%, and 75%

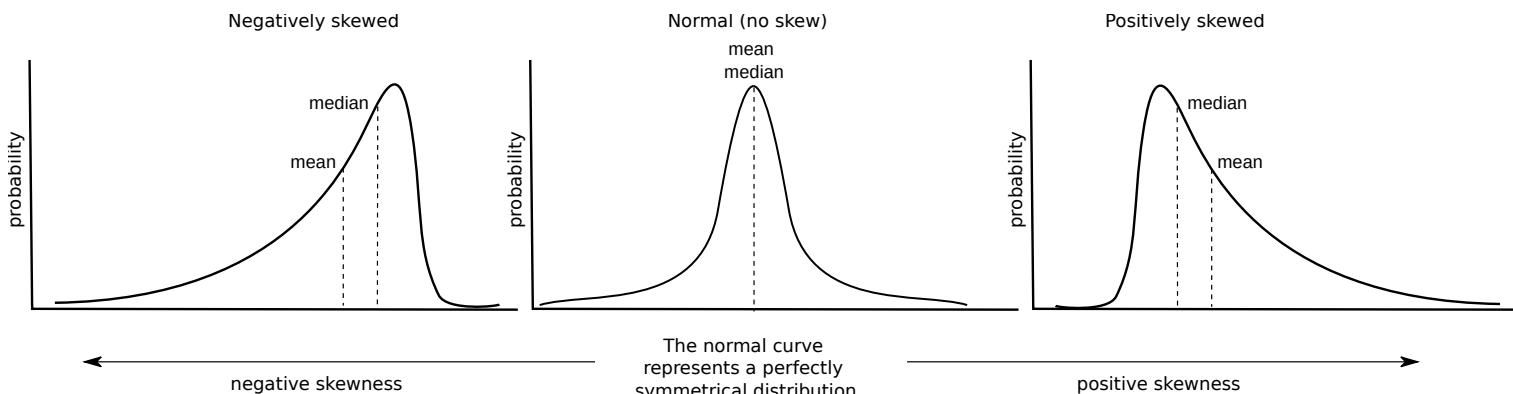
$$\pi_p = \min(y_{j \geq pN})$$



Descriptive Statistics (2)

- **Skewness:**

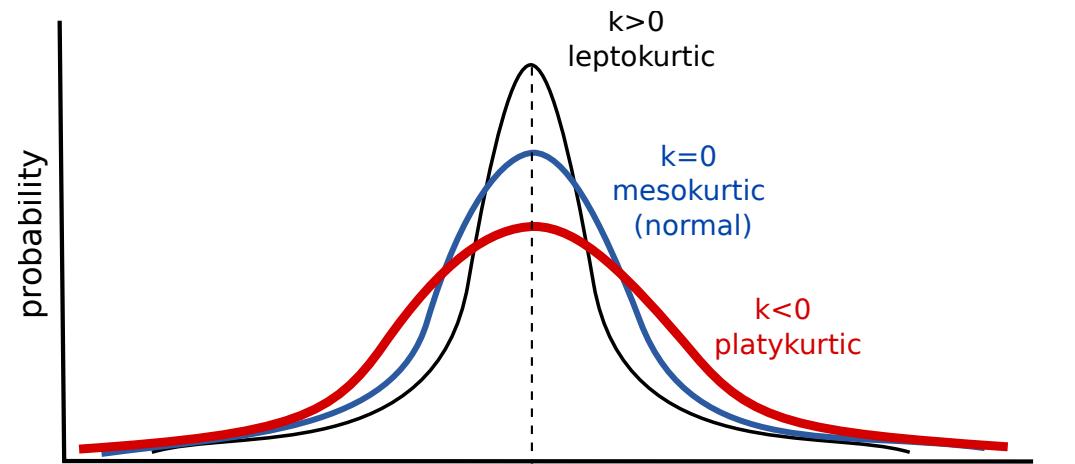
- ◆ Measure of the asymmetry of the probability distribution about its mean.



- **Excess Kurtosis:**

- ◆ Measure of the "tailedness" of the probability distribution.
- ◆ "-3" constant is used to normalize kurtosis to zero for a normal distribution.

$$b_1 = \frac{m_3}{\sigma^3} = \frac{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^3}{[\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2]^{3/2}}$$



$$k = \frac{m_4}{\sigma^4} = \frac{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^4}{[\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2]^2} - 3$$



Descriptive Statistics (3)

- Covariance

- ◆ Metric that quantifies how much two random variables have simultaneous variations:

$$\text{Cov}_{X,Y} = \frac{1}{N} \sum_{i=1}^N (x_i - \mu_X)(y_i - \mu_Y)$$

- Correlation coefficient

- ◆ Normalized covariance, varies between -1 and 1:

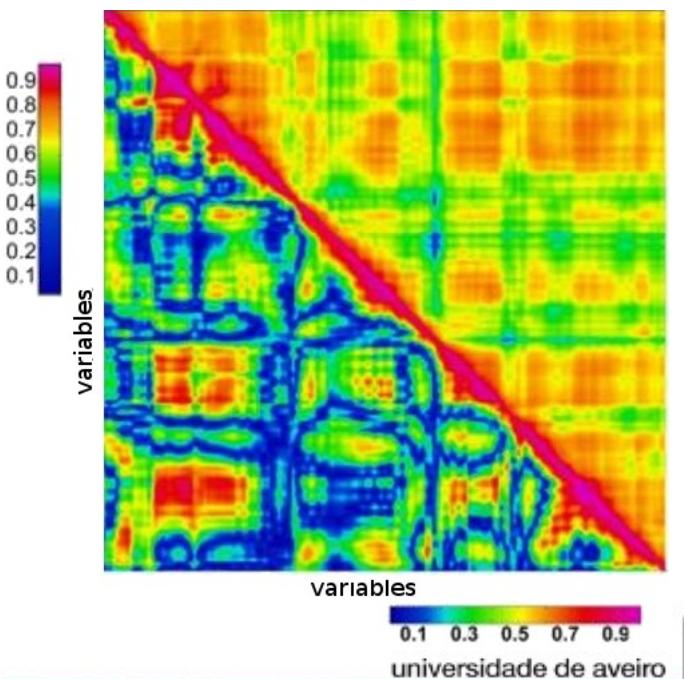
$$\rho_{X,Y} = \frac{\text{Cov}_{X,Y}}{\sigma_X \sigma_Y} \quad \sigma_X = \sqrt{\text{Var}(X)}$$

- Correlation matrix

- ◆ Defined by a (MxM) matrix, to quantify the correlation between M variables X_i :

$$C = \{c_{i,j}\}, i, j = 1, \dots, M$$

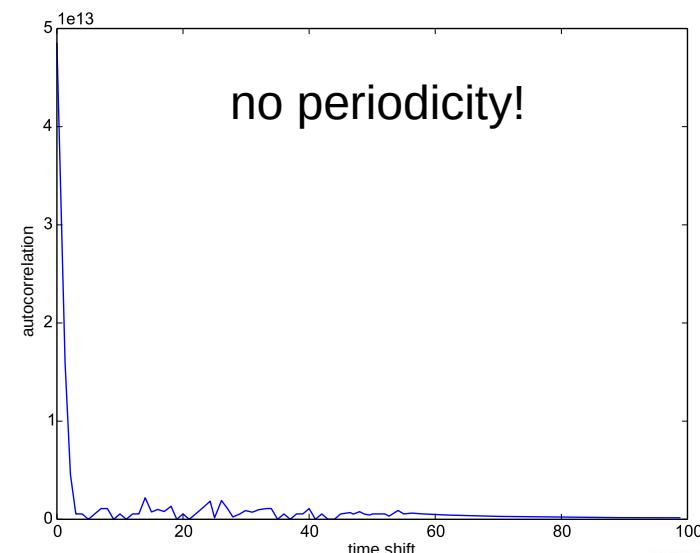
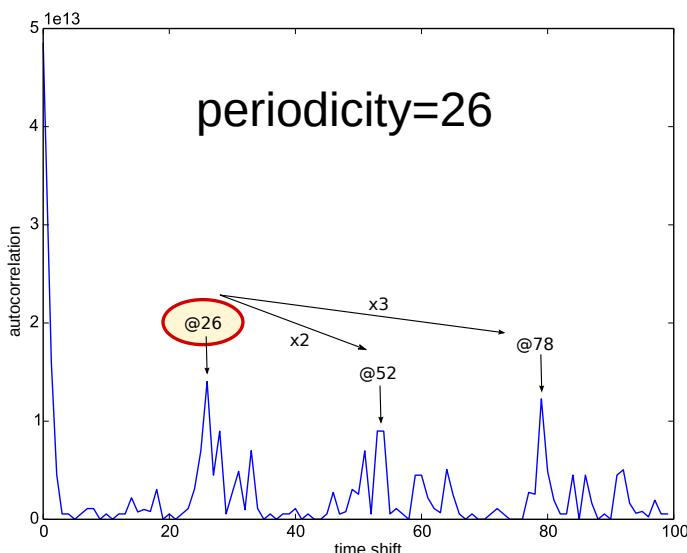
$$c_{i,j} = \rho_{X_i, X_j}$$



Periodicity Analysis (1)

Autocorrelation

- Autocorrelation
 - ◆ Correlation between the process and a shifted version (in time, by k samples) of the same process:
- Autocorrelation local maximums (peaks), reveal periodicity.
 - ◆ Differences between positions (k) of local maximums give periodicity.



Periodicity Analysis (2)

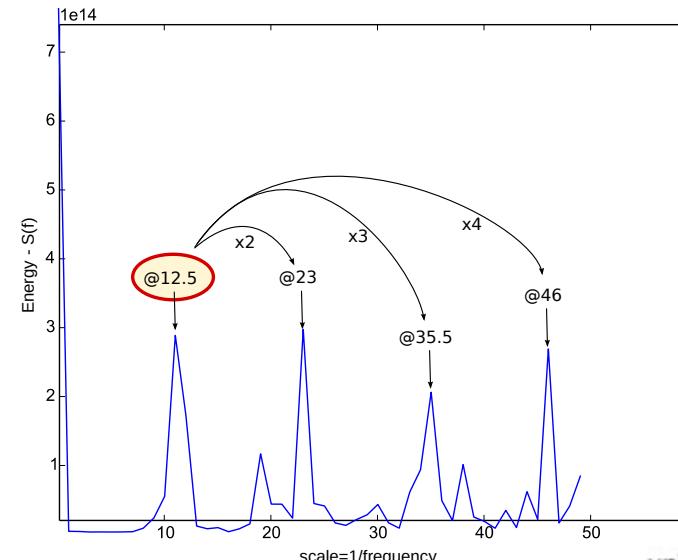
Periodograms

- Periodogram

- Frequency analysis → Spectral density estimation: Energy per frequency.
- Given by the modulus squared of the discrete Fourier transform.
 - For a signal x_i sampled every Δt :

$$S(f) = \frac{\Delta t}{N} \left| \sum_{n=1}^N x_n e^{-j2\pi n f} \right|^2, \quad -\frac{1}{2\Delta t} < t \leq \frac{1}{2\Delta t}$$

- The inverse of the frequencies with higher energy give the different periods (of periodicity).



Periodicity Analysis (3)

Scalograms

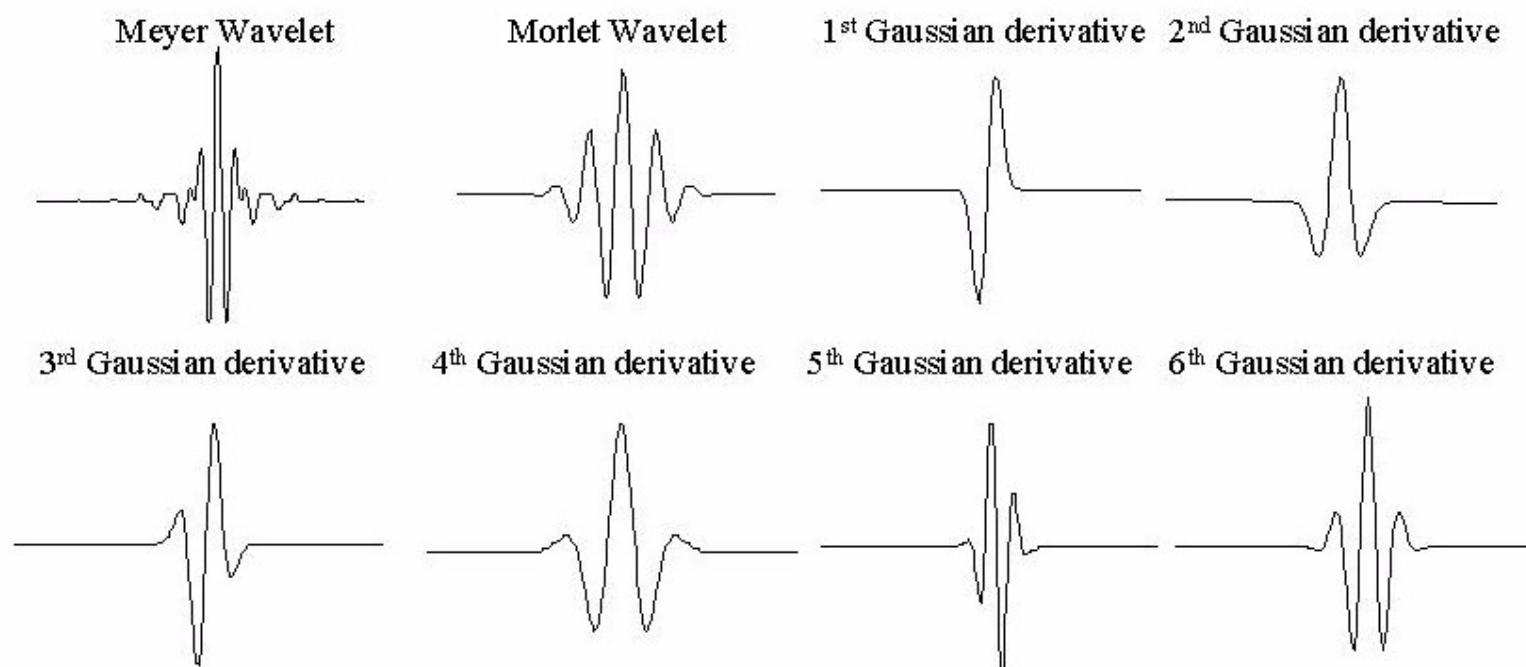
- Scalogram

- Joint Frequency/Time analysis → Wavelet Analysis
 - Energy per frequency/time.

$$\Psi_x^\psi(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{+\infty}^{-\infty} x(t)\psi^*\left(\frac{t-\tau}{s}\right)dt$$

Wavelet
functions

$$\psi^*(t)$$



Periodicity Analysis (4)

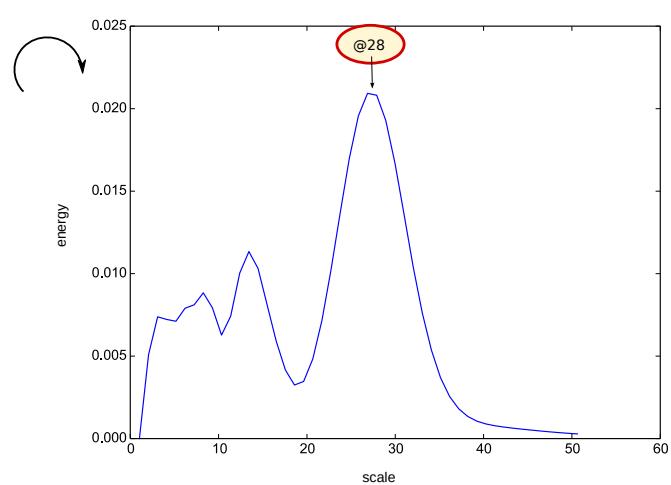
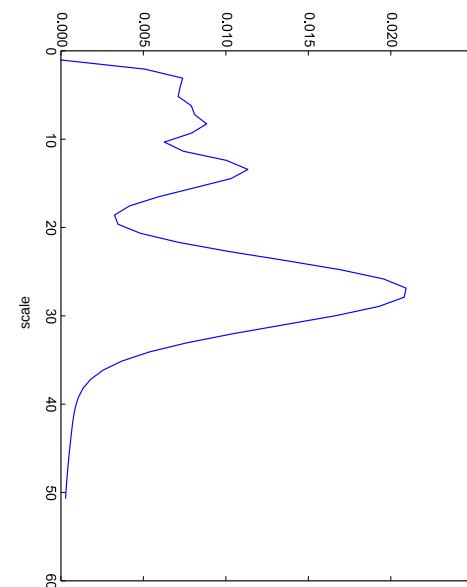
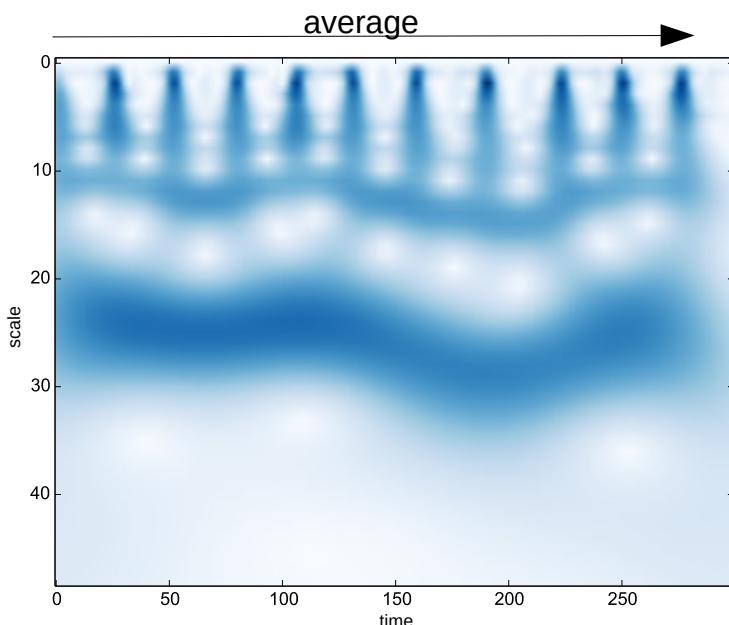
Scalograms

- Given by the normalized modulus squared of the Wavelet transform.

$$\hat{E}_x(\tau, s) = \frac{|\Psi_x^\psi(\tau, s)|^2}{\sum_{\tau' \in T} \sum_{s' \in S} |\Psi_x^\psi(\tau', s')|^2}$$

- Averaged over time.

$$\bar{e}_x(s) = \frac{1}{|T|} \sum_{\tau \in T} \hat{E}_x(\tau, s), \forall s \in S$$



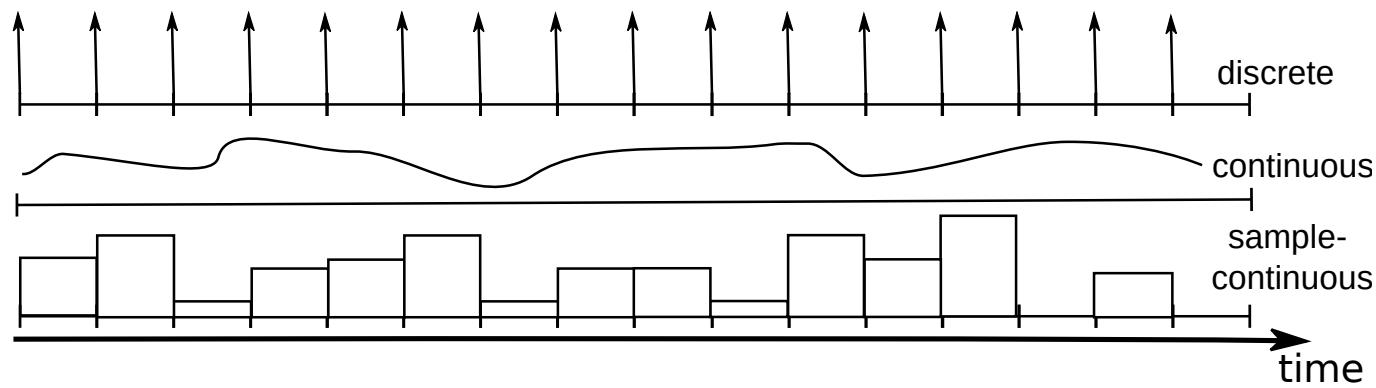
Stochastic Process

- A collection of variables indexed by a time variable, representing the evolution of some system over time.

$$X = \{x_t = a, t \in T\}$$

- ◆ Discrete variables: $a \in A, A = \{\alpha_1, \alpha_2, \dots, \alpha_S\}$
- ◆ Continuous variables: $a \in \mathbb{R}$
- ◆ Discrete time: $T = \{T_0 + k\Delta t, k \in \mathbb{N}_0\}$
- ◆ Continuous time: $T = \mathbb{R}_0$
- ◆ A continuous time process never exists in practice, what exists is a Sample-Continuous time process:

$$x_t = x'_{T_k}, t \in \mathbb{R}, T_k \leq t < T_{k+1}$$



Multivariate Stochastic Processes

- Variables belong to a multidimensional space of dimension N.

$$X = \{x_t = \vec{a}, t \in T\}$$

- ◆ Discrete variables:

$$\vec{a} \in A, A = \{\vec{\alpha}_1, \vec{\alpha}_2, \dots, \vec{\alpha}_S\}, \vec{\alpha}_i \in \mathbb{R}^N$$

- ◆ Continuous variables:

$$\vec{a} \in \mathbb{R}^N$$



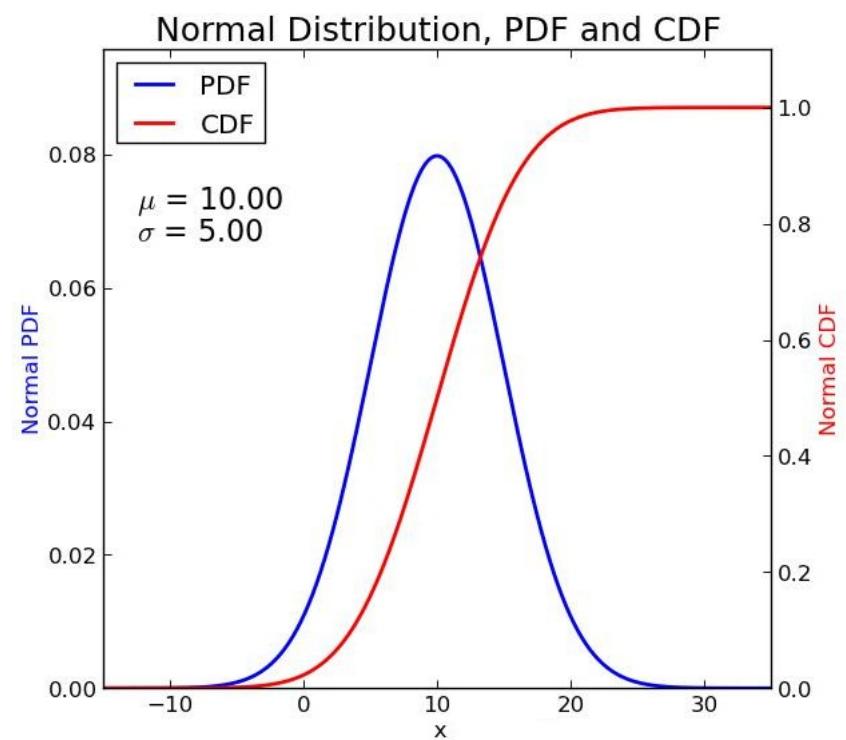
Probability Functions (1)

- Discrete

- Probability Mass Function (PMF)
- $\text{pmf}_X(a) = \Pr[X = a], a \in A, A = \{\alpha_1, \alpha_2, \dots, \alpha_N\}$
- $\sum_{\forall a \in A} \text{pmf}_X(a) = 1$

- Continuous

- Probability Density Function (PDF)
- $f_X(a) = \Pr[X = a], a \in \mathbb{R}$
- $\int_{-\infty}^{+\infty} f_X(x)dx = 1$
- Cumulative Density Function (CDF)
- $F_X(a) = \Pr[X \leq a] = \int_{-\infty}^a f_X(x)dx$



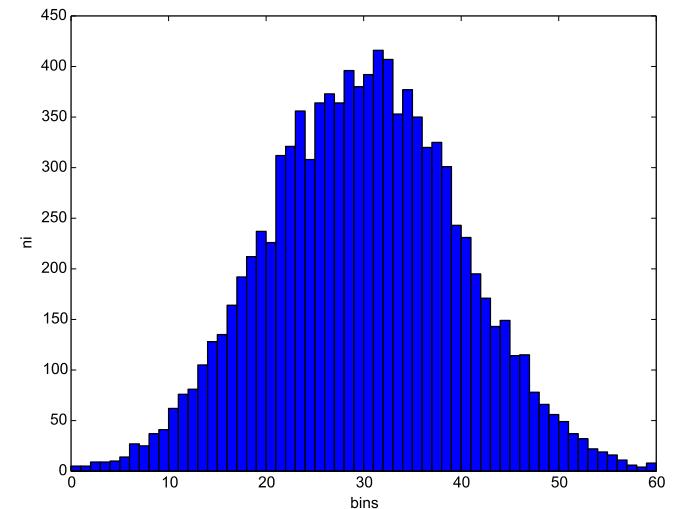
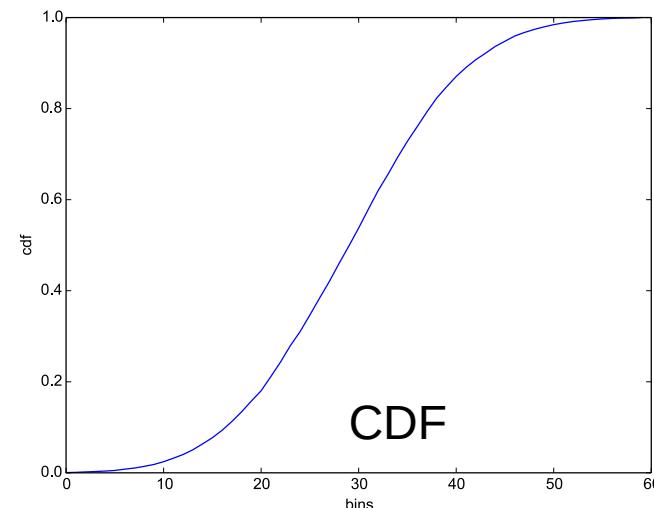
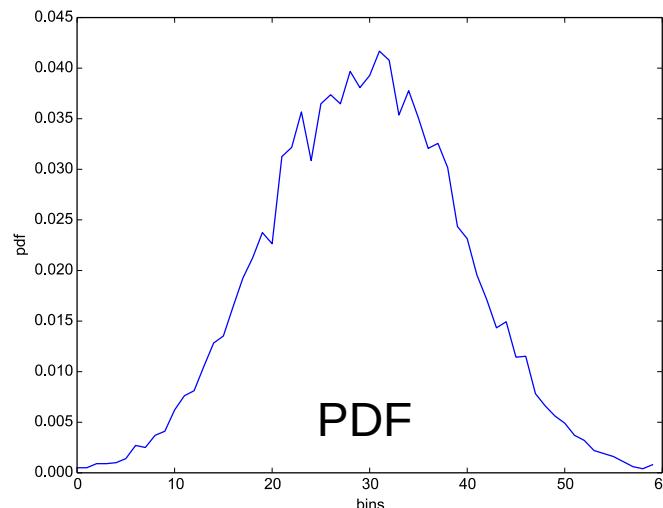
Probability Functions (2)

- Inference and interpretation
 - ◆ Histogram with bins $B = \{b_1, b_2, \dots, b_{M+1}\}$

$$n_i = \text{count}(b_i \leq X < b_{i+1}), i = 1, 2, \dots, M$$

$$f_X(a) = \frac{n_i}{N(b_{i+1} - b_i)}, \exists i, b_i \leq a < b_{i+1}$$

$$F_X(a) = \sum_{i=1}^j \frac{n_i}{N(b_{i+1} - b_i)}, \max_j : a < b_{j+1}$$



Statistical Univariate Distributions

- Most commonly used distributions:

- ◆ Discrete

- ◆ Uniform: $\text{pmf}_X(a) = \begin{cases} \frac{1}{N}, & a \in A \\ 0, & a \notin A \end{cases}, A = \{\alpha_1, \alpha_2, \dots, \alpha_N\}$

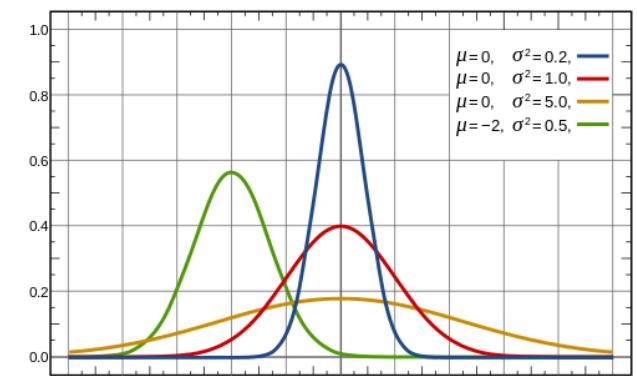
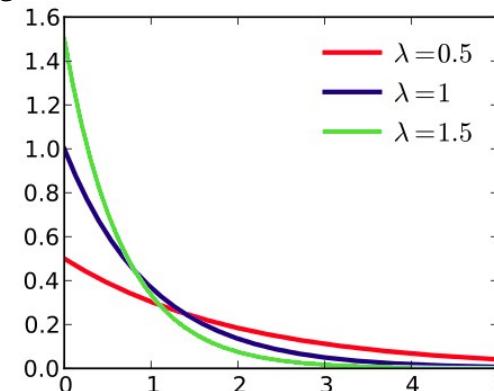
- ◆ Poisson: $\text{pmf}_X(a) = \frac{\lambda^a e^{-\lambda}}{a!}, \lambda > 0$

- ◆ Continuous

- ◆ Uniform: $f_X(a) = \begin{cases} \frac{1}{a_{\max}-a_{\min}}, & a \in [a_{\min}, a_{\max}] \\ 0, & \text{otherwise} \end{cases}$

- ◆ Normal/Gaussian: $f_X(a) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(a-\mu)^2}{2\sigma^2}}$

- ◆ Exponential: $f_X(a) = \lambda e^{-\lambda a}$



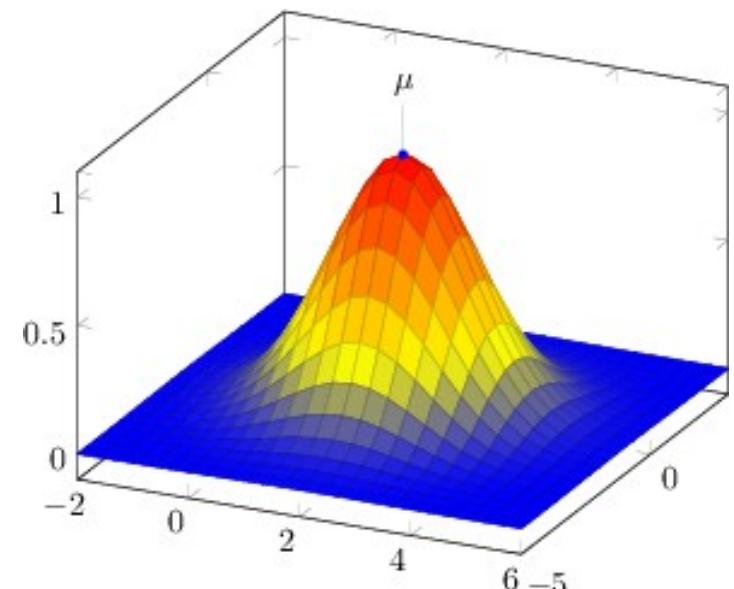
Multivariate Distributions

- Joint probability of a multidimensional variable.
- Incorporates correlation (ρ) between dimensions.
- E.g., 2-Dimensions Gaussian:



$$f_X((a_1, a_2)) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} e^{-\frac{z}{2(1-\rho^2)}}$$

$$z = \frac{(a_1 - \mu_1)^2}{\sigma_1^2} - \frac{2(a_1 - \mu_1)(a_2 - \mu_2)}{\sigma_1\sigma_2} + \frac{(a_2 - \mu_2)^2}{\sigma_2^2}$$



Aprendizagem Aplicada à Segurança

(Mestrado em Cibersegurança-DETI-UA)



**LECTURE 1 ML MODULE
Supervised learning –
Linear Regression**

Petia Georgieva
(petia@ua.pt)

DETI/IEETA – UA

LINEAR REGRESSION - Outline

1. Univariate linear regression

- Cost (loss) function - Mean Squared Error (MSE)
- Cost function convergence
- Gradient descent algorithm

2. Multivariate linear regression

- Overfitting problem

3. Regularization => way to deal with overfitting

Supervised Learning - CLASSIFICATION vs REGRESSION

Classification- the label is an integer number.
(e.g. 0, 1 for binary classification)

Regression - the label is a real number.

Examples of regression problems:

- Weather forecast
- Predicting wind velocity from temperature, humidity, air pressure
- Time series prediction of stock market indices
- Predicting sales amounts of new product based on advertising expenditure

Standard Notations in this module

x – input vector of features, attributes

y – output vector of labels, ground truth, target

m - number of training examples

n – number of features

$h_\theta(x)$ - model (hypothesis)

θ - vector of model parameters

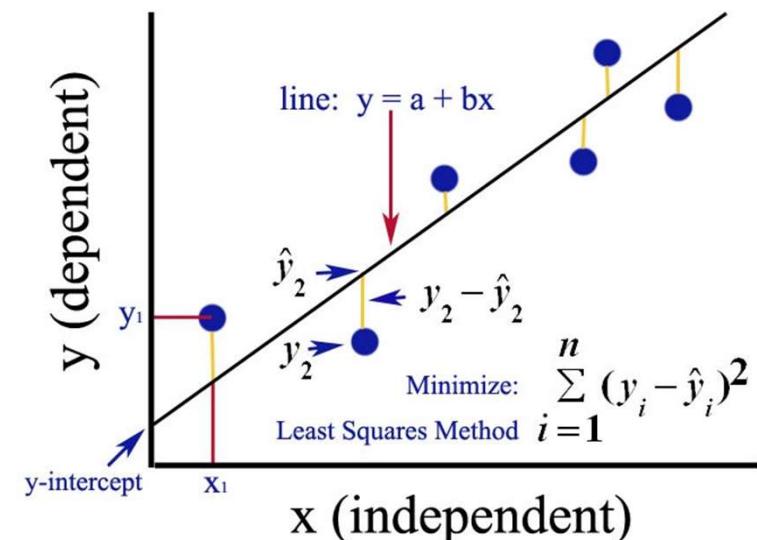
Training set: data matrix X (m rows, n columns)

	feature x_1	feature x_2	feature x_n	output(label) y
Example 1	$x_1^{(1)}$			$x_n^{(1)}$	$y^{(1)}$
Example 2	$x_1^{(2)}$			$x_n^{(2)}$	$y^{(2)}$
...					
Example i	$x_1^{(i)}$			$x_n^{(i)}$	$y^{(i)}$
...					
...					
Example m	$x_1^{(m)}$			$x_n^{(m)}$	$y^{(m)}$

Supervised Learning – univariate regression

Problem: Learning to predict the housing prices (output) as a function of the living area (input/feature)

Living area (feet ²)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
:	:



Supervised Learning – univariate linear regression

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 = [1 \quad x_1] \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \vec{x}^T \vec{\theta}$$

=> in Python => np.dot(X, Theta)

$$\vec{x} = \begin{bmatrix} x_0 = 1 \\ x_1 \end{bmatrix}$$

	x_0 (extra column)	feature x_1 (living area)	output(label) y (price)
Example 1=>	1	$x^{(1)}$	$y^{(1)}$
Example 2=>	1	$x^{(2)}$	$y^{(2)}$
	1		
Example m=>	1	$x^{(m)}$	$y^{(m)}$

Mean Square Error (MSE)

Linear Model (hypothesis) =>

$$h_{\theta}(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

Cost (loss) function
(Mean Square Error)

=>

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

m – number of training examples

Goal =>

$$\min_{\theta} J(\theta)$$

Gradient descent algorithm =>
iterative algorithm; at each
iteration all parameters (theta)
are updated simultaneously

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

alpha – learning rate > 0

Linear Regression (computing the gradient)

Cost function =>

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost function gradients =>
vector with parcial derivatives of J with respect to each parameter for one example ($m=1$)

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_\theta(x) - y) x_j\end{aligned}$$

Cost function gradients =>
for m examples

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Linear Regression – iterative gradient descent algorithm (summary)

Initialize model parameters (e.g. $\theta = 0$)

Repeat until J converge {

Compute Linear Regression Model =>

Compute cost function =>

$$h_{\theta}(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

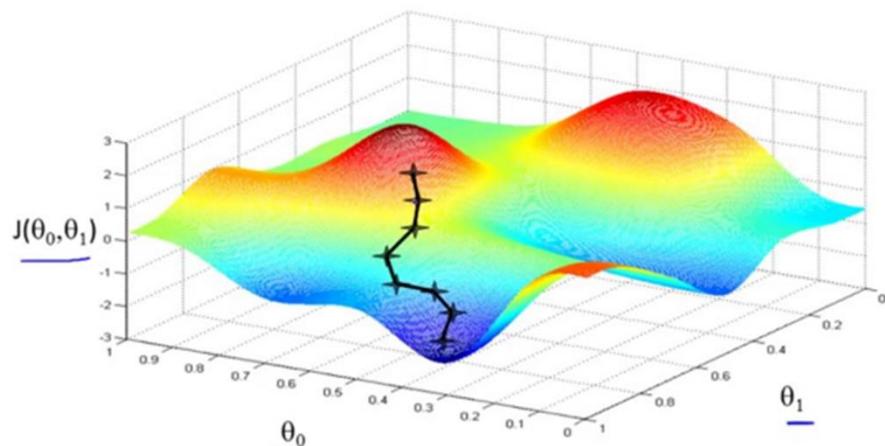
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Compute cost function gradients =>

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Update parameters =>
}

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$



Batch/mini batch/stochastic gradient descent for parameter update

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Batch learning (classical approach):

update parameters after all training examples have been processed, repeat several iterations until convergence

Mini batch learning (if big training data):

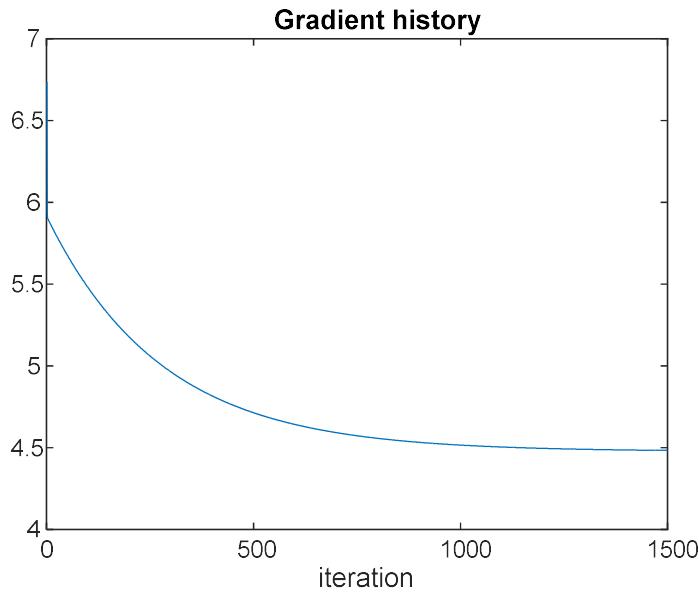
divide training data into small batches update parameters after each mini batch has been processed, repeat until convergence

Stochastic (incremental) learning (large-scale ML problems):
update parameters after every single training example has been processed.

Stochastic Gradient Descent (SGD): the true gradient is approximated by a gradient at a single example. Adaptive learning rate.

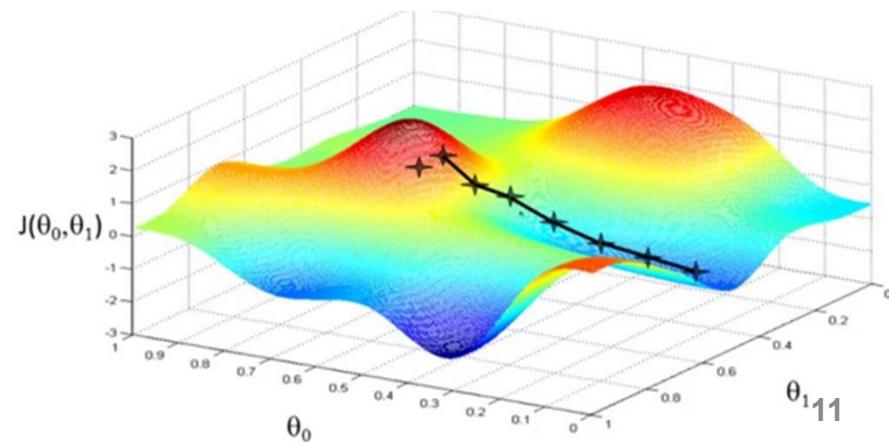
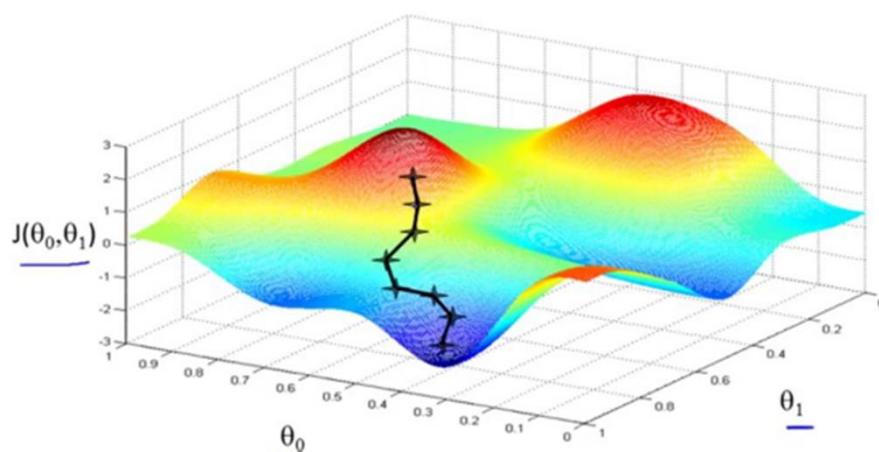
Cost function convergence

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$



Linear Regression (LR):

starting from different initial values of the parameters the cost function J should always converge (maybe to a local minimum !!!) if LR works properly.

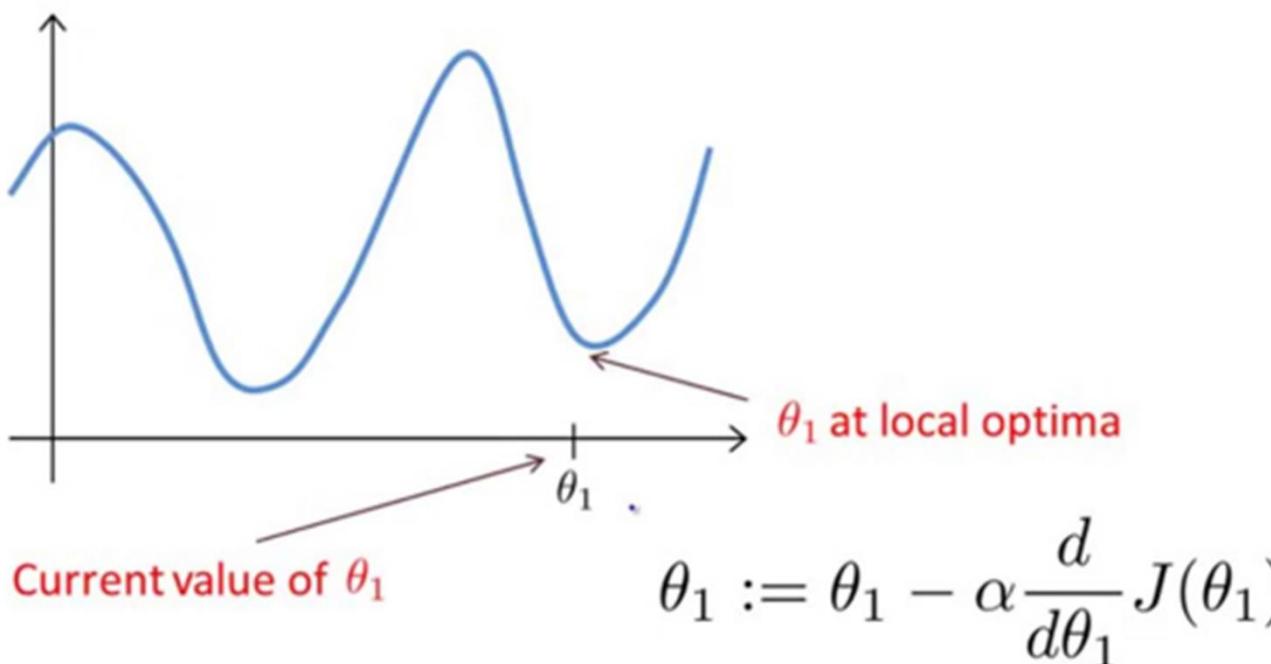


Lin Reg Cost function – local minimum

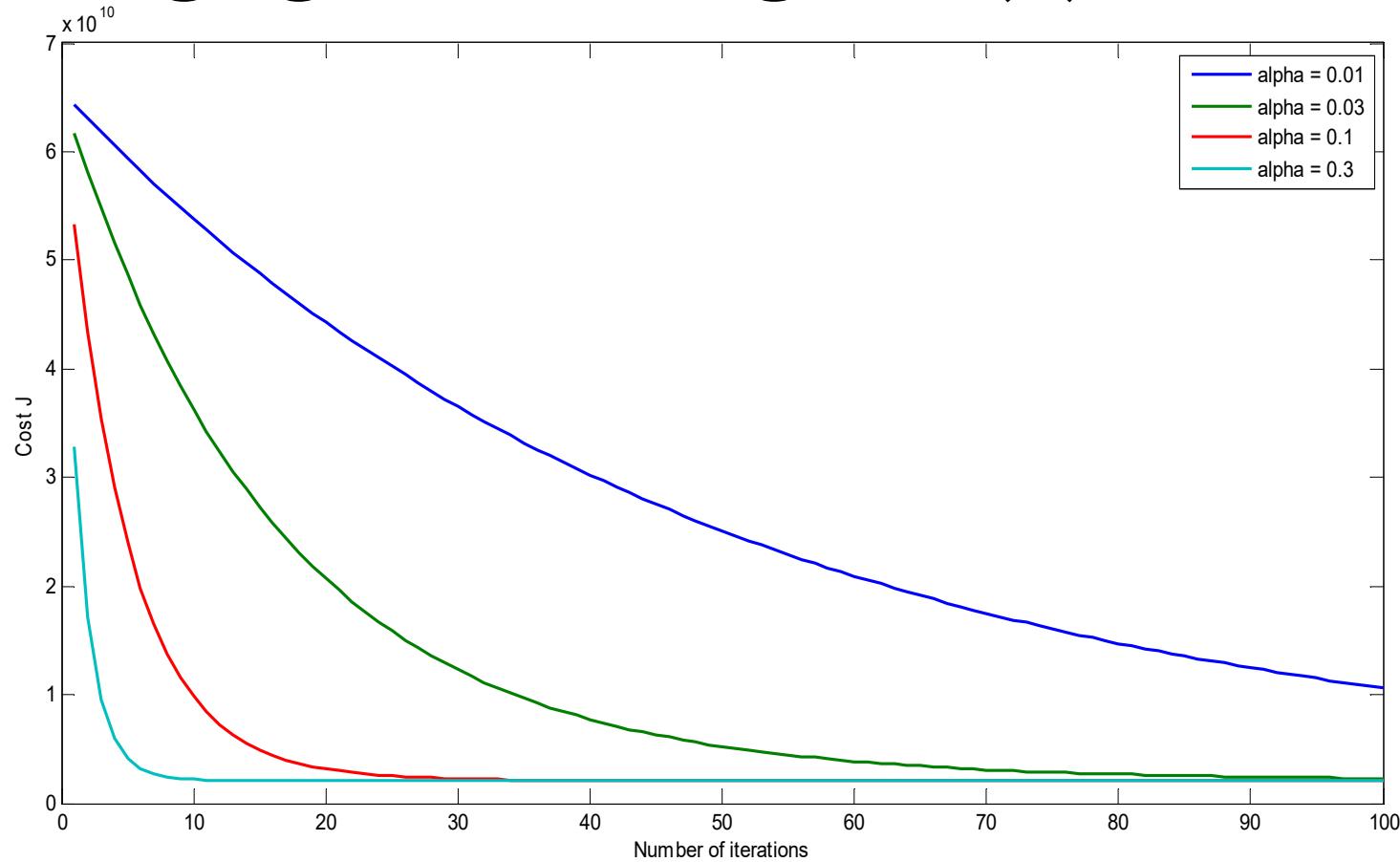
Suppose θ_1 is at a local optima as shown in the figure.

What will one step of Gradient Descent do ?

- 1) Leave θ_1 unchanged
- 2) Change θ_1 in a random direction
- 3) Decrease θ_1
- 4) Move θ_1 in direction to the global minimum of J



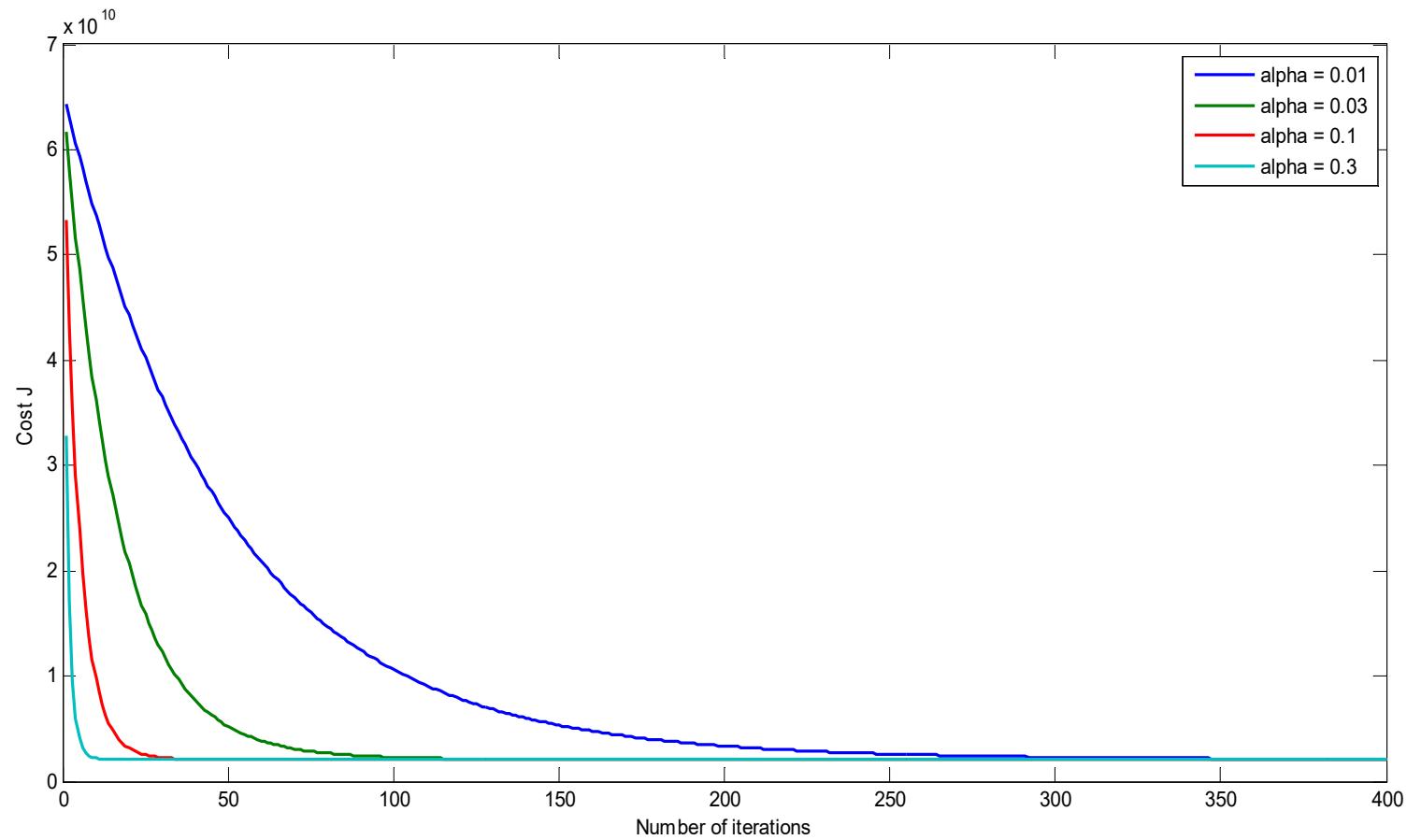
Cost function convergence changing the learning rate (α) -100 iter.



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

If α too small : slow convergence of the cost function J (the Gradient Descent optimization can be slow)

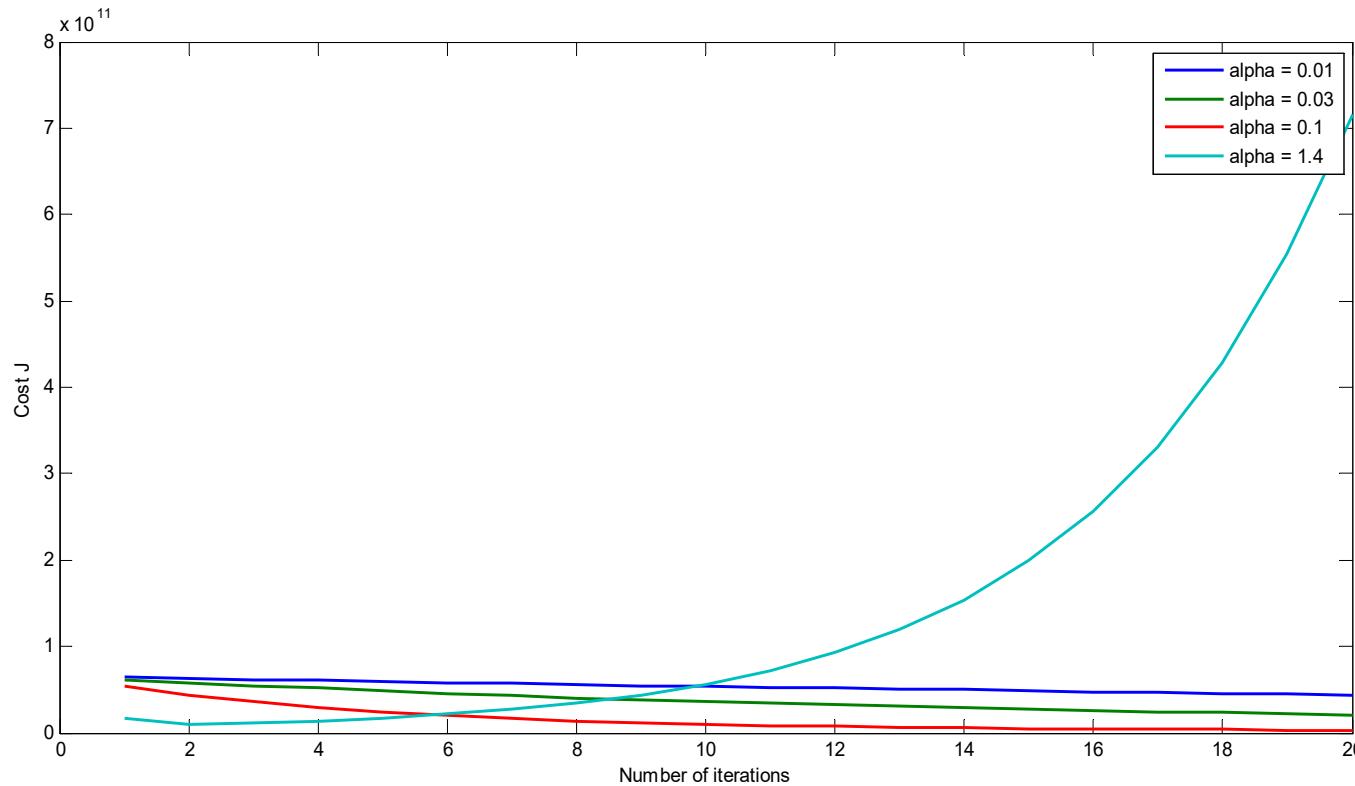
Cost function convergence changing the learning rate (α) -400 iter.



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

14

LinReg Cost function convergence - learning rate variation (α)

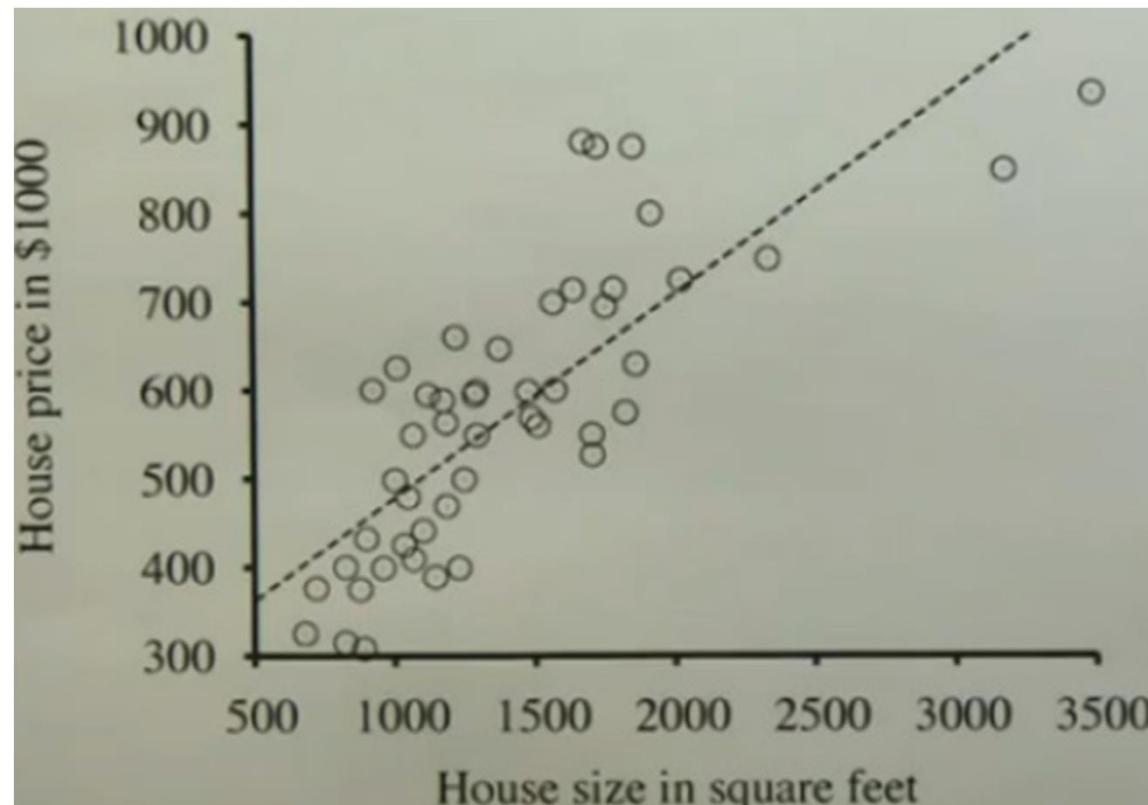


If α too large: the cost function J may no converge (decrease at each iteration). It may diverge !

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Univariate Regression

$$h_{\theta}(x) = \theta^T x = \theta_0 + \theta_1 x_1$$



Given the house area, what is the most likely house price?

If univariate linear regression model is not sufficiently good model,
add more data (ex. # bedrooms).

Multivariate Regression

Problem: Learning to predict the housing price as a function of living area & number of bedrooms.

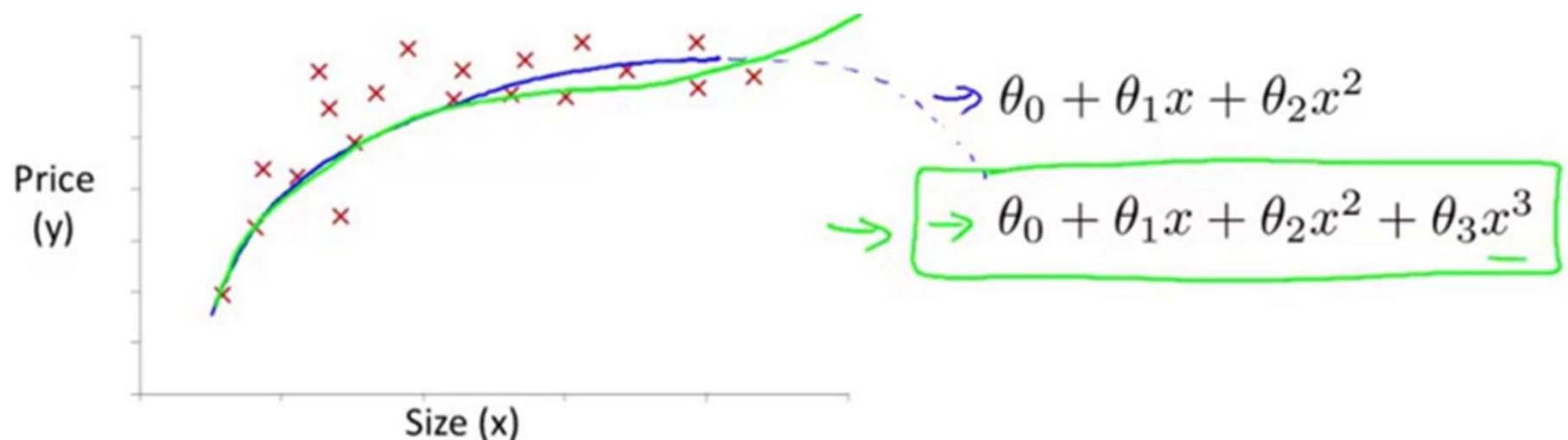
Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
:	:	:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = [\theta_0 \quad \theta_1 \quad \theta_2] \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \end{bmatrix} = \vec{\theta}^T \vec{x}$$

Polynomial Regression

If univariate linear regression model is not a good model, try polynomial model.

Univariate ($x_1=\text{size}$) housing price problem transformed into multivariate (still linear !!!) regression model $x=[x_1=\text{size}, x_2=\text{size}^2, x_3=\text{size}^3]$



$$\begin{aligned} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3 \end{aligned}$$

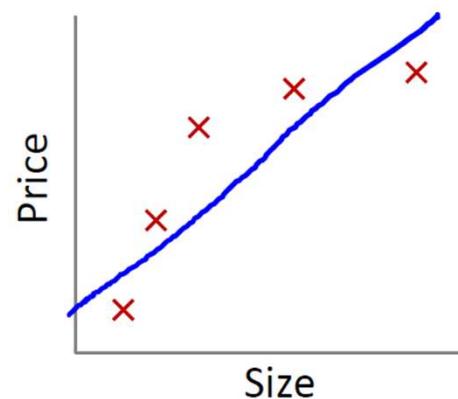
$$x_1 = (\text{size})$$

$$x_2 = (\text{size})^2$$

$$x_3 = (\text{size})^3$$

Overfitting problem

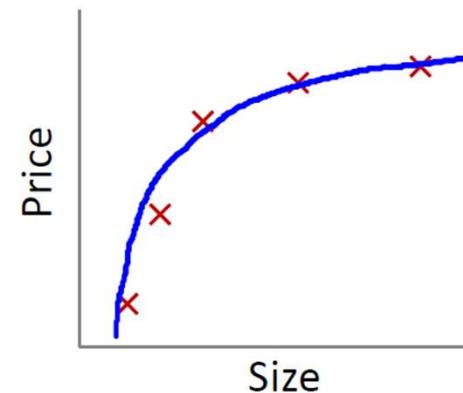
Overfitting: If we have too many features (e.g. high order polynomial model), the learned hypothesis may fit the training set very well but fail to generalize to new examples (predict prices on new examples).



underfit

(1st order polin. model)

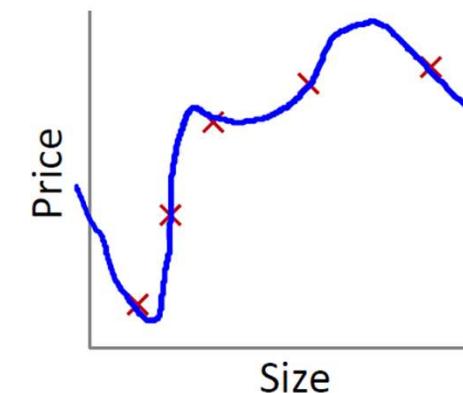
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



just right

(3rd order polinom. model)

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$



overfit

(higher ord. polinom. Model)

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_{16} x^{16}$$

Overfitting problem

Overfitting: If we have too many features (x_1, \dots, x_{100}) the learned model may fit the training data very well but fails to generalize to new examples.

x_1 = size of house

x_2 = no. of bedrooms

x_3 = no. of floors

x_4 = age of house

x_5 = average income in neighborhood

x_6 = kitchen size

:

:

x_{100}

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \vec{\theta}^T \vec{x}$$

How to deal with overfitting problem ?

1. Reduce number of features.

- Manually select which features to keep.
- Algorithm to select the best model complexity.

2. Regularization (add extra term in cost function)

Regularization methods shrink model parameters θ towards zero to prevent overfitting by reducing the variance of the model.

2.1 Ridge Regression (L2 norm)

- Reduce magnitude of θ (but never make them =0) => keep all features
- Works well when all features contributes a bit to the output y .

2.2 Lasso Regression (L1 norm)

- May shrink some of the elements of vector θ to become 0.
- Eliminate some of the features => Serve as feature selection

Regularized Linear Regression (cost function)

Unregularized cost function =>

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Regularized cost function

(add extra regularization term
don't regularize θ_0)

Ridge Regression

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

$$\min_{\theta} J(\theta)$$

Regularized Linear Regression (cost function gradient)

**Unregularized cost
function gradients =>**

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

**Regularized cost
function gradients =>**

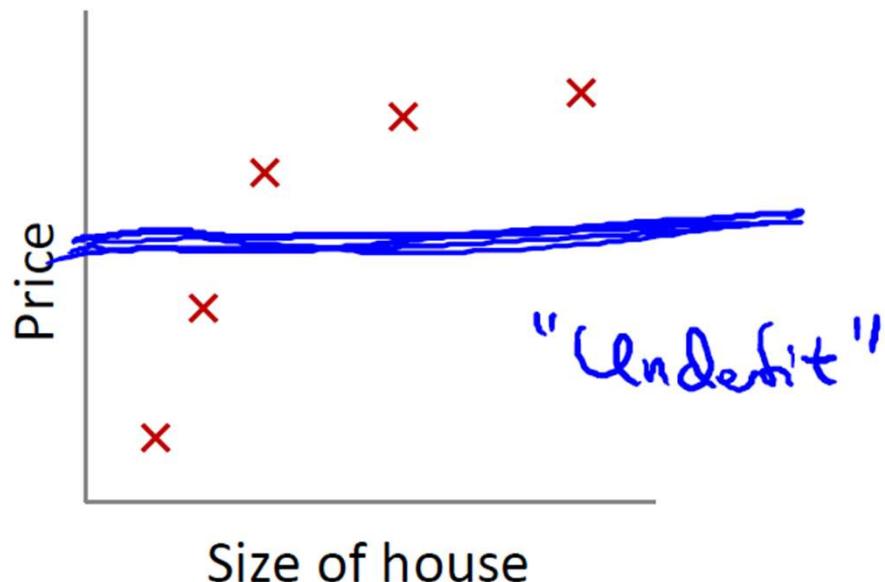
$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \quad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \quad \text{for } j \geq 1$$

Regularized Linear Regression

What if lambda is set to an extremely large value ?

- Algorithm fails to eliminate overfitting.
- Algorithm results in under-fitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.



Regularization: Lasso Regression

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \left[-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n |\theta_j|$$

Ridge Regression shrinks θ towards zero, but never equal to zero => all features are included in the model no matter how small are the coefficients.

Lasso Regression is able to shrink coefficients to exactly zero => reduces the number of features. This makes Lasso Regression useful in cases with high dimension.

Lasso Regression involves absolute values (not differentiable)=> computing is difficult => relevant algorithms available in sklearn Python library.

JUPYTER NOTEBOOK

Project Jupyter - 2014 by [Fernando Pérez](#)

Reference to the 3 core programming languages supported
[Julia](#), [Python](#), [R](#)

[open-source software](#), open-standards, language agnostic
[web-based interactive](#) computational environment for creating
notebook documents.

Ordered list of input/output cells which can contain code, text
(using Markup language – html, latex,), maths, plots

MATLAB-BASED IPYTHON NOTEBOOKS:

<https://anneurai.net/2015/11/12/matlab-based-ipython-notebooks/>.

Aprendizagem Aplicada à Segurança

(Mestrado em Cibersegurança-DETI-UA)



LECTURE 2

Supervised learning – Classification

Petia Georgieva
(petia@ua.pt)

DETI/IEETA – UA

OUTLINE

- **Logistic Regression (logit model)**
- **Support Vector machines (SVM)**
- **K- Nearest-Neighbor (k-NN)**
- **Decision Tree (DT)**

Classification –

LOGISTIC REGRESSION (LOGIT)

Binary vs Multiclass Classification

Email: Spam /Not Spam ?

Tumour: Malignant /Benign ?

Online Transactions: Fraudulent (Yes / No) ?

Binary classification:

$y = 1$: “positive class” (e.g. malignant tumour)

$y = 0$: “negative class” (e.g. benign tumour)

Find a model $h(x)$ that outputs values between 0 and 1

$$0 \leq h(x) \leq 1$$

if $h(x) \geq 0.5$, predict “ $y=1$ ”

if $h(x) < 0.5$, predict “ $y=0$ ”

Multiclass classification (K classes) $\Rightarrow y = \{0, 1, 2, \dots\}$

Build K binary classifiers, for each classifier one of the classes has label 1 all other classes take label 0 .

Logistic Regression

Given labelled data of m examples, n features

Labels $\{0,1\} \Rightarrow$ binary classification

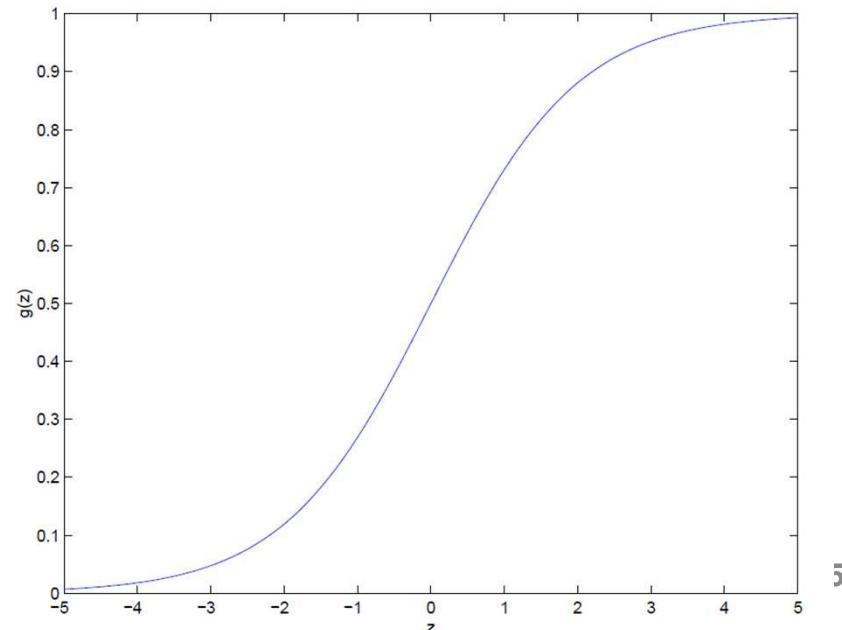
x –vector of features; θ – vector of model parameters;

$h(x)$ – logistic (sigmoid function) model – logit model

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}} = \frac{1}{1 + e^{-z}} = g(\theta^T x) = g(z)$$

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Logistic (sigmoid) function



Logistic Regression Cost Function

Linear regression model =>

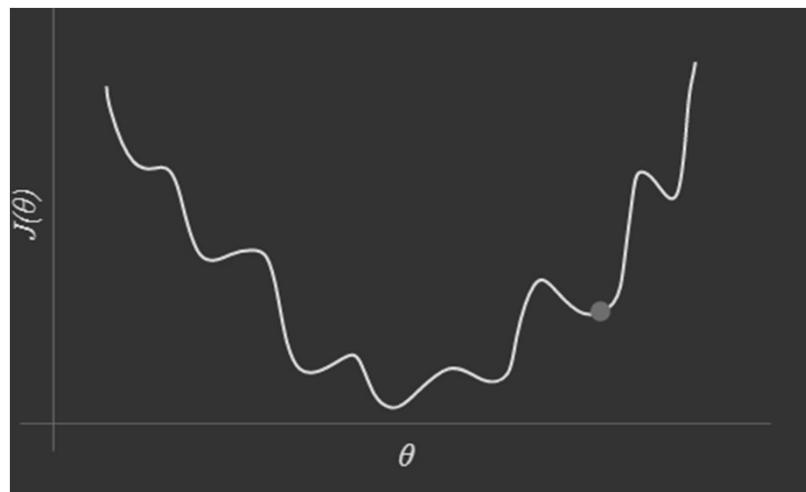
$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \vec{\theta}^T \vec{x}$$

Lin Regr. cost (loss) function (MSE) => $J = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

Nonlinear logit model =>

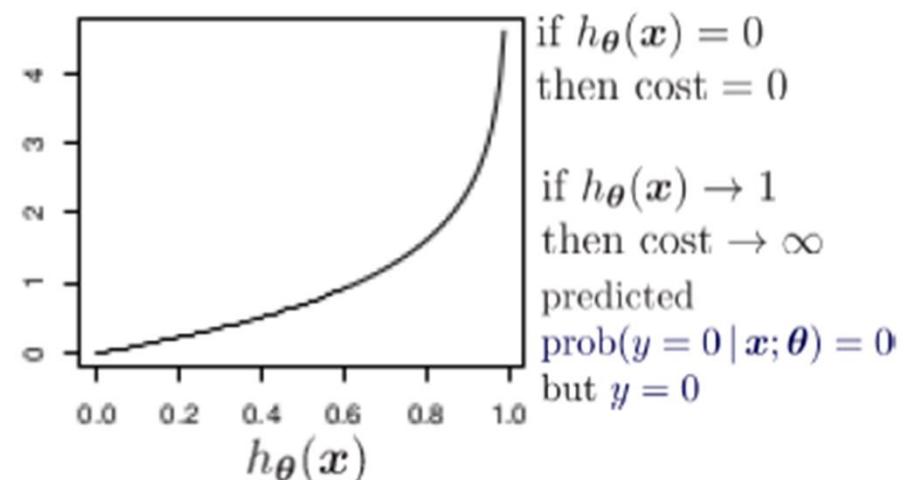
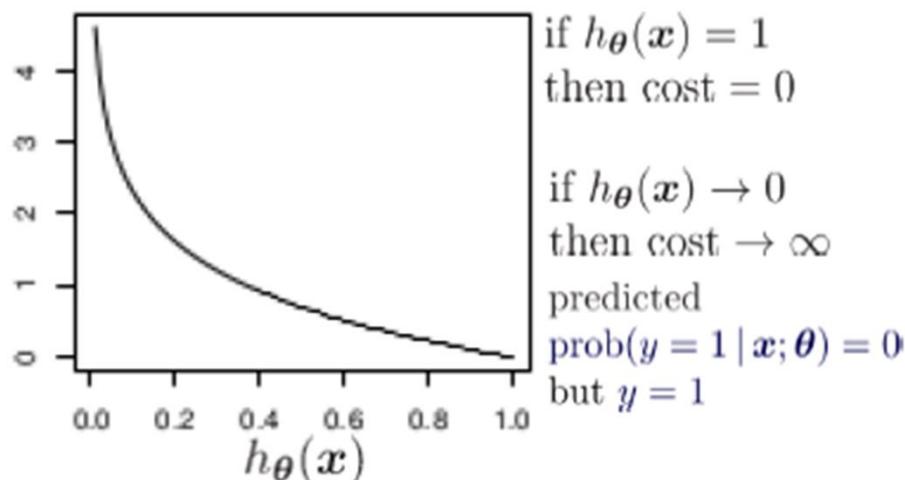
$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

If we use the same cost function as with linear regression, but now we have the nonlinear logit model, $J(\theta)$ will be a non-convex function (has many local minima)=> **not efficient for optimization !**



Logistic Regression Cost Function

$$\text{cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



**Logistic regression cost function combined into one expression :
(also known as *binary Cross-Entropy or Log Loss function*)**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

Logit with gradient descent learning

Initialize model parameters (*e.g.* $\theta = 0$)

Repeat until J converge {

Compute Logit Model prediction =>
(different from linear regression model)

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Compute Logit cost function =>
(different from linear regression cost function)

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_{\theta}(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Goal =>

$$\min_{\theta} J(\theta)$$

Compute cost function gradients =>
(same as linear regression gradients)

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Update parameters =>
(same as linear regression parameter update)

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

Optimization algorithms

Gradient descent (learned in class) -

updates the parameters in direction in which the gradient decreases most rapidly.

2. Other optimization algorithms

- Conjugate gradient
- AdaGrad, RMSProp
- Stochastic gradient descent with momentum
- ADAM (combination of RMSProp and stochastic optimization)
- BFGS (Broyden–Fletcher–Goldfarb–Shanno)
- Quasi-Newton methods (approximate the second derivative)

Characteristics

- Adaptive learning rate (alfa);
- Often faster than gradient descent; better convergence;
- Approximate (estimate) the true gradient over a mini-batch and not over the whole data;
- More complex algorithms

Logistic regression - example

Ex.: We have applicant's scores on two admission exams in the univ (these are the features x_1 and x_2) and we know the final decision (admitted or not admitted – the labels y). Build a logistic regression model to tell what are the chances for new candidates to be admitted into the university if we know their exam scores.

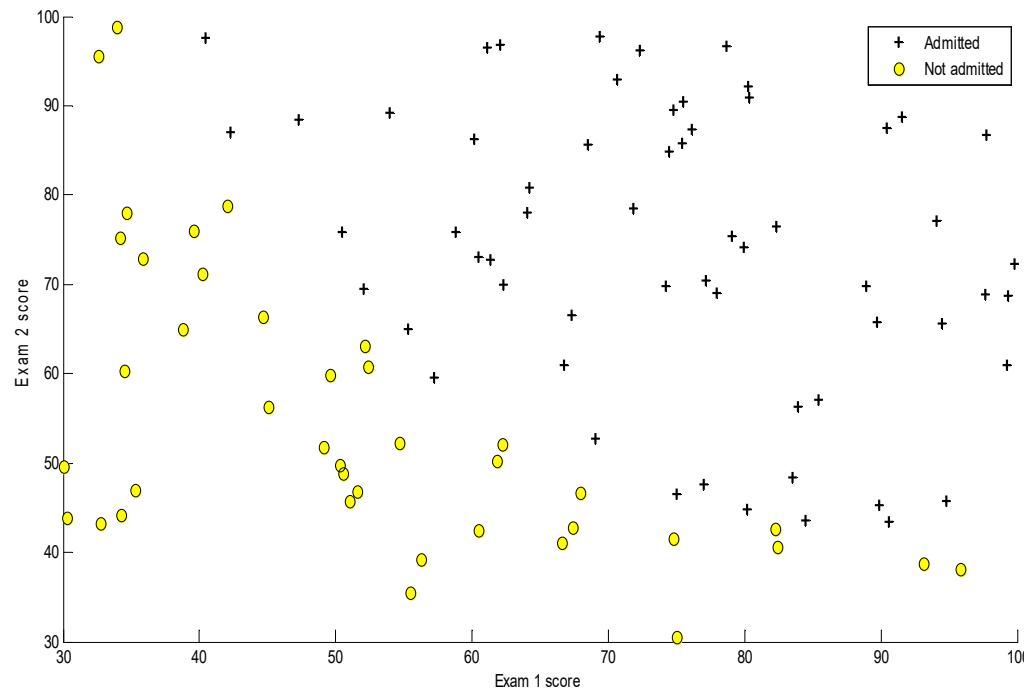


Fig. 1 Training Data

Logistic regression - example

$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \Rightarrow$ decision boundary

if $z > 0 \Rightarrow g(z) > 0.5 \Rightarrow$ predict class = 1

$$g(z) = \frac{1}{1+e^{-z}}$$

if $z < 0 \Rightarrow g(z) < 0.5 \Rightarrow$ predict class = 0

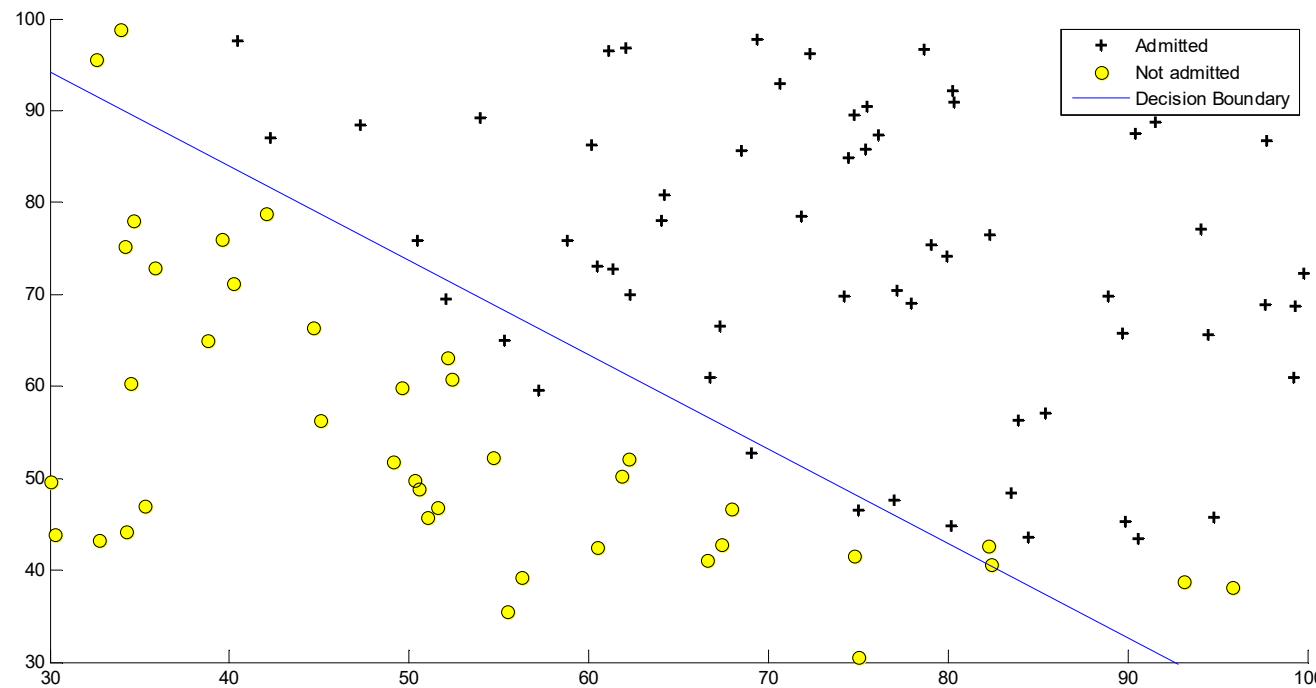
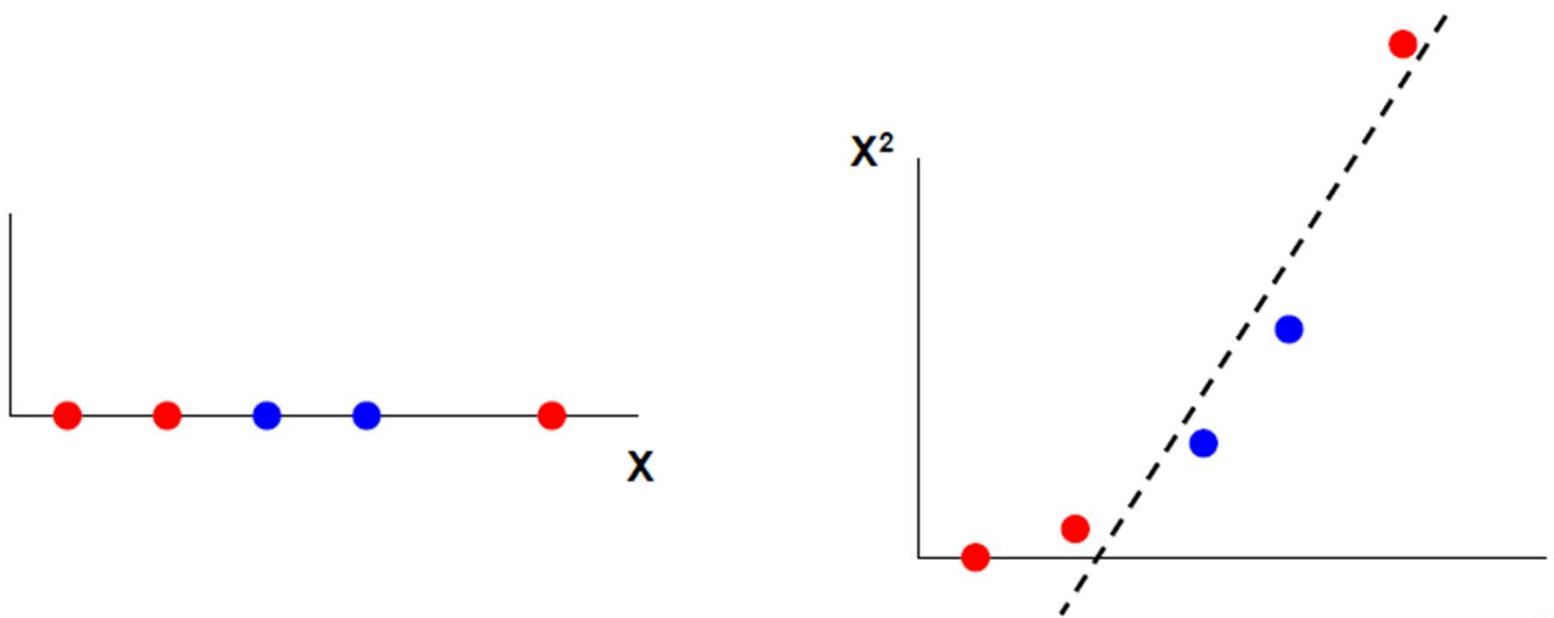


Fig. Training data and linear decision boundary with the optimized parameters (θ)

Nonlinearly Separable Data

**Linear classifier cannot
classify these examples.**

And now ?



$$z = \theta^T x = \theta_0 + \theta_1 x + \theta_2 x^2 = 0 \Rightarrow$$

Nonlinear decision boundary (in the original feature space x)

Linear decision boundary (in the extended feature space x, x²)

if $z > 0 \Rightarrow g(z) > 0.5 \Rightarrow$ predict class = 1

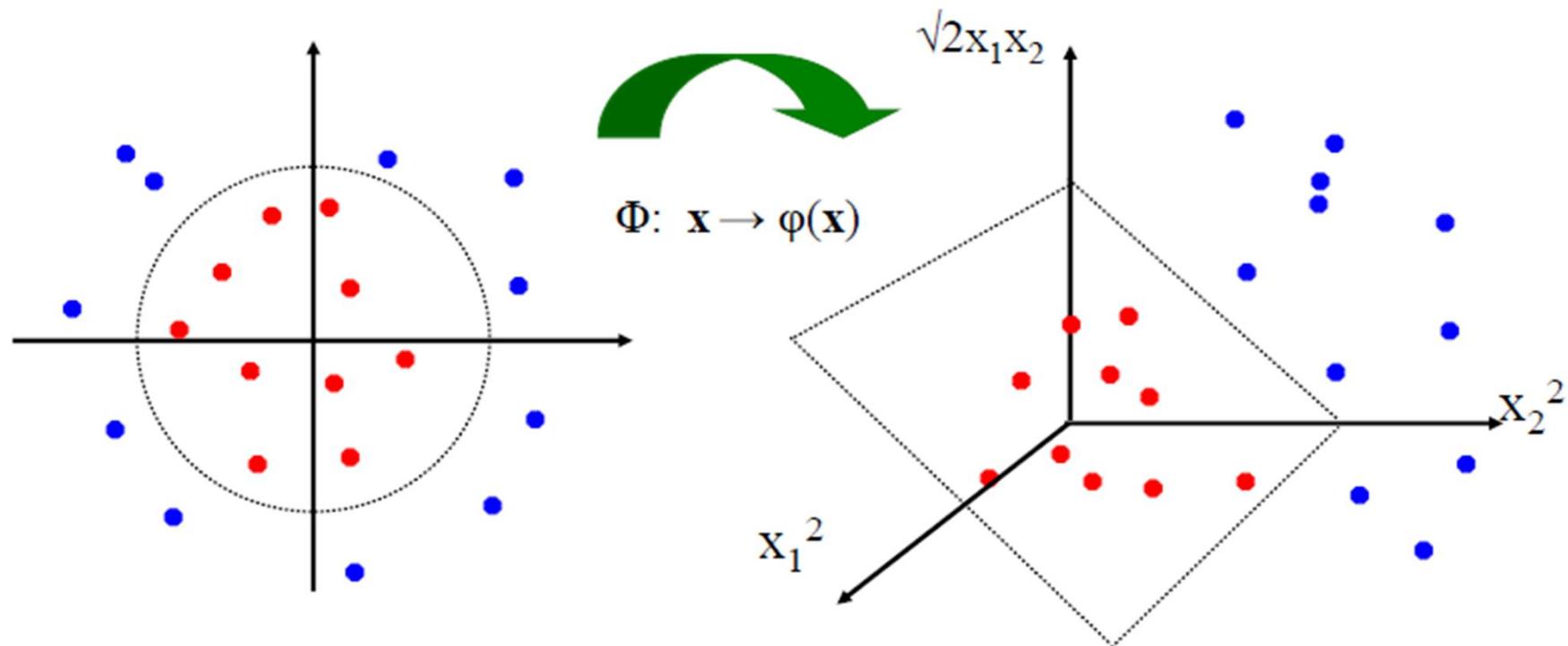
if $z < 0 \Rightarrow g(z) < 0.5 \Rightarrow$ predict class = 0

Nonlinearly Separable Data

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(x)$) where the training set is separable:

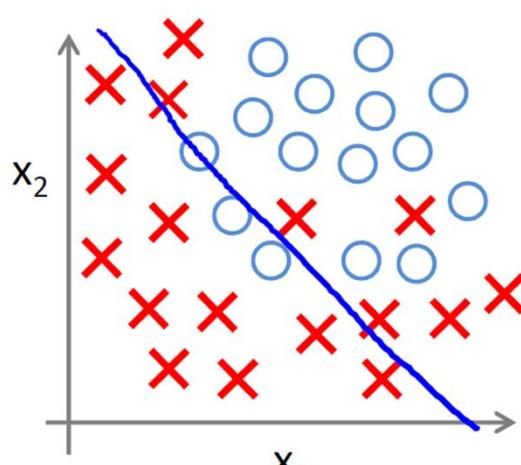
$$x = (x_1, x_2)$$

$$\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

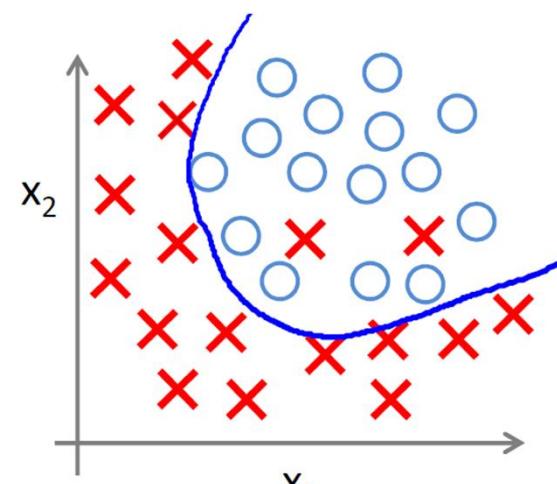


Overfitting problem

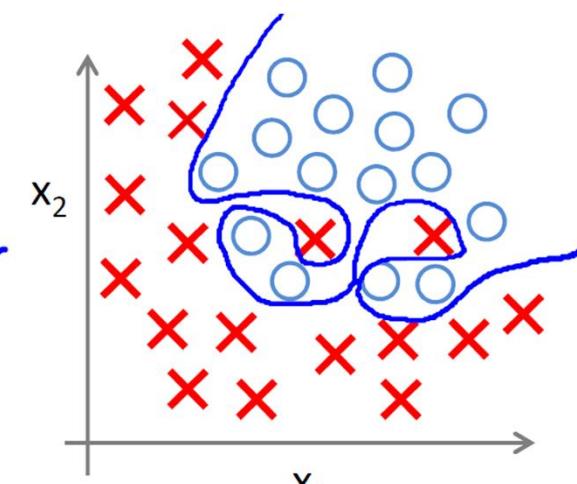
Overfitting: If we have too many features, the learned model may fit the training data very well but fail to generalize to new examples.



underfit - high bias model



ok model



overfit – high variance

Regularization

Regularization to prevent overfitting.

1 Ridge Regression

- Keep all the features, but reduces the magnitude of θ .
- Works well when each of the features contributes a bit to predict y.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

2 Lasso Regression

- May shrink some coefficients of θ to exactly zero.
- Serve as a feature selection tools (reduces the number of features).

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n |\theta_j|$$

Regularized Logistic Regression

Unregularized Logit cost function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

Regularized Logit cost function (ridge regression)

λ is the regularization parameter (hyper-parameter) that needs to be selected

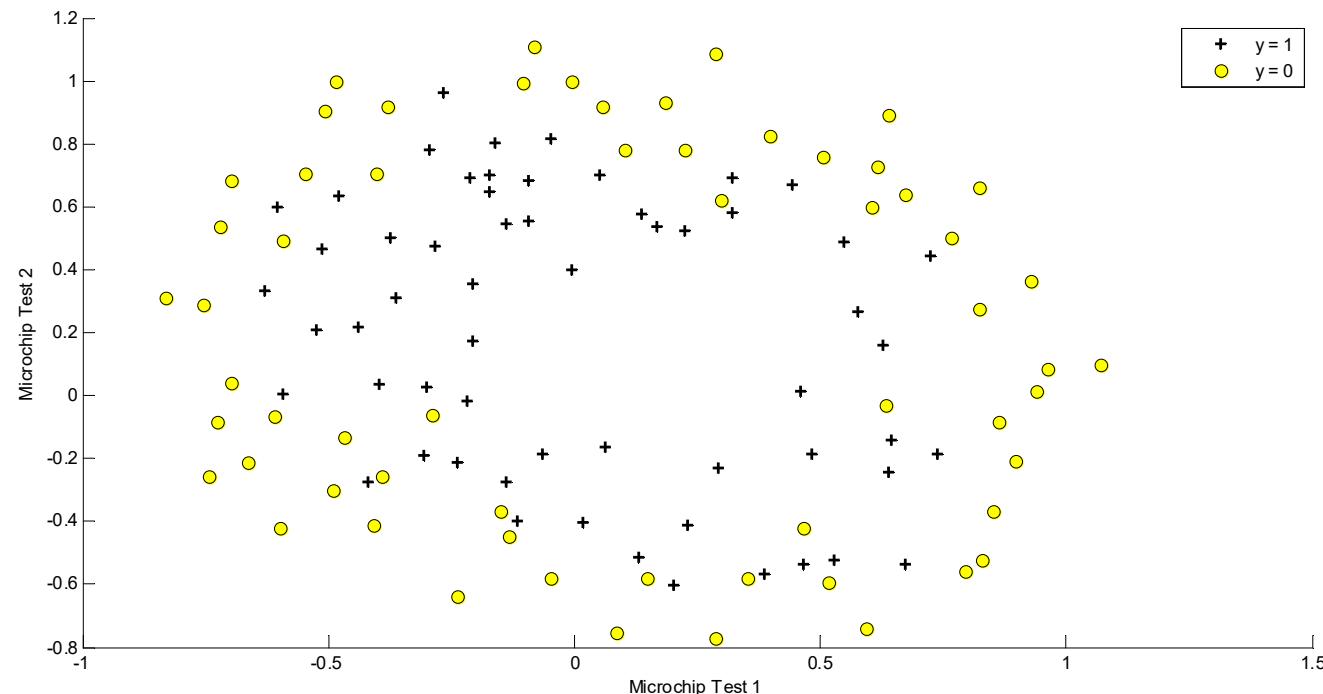
$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Small $\lambda \Rightarrow$ lower bias, higher variance

High $\lambda \Rightarrow$ higher bias, lower variance

Regularized Log Reg -example

Predict whether microchips from a fabrication plant passes quality assurance (QA). During QA, each microchip goes through various tests to ensure it is functioning correctly. Suppose we have the test results for some microchips on two different tests. From these two tests, we would like to determine whether the microchips should be accepted ($y=1$) or rejected ($y=0$).



Regularized Log Reg -example

Dataset is not linearly separable => logistic regression will only be able to find a linear decision boundary. One way to fit the data better is to create more features. For example add polynomial terms of x_1 and x_2 .

$$\text{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

NONLINEAR decision boundary \Rightarrow

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \dots + \theta_{28} x_2^6 = 0$$

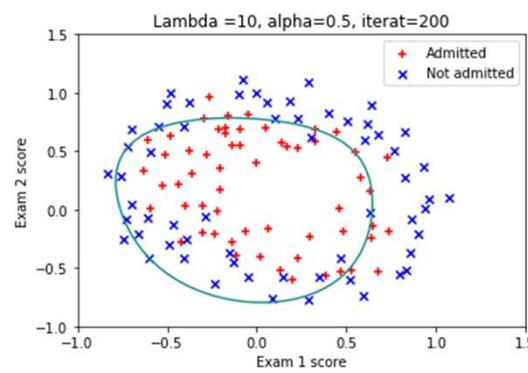
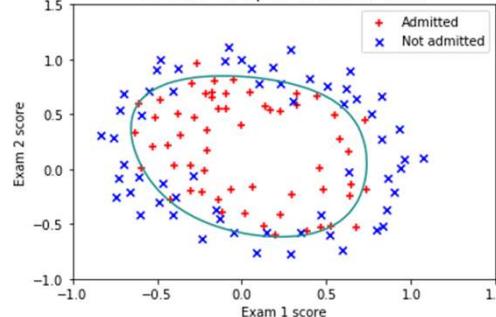
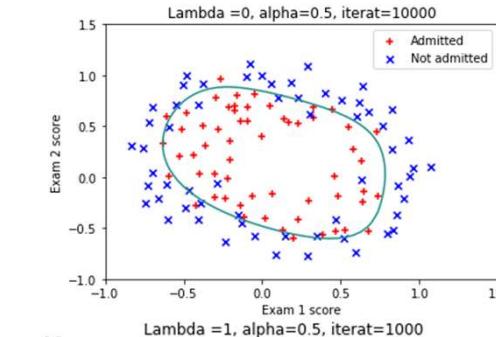
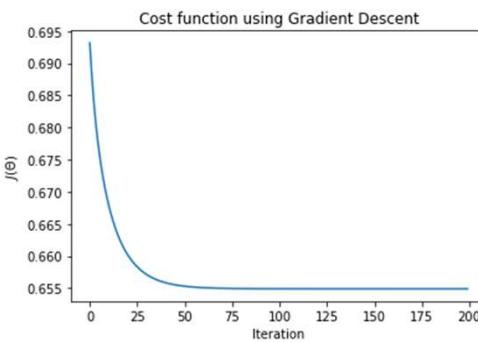
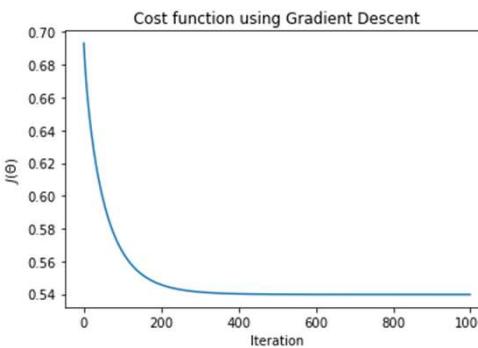
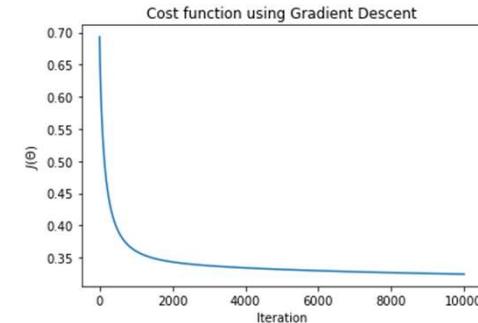
if $z > 0 \Rightarrow g(z) > 0.5 \Rightarrow$ predict class = 1

if $z < 0 \Rightarrow g(z) < 0.5 \Rightarrow$ predict class = 0

ML

Regularized Log Reg -example

Accuracy on training data: :84.75% ($\lambda=0$) | 83.90 % ($\lambda=1$) | 71.2 % ($\lambda=10$))



Multiclass Classification

Exs. *Email division* (work, friends, family, hobby)

Medical diagnosis (not ill, cold, flu) ; *Weather* (sunny, cloudy, rain, snow)

One-versus-all strategy :

For K classes train K binary classifiers:

for c=1:K

 Make $y_{\text{binary}}=1$ (only for examples of class c)

$y_{\text{binary}}=0$ (for examples of all other classes)

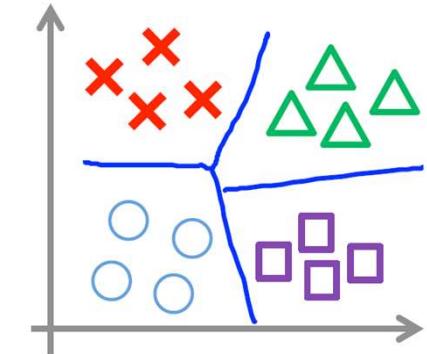
 theta=Train classifier with training data X and output y_{binary} .

 Save the learned parameters of all classifiers in one matrix

 where each row is the learned parameters of one classifier:

 theta_all(c,:)=theta

end



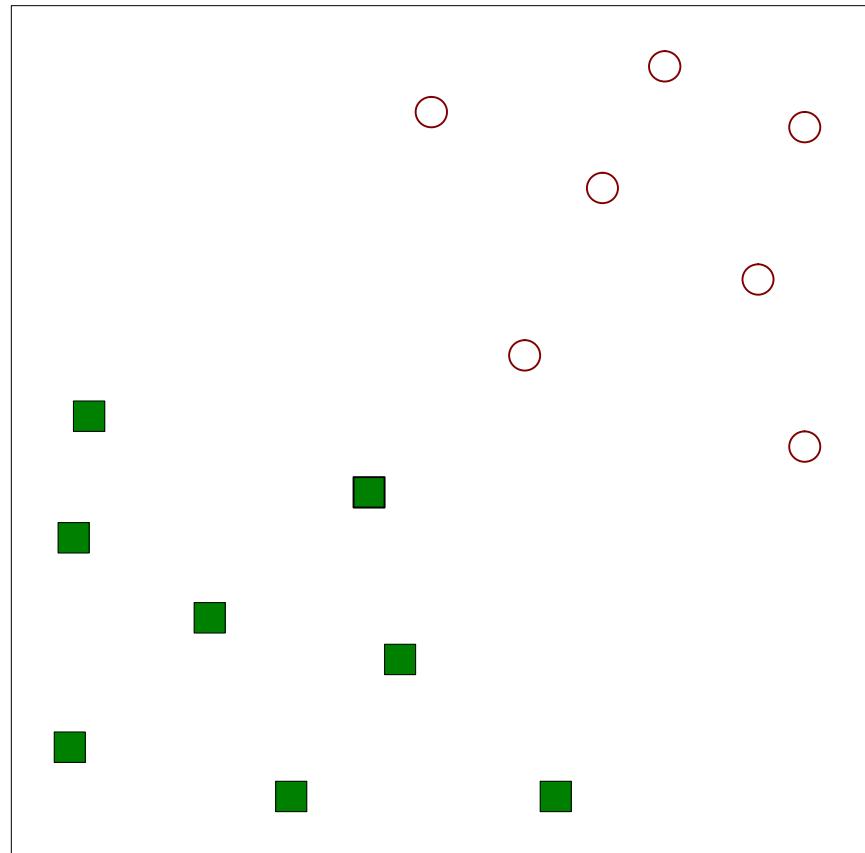
New example: winner-takes-all strategy, the binary classifier with the highest output score assigns the class.

Classification –

SUPPORT VECTOR MACHINES (SVM)

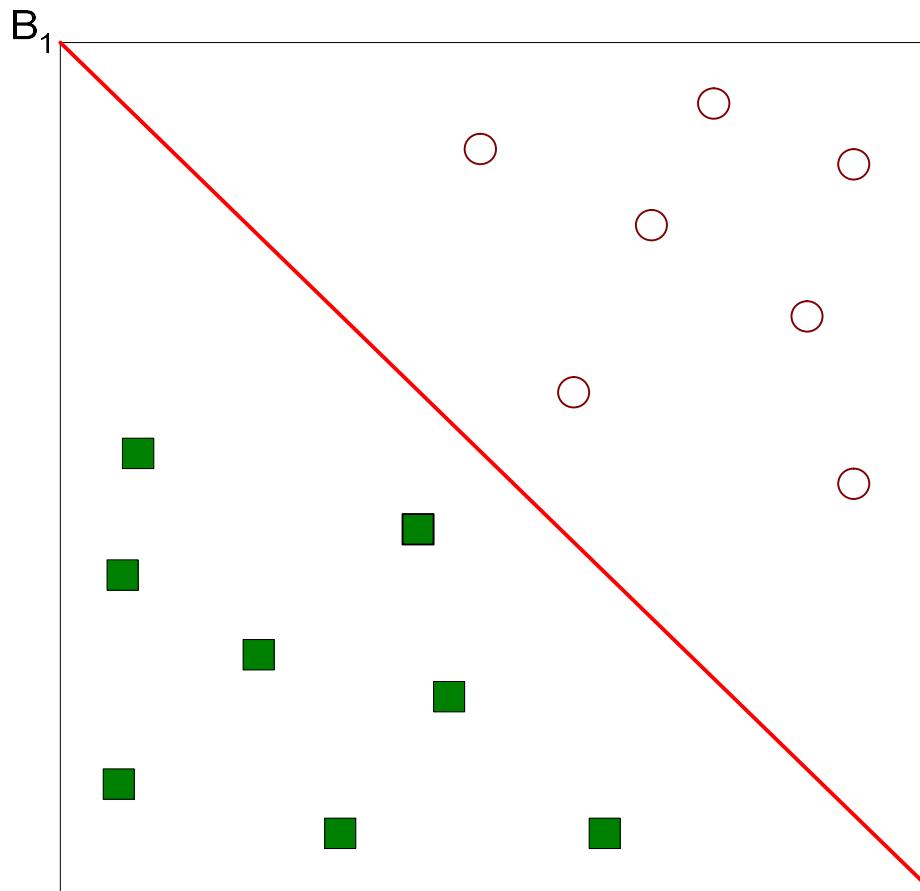
Proposed by Vladimir N. Vapnik and Alexey Chervonenkis, 1963

Linearly separable classes



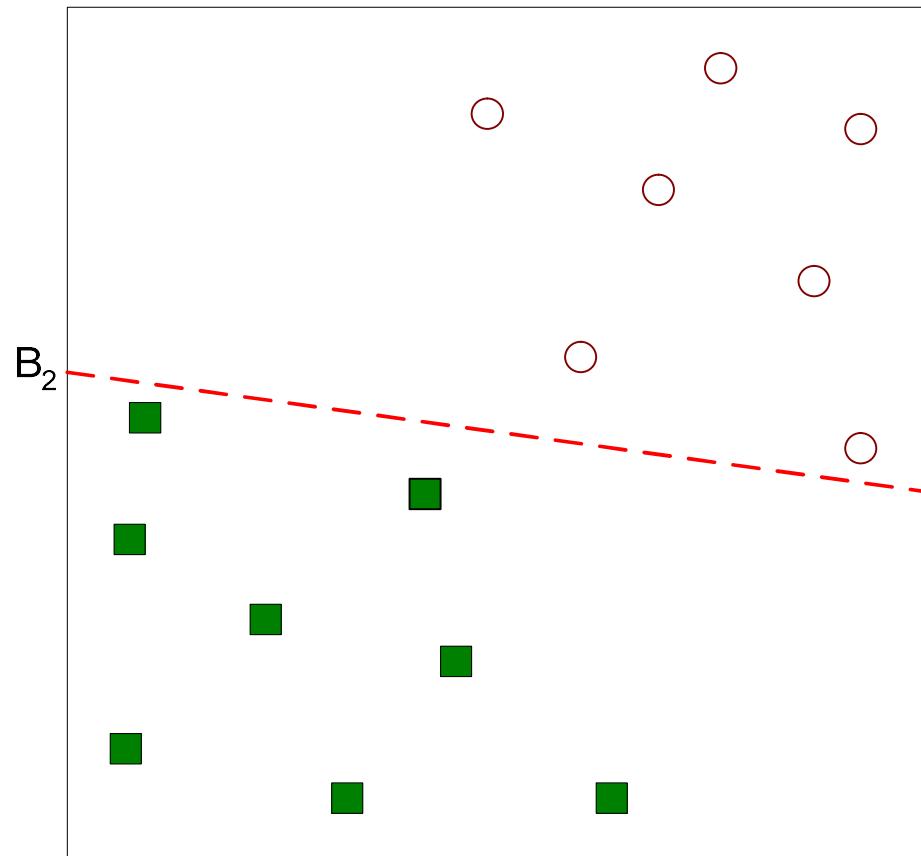
Find a decision boundary to separate data

Linearly separable classes



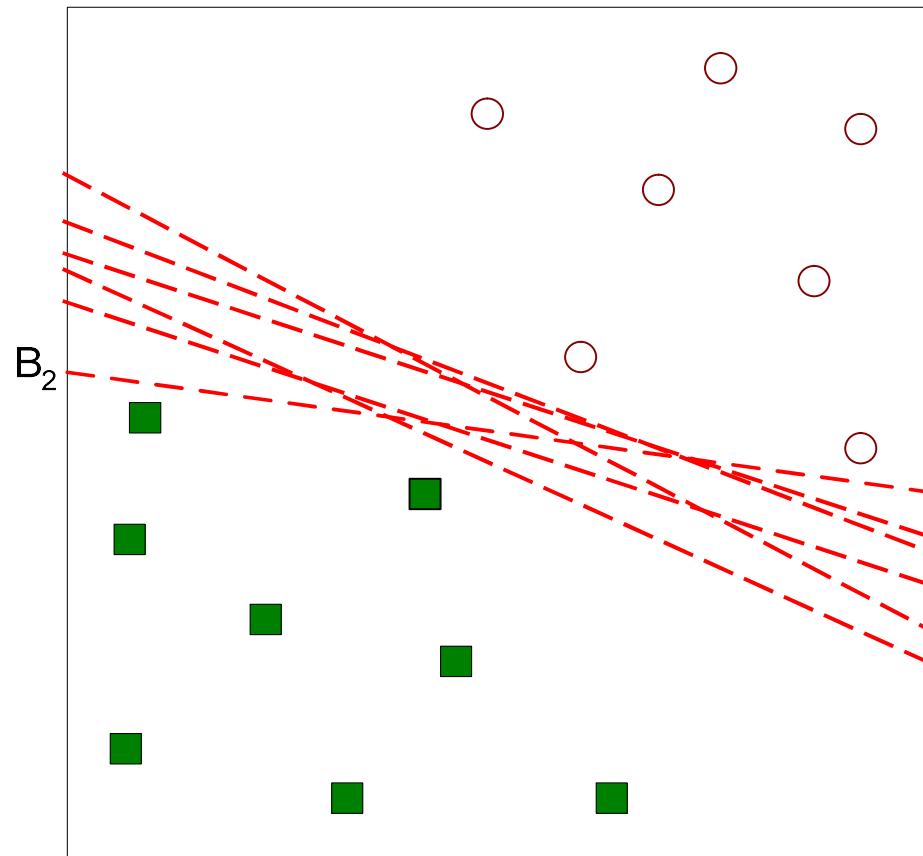
One Possible Solution

Linearly separable classes



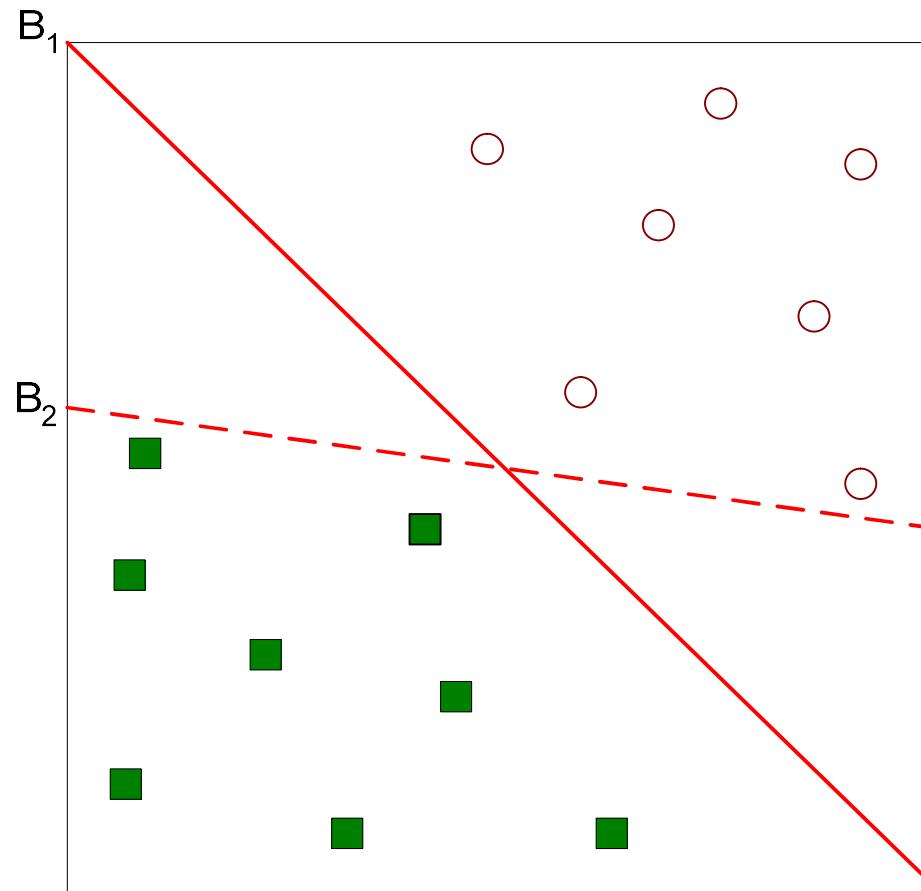
Another possible solution

Linearly separable classes



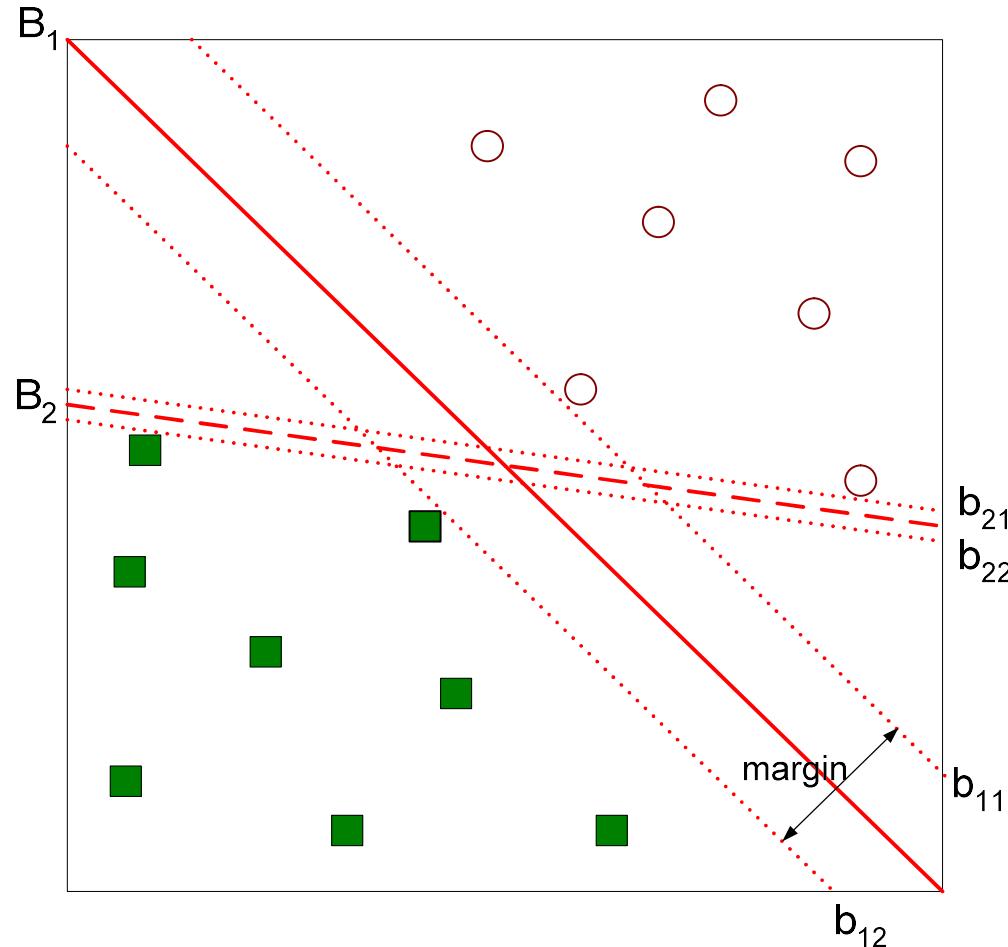
Many possible solutions

Linearly separable classes



Which one is better? B_1 or B_2 ?

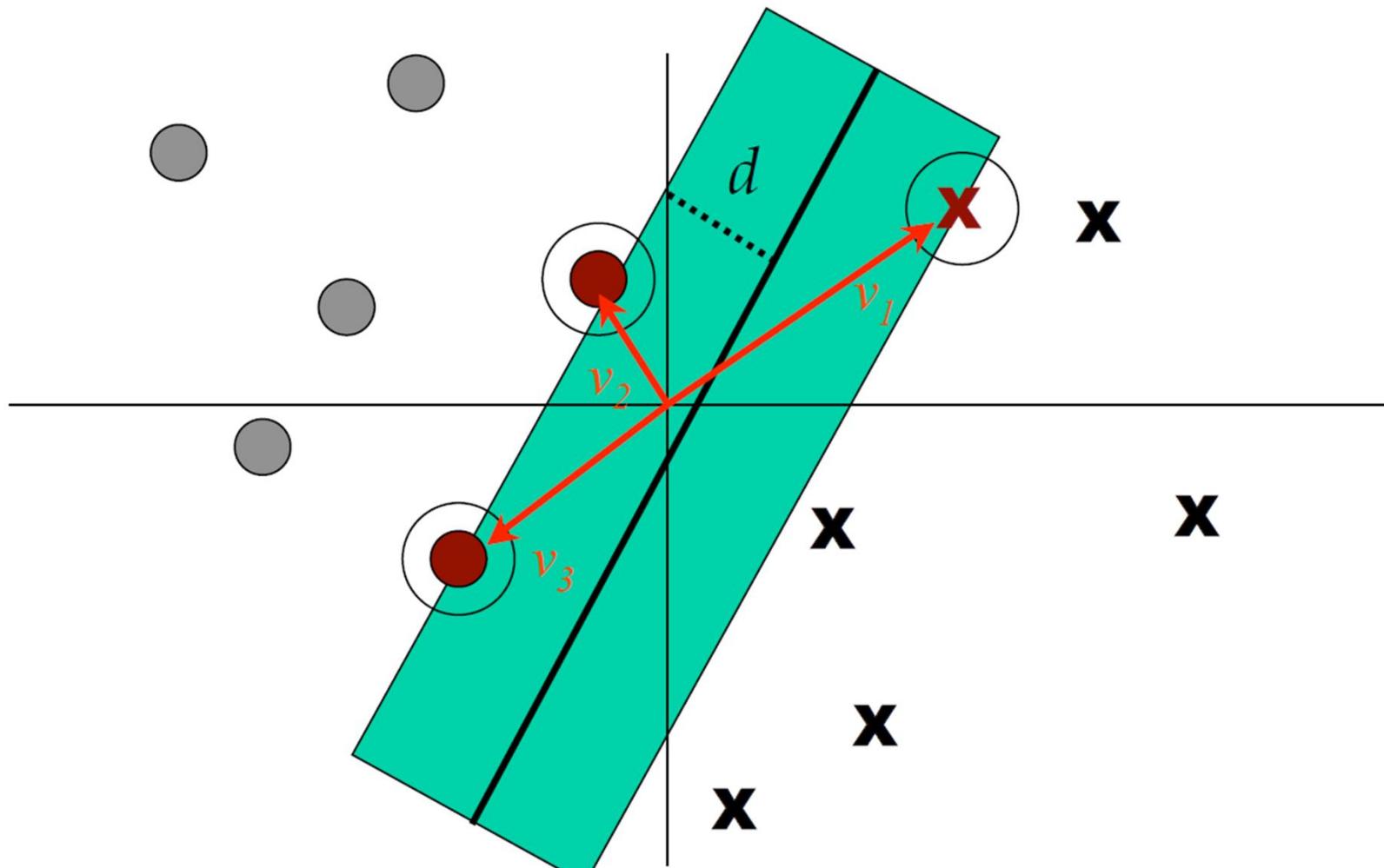
SVM - Large margin classifier



Find a boundary that **maximizes** the margin => B1 is better than B2

SUPPORT VECTORS (v_1, v_2, v_3)

Only the closest points (support vectors) from each class are used to decide which is the optimum (the largest) margin between the classes.

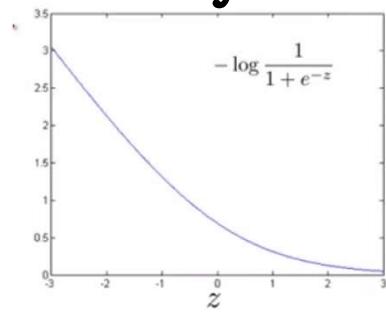


SVM cost function

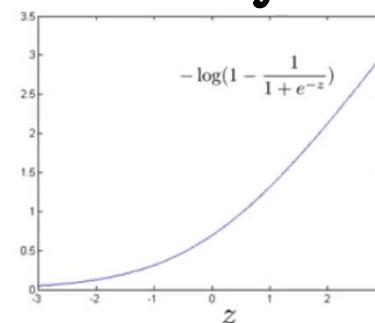
Regularized LogReg cost function:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \left(-\log h_{\theta}(x^{(i)}) \right) + (1 - y^{(i)}) \left((-\log(1 - h_{\theta}(x^{(i)}))) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

for $y=1$

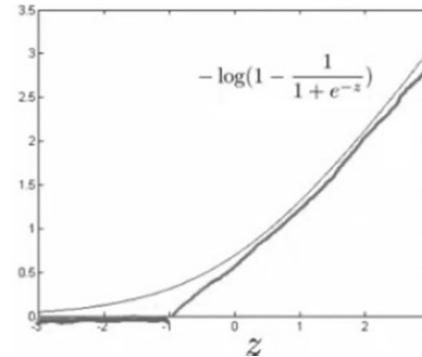
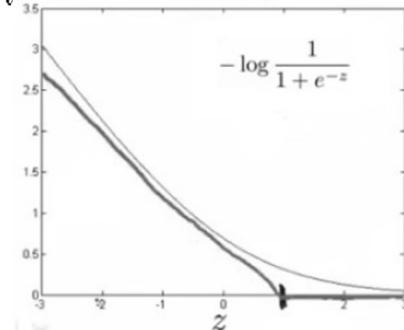


for $y=0$



Regularized SVM cost function (Modification of Logit cost function.
cost0 & **cost1** are asymptotic safety margins with computational advantages)

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$



$$z = \theta^T x$$

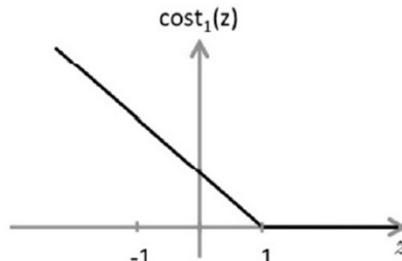
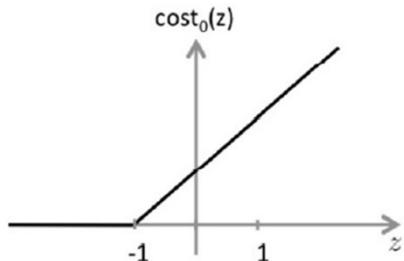
SVM cost function

Regularized LogReg cost function:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \left(-\log h_{\theta}(x^{(i)}) \right) + (1 - y^{(i)}) \left((-\log(1 - h_{\theta}(x^{(i)}))) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Regularized SVM cost function

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$



$$z = \theta^T x$$

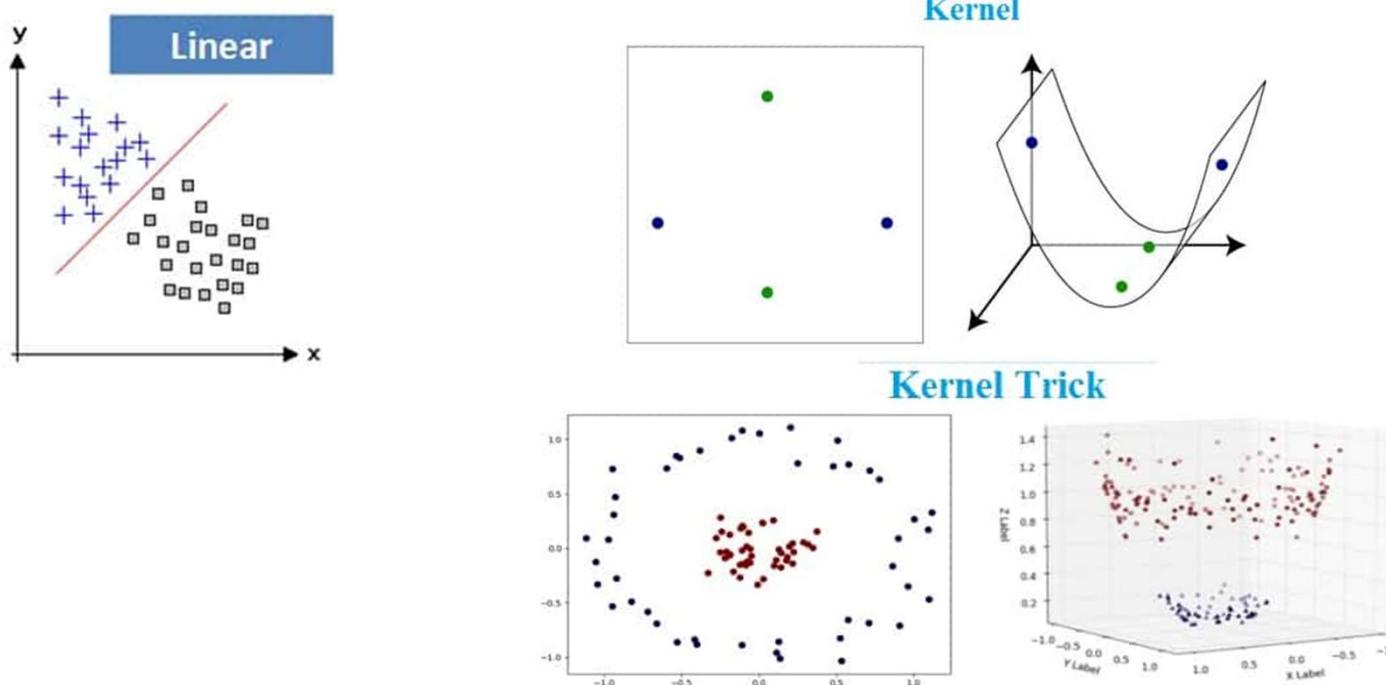
Different way of parameterization: C is equivalent to $1/\lambda$.

$C > 0$ - parameter that controls the penalty for misclassified training examples.

Increase C more importance to training data fitting.

Decrease C – more importance to generalization properties (combat overfitting).

Nonlinearly separable data – kernel SVM



Kernel: function which maps a lower-dimensional data into higher dimensional data.

Typical Kernels:

- Polynomial Kernel - adding extra polynomial terms
- Gaussian Radial Basis Function (RBF) kernel – the most used kernel
- Laplace RBF kernel
- Hyperbolic tangent kernel
- Sigmoid kernel, etc.

Nonlinear SVM – Gaussian RBF Kernel

$$k(x_i, x_j) = e^{-\gamma \|x^{(i)} - x^{(j)}\|^2}, \quad \gamma > 0, \gamma = 1/2\sigma^2, \quad \sigma - \text{variance}$$

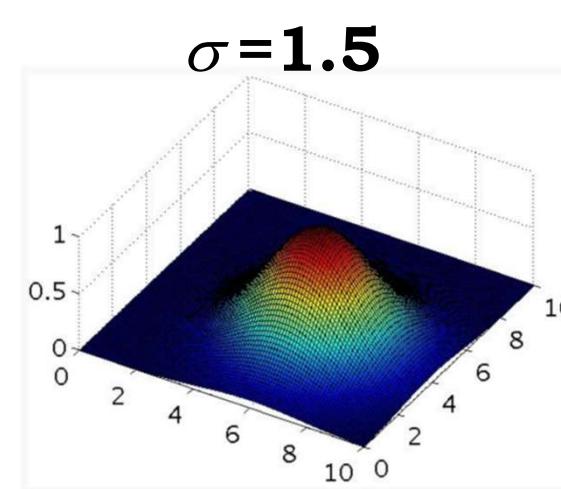
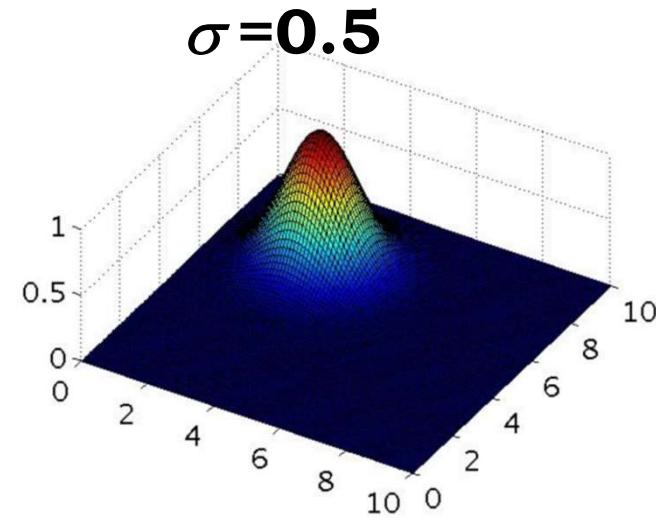
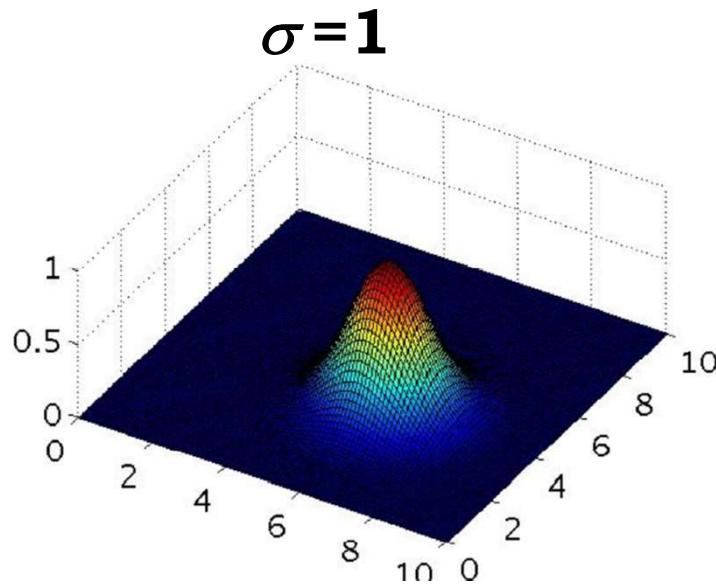
The RBF kernel is a metric of similarity between examples, $x^{(i)}$ and $x^{(j)}$
Substitute the original features with similarity features (kernels).

Note: the original ($n+1$ dimensional) feature vector is substituted
by the new ($m+1$ dimensional) similarity feature vector.

m – number of examples, **m>>n !!!**

Gaussian RBF Kernel – Parameter σ

$$k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \quad \gamma = \frac{1}{2\sigma^2} > 0$$



**σ determines how fast
the similarity metric decreases
to 0 as the examples go away of each other.**

Large σ : kernels vary more smoothly (combat overfitting)

Small σ : kernels vary less smoothly (more importance to training data fitting)

SVM parameters

How to **choose hyper-parameter C**:

Large C: lower bias, high variance (equivalent to small regular. param. λ)

Small C: higher bias, lower variance (equivalent to large regular. param. λ)

How to **choose hyper-parameter σ** :

Large σ : features vary more smoothly. Higher bias, lower variance

Small σ : features vary less smoothly. Lower bias, higher variance

SVM implementation

Use SVM software packages to solve SVM optimization !!!

In Python, use Scikit-learn (sklearn) machine learning library and

Import SVC (Support Vector Classification):

```
from sklearn.svm import SVC  
classifier = SVC(kernel="rbf",gamma =?)
```

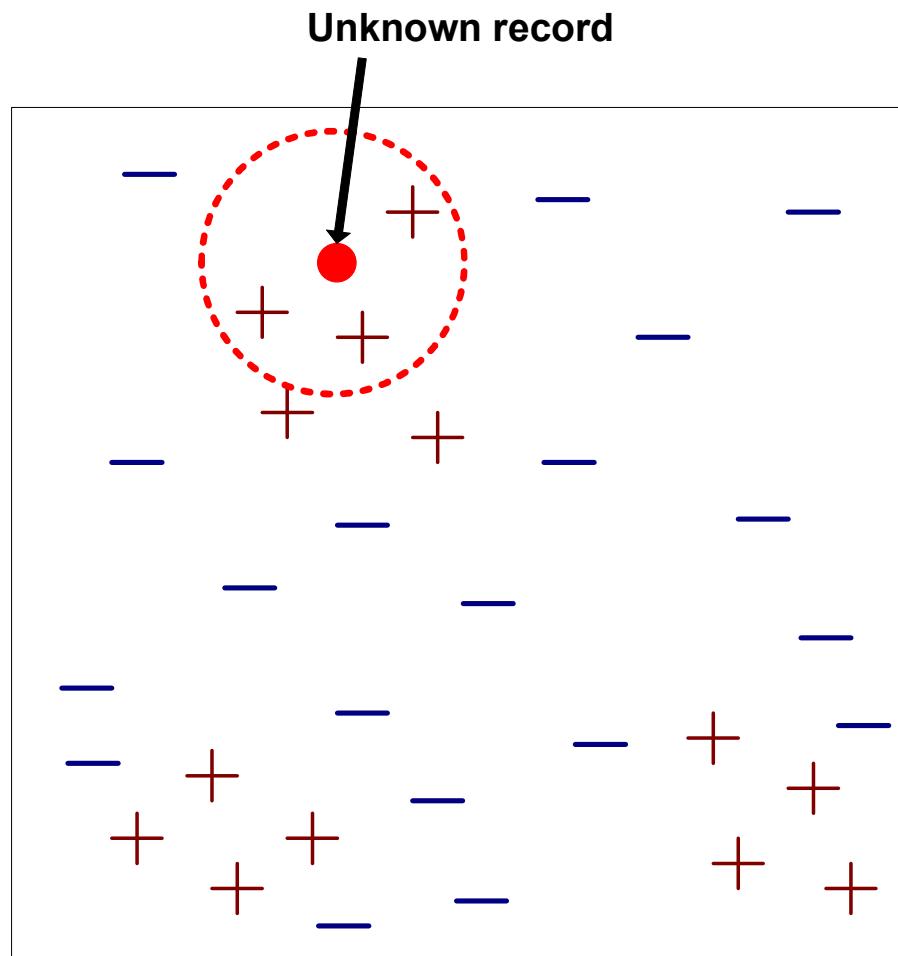
"rbf" (Radial Basis Function) corresponds to the Gaussian kernel.

gamma = $1/\sigma$.

Classification –

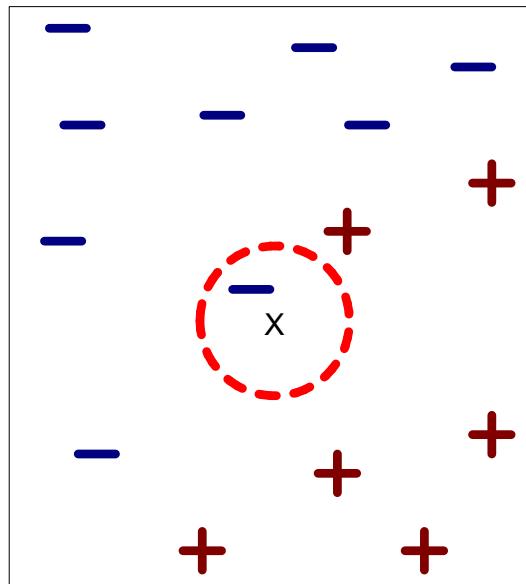
K- Nearest-Neighbor (k-NN)

K- Nearest-Neighbor (k-NN) Classifier

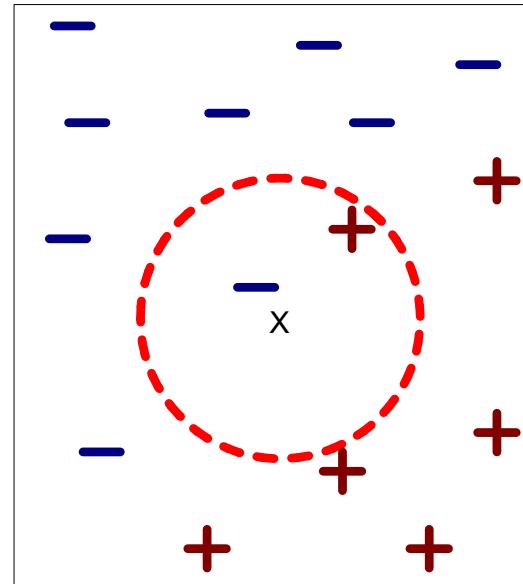


- KNN requires:
 - Set of labeled records.
 - Measure to compute distance (similarity) between records.
 - K is the number of nearest neighbors (the closest points).
- To classify a new (unlabeled) record:
 - Compute its distance to all labeled records.
 - Identify k nearest neighbors.
 - The class label of the new record is the label of the majority of the nearest neighbors.

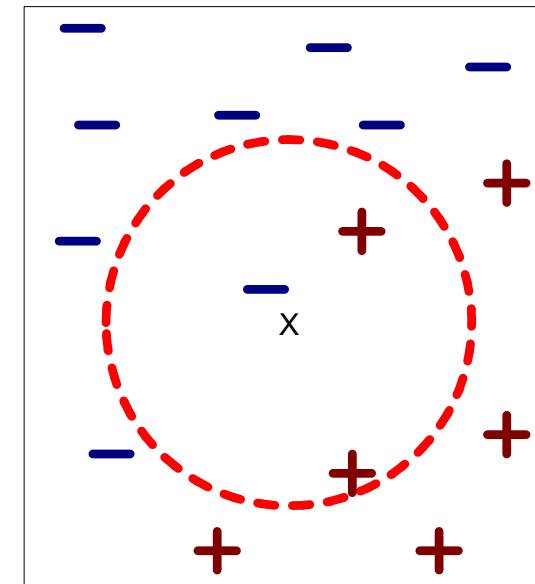
K-NN- choice of k



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

K- Nearest Neighbors of the new point x are the points that have the smallest distance to x

Lab work - Spam Classification

- Labelled data set : SpamAssassin Public Corpus
- Convert the email into a binary feature vector:
 - Clean (remove slash , dots, coms)
 - Tokenize (parse) into words
 - Count the word frequency
 - Create dictionary with most frequent words (e.g. 10000 to 50000 words)
- **Feature space.**
 - Substitute the words with the dictionary indices.
 - Extract binary features from emails - binary (sparse) feature for an email corresponds to whether the i-th word in the dictionary occurs in the email.
- **Apply classifier** (e.g. SVM)

Dictionary => Email with dictionary indices => binary features

1 aa
2 ab
3 abil
...
86 anyon
...
916 know
...
1898 zero
1899 zip

86 916 794 1077 883
370 1699 790 1822
1831 883 431 1171
794 1002 1893 1364
592 1676 238 162 89
688 945 1663 1120
1062 1699 375 1162
479 1893 1510 799
1182 1237 810 1895
1440 1547 181 1699
1758 1896 688 1676
992 961 1477 71 530
1699 531

$$x = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^n$$

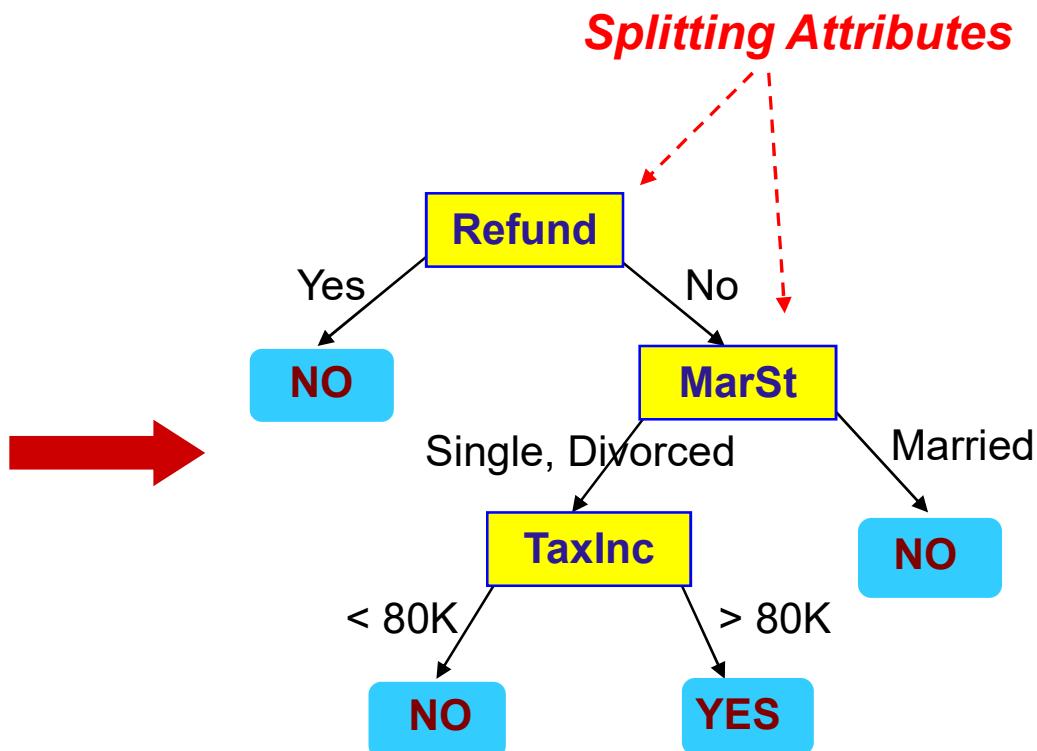
Classification –

Decision Trees

Classification by Decision Tree – model 1

Tid	Categorical				continuous class
	Refund	Marital Status	Taxable Income	Cheat	
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	

Training Data

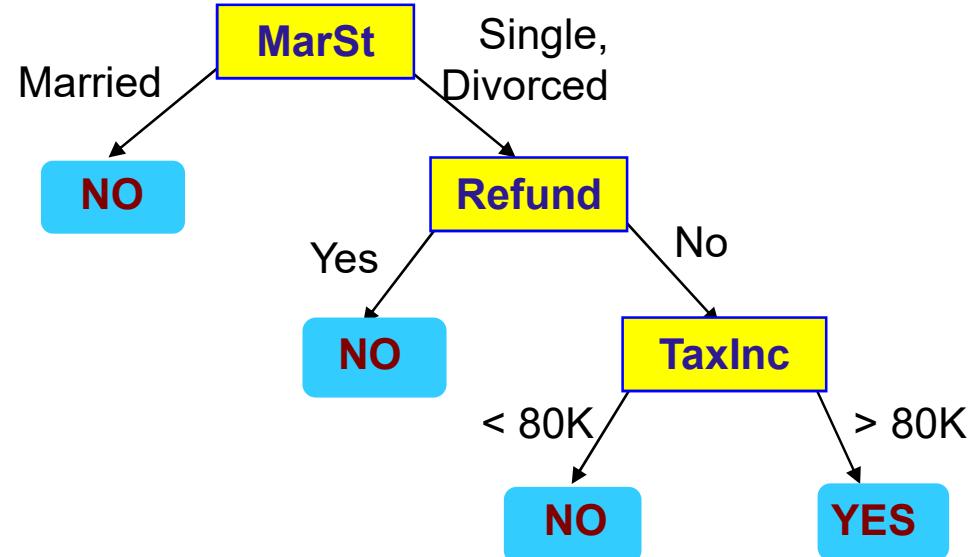


Model: Decision Tree

Classification by Decision Tree – model 2

Tid	Refund	Marital Status	Taxable Income	class	
				categorical	categorical
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	

Training Data



There could be more than one tree that fits the same data!

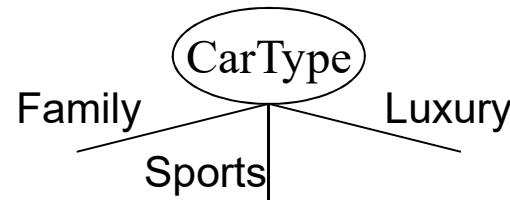
Decision Tree

Decision Tree has 2 basic type of nodes:

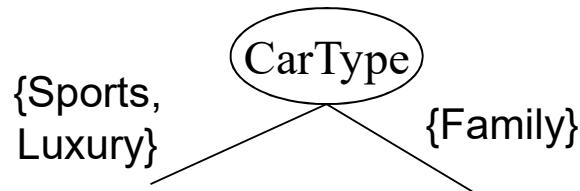
- **Leaf node** - a class label, determined by majority vote of training examples reaching that leaf.
- **Node** is a question on one feature. It branches out according to the answers.
 - **root node**: the first (top) node of the tree
 - **child node**: the next node to a current node

Splitting of Categorical Features

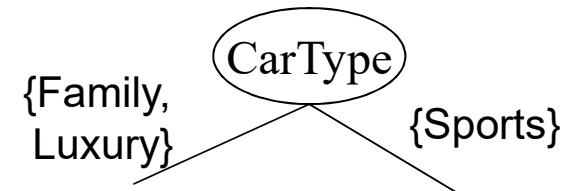
- **Multi-way split:** Use as many partitions as distinct values.



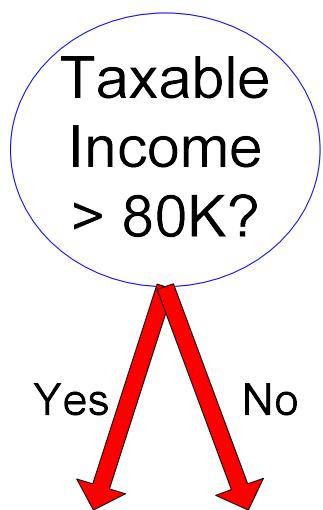
- **Binary split:** Divides values into two subsets.



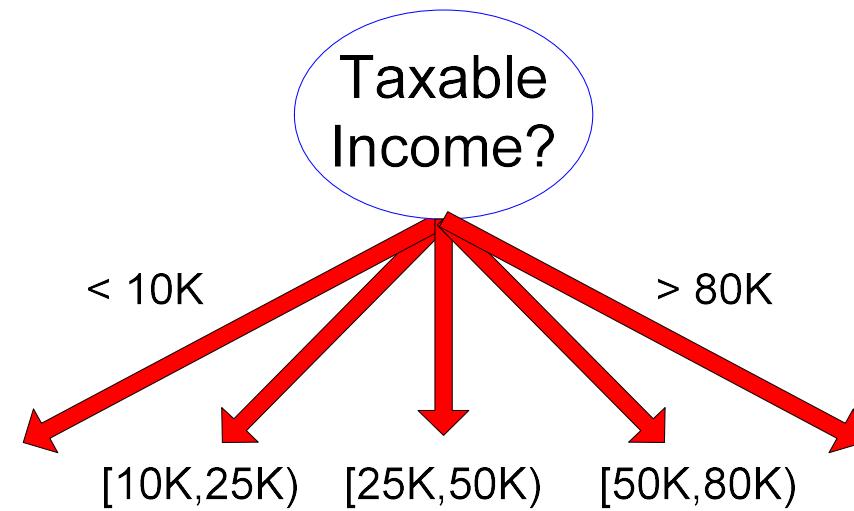
OR



Splitting of Continuous Features



(i) Binary split



(ii) Multi-way split

How to determine the best split ?

We want:

- To get the smallest tree
- Pure leaf nodes, i.e. all examples having (almost) the same class (ex. C0 or C1).
- Choose the feature that produces the “purest” (the most homogeneous) nodes
- We need a measure of node impurity (homogeneity).

C0: 5
C1: 5

Non-homogeneous,
High degree of impurity

C0: 9
C1: 1

Homogeneous,
Low degree of impurity

Measures of Node Impurity

- **Gini Index** at a given node:

$$GINI(node) = 1 - \sum_{Class_j} [p(Class_j | node)]^2$$

- **Classification error** at a given node:

$$Error(node) = 1 - \max_{Class_j} p(Class_j | node)$$

- **Entropy (H)** at a given node:

$$H(node) = - \sum_{Class_j} p(Class_j | node) \log p(Class_j | node)$$

$p(Class_j / node)$: probability of $Class_j$ at a given $node$

Computing GINI - example

$$GINI(node) = 1 - \sum_{Class_j} [p(Class_j | node)]^2$$

(*p* - probability)

C1	0
C2	6

$$p(C1) = 0/6 = 0 \quad p(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - p(C1)^2 - p(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$p(C1) = 1/6 \quad p(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$p(C1) = 2/6 \quad p(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

Computing Classification Error - example

$$Error(node) = 1 - \max_{Class_j} p(Class_j | node)$$

C1	0
C2	6

$$p(C1) = 0/6 = 0 \quad p(C2) = 6/6 = 1$$
$$\text{Error} = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	1
C2	5

$$p(C1) = 1/6 \quad p(C2) = 5/6$$
$$\text{Error} = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

C1	2
C2	4

$$p(C1) = 2/6 \quad p(C2) = 4/6$$
$$\text{Error} = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

Entropy



$p(\text{head})=0.5$
 $p(\text{tail})=0.5$
 $H=1$



$p(\text{head})=0.51$
 $p(\text{tail})=0.49$
 $H=0.9997$

- Entropy is a probabilistic measure of information uncertainty.
In DTree we use it to measure the purity of a node.
- The node is pure if it has only examples that belongs to one class
=> entropy $H = 0$, no uncertainty (this is what we want !)
- If the node has equal number of examples of all classes
=> entropy reaches maximum $H = 1$, the result is very uncertain.

Computing Entropy - example

$$H(node) = - \sum_{Class_j} p(Class_j | node) \log p(Class_j | node)$$

C1	0
C2	6

$$\begin{aligned} P(C1) &= 0/6 = 0 & P(C2) &= 6/6 = 1 \\ H &= -0 \log(0) - 1 \log(1) = -0 - 0 = 0 \end{aligned}$$

C1	1
C2	5

$$\begin{aligned} P(C1) &= 1/6 & P(C2) &= 5/6 \\ H &= -(1/6) \log_2(1/6) - (5/6) \log_2(1/6) = 0.65 \end{aligned}$$

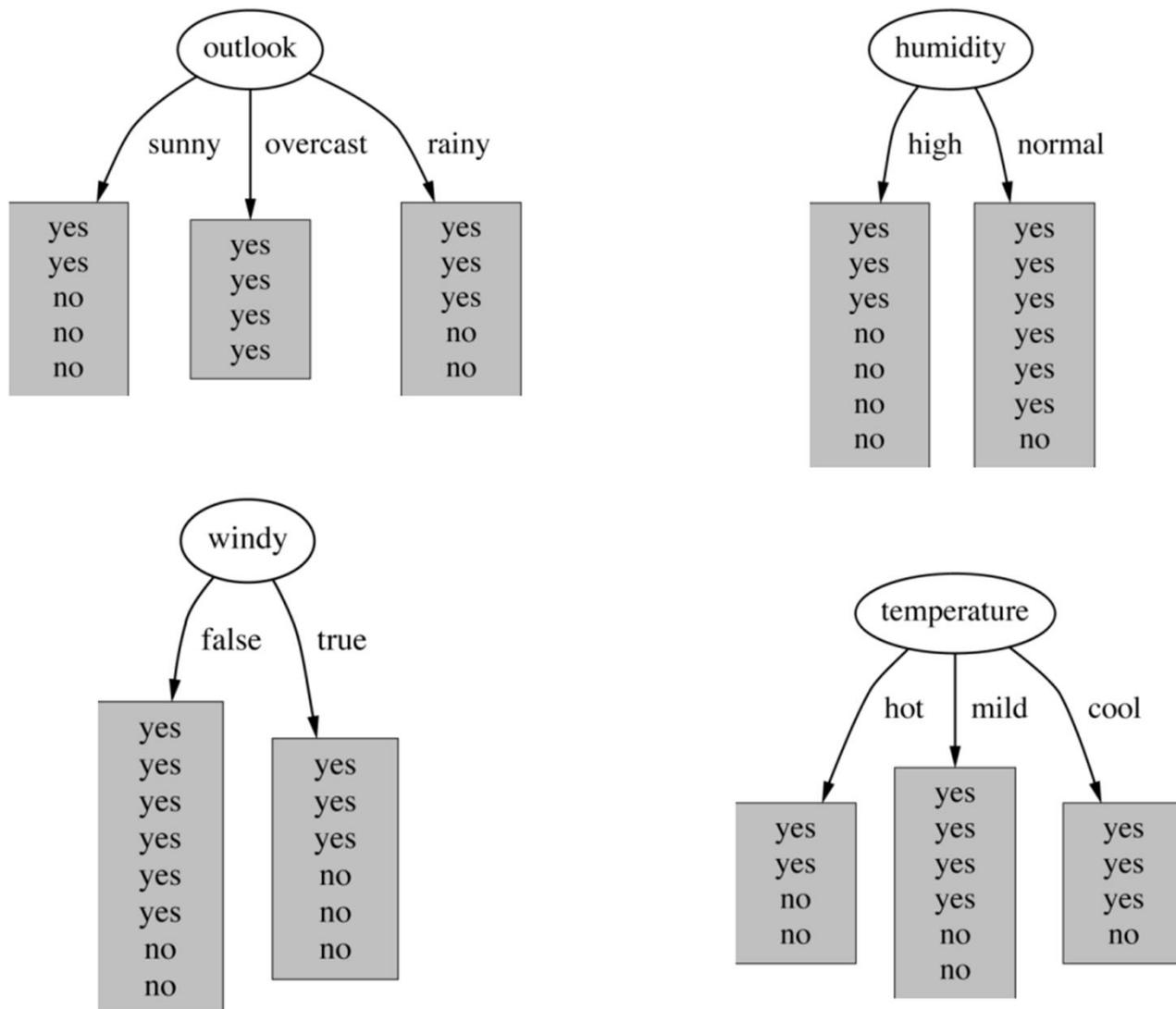
C1	2
C2	4

$$\begin{aligned} P(C1) &= 2/6 & P(C2) &= 4/6 \\ H &= -(2/6) \log_2(2/6) - (4/6) \log_2(4/6) = 0.92 \end{aligned}$$

Example – Weather data (Play golf or Not)

OUTLOOK	TEMP	HUMIDITY	WINDY	PLAY
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Which feature to select as root node?

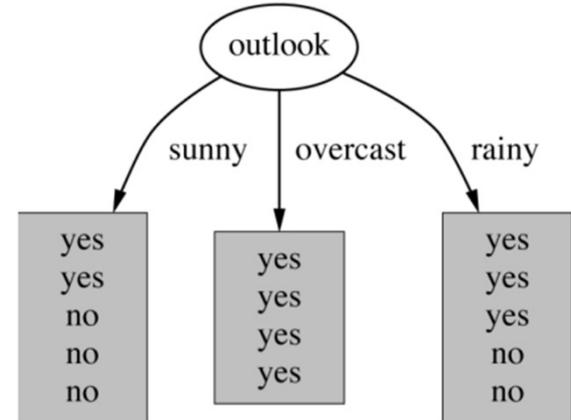


Information Gain

Information Gain = Entropy (H) before split -
Entropy (H) after split at one node (one feature).
How much uncertainty was reduced after splitting
on a given feature.

$$H(node) = - \sum_{Class_j} p(Class_j | node) \log p(Class_j | node)$$

Feature OUTLOOK:



$$H(\text{before split}) = -[(9/14)\log_2(9/14) + (5/14)\log_2(5/14)] = 0.9403$$

(14 example: 9 ex. class Yes ; 5 ex. class No)

$$H(\text{Sunny}) = -[(2/5)\log_2(2/5) + (3/5)\log_2(3/5)]$$

(5 ex. with Outlook=Sunny: 2 ex. class Yes; 3 ex. class No)

$$H(\text{Overcast}) = -[4/4\log_2(4/4)]$$

(4 ex. with Outlook=Overcast - 4 ex. class Yes ; 0 ex. class No)

$$H(\text{Rainy}) = -[(2/5)\log_2(2/5) + (3/5)\log_2(3/5)]$$

(5 ex. with Outlook=Rainy - 3 ex. class Yes; 2 ex. class No)

$$H(\text{after split}) = (5/14)H(\text{Sunny}) + (4/14)H(\text{Overcast}) + (5/14)H(\text{Rainy}) = 0.6935$$

Building a Decision Tree - example

Feature OUTLOOK: Information Gain = 0.2467

Feature TEMPERATURE : Information Gain = 0.029

Feature HUMIDITY: Information Gain = 0.157

Feature WINDY: Information Gain = 0.048

Choose feature that maximizes the information gain (minimizes the node impurity) !

Decision: Choose OUTLOOK as the root node.

This procedure is repeated for each node.

Splitting stops when data cannot be split any further or

the class is defined by the majority of elements of a subset.

Computer Security Career Paths

Path 1

**20
Years**

Computer Security Expert

Forensic Analyst

Forensic Lab Director

Chief Security Official

Highly-Paid Security Consultant

Path 2

**2
Years**
(14 months
with good
behavior)

Hacker

Criminal

Convict

Aprendizagem Aplicada à Segurança

(Mestrado em Cibersegurança-DETI-UA)



LECTURE 3

MODEL SELECTION AND VALIDATION – BIAS VS. VARIANCE

Petia Georgieva
(petia@ua.pt)

DETI/IEETA – UA

Outline

Model performance evaluation: perf. metrics

- **Model selection: Bias vs. variance**
- **Learning curves**
- **K -fold Cross Validation**

Performance Evaluation – Confusion Matrix

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Python: from sklearn.metrics import confusion_matrix

Performance metric - Accuracy

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	(TP)	(FN)
	Class>No	(FP)	(TN)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy - fraction of examples correctly classified.

1-Accuracy: Error rate (misclassification rate)

Limitation of Accuracy

- Consider binary classification (**Unbalanced data set**)
 - Class 0 has 9990 examples
 - Class 1 has 10 examples
- If model classify all examples as class 0, accuracy is $9990/10000 = 99.9\%$
- Accuracy is misleading metrics because model does not classify correctly any example of class 1
 - => Use other performance metrics.
 - => Find a way to balance the data set
(re-sampling methods: oversampling, under-sampling)

Other Performance Metrics

True Positive Rate (TPR), Sensitivity, Recall
of all positive examples the fraction of correctly classified

$$TPR = \frac{TP}{TP + FN}$$

True Negative Rate (TNR), Specificity
of all negative examples the fraction of correctly classified

$$TNR = \frac{TN}{TN + FP}$$

False Positive Rate (FPR) - how often an actual negative instance will be classified as positive, i.e. “false alarm”

$$FPR = 1 - TNR = \frac{FP}{FP + TN}$$

Precision - the fraction of correctly classified positive samples from all classified as positive

$$\text{Precision} = \frac{TP}{TP + FP}$$

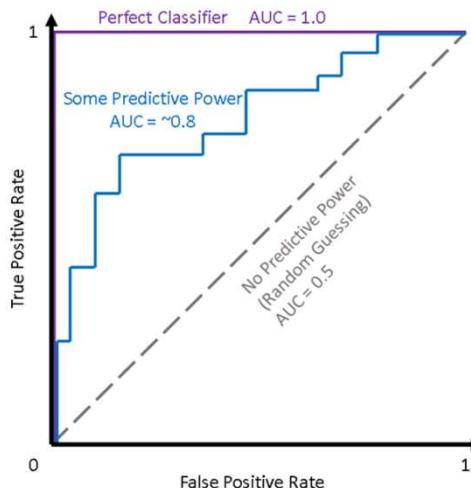
Combined performance metrics

F1 Score - weighted average of Precision and Recall

$$F1 = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

Balanced Accuracy= $(\text{Recall} + \text{Specificity}) / 2$

Receiver Operating Characteristic (ROC) curve



ROC curve: **True Positive Rate (TPR)** against **False Positive Rate (FPR)** for a binary classifier changing the **thresholds** between positive and negative.

For example, in logistic regression, if an observation is predicted to be > 0.5 , it is labelled as positive.

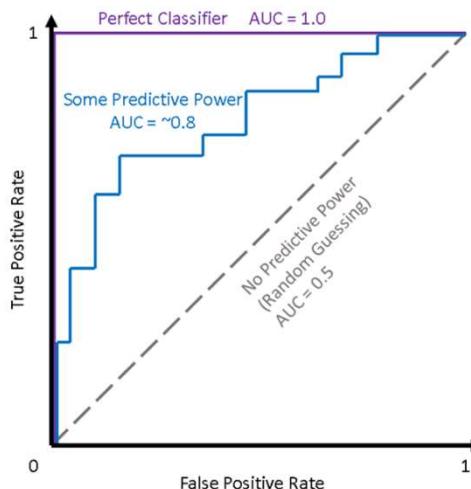
But, we can choose any threshold between 0 and 1 (0.1, 0.3, 0.6, 0.99, etc.).

ROC curves visualize how these choices affect classifier performance.

For multi K-class problem, draw K ROC curves.

Python: from sklearn.metrics import roc_curve

Area Under the (ROC) Curve - AUC



ROC curve is useful for visualization, but it's good to have also a single metric => AUC.
The higher the AUC score, the better a classifier performs for the given task.
For a classifier with no predictive power (i.e., random guessing) => AUC = 0.5.
For a perfect classifier => AUC = 1.0.
Most classifiers fall between 0.5 and 1.0.

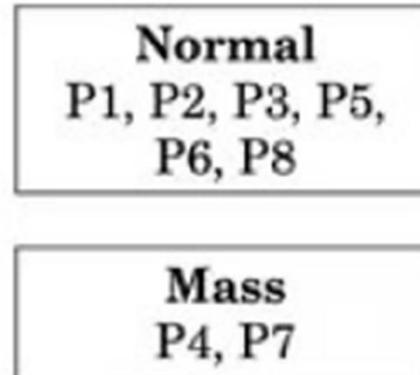
Python: from sklearn.metrics import auc

Class Imbalance problem

Solution: Re-sampling methods (under-sampling, oversampling)

Examples

P1 Normal
P2 Normal
P3 Normal
P4 Mass
P5 Normal
P6 Normal
P7 Mass
P8 Normal



Re-Sampled

P3 Normal
P6 Normal
P1 Normal
P8 Normal
P7 Mass
P4 Mass
P7 Mass
P4 Mass

Sample 4 →

Sample 4 →

Definitions for Epoch /Batch Size / Iterations / Train step

One Epoch is when an ENTIRE dataset is passed through the model (e.g. forward and backward in a neural network) only ONCE.
If data is too big to feed to the computer at once one epoch is divided in several smaller batches.

Batch Size: Total number of training examples present in a single batch.

Iterations is the number of batches needed to complete one epoch.

Example: Let's say we have 2000 training examples.
We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.

Training run/step - is one update of the model parameters.
We update the parameters after one batch or after one epoch.

Deciding what to do next ?

Suppose you have trained a ML model on some data. When you test the trained model on a new set of data, it makes unacceptably large errors. What should you do ?

- **Get more training examples ?**
- **Try smaller sets of features (feature selection) ?**
- **Try getting additional features (feature engineering) ?**
- **Try using different/nonlinear kernels ?**
- **Try other values of the hyper parameters (e.g. regul. parameter) ?**

Run tests to gain insight what isn't working with the learning algorithm and how to improve its performance.

Diagnostics is time consuming , but can be a very good use of your time.

Simplest division: Train & Test subsets

- Training set (70%-80 %) : used to train the model
- Test set (30%-20%) : used to test the trained model

- **Optimize the model parameters with training data**
(minimize some cost/loss function J)

After the training stage is over (i.e. the cost function J converged)

- Compute the MSE on test data (for regression problems)

$$E_{test}(\theta) = \frac{1}{m_{test}} \left[\sum_{i=1}^{m_{test}} \left(h_{\theta}(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2 \right]$$

or

- Compute the model accuracy or some other metric from the confusion matrix, on test data (for classification problems)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3 way split: Train/Dev/Test Sets

Choose ML model: Logit, SVM, K-NN, etc. ?

Choose model hyper-parameters:

- What is the best learning rate ?
- What is the best regularization parameter (λ) ?
- What is the best polynomial degree ?
-

Divide dataset in 3 sub-sets:

- Training set
- Cross Validation (CV) set = Development set = ‘dev’ set
- Test set

Traditional division for Small data set (up to 10000 examples) :

60% - 20% - 20%

Big data (1 million. examples): 98% - 1% - 1%

Model /hyper parameter selection

Step 1: Optimize parameters θ (to minimize some cost function J) using the same training set for all models. Compute some perf. metrics with the training data (i.e. error, accuracy) :

Training error =>
$$E_{train}(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \right]$$

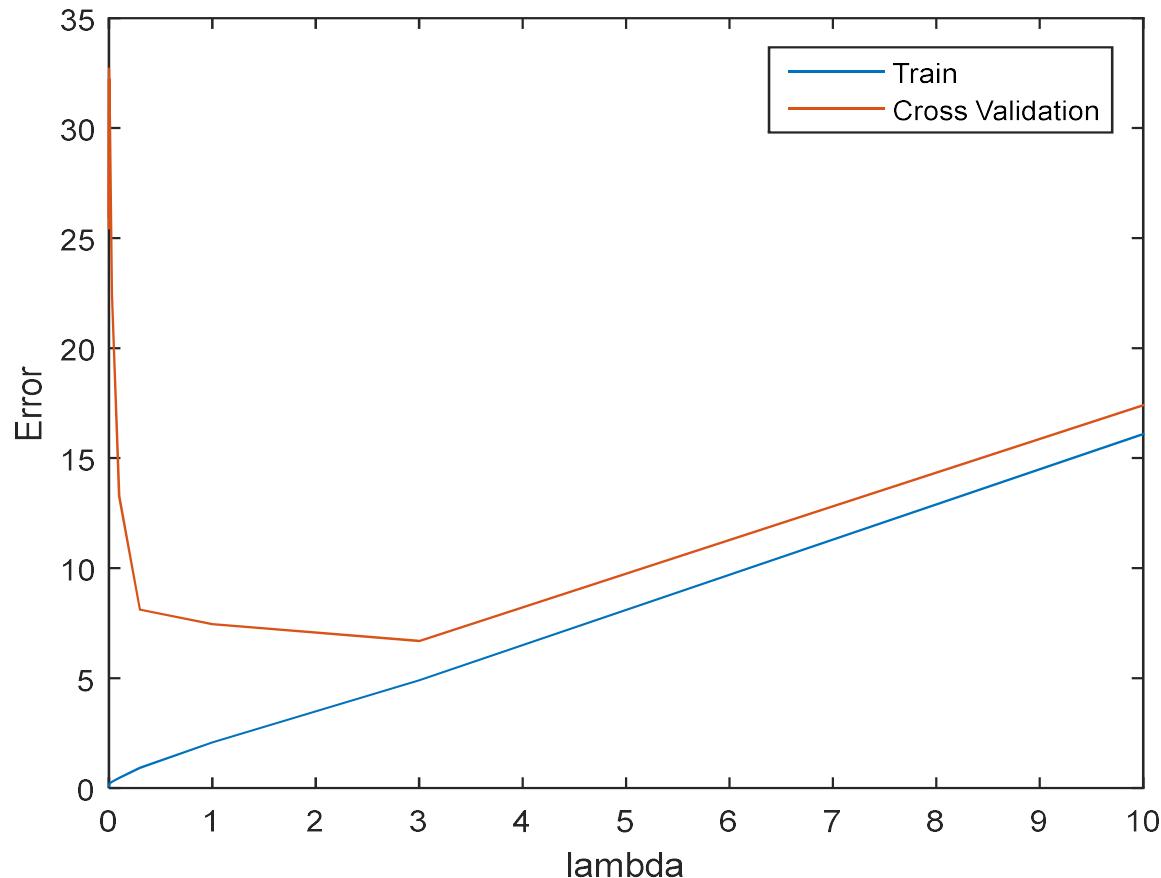
Step 2: Test the optimized models from step 1 with the CV set and choose the model with the min CV error (or other performance metric with dev data):

Cross validation (CV)/dev error =>
$$E_{cv}(\theta) = \frac{1}{2m_{cv}} \left[\sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2 \right]$$

Step 3: Retrain the best model from step 2 with both train and CV sets starting from the parameters got at step 2. Test the retrained model with test set and compute test data perf. metric (***the real model performance !!!***):

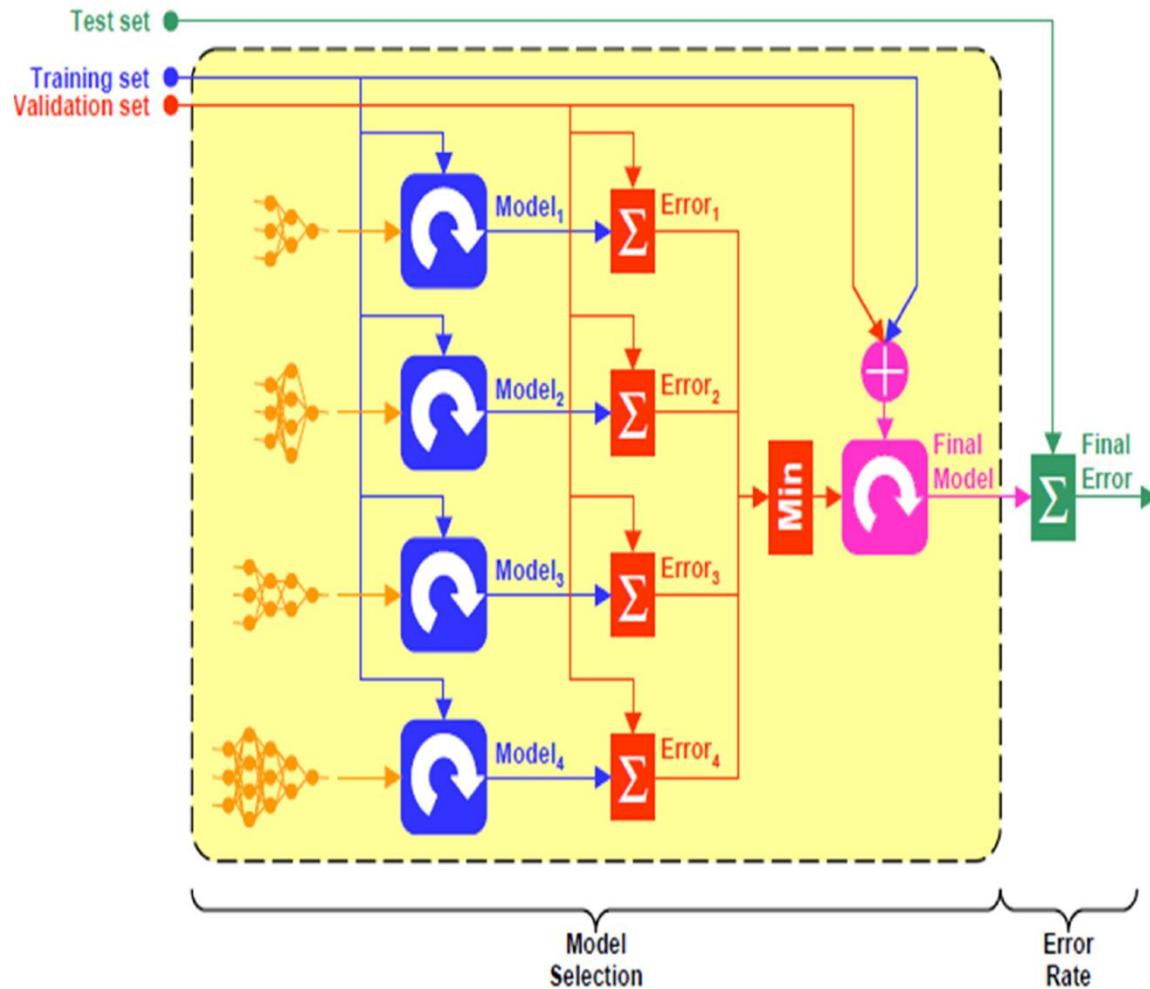
Test error =>
$$E_{test}(\theta) = \frac{1}{2m_{test}} \left[\sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2 \right]$$

Example: Select best λ



Best $\lambda = 3$

Training/Valid (Dev)/Test subsets



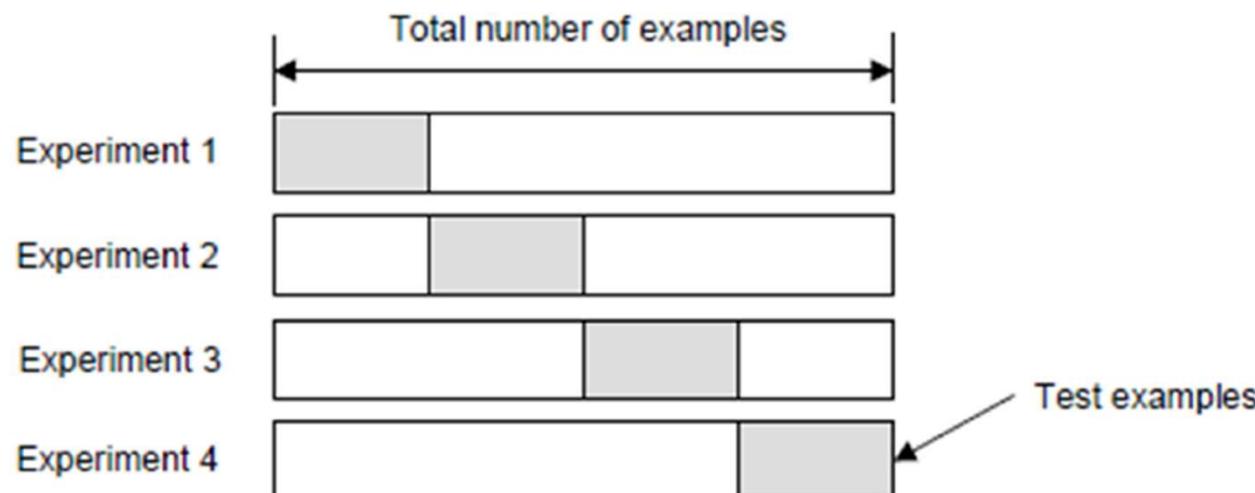
1

**The most credible is the performance metric with test data,
not used for training or validation of the model.**

K -fold Cross Validation

- Divide data into Training and Test subsets.
- Divide Training data into K subsets (K-fold).
- Use K-1 subsets for training and the remaining subset for CV.
- The final validation error is the average CV error of K experiments.
- Choose the best model /hyper-parameter the one that minimise the average CV error.

$$E_{cv} = \frac{1}{K} \sum_{i=1}^K E_{testi}$$

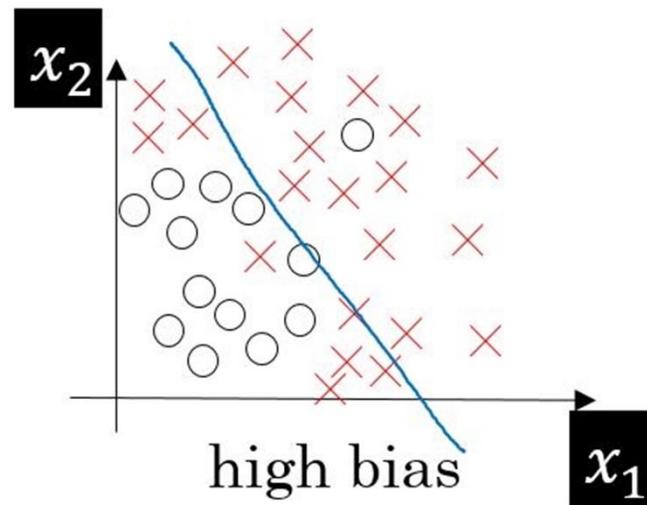


Bias vs. Variance

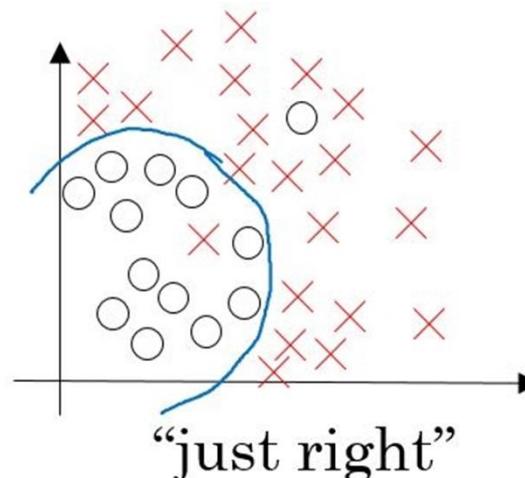
An important concept in ML is the bias-variance tradeoff.

Models with **high bias** are not complex enough and **underfit** the training data.

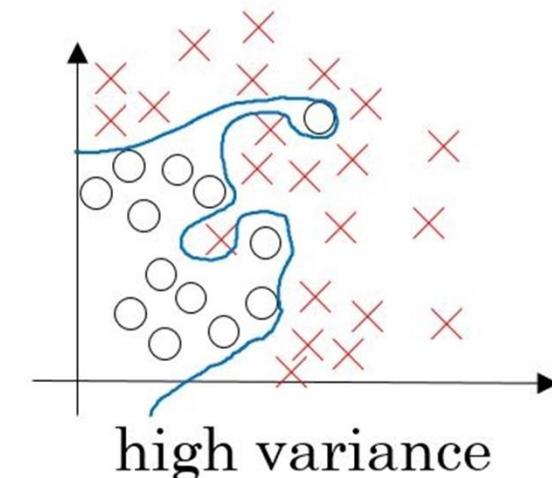
Models with **high variance** are too complex and **overfit** the training data.



underfitting data
(very simple model)



(good model)



overfitting data
(very complex model)

Diagnosing Bias vs. Variance

How to diagnose if we have a high bias problem or high variance problem ?

High Bias (underfitting) problem:

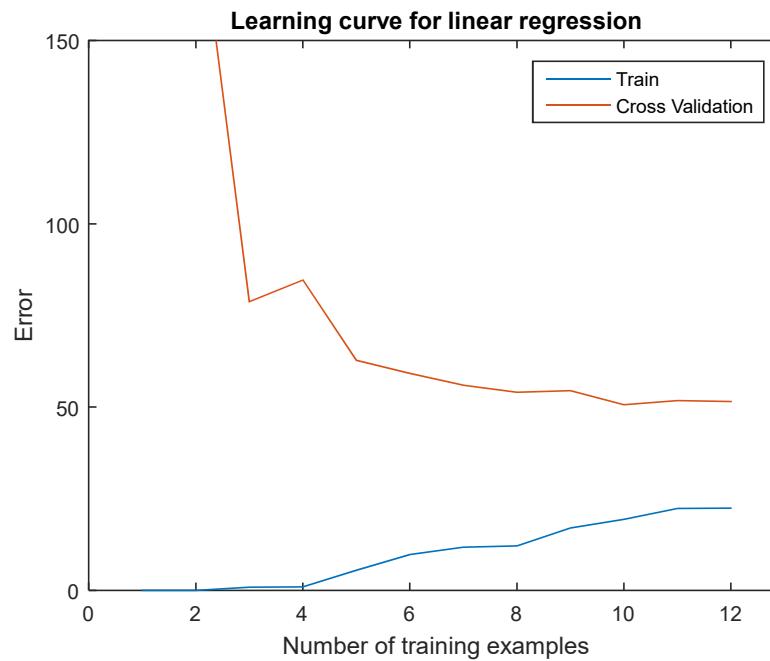
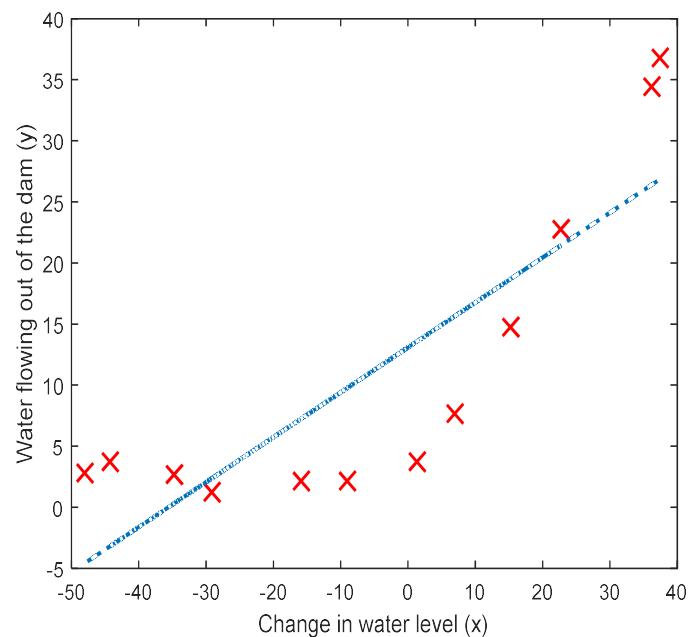
Training error (E_{train}) and Validation/dev error (E_{cv}) are both high

High Variance (overfitting) problem:

Training error (E_{train}) is low
and Validation/dev error (E_{cv}) is much higher than E_{train}

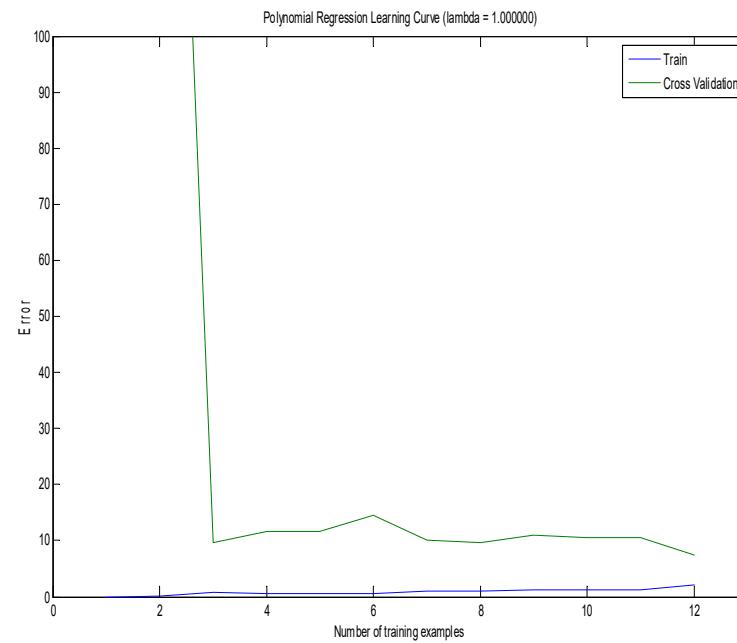
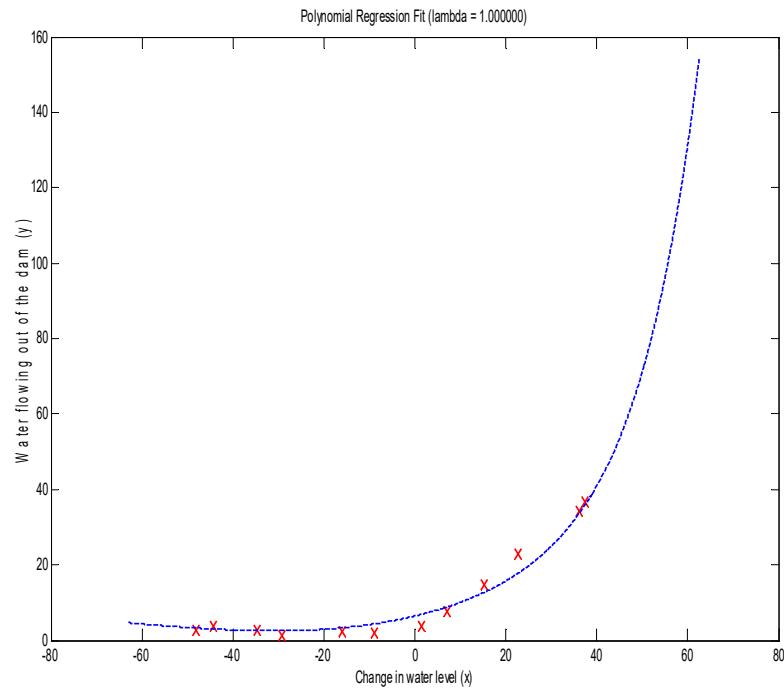
Learning Curves

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



**If a learning algorithm is suffering from high bias,
getting more training data will not help much**

Learning Curves



**If a learning algorithm is suffering from high variance,
getting more training data is likely to help**

Hints to improve ML model

Suppose you have learned a data model (hypothesis). However, when you test your hypothesis on a new set of data, you find that it makes unacceptably large errors in its prediction (regression or classification). What should you try next?

- **Get more training examples – fixes high variance**
- **Try smaller sets of features – fixes high variance**
- **Try getting additional features – fixes high bias**
- **Try adding polynomial features - fixes high bias**
- **Try decreasing λ – fixes high bias**
- **Try increasing λ – fixes high variance**

Aprendizagem Aplicada à Segurança

(Mestrado em Cibersegurança-DETI-UA)



LECTURE 4

Anomaly detection

Petia Georgieva
(petia@ua.pt)

DETI/IEETA – UA

Outline

- **Univariate Gaussian Distribution**
- **Anomaly Detection Algorithm**
- **Multivariate Gaussian Distribution –**
- **Covariance Matrix**

Popular applications of anomaly detection

Fraud detection - $x^{(i)}$ – vector of the following features of user i

x_1 - how often does the user login

x_2 # of web pages visited / # of transactions

x_3 - the typing speed of the user

How to identify suspicious computer users ?

Matrix X	feature x_0	feature x_1	feature x_n
User1 /Motor1	I	$x^{(1)}$		$x_n^{(1)}$
User2 /Motor2	I	$x^{(2)}$		$x_n^{(2)}$
	I			
User i /Motor i				$x_n^{(i)}$
User m /Motor m	I	$x^{(m)}$		$x_n^{(m)}$

Manufacturing (e.g. aircraft engine features)

x_1 = heat generated

x_2 =vibration intensity,

x_3, x_4, \dots other features

How to identify anomalous production (engines) for quality control ?

Monitoring computers in data center: $x^{(i)}$ = features of machine i

x_1 =memory use of computer

x_2 =number of disc accesses / sec

x_3 =CPU load

x_4 =network traffic &....other features...

How to identify if a computer is doing something strange and further inspect it?

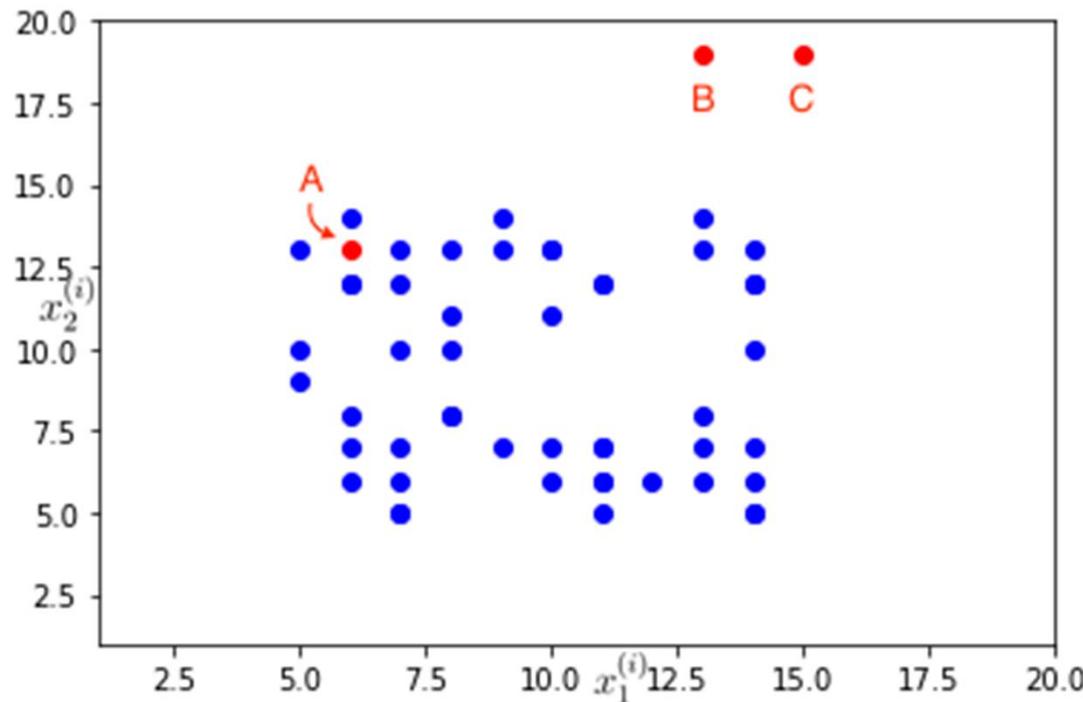
Anomaly detection

We have a dataset of examples (blue points): $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

Each example has two features x_1 and x_2 .

We get new examples (red points): A, B, C

How to decide that A is not an outlier, B and C are anomalous (outliers) ?



Anomaly detection – How ?

From given regular (not anomalous) data get a model of what is considered as normal. For example – a probability model $p(x)$.

Identify anomalous case by checking if

$$p(x_{test}) < \epsilon \text{ (flag as anomaly)}$$

$$p(x_{test}) \geq \epsilon \text{ (flag as normal)}$$

Gaussian (Normal) distribution

If $x \in \mathbb{R}$, and x follows Gaussian distribution with mean, μ and variance σ^2 , denoted as,

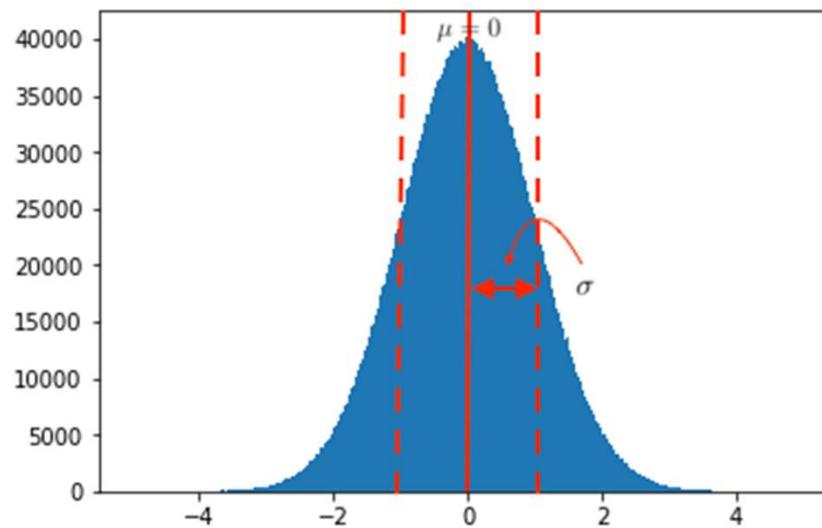
$$x \sim \mathcal{N}(\mu, \sigma^2)$$

Standard normal Gaussian distribution ($\mu=0$, standard deviation $\sigma=1$).

Density is higher around μ and reduces as distance from mean increases.

If we know parameters μ and σ , the probability of x in Gaussian distribution is:

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

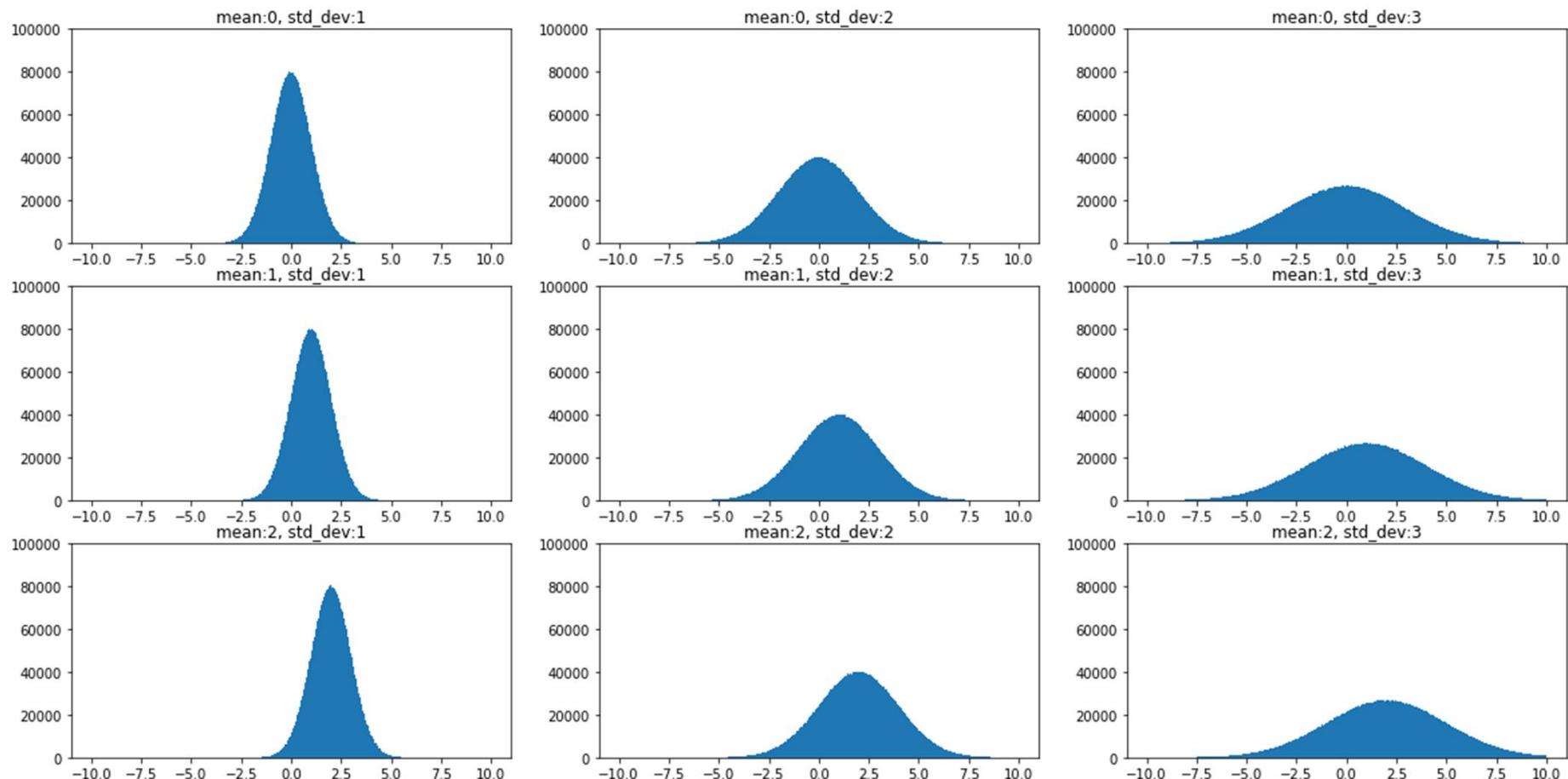


Effect of μ and σ on Gaussian curve

Mean (μ) defines the centering of the distribution.

Standard deviation (σ) defines the spread of the Gaussian distribution.

As the spread increases the height of the plot decreases, because the total area under a probability distribution should be = 1.



Parameter estimation of Gaussian curve

Parameters (μ and σ) of the Gaussian distribution are estimated based on given data

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)^2$$

Density Estimation Algorithm

Given m training examples =>

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

Each example i has n features =>

$$\{x_2^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}\}$$

Major assumptions:

The features have Gaussian distribution and are independent.

Compute μ and σ of each feature.

Compute $p(x_j)$ of each feature.

$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

$$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

:

$$x_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$$

:

$$x_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

$$p(x) = p(x_1; \mu_1, \sigma_1^2)p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

Anomaly Detection Algorithm

1. Choose features that you think might be indicative of anomalous examples.
2. Fit Gaussian distribution parameters (μ and σ) for each feature.

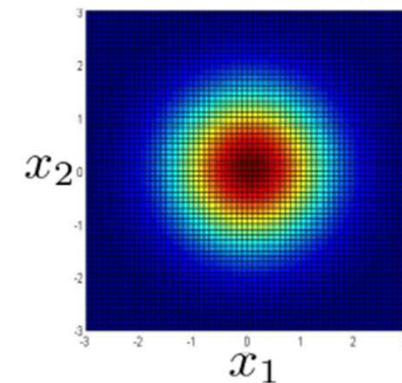
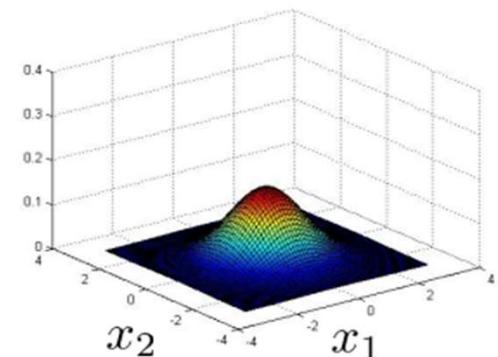
$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

3. Given new example (x_{new}), compute $p(x_{\text{new}})$

4. Anomaly if $p(x_{\text{new}}) < \epsilon$ (some threshold)

x_1, x_2 - features

z axis is the probability density function



Evaluation of Anomaly Detection System

Let we have some *labelled data*, of many normal examples and a much less anomalous examples.

Train set (take ONLY normal examples !!!): $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

Cross validation (CV) set (normal & anomalous exs.):

$$(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

Test set (normal & anomalous exs.): $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Ex. Manufacturing (e.g. aircraft engine features)

10000 good (normal) engines & 20 anomalous engines

Train set (only good examples): 6000 good engines ($y=0$)

CV set: 2000 good engines ($y=0$), 10 anomalous ($y=1$)

Test set: 2000 good engines ($y=0$), 10 anomalous ($y=1$)

Evaluation of Anomaly Detection System

1) Fit $p(x)$ on train set (ONLY normal examples)

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

2) Check the performance on cross-validation (CV) set.

From a range of possible values of ϵ , choose the best threshold (ϵ) to optimize some performance metric.

$$p(x_{\text{CV}}) < \epsilon \text{ (anomaly)}$$

$$p(x_{\text{CV}}) > \epsilon \text{ (normal)}$$

Possible performance metrics:

- True positive, true negative, false positive, false negative
- Precision/Recall
- F1-score

Accuracy is not a good perf. measure because data is unbalanced (much more normal examples than anomalous).

3) Test the final model on test set.

$$p(x_{\text{test}}) < \epsilon \text{ (anomaly)}$$

$$p(x_{\text{test}}) > \epsilon \text{ (normal)}$$

Anomaly Detection vs. Supervised Learning

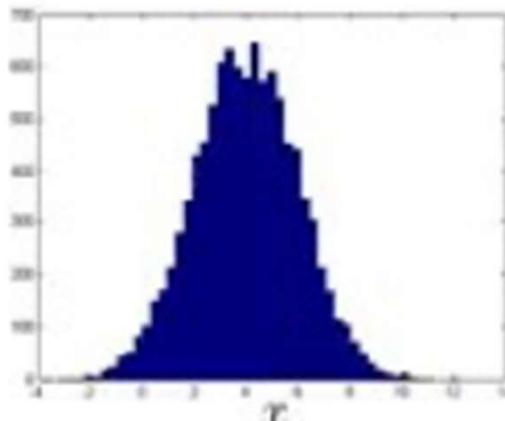
- Very small number of positive (anomalous) examples.
- Large number of negative (normal) examples.
- Many different “types” of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like.
- Future anomalies may look nothing like any of the anomalous examples seen before.
- Usually large number of both positive and negative examples
- Enough positive examples for the algorithm to get a sense of what positive examples are like.
- Future positive examples likely to be similar to ones in training set.

Exs.: Spam/fishing email classification; cancer prediction/classification

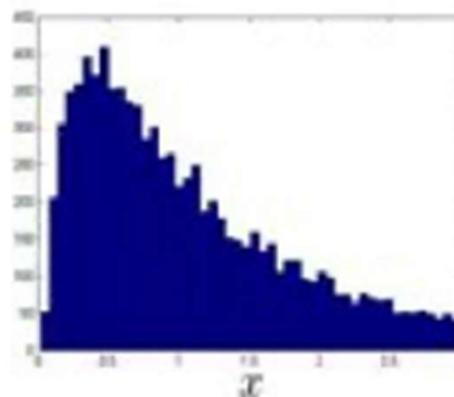
Exs.: fraud detection, monitoring computers in data centers, suspicious computer;

Choosing Features

If features have Gaussian distribution =>
apply the anomaly detection algorithm .



If features do not have Gaussian distribution =>
apply some transformation of data to get close to Gaussian curve.

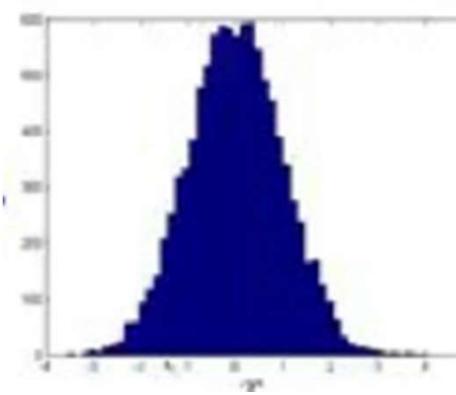
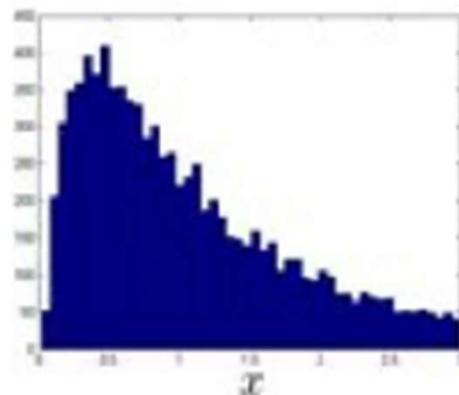


Choosing Features

Popular feature transformations =>

- $\log(x)$
- $\log(x + c)$
- \sqrt{x}

for example: $x \Rightarrow \log(x) \Rightarrow$ much more Gaussian curve



Error Analysis for Anomaly Detection

Want: $p(x)$ large for normal examples x ;
 $p(x)$ small for anomalous examples x .

Most common problem:

$p(x)$ is comparable (say both large) for normal and anomalous examples.

Solution: Make error analysis to create new features

Look at the anomalies the algorithm did not flag correctly (the mistakes) and try to create some new feature that may take unusually different (large or small) values in the event of anomaly and thus distinguish the abnormal ex.

Ex. Monitoring computers in data center

x_1 =memory use of computer

x_2 =number of disc accesses /sec

x_3 =CPU load

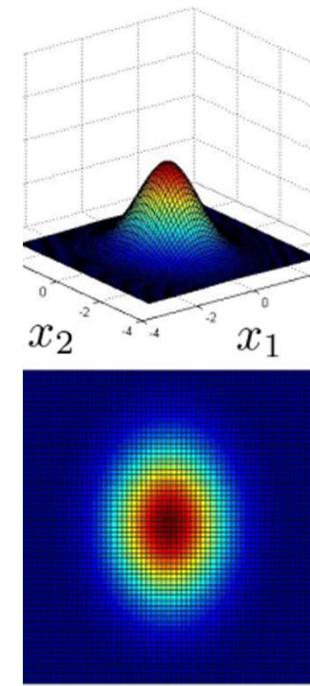
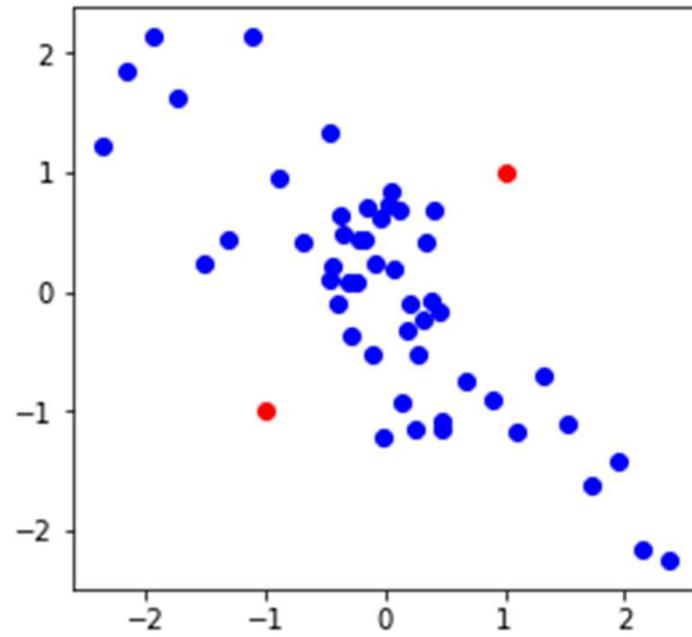
x_4 =network traffic

Feature engineering (new features)

x_5 =CPU load /network traffic

x_6 = $(\text{CPU load})^2$ /network traffic

Multivariate Gaussian Distribution

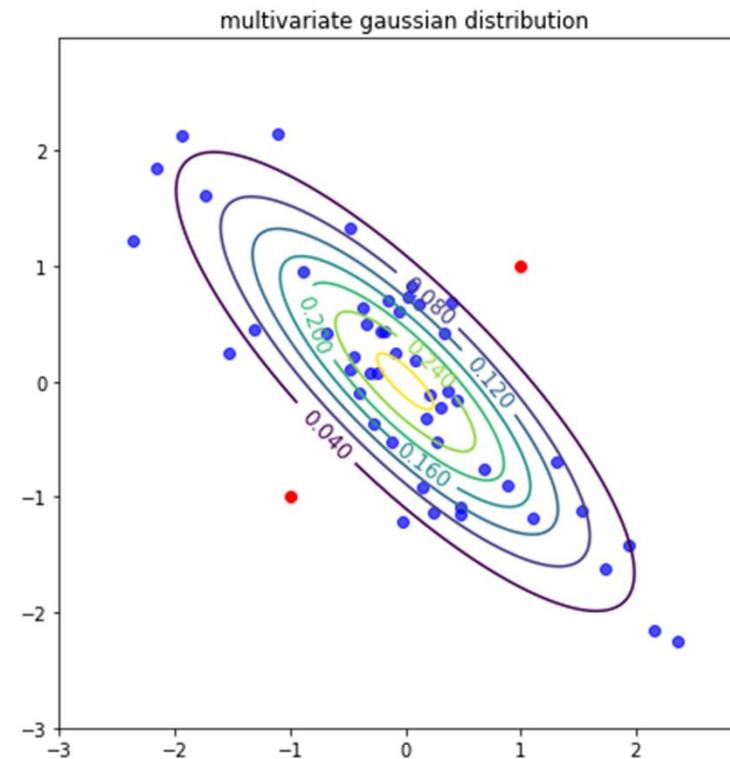
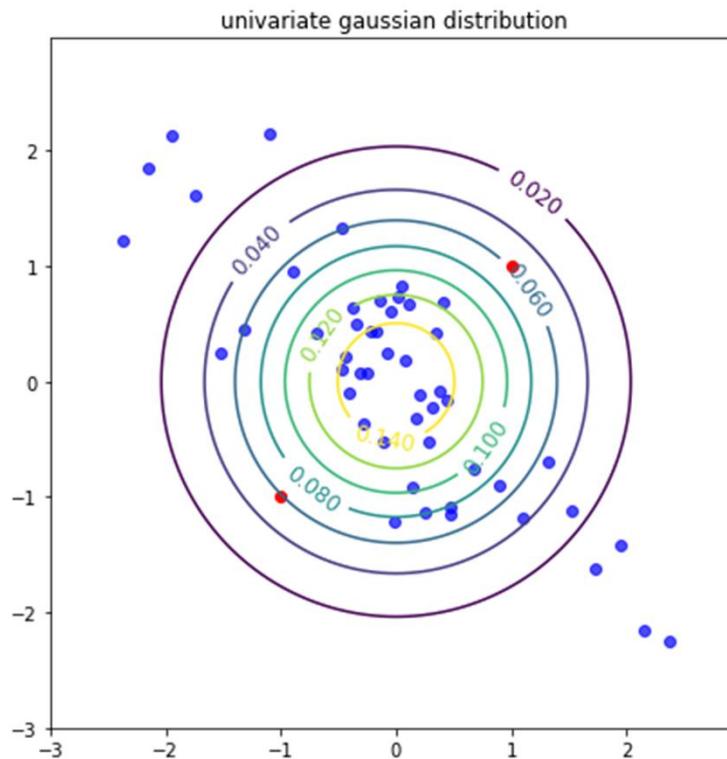


If we use univariate Gaussian distribution for this data, the contour plots (curves with the same probability value) will be circles (if both variances are the same) or axes aligned ellipses (if the variances are different). The red points will have relatively high probability => not flagged as outliers.

But features x_1 and x_2 are negatively correlated (one increases, the other decreases). The assumption of independance is violated.

Red points are outliers.

Univariate vs. Multivariate Gaussian Distribution



Univariate Gaussian distribution considers separately probability models for each $p(x_1)$, $p(x_2)$ => it will not flag the red points as outliers.
Better use Multivariate Gaussian distribution.

Multivariate density estimation

Given training data, estimate μ (nx1 vector) and **Σ (nxn covariance matrix)**:

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

For a new example x , compute:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

Flag as anomaly if $p(x) < \epsilon$.

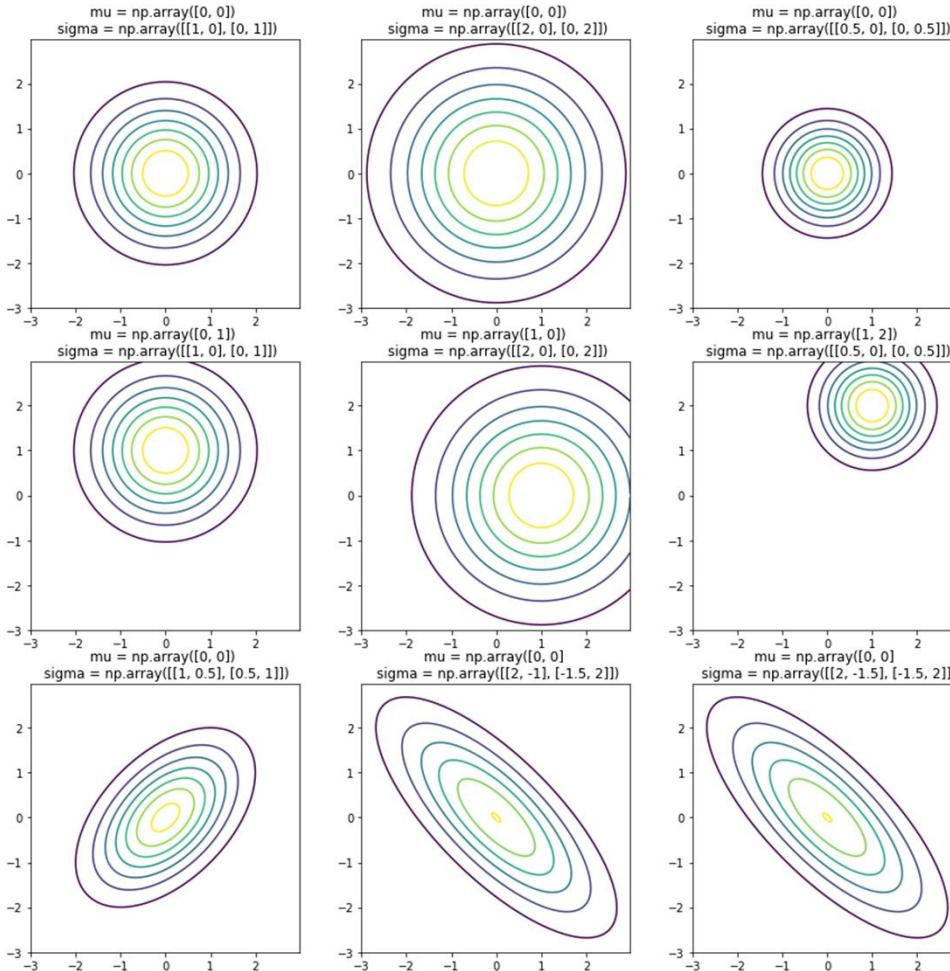
Σ – nxn covariance matrix

$|\Sigma|$ - determinant of matrix Σ .

Σ - symmetric about the main diagonal.

Σ - major difference between univariate and multivariate Gaussian !!!

Effect of Mean and Covariance Matrix Shifting



μ shifts the center of the distribution.

Diagonal elements of Σ vary the spread of the distribution along the corresponding features

Off-diagonal elements of Σ show the correlation among the features:

Positive off-diagonal values of Σ
=> positive correlation

Negative off-diagonal values of Σ
=> negative correlation

Univariate Gaussian distribution is a special case when off-diagonal values of Σ are 0.

Original vs. Multivariate Gaussian models for Anomaly detection

Multivariate Gaussian

Gaussian model with independent features

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)^2$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

$$p(x) = p(x_1; \mu_1, \sigma_1^2)p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

Manually create features to capture anomalies
(e.g. $x_{\text{new}}=\text{CPU load / network traffic}$)

Computationally cheaper, scales better to large number of features (n)



$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

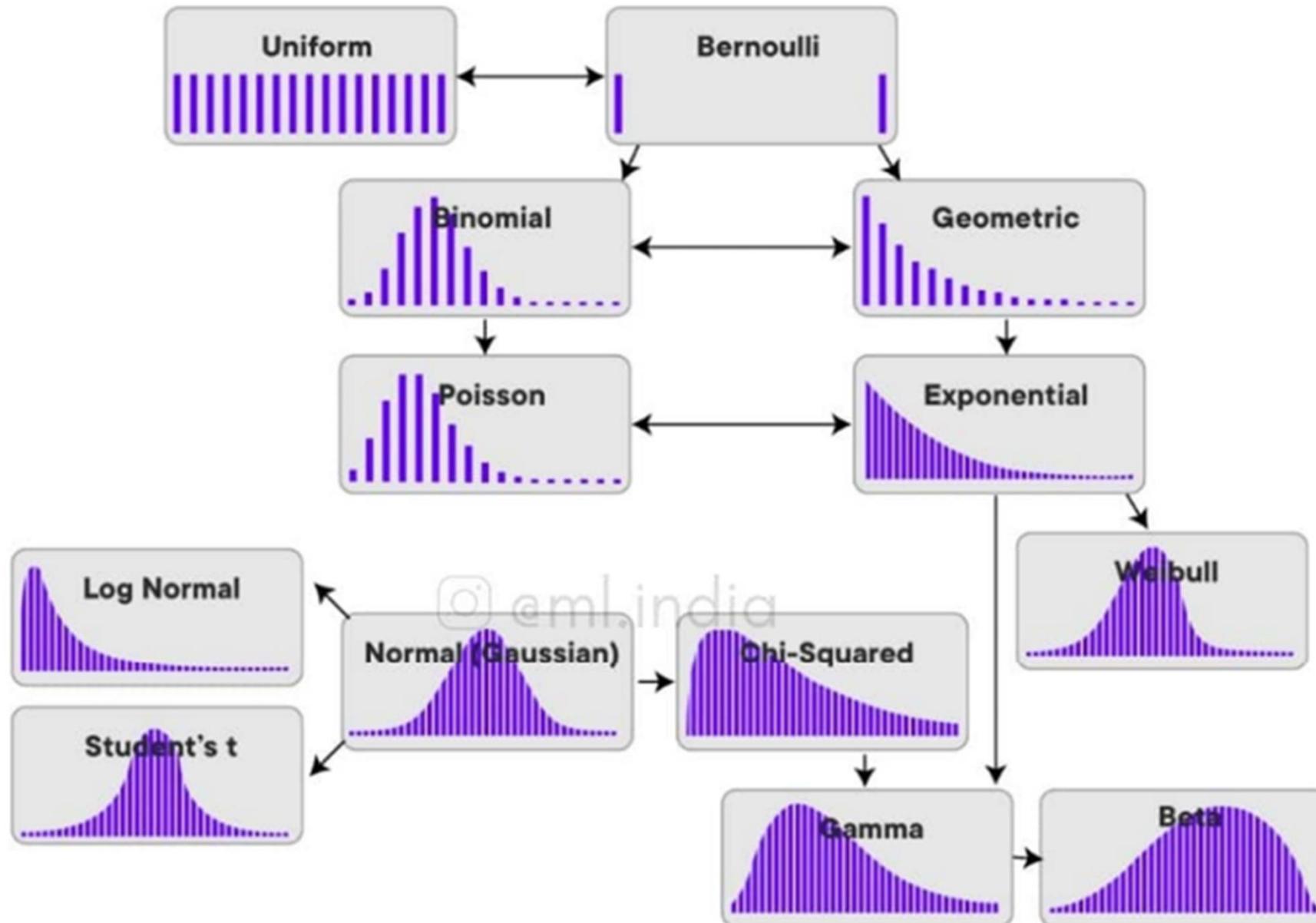
$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

Automatically captures correlations between features

Computationally more expensive, takes time to compute inverse of Σ if number of features (n) is large

Must have training set size (m) >> number of features (n), otherwise Σ is singular and not invertible.
In practice $m > 10 * n$

Distribution Types



Aprendizagem Aplicada à Segurança

(Mestrado em Cibersegurança-DETI-UA)



LECTURE 5

Unsupervised Learning

(K-means Clustering and PCA)

Petia Georgieva
(petia@ua.pt)

DETI/IEETA – UA

Outline

Unsupervised learning

1. K-means clustering

2. Data dimensionality reduction

- data compression / data visualization

3. Principal Component Analysis (PCA)

SUPERVISED vs. UNSUPERVISED LEARNING

Supervised Learning - (given DATA + LABELS):

ML method is trained with labeled data to predict the labels of new examples (learning by labeled examples)

Matrix X	feature x_n	feature x_1	feature x_n	Vector y - output (label)
Example 1	1	$x^{(1)}$		$x_n^{(1)}$	$y^{(1)}$
Example 2	1	$x^{(2)}$		$x_n^{(2)}$	$y^{(2)}$
	1				
				$x_n^{(i)}$	
Example m	1	$x^{(m)}$		$x_n^{(m)}$	$y^{(m)}$

Unsupervised Learning - given UNLABELED DATA

ML method to discover the data internal (statistical) structure

Matrix X	feature x_1	feature x_2	feature x_n
Example 1	$x_1^{(1)}$	$x_2^{(1)}$		$x_n^{(1)}$
Example 2	$x_1^{(2)}$	$x_2^{(2)}$		$x_n^{(2)}$
...				
Example i	$x_1^{(i)}$	$x_2^{(i)}$		$x_n^{(i)}$
...				
...				
Example m	$x_1^{(m)}$	$x_2^{(m)}$		$x_n^{(m)}$

Unsupervised learning -

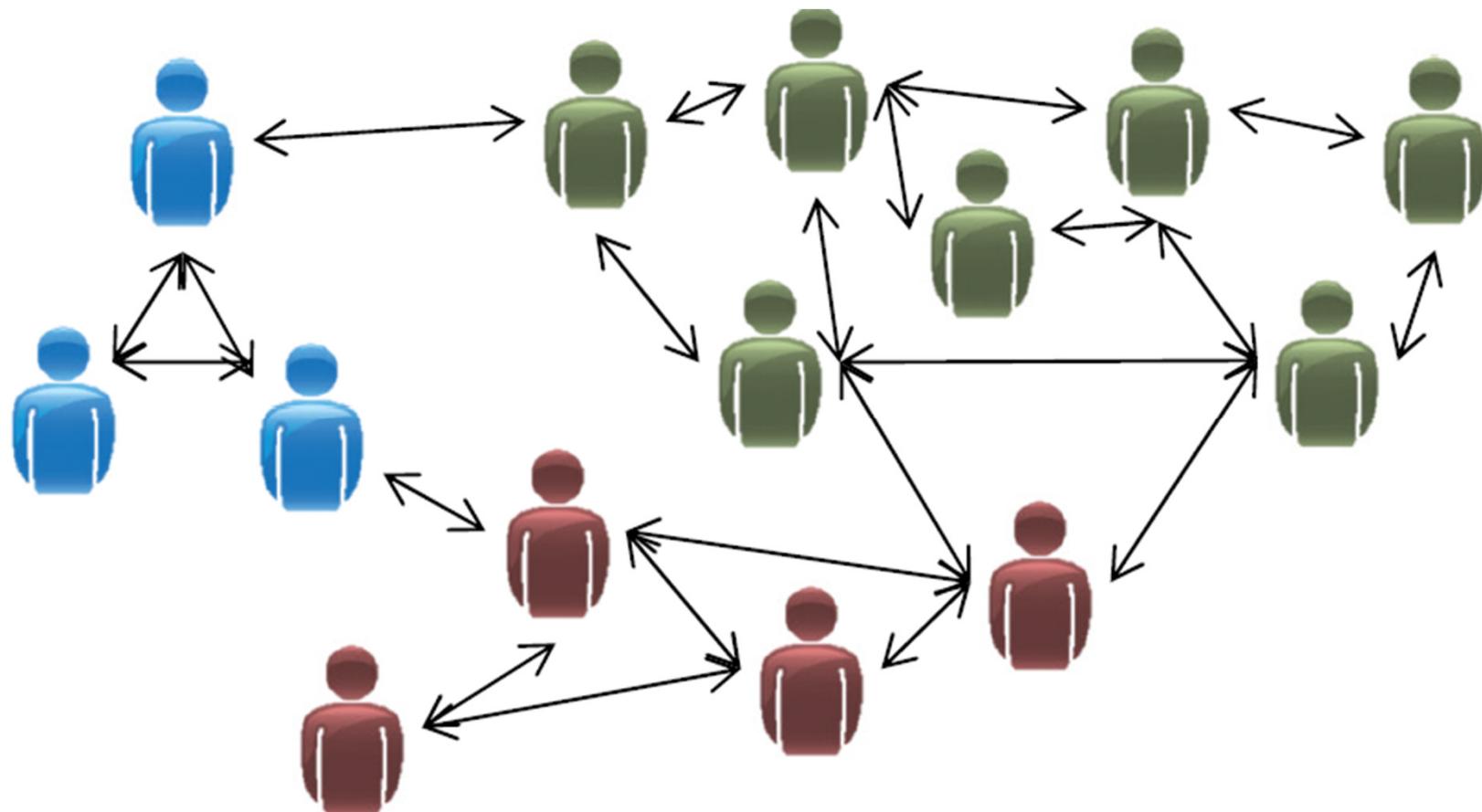
Market segmentation: data base of customers => division in target groups

Features: education, job, age, marital status, etc.

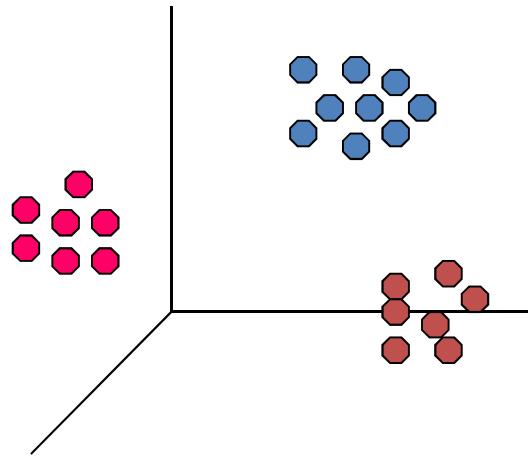


Unsupervised learning

Social network analysis: user grouping, group-specific advertising



Clustering intuition

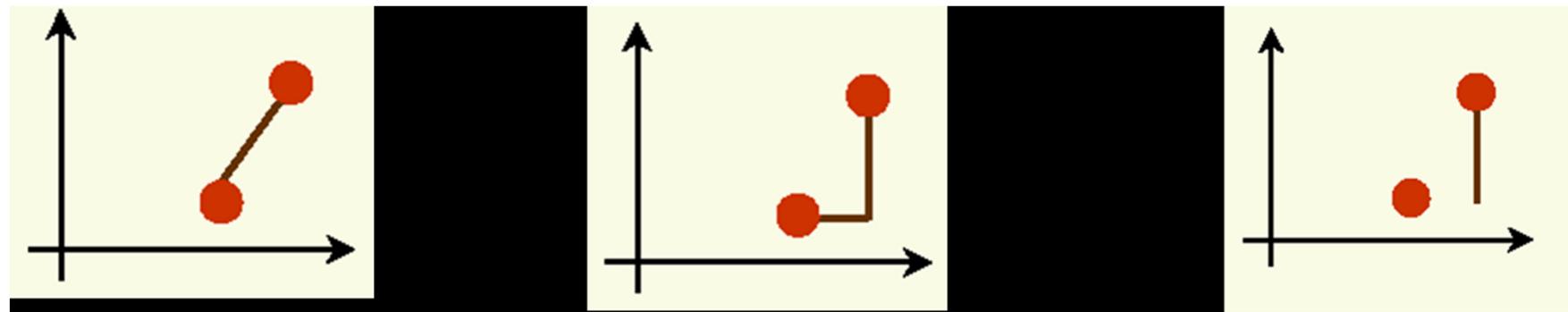


- Given a set of not labeled examples
- Find a relevant grouping of the examples into clusters such that:
 - Examples in the same cluster have **high similarity**
 - Examples from different clusters have **high dissimilarity**

Similarity measures –

Euclidian distance; Chebyshev distance; Manhattan distance

Distance (similarity) measures



**Euclidian Distance
(L2 norm)**

$$d(p,q)=\sqrt{(x_p-x_q)^2+(y_p-y_q)^2}$$

**Manhattan Distance
(L1 norm)**

$$d(p,q)=|x_p - x_q| + |y_p - y_q|$$

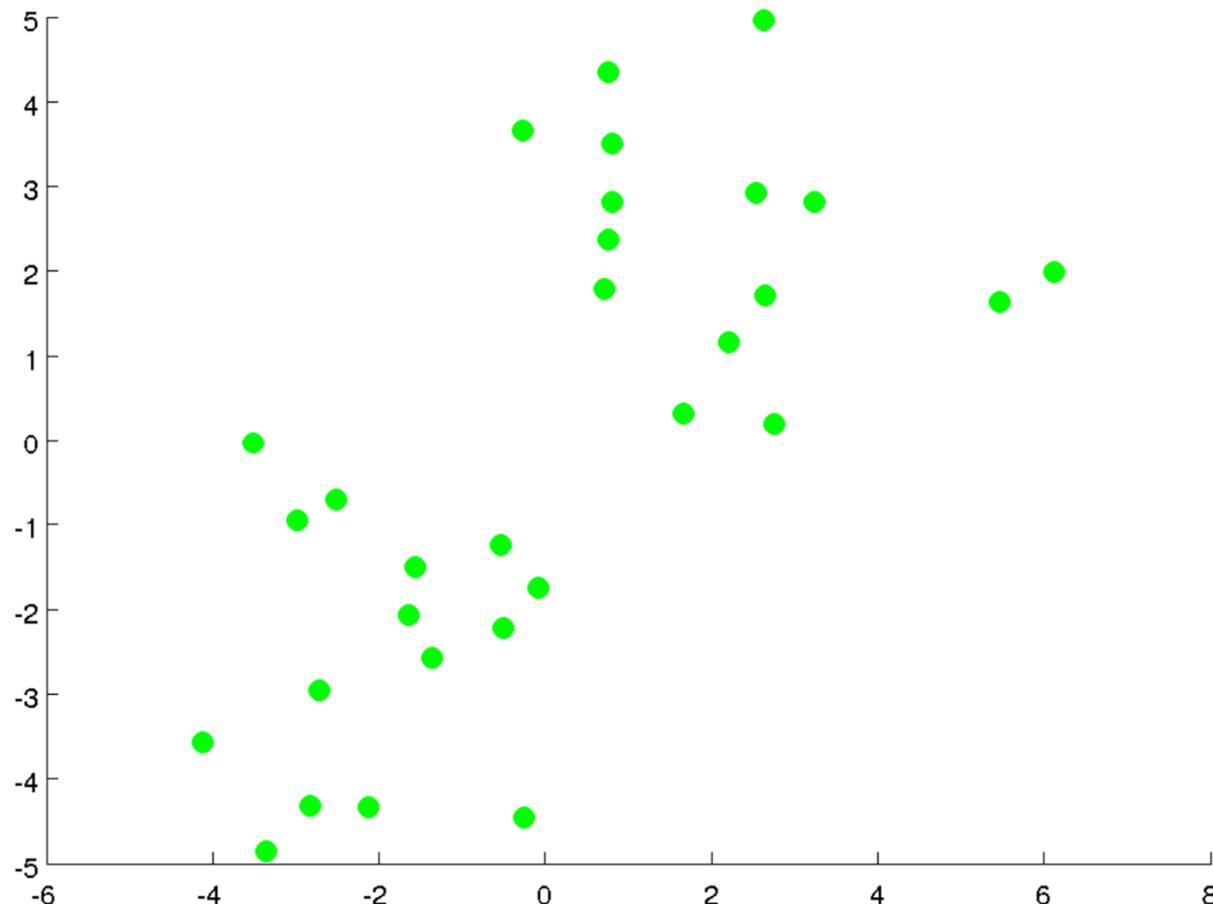
Chebyshev distance

$$d(p,q)=\max|(x_p - x_q),(y_p - y_q)|$$

K-means algorithm

Given:

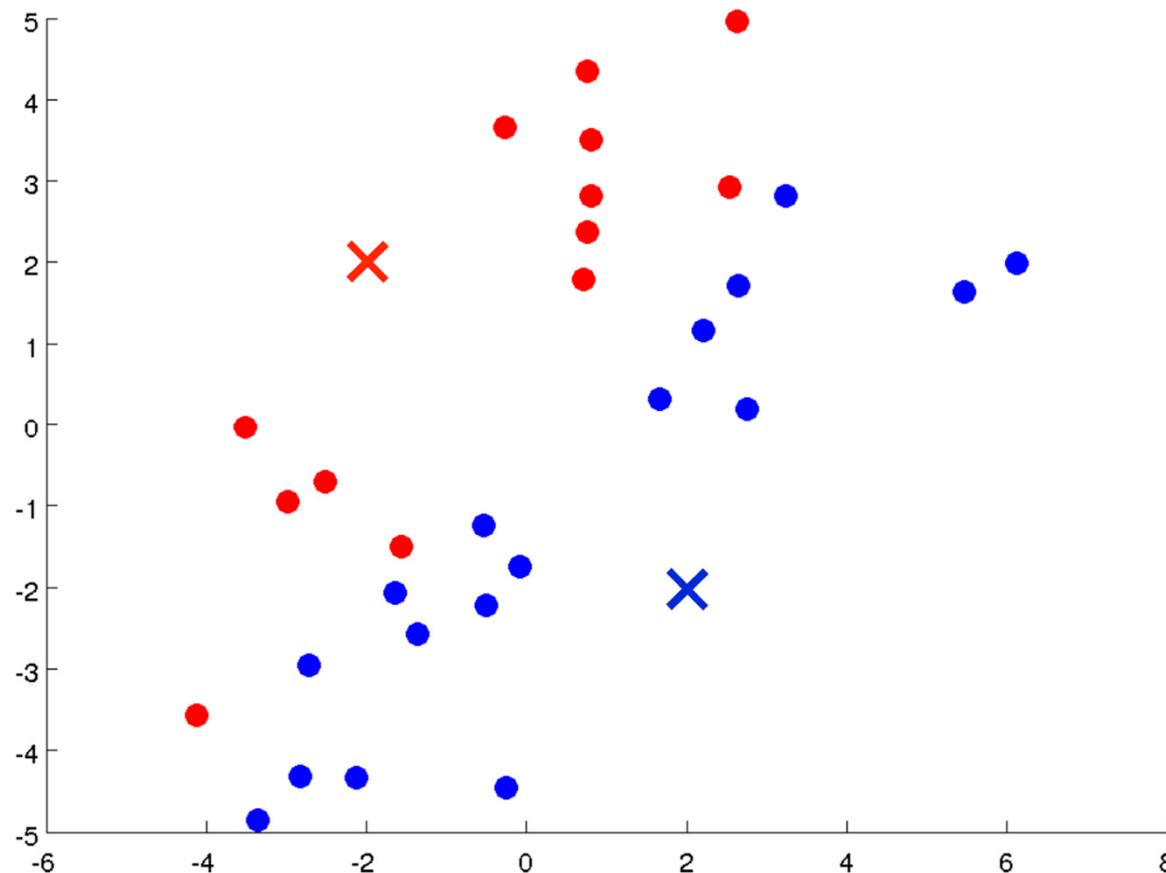
- K – the number of clusters
- Training set - no labels



K-means algorithm

Randomly initialize K cluster centroids (e.g. K=2)

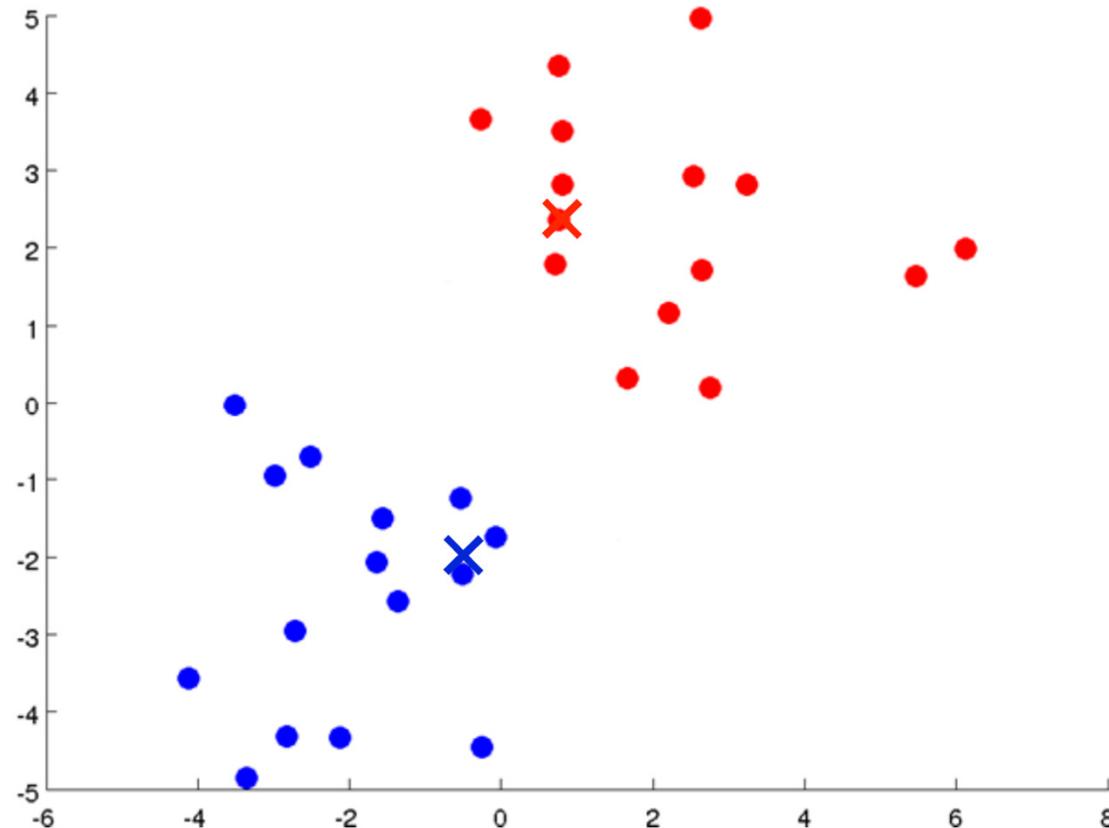
Assign data points to their closest centroid (Euclidian distance)



K-means algorithm

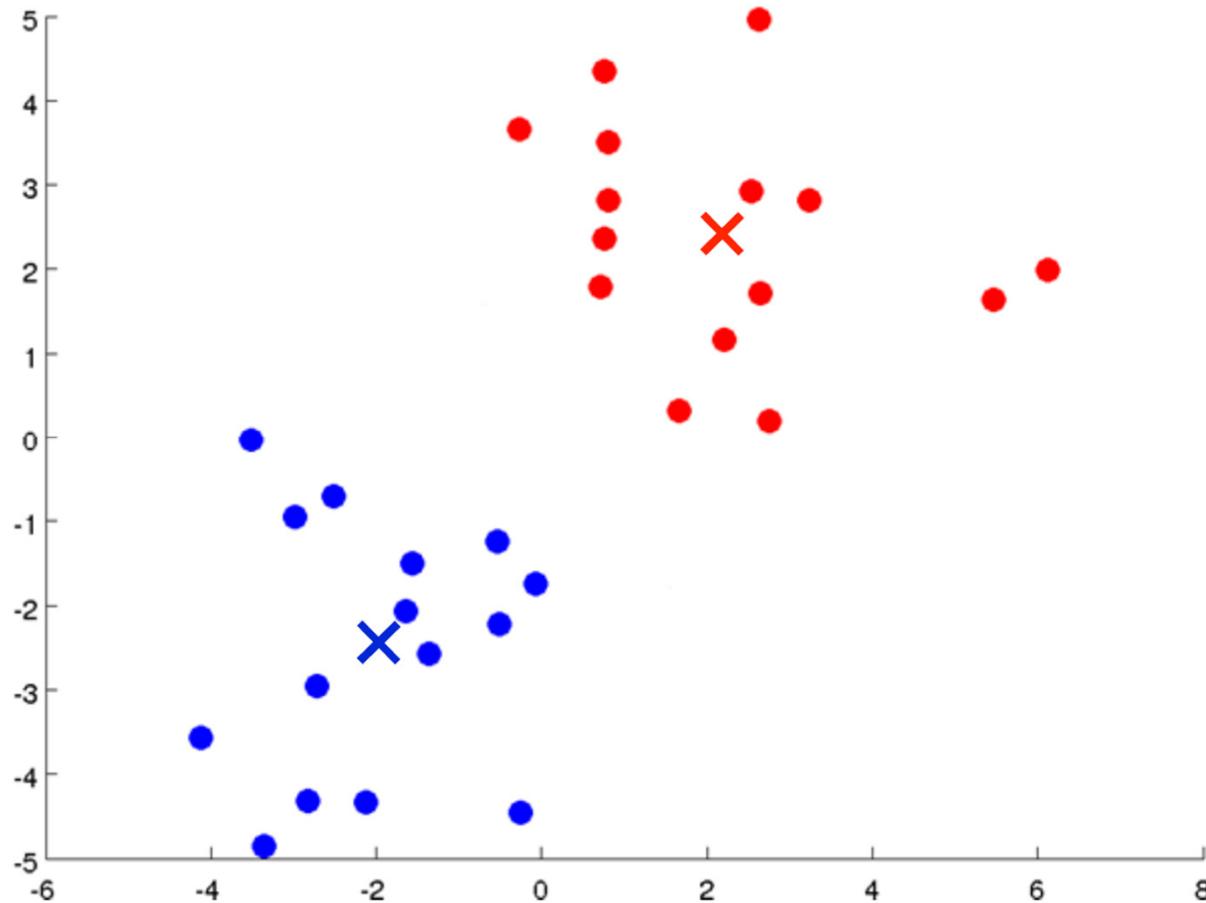
Compute new centroids = mean of the points assigned to that cluster.

Assign data points to the new closest centroid.



K-means algorithm

Repeat until convergence



K-means algorithms

Input:

- K (number of clusters)
- Training set (no labels)

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

*Cluster
assignment =>
step*

 for $i = 1$ to m
 $c^{(i)} :=$ index (from 1 to K) of cluster centroid
 closest to $x^{(i)}$

*Move centroid =>
step*

 for $k = 1$ to K
 $\mu_k :=$ average (mean) of points assigned to cluster k

}

K-means optimization objective (distortion = average distance)

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$\min_{\substack{c^{(1)}, \dots, c^{(m)}, \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

Stop K-means learning (different criteria):

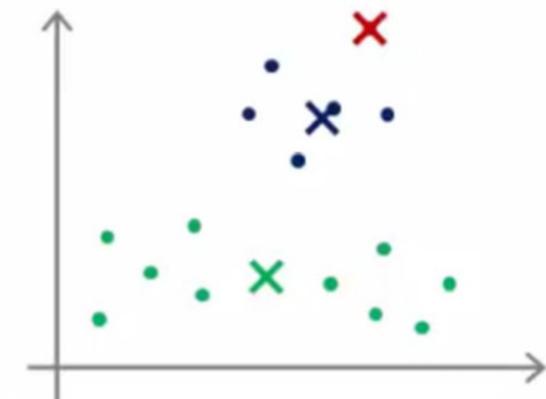
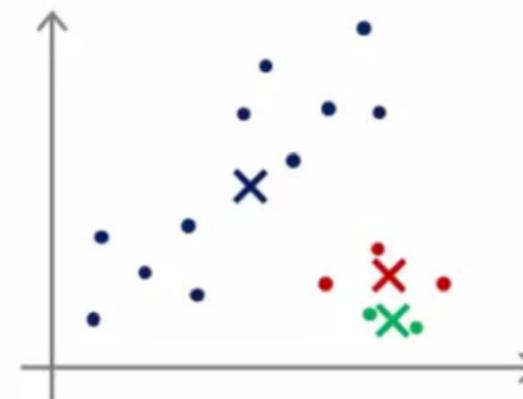
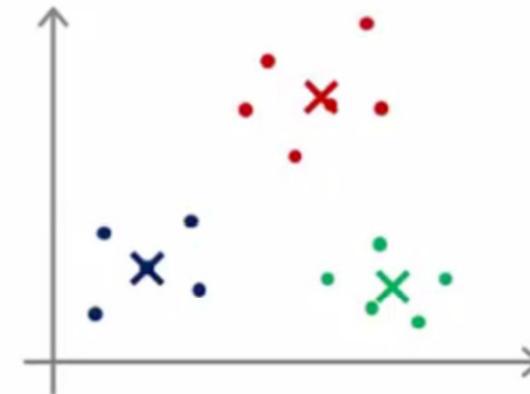
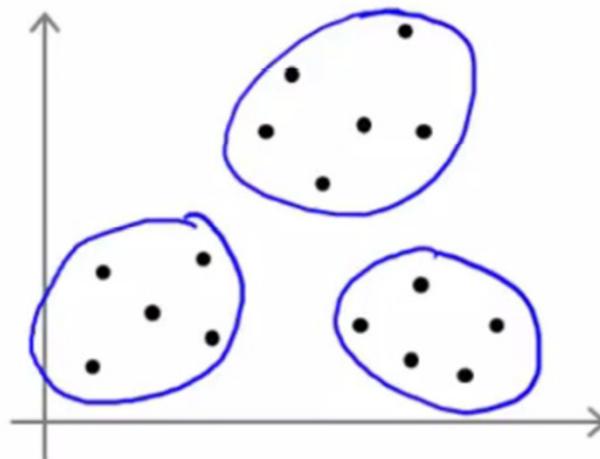
- Achieved Max number of iterations
- $J <$ some threshold
- No improvement of J between subsequent iterations

Single (Random) Initialization

Choose number of clusters K

Initialize K cluster centroids = randomly picked K training examples

Local optima



Repeat Random Initializations

For i = 1 to 100 {

 Randomly initialize K-means.

 Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

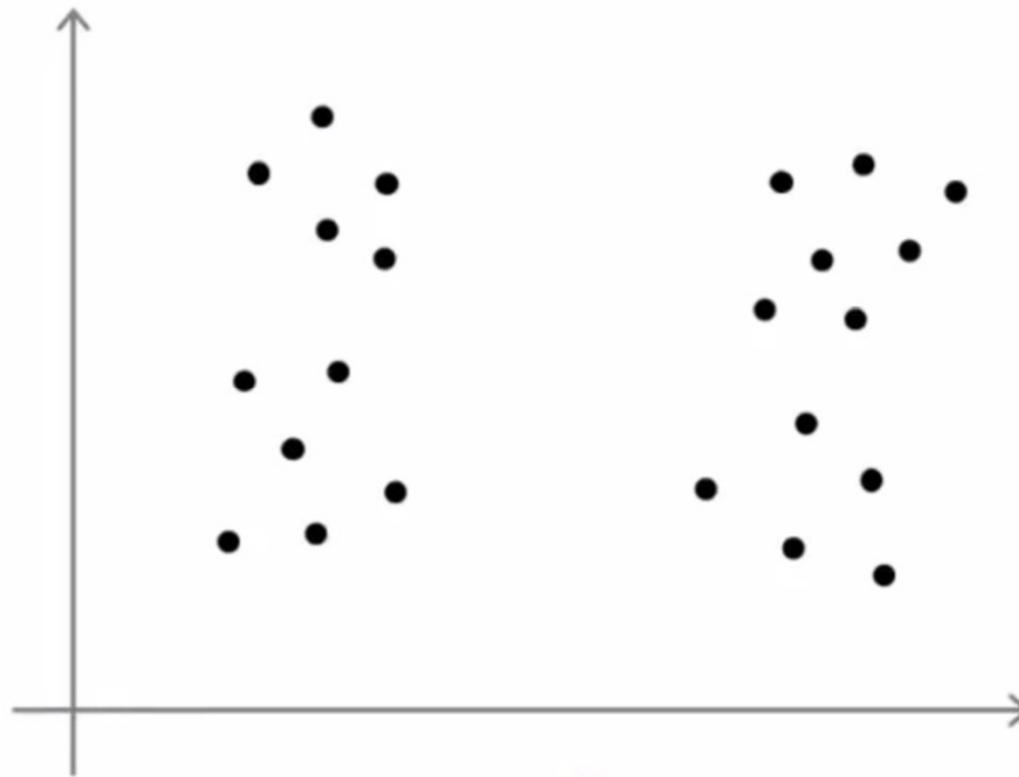
 Compute cost function (distortion)

$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

}

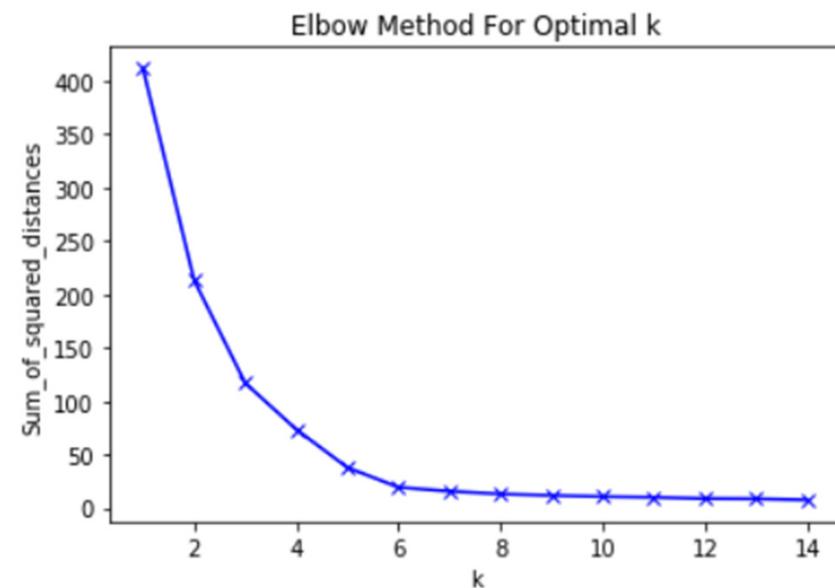
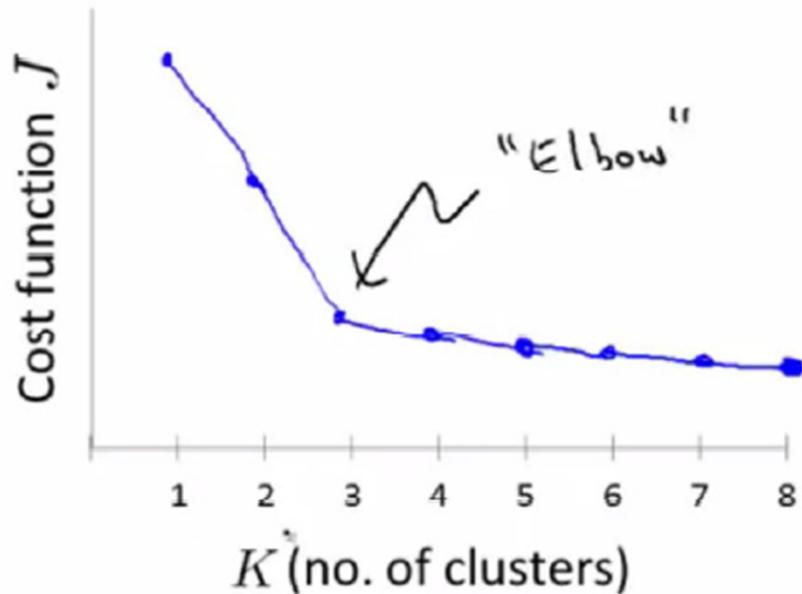
Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

How to Choose # of clusters



- Choose K by data visualisation (if possible)
- Ask domain experts (highly recommendable) , e.g. anomaly detection (experts should know how many types of anomalies are expected)
- Choose K automatically (e.g. Elbow method)

Choosing the number of clusters (Elbow method)

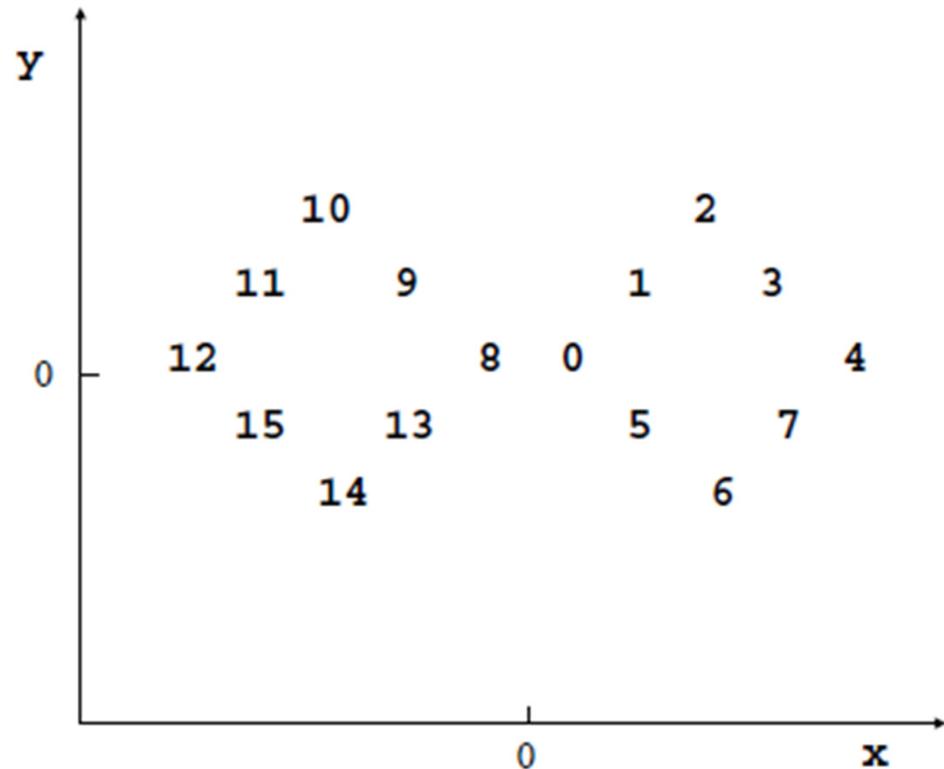


[Tutorial: How to determine the optimal number of clusters for k-means clustering | by Tola Alade | Cambridge Spark](#)

 <https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f> 17

K-MEANS CLUSTERING – Example

Id	x	y
0:	1.0	0.0
1:	3.0	2.0
2:	5.0	4.0
3:	7.0	2.0
4:	9.0	0.0
5:	3.0	-2.0
6:	5.0	-4.0
7:	7.0	-2.0
8:	-1.0	0.0
9:	-3.0	2.0
10:	-5.0	4.0
11:	-7.0	2.0
12:	-9.0	0.0
13:	-3.0	-2.0
14:	-5.0	-4.0
15:	-7.0	-2.0



- find the best 2 clusters

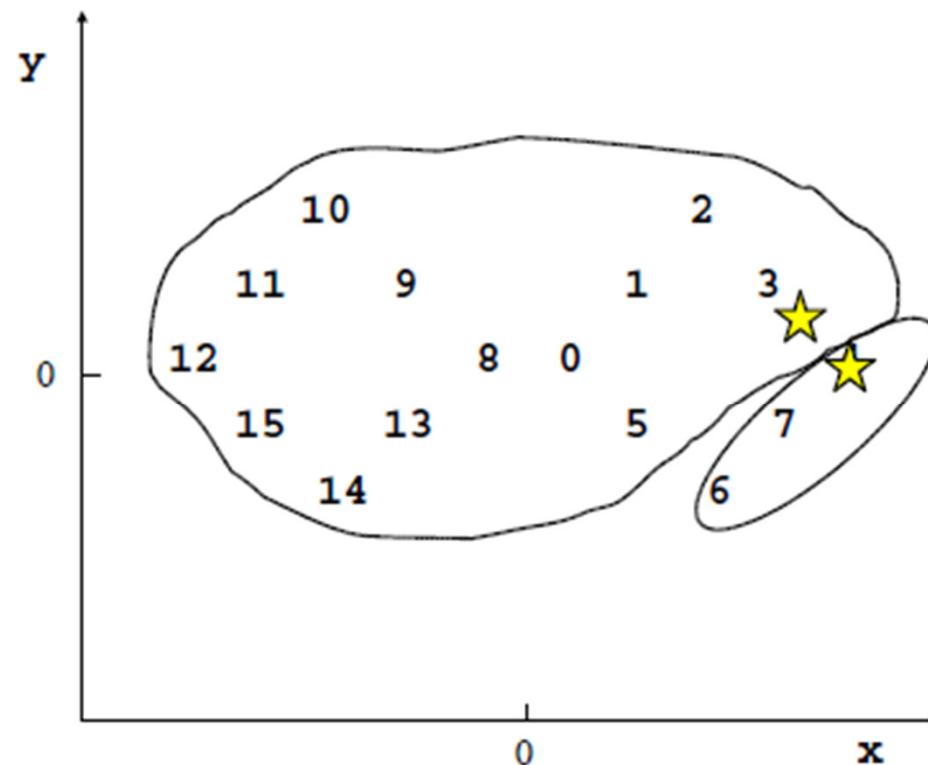
K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887



K-MEANS CLUSTERING – Example

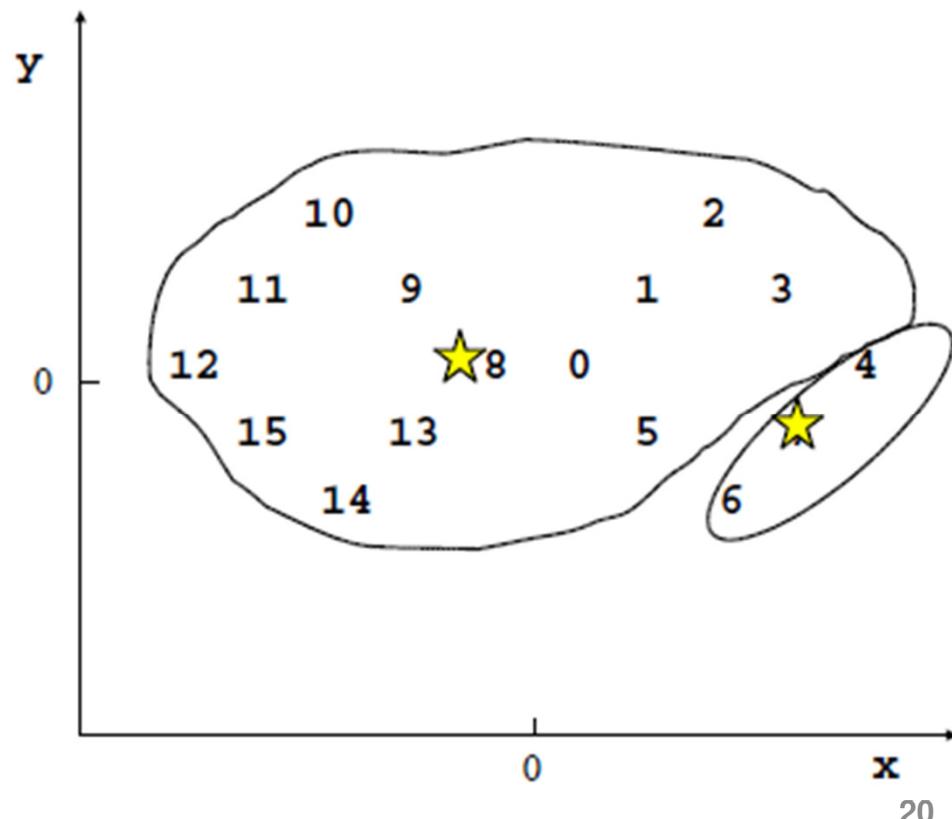
Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

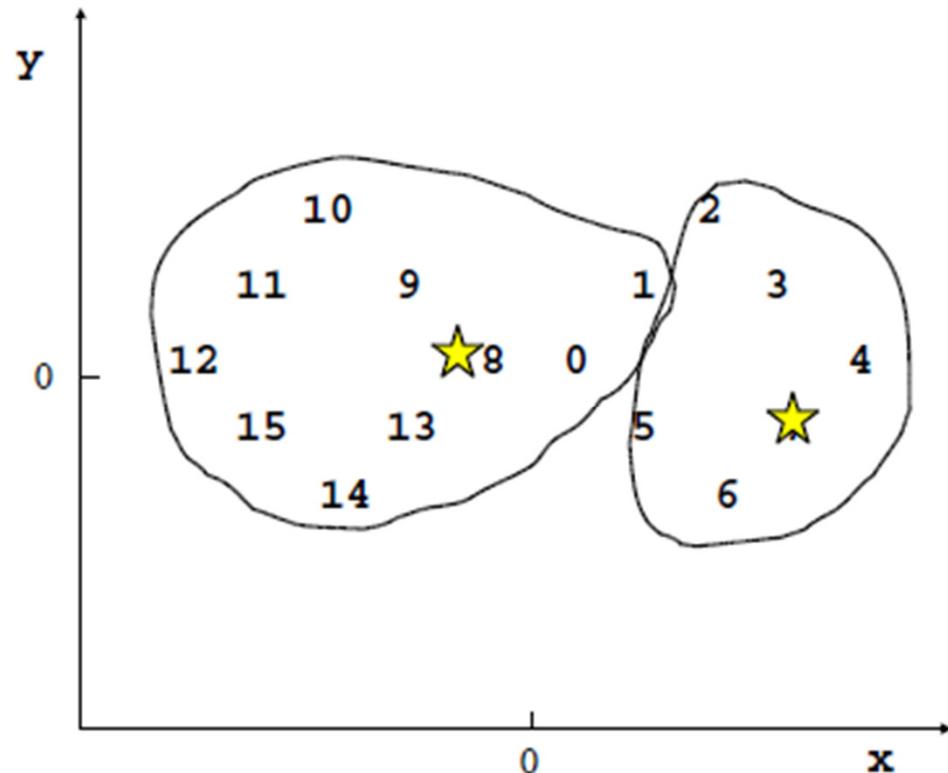
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

Average Distance: 3.6928



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

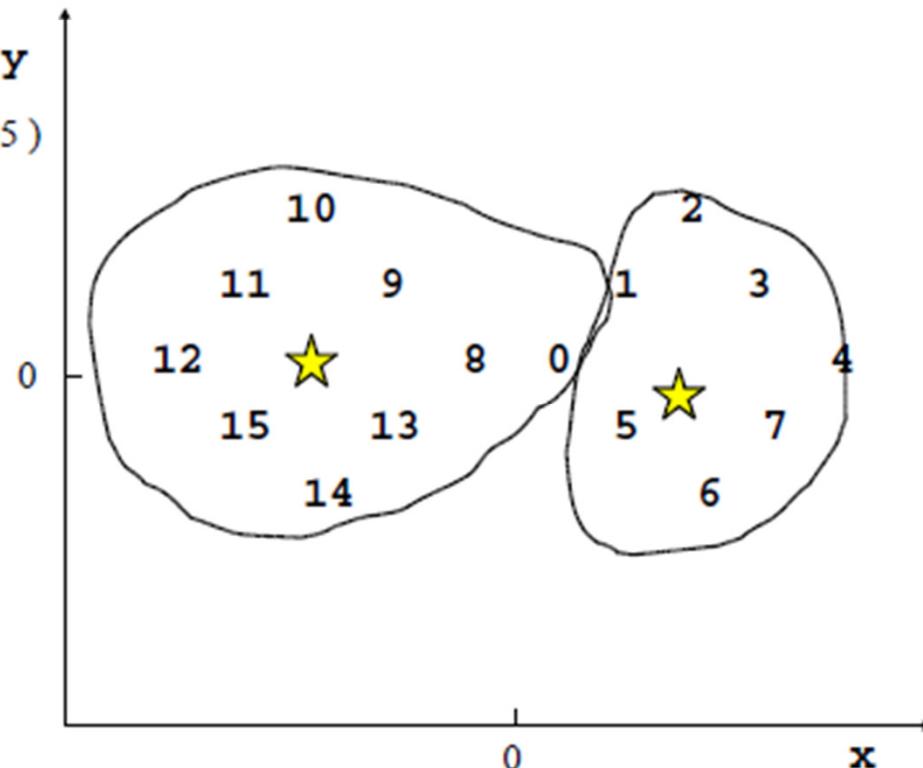
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

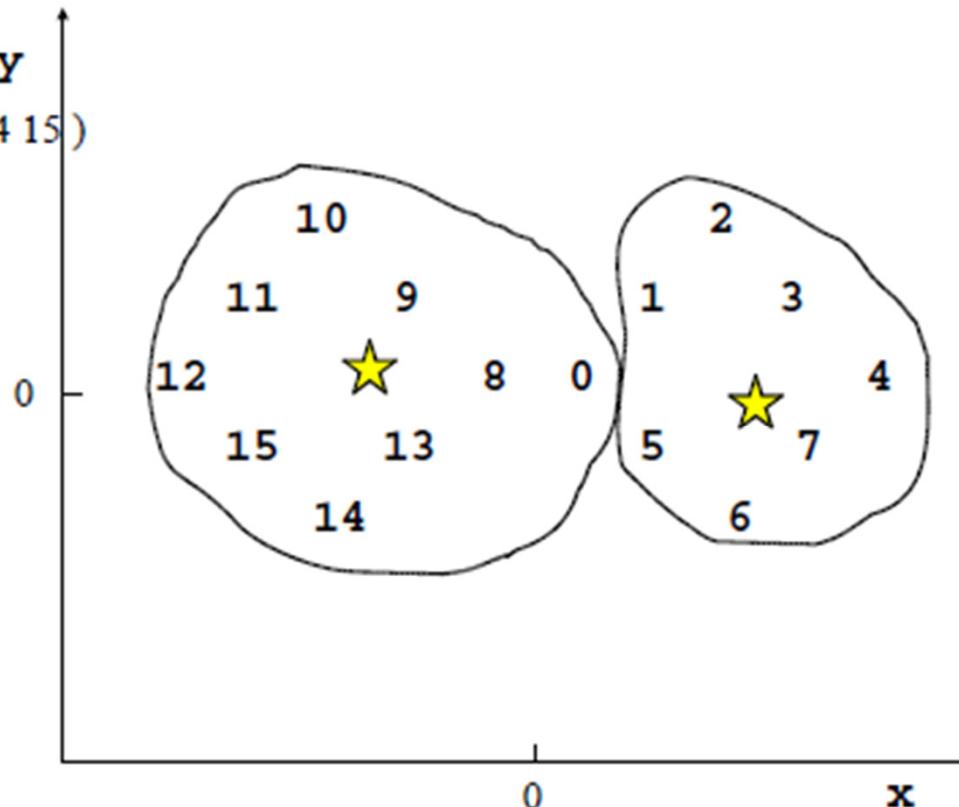
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)

Cluster Centers: (5.57143 0.0) (-4.33334 0.0)

Average Distance: 3.49115



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

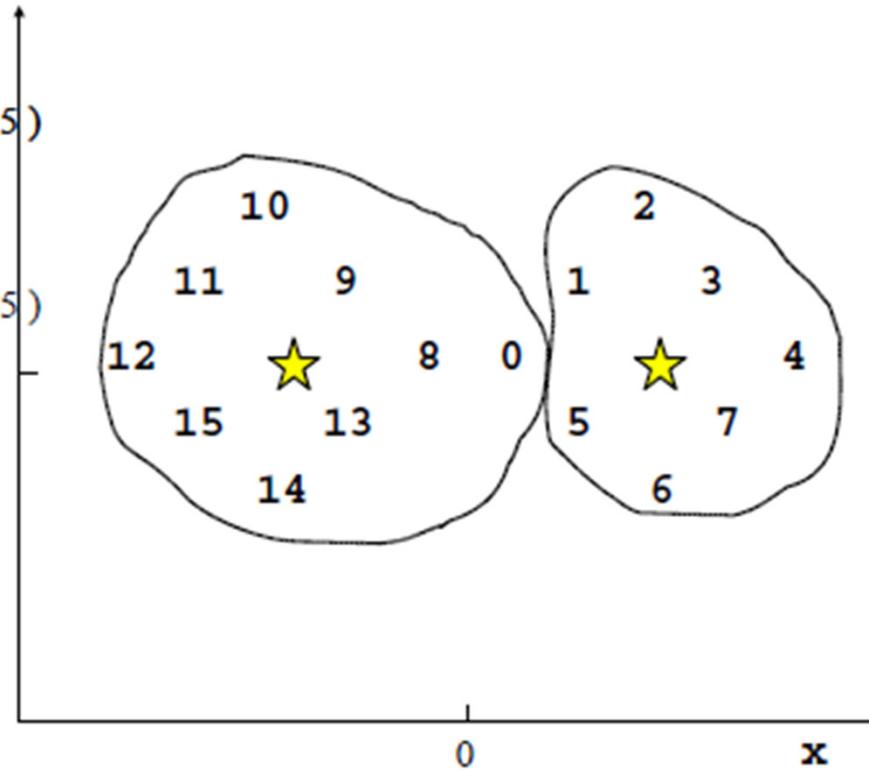
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)

Cluster Centers: (5.57143 0.0) (-4.33334 0.0)

Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)

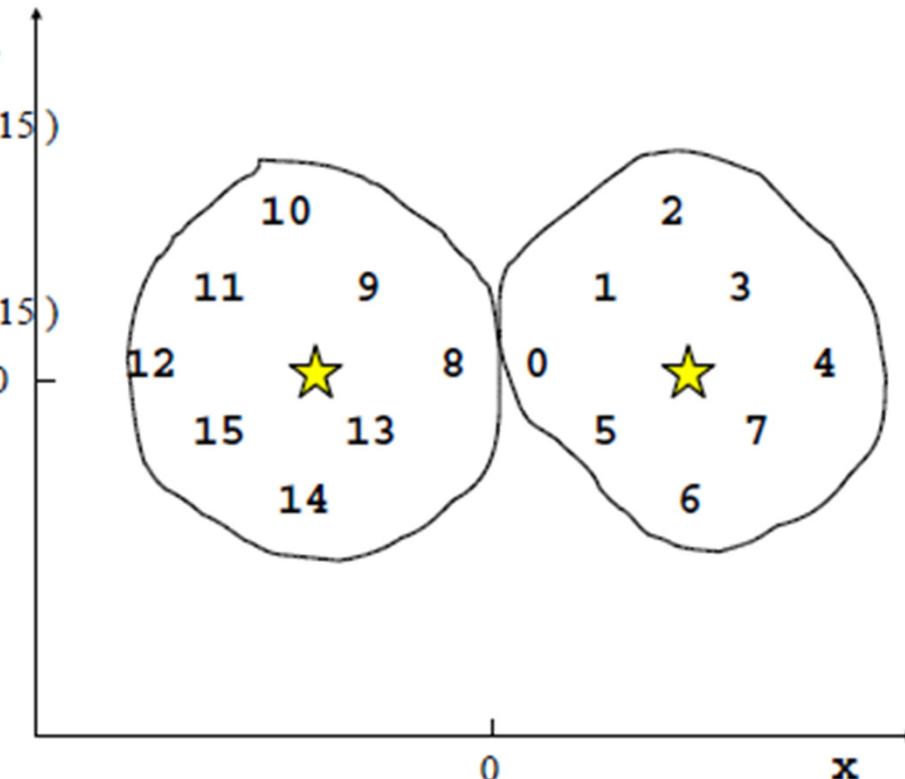
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)

Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)

Cluster Centers: (5.0 0.0) (-5.0 0.0)

Average Distance: 3.41421



K-MEANS CLUSTERING – Example

Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)

Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)

Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)

Cluster Centers: (5.57143 0.0) (-4.33334 0.0)

Average Distance: 3.49115

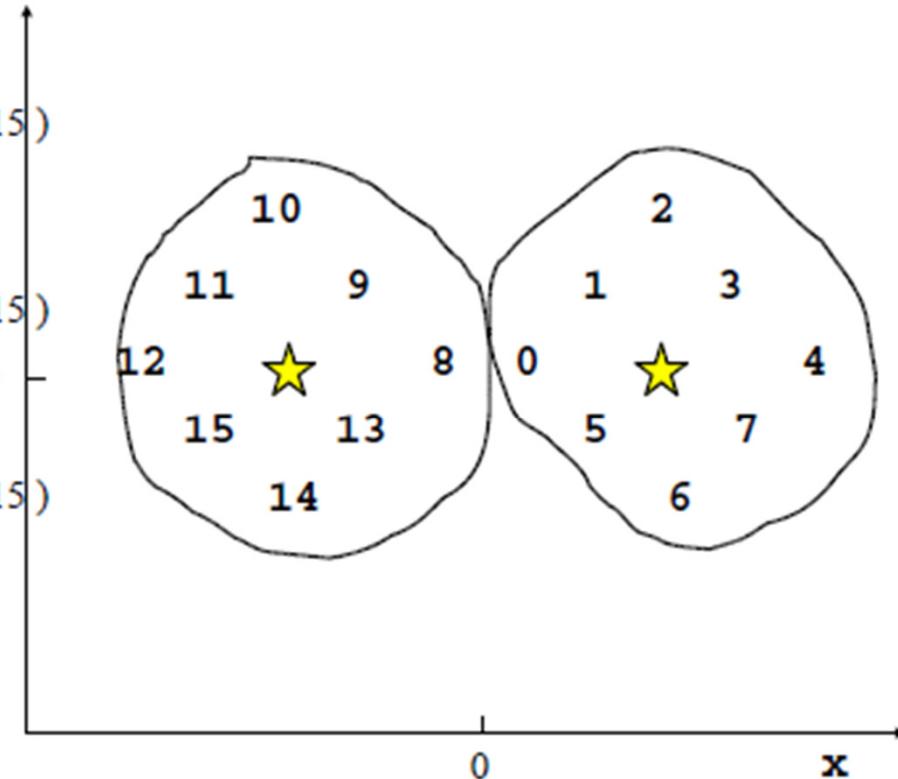
Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)

Cluster Centers: (5.0 0.0) (-5.0 0.0)

Average Distance: 3.41421

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)

No improvement.



K-MEANS -summary

- The most popular clustering method.
- Need to know K.
- May converge to a Local Minimum .
- High number of computations.

K-means for dimensionality reduction

dimensionality reduction is useful for:

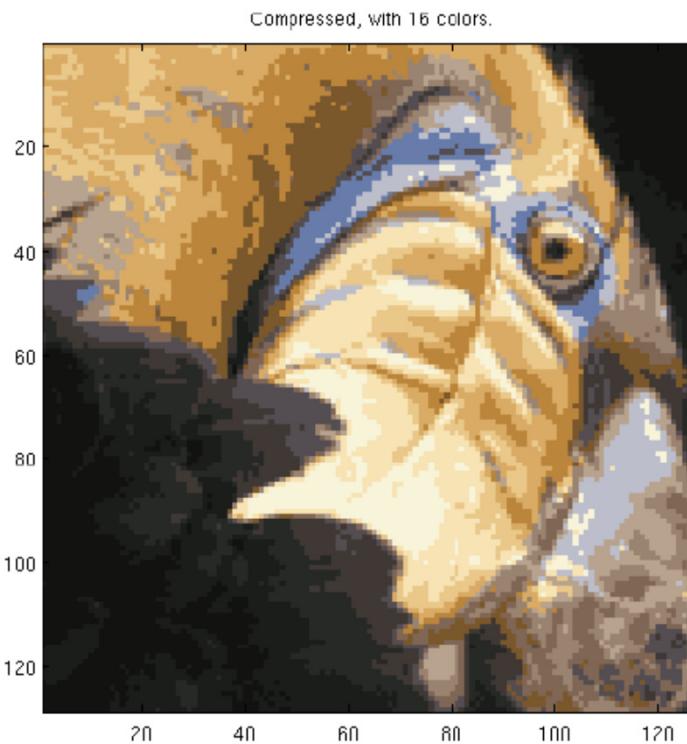
Data compression (from 10000-1000 D to) 100 D

Reduce memory/ disk needed to store data
Speed up learning algorithm

Data visualization (from 100-50D to 2-3D)

Image compression with K-means

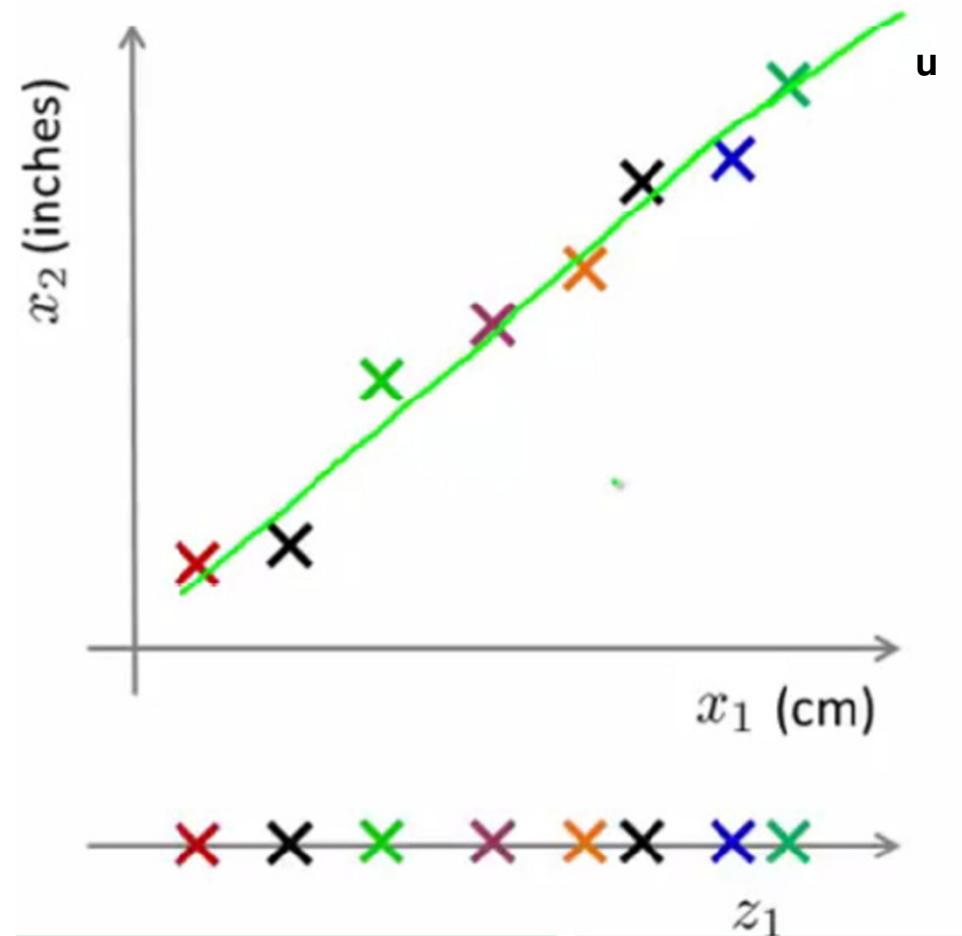
RGB image: 3*8 bits/pixel Compressed image: 16 colors(clusters) => 4 bits



DATA COMPRESSION

Example: reduce data from 2D to 1D

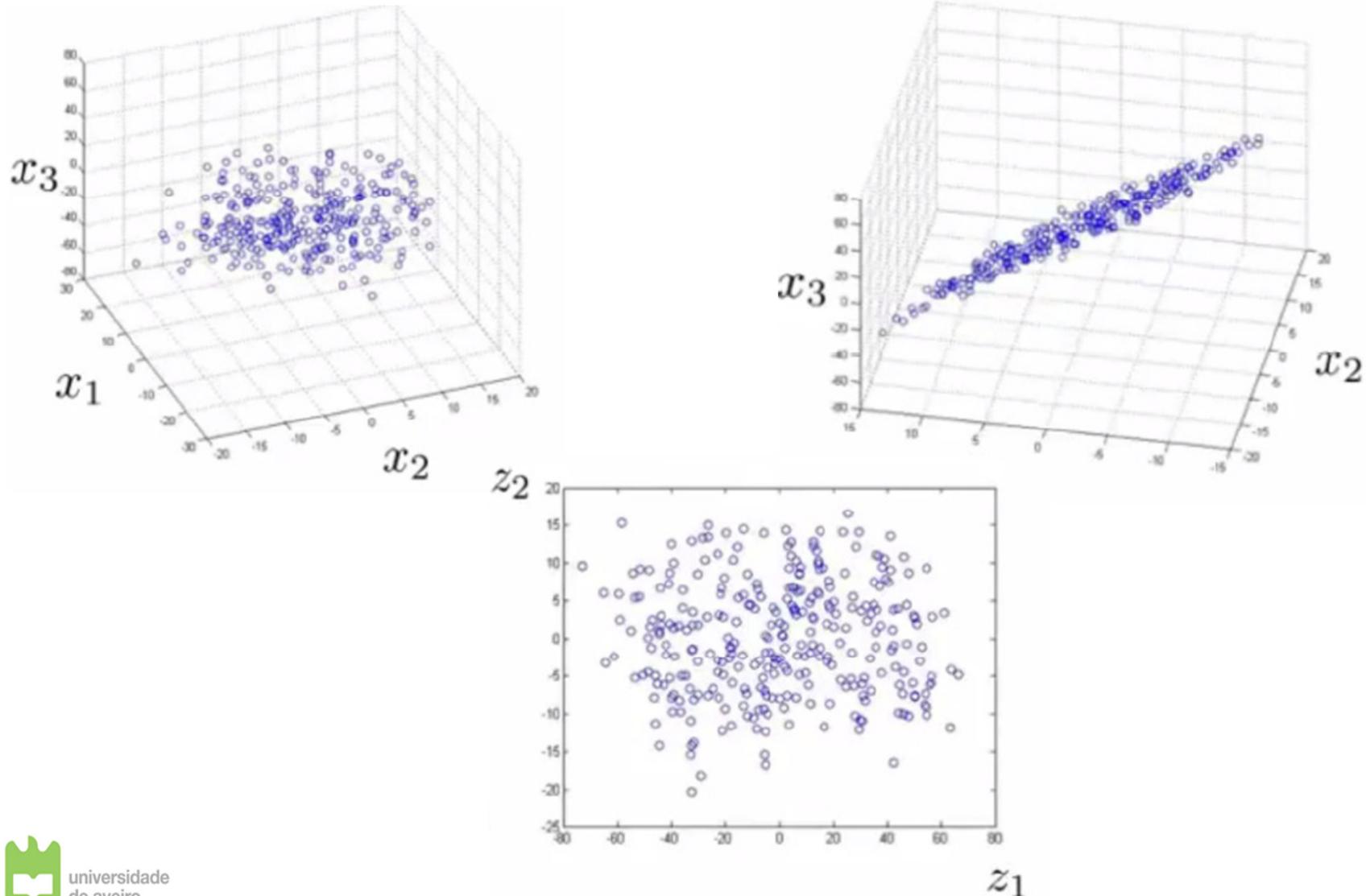
If 2D data is located along a line, the second dimension is redundant



DATA COMPRESSION

Example: reduce data from 3D to 2D

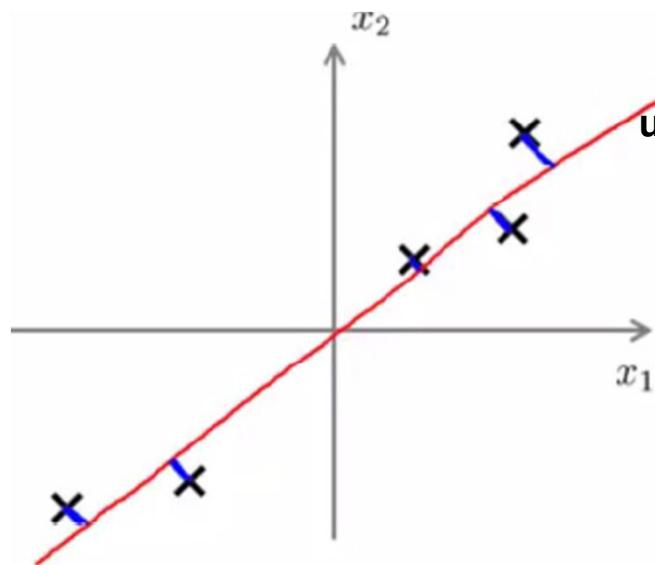
If 3D data is located along a plane, the 3rd dimension is redundant



PRINCIPAL COMPONENT ANALYSIS (PCA)

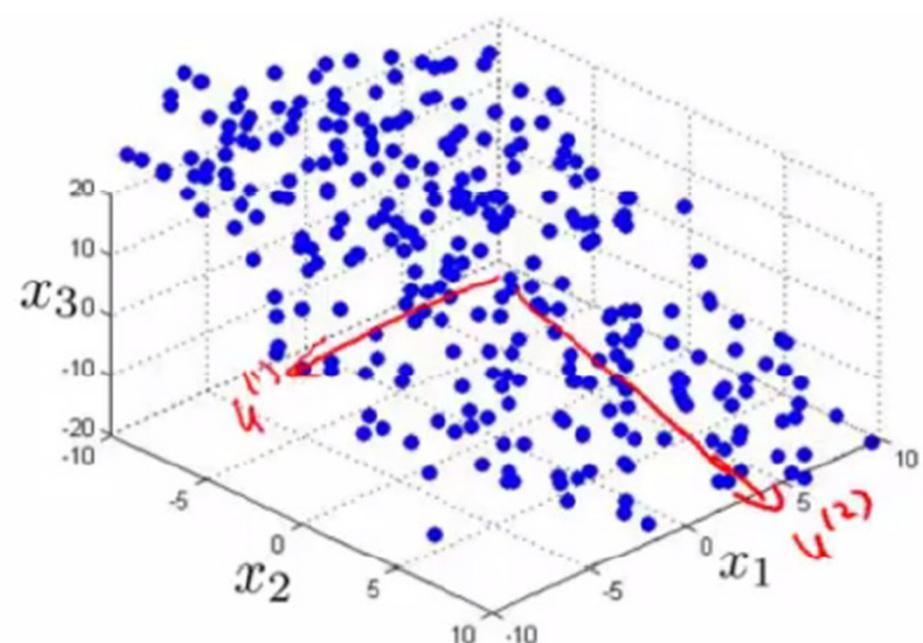
Reduce from 2D to 1D:

find the best direction (vector u) onto which to project data such that to minimize the projection error



Reduce from 3D to 2D:

find the orientation of the best plane (vectors u_1, u_2) onto which to project data such that to minimize the projection error

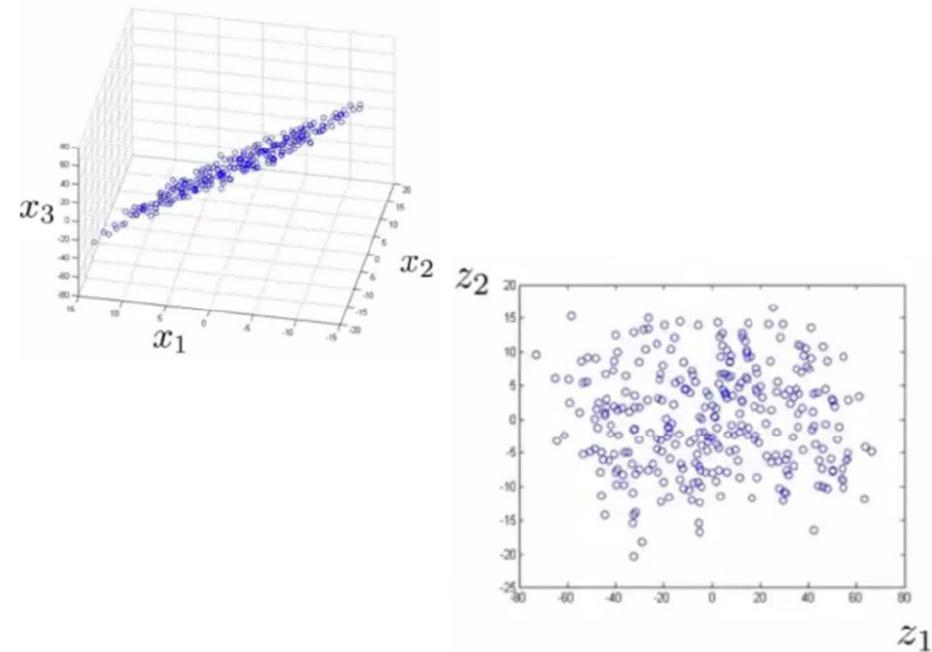
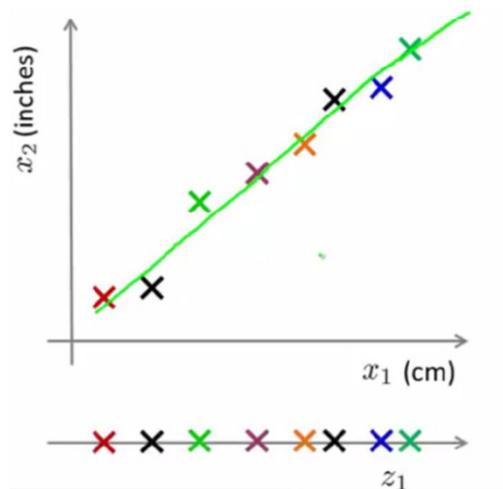


PRINCIPAL COMPONENT ANALYSIS (PCA)

Reduce from n-dimension to k-dimension ($k < n$):

Two tasks:

- **Task 1:** Compute the coordinate system of the best k-dimensional surface (represented by vectors u_1, \dots, u_k) onto which to project data such that to minimize the projection error.
- **Task 2:** Compute the values of the projected data in the lower dimensional space (z values)



PCA – DATA PREPROCESSING (step 1)

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

Thus, all features have zero mean !

If the features have significantly different range of values, normalize them., e.g. in the interval [0,1] or [-1,1] or mean=0 & std=1

matrix \mathbf{X} ($m \times n$)	feature x_1	feature x_2	feature x_n
Example 1	$x_1^{(1)}$	$x_2^{(1)}$		$x_n^{(1)}$
Example 2	$x_1^{(2)}$	$x_2^{(2)}$		$x_n^{(2)}$
...				
Example i	$x_1^{(i)}$	$x_2^{(i)}$		$x_n^{(i)}$
...				
...				
Example m	$x_1^{(m)}$	$x_2^{(m)}$		$x_n^{(m)}$

DATA NORMALIZATION

```
from sklearn.preprocessing import MinMaxScaler
```

```
mms = MinMaxScaler()  
mms.fit(data)  
data_transformed = mms.transform(data)
```

MinMaxScaler?

```
MinMaxScaler(feature_range=(0, 1), copy=True)
```

Transforms features by scaling each feature to a given range.

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()  
sc.fit(data)  
data_transformed =sc.transform(data)
```

Standardize features by removing the mean and scaling to unit variance.

PCA – Singular Value Decomposition (step 2)

- Compute Covariance matrix of the mean normalized data matrix X (dimension $m \times n$ - m examples, n features):

$$\mathbf{Cov} = \mathbf{X}^T * \mathbf{X}$$

- Compute Singular Value Decomposition(SVD) of Covariance matrix:

$$\mathbf{Cov} = \mathbf{U} * \mathbf{S} * \mathbf{V}$$

U ($n \times n$) - matrix of eigenvectors:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

S ($n \times n$)– diagonal matrix of singular values in decreasing order:

$$S_{n \times n} = \begin{bmatrix} S_{11} & 0 & \dots & 0 \\ 0 & S_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & S_{nn} \end{bmatrix}$$

PCA

The projection vectors are the first k columns of U ($k < n$): - **Task 1**

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n} \quad Ureduce_{nxk} = U(:,1:k)$$

Step 3: Compute the new (projected) data matrix Z (m examples, k features): - **Task 2**

$$Z_{mxk} = X_{mxn} * Ureduce_{nxk}$$

Step 4: Reconstruct data matrix X from the projected Z matrix :

$$X_{approx(mxn)} = Z_{mxk} * Ureduce_{kxn}^T$$

Choosing k (number of principal components)

Average squared approximation error:

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2$$

Total data variation:

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$$

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01 \quad (1\%)$$

“99% of variance is retained”

(typically the desired retained variance is between 90-99%)

Choosing k (number of principal components)

Compute SVD once: $[U, S, V] = \text{svd}(\text{Cov})$

Choose $k < n$ such that you get the desired retained variance
(e.g. 99 %):

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.99$$

$$S_{nxn} = \begin{bmatrix} S_{11} & 0 & \cdots & 0 \\ 0 & S_{22} & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & S_{nn} \end{bmatrix}$$

Aprendizagem Aplicada à Segurança

(Mestrado em Cibersegurança-DETI-UA)



LECTURE 6

Introduction do Deep Learning

Petia Georgieva
(petia@ua.pt)

DETI/IEETA – UA

Outline

- **Deep Neural Networks**
- **Convolutional Neural Networks (CNN)**
- **YOLO (You Only Look Once)**
- **Sequence models - Long-Short Term Memory (LSTM)**
- **Time Series Forecasting**

Deep Learning – major applications

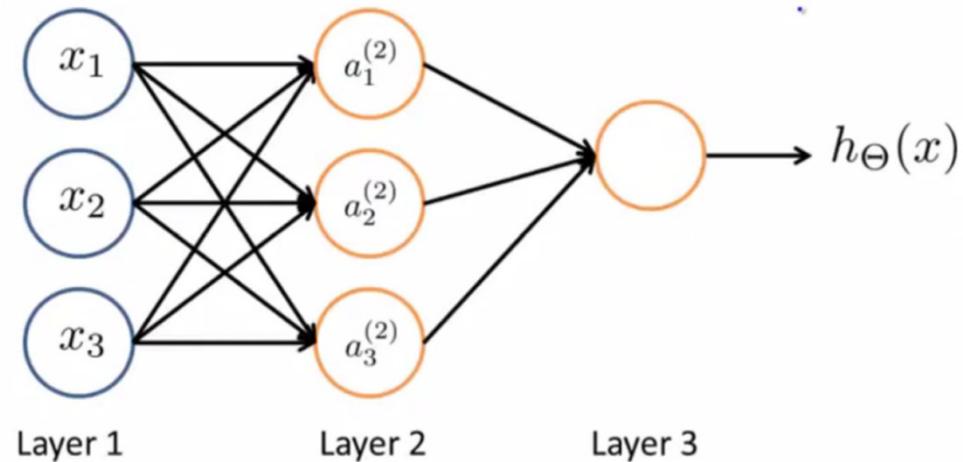
Image, Video, Speech, Text processing (large scale, big data) :

- Computer Vision
- Speech Recognition
- Natural Language Processing (topic classification, sentiment analysis, language translation, etc.)
- Bioinformatics (genomics, drug discovery)
- 5G+ communications
- Time series processing

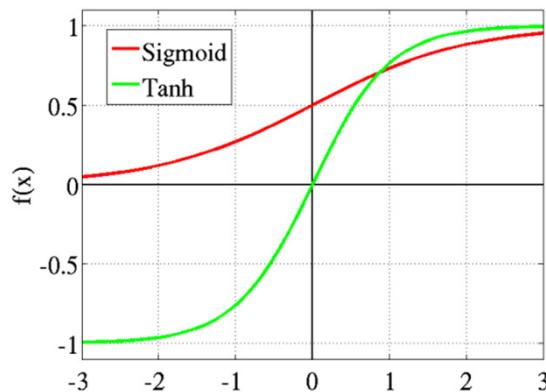
Machine Learning - Shallow (3 layers) Neural Network

50 x 50 pixel images \rightarrow 2500 pixels
 $n = 2500$ (7500 if RGB)

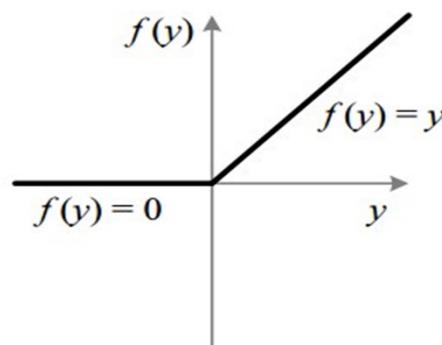
$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$



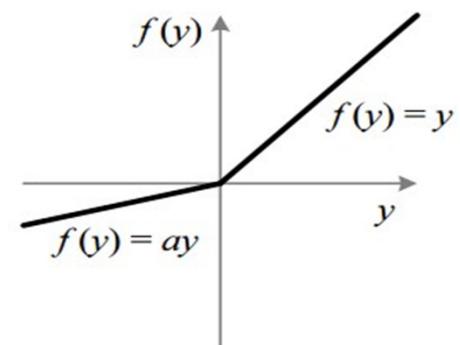
Typical Activation functions



Sigmoid (logistic)/Hyperbolic tangent



ReLU (Rectified Linear Unit)



Leaky ReLU

Why deep learning ?

Hardware get smaller.

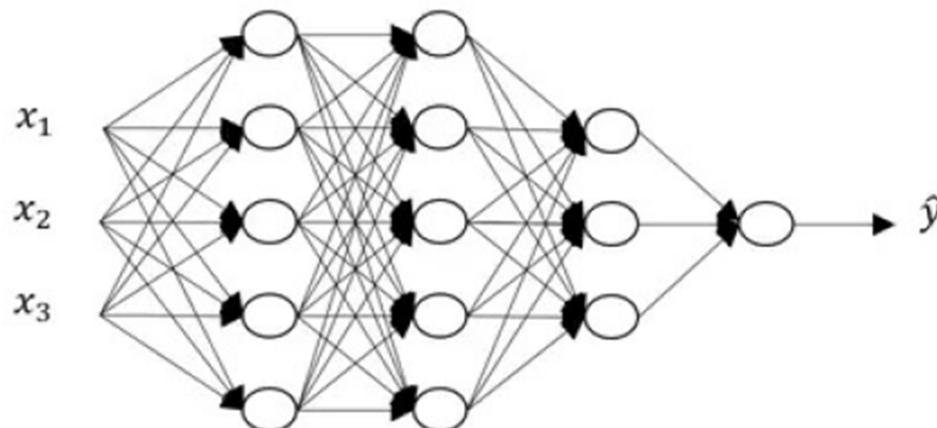
Sensors get cheaper, widely available IoT devices with high sample-rate.

Data sources: sound, vibration, image, electrical signals, accelerometer, temperature, pressure, LIDAR, etc.

Big Data: Exponential growth of data, (IoT, medical records, biology, engineering, etc.)

How to deals with **unstructured data** (image, voice, text) =>
needs for feature extraction (data mining).

Deep Neural Networks: first extract (automatically) the features, then solve
ML tasks (classification, regression, clustering)



Conventional ML vs. Deep Learning

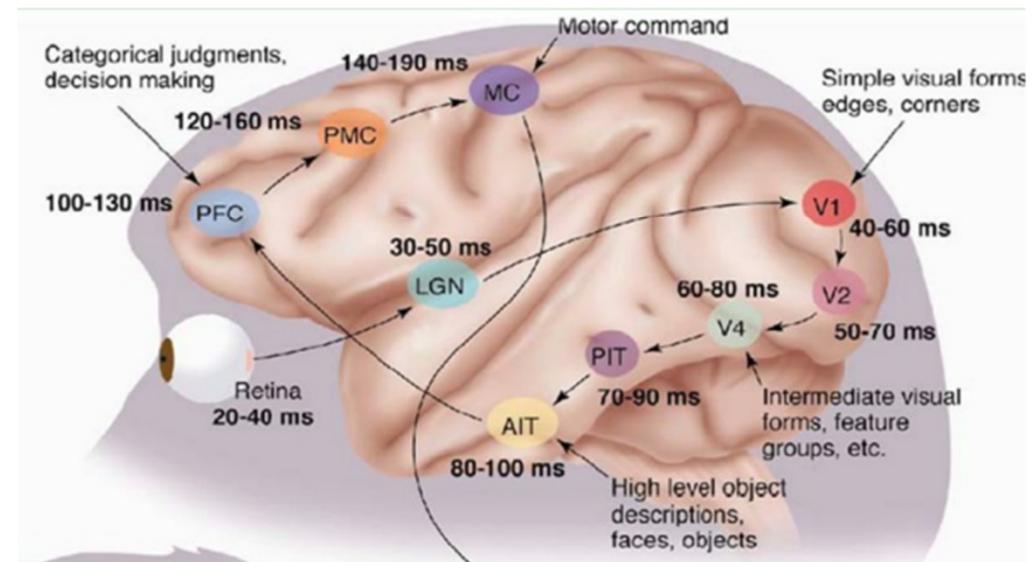
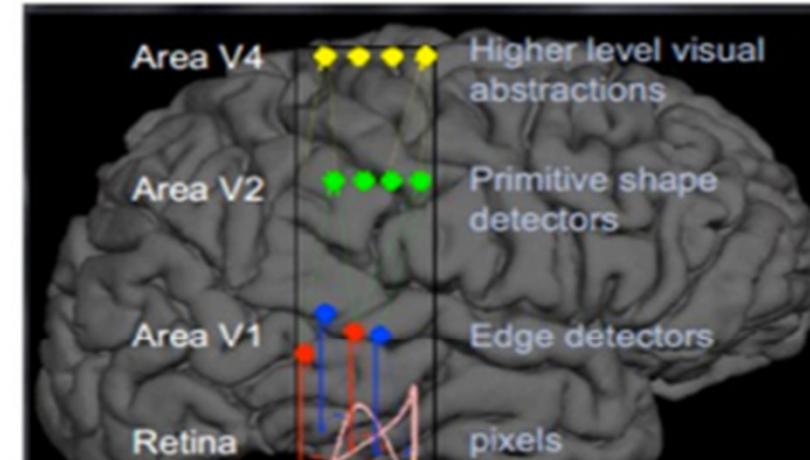
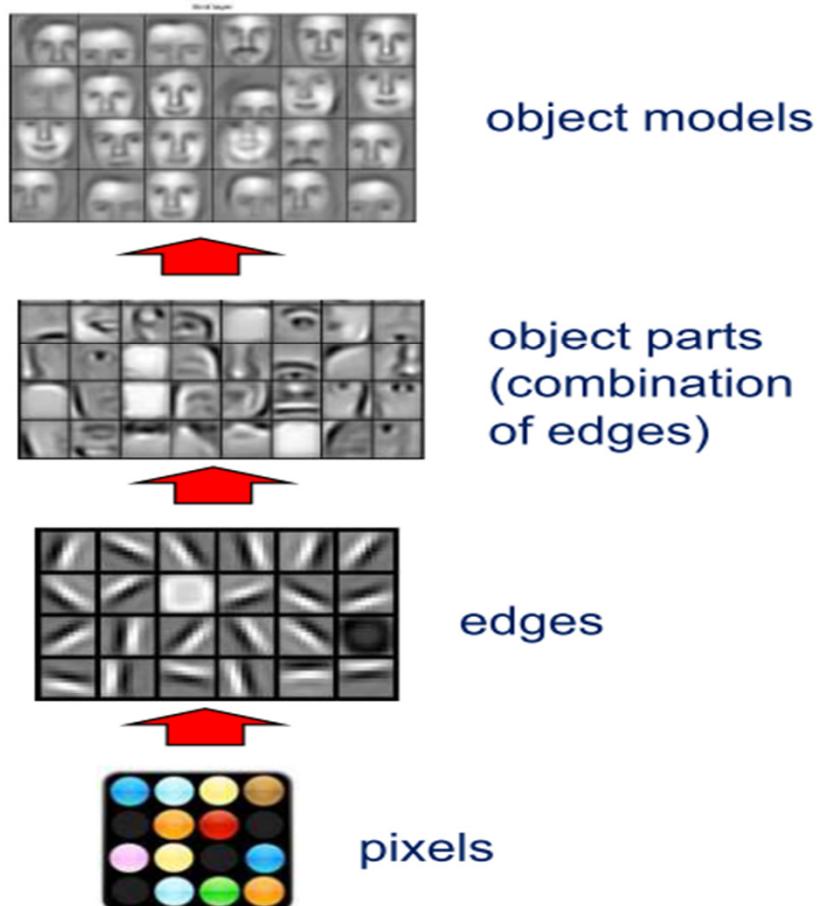
Conventional Machine Learning (ML)

- Needs feature engineering – hard, time-consuming task, requires expert knowledge.
- Don't work well with raw data (e.g. pixels/voxels of images).
- Better than DL if good features are found.

Deep Learning

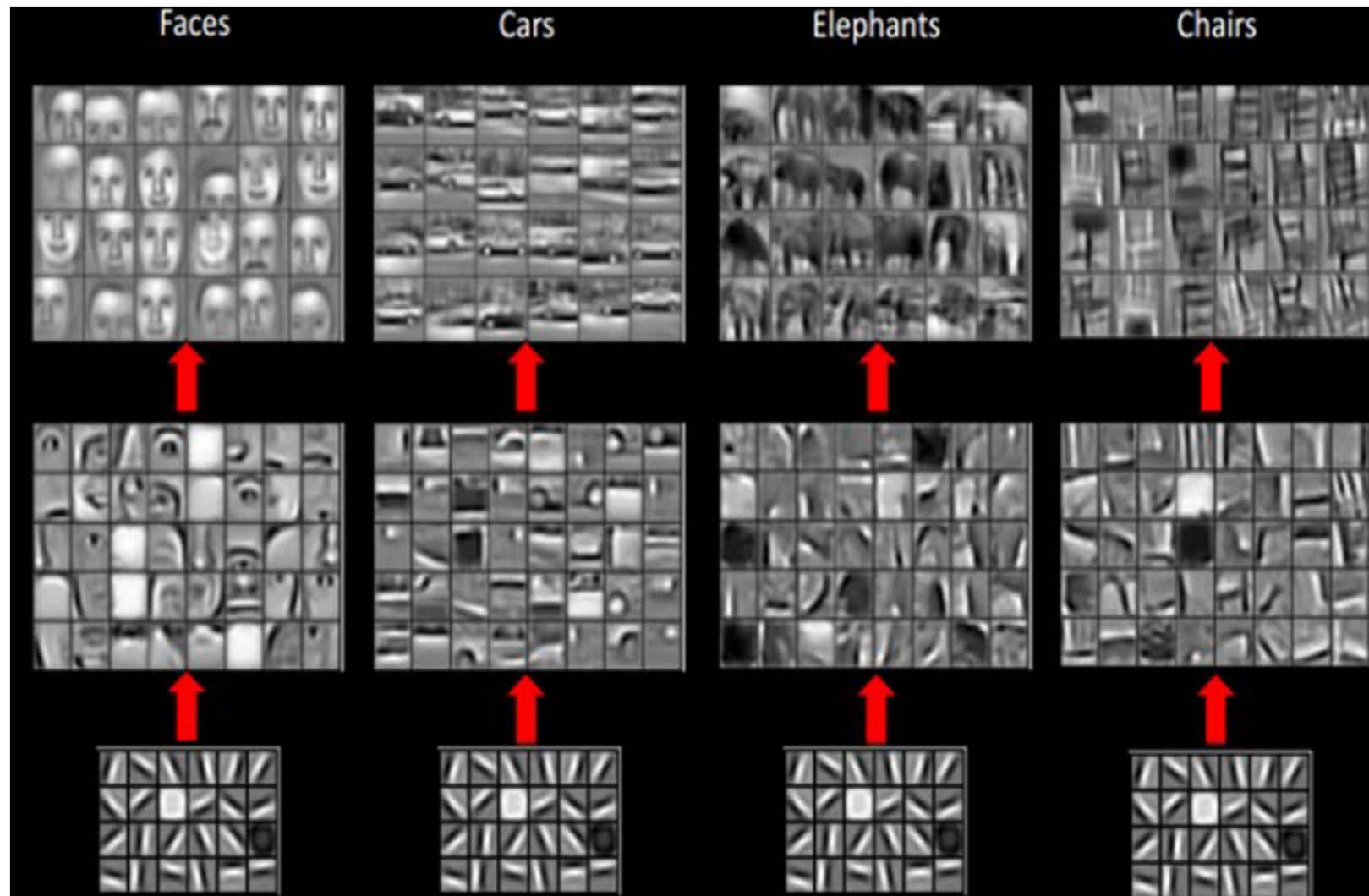
- Representation-learning methods with multiple levels of representation.
- Layers that extract automatically features from raw data (latent structures not seen/difficult to be designed by humans)
- Needs large amounts of labelled training data
- Needs massive computational resources (e.g. GPUs, parallel solvers)

Representation Learning inspired by visual neuroscience



By Yann LeCun, Yoshua Bengio & Geoffrey Hinton
Nature 521, 436–444, 2015, doi:10.1038/nature14539

Representation Learning



By Yann LeCun, Yoshua Bengio & Geoffrey Hinton
Nature 521, 436–444, 2015, doi:10.1038/nature14539

Convolutional Neural Networks (CNN)

Why Convolution Learning ?

Deep learning on large images

If the image has 1000x1000x3 (RGB) pixels=3 million features (inputs)

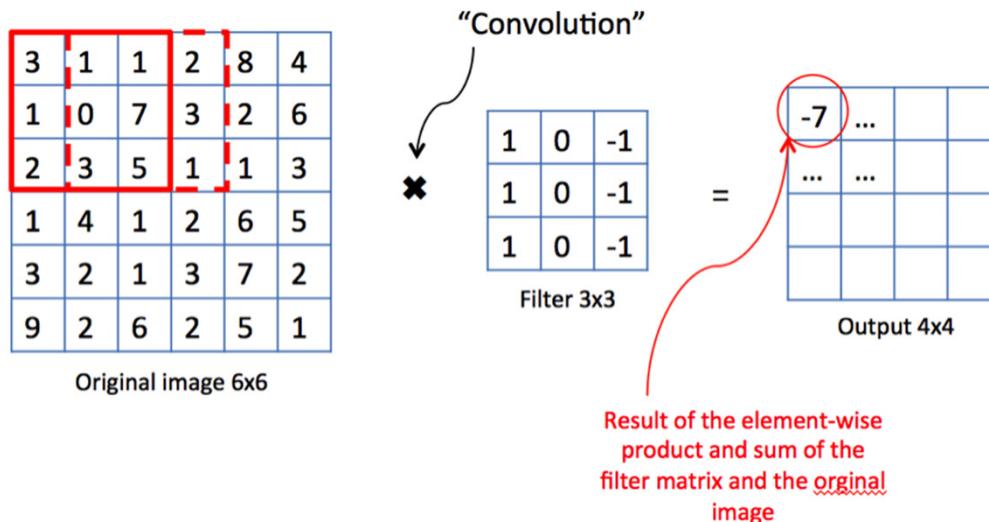
If the first hidden layer has 1000 nodes =>

The parameter matrix between the input and the hidden layer has (1000x3million) 3billion parameters.

1st problem: Difficult to get enough data to prevent model overfitting

2nd problem: Computational (memory) requirements to train such networks are not feasible.

Solution: implement convolution operation



OPERATION CONVOLUTION

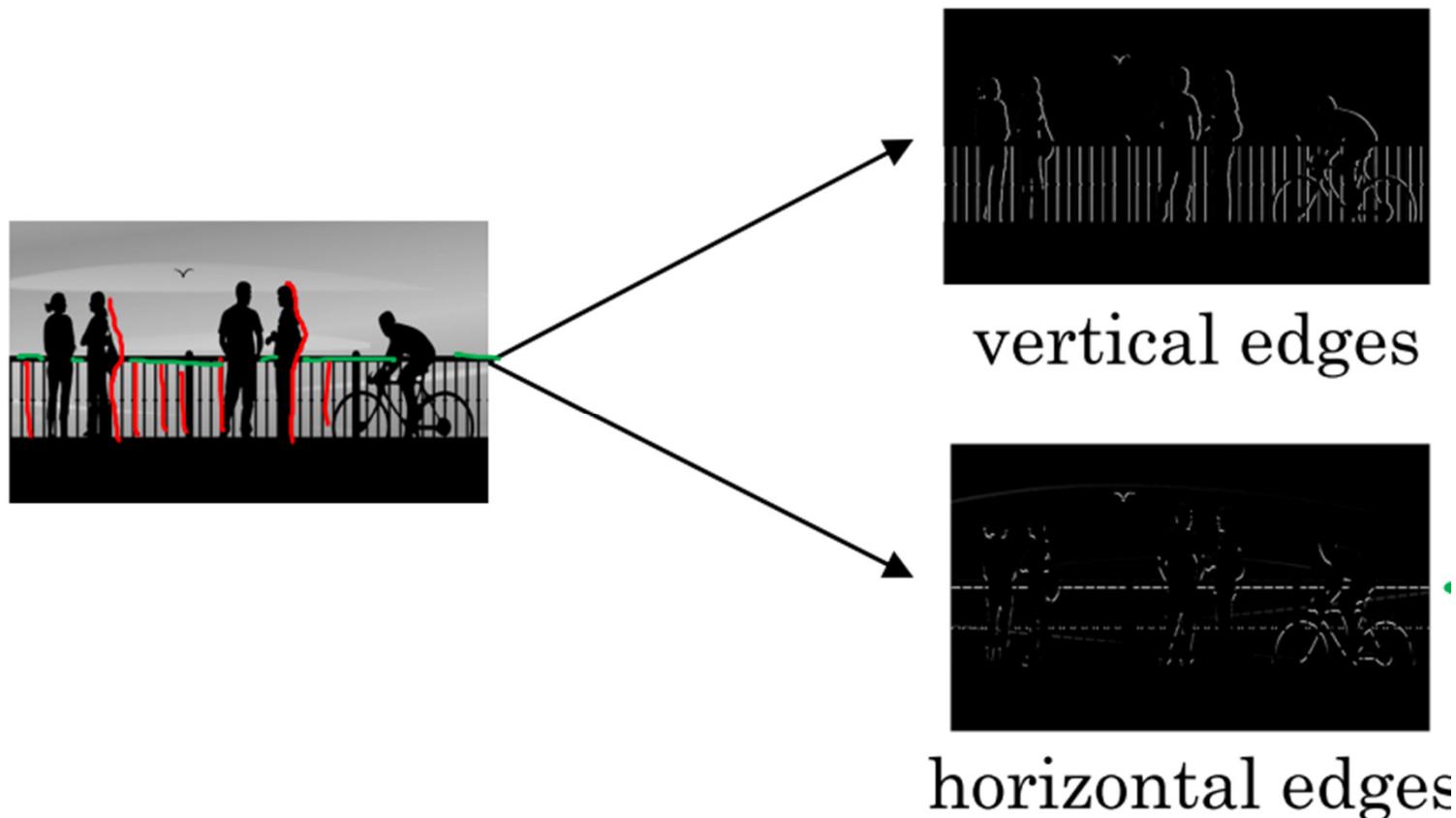
1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Detect horizontal/vertical edges



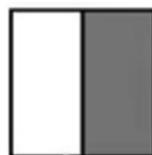
VERTICAL EDGES DETECTOR

Illustrative example:

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

$$\begin{array}{ccc} * & \begin{array}{ccc} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{array} & = \end{array}$$

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Detection of bright to dark transition (+30)

Hand-picked convolutional filters(kernels)

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

=> **Horizontal edge detector**

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

⇒ **Sobel filter**

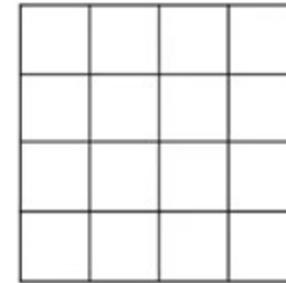
$$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$$

⇒ **Sharr filter**

CONVOLUTIONAL FILTERS (KERNELS)

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Hand-picking the filter values is difficult.

Why not let the computer to learn them ?

Treat the filter numbers as network trainable parameters (weights), and let the computer learn them automatically.

Other than vertical and horizontal edges, such computer-generated filter can learn information from different angles (e.g. 45°, 70°, 73°) .

By convention the conv filter is a square matrix with odd size (typically 3x3; 5x5; 7x7, also 1x1) .

It is nice to have a central pixel and it facilitates the padding.

PADDING

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}$$

Ex. Take 6×6 image, apply 3×3 conv filter, get 4×4 output matrix, because we shift the filter one row down or one column right and therefore we have 4×4 possible positions for the 3×3 filter to appear in the 6×6 input matrix.

In general: given $n \times n$ input matrix and $f \times f$ filter matrix, the convolution operation will compute $(n-f+1) \times (n-f+1)$ output matrix by applying one pixel up/down left/right rule.

1st problem: Shrink the matrix size as we continue to further apply convolution. The image will get very small if we have many convolution layers.

2nd problem: Pixels on the corner of the image are used only once while the pixels in the centre of the image are used many times. This is uneven, loose of inf.

Solution: Padding.

SYMMETRIC PADDING

Add p extra columns and rows at the image borders with 0 values (zero padding). Output matrix size:

$$(n+2p-f+1) \times (n+2p-f+1)$$

“valid” convolution =>

no padding



		Blue			
		Green			2
Red	123	94	83	4	30
	34	44	187	92	124
	34	76	232	124	142
	67	83	194	202	

“same” convolution =>

Pad so that output size is the same as the input size. Formula for choosing p
(f is usually odd number!):

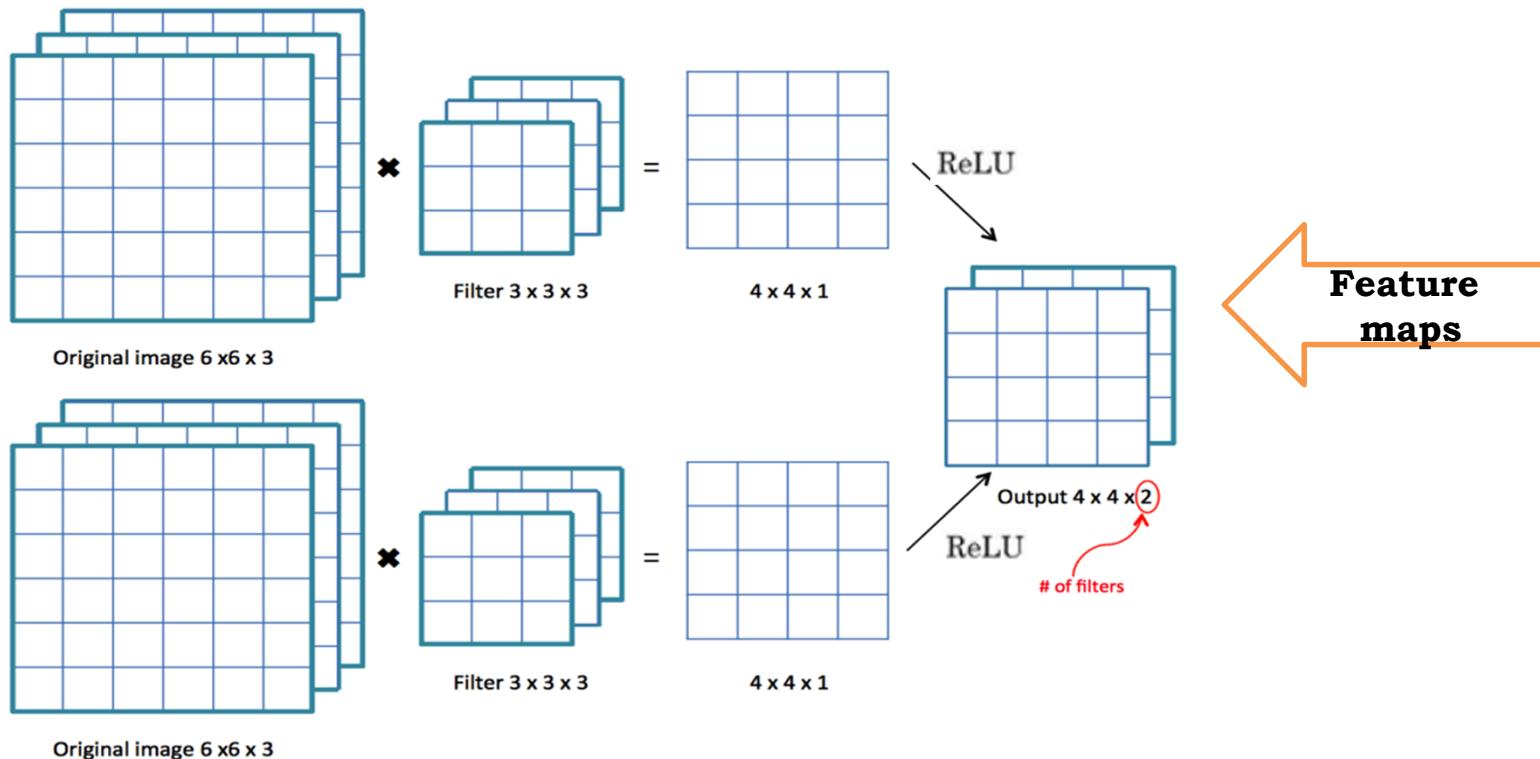
$$\begin{aligned} n + 2p - f + 1 &= n \\ 2p - f + 1 &= 0 \\ 2p &= f - 1 \\ p &= (f - 1)/2 \end{aligned}$$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	123	94	83	2	0	0	0
0	0	34	44	187	92	0	0	0
0	0	34	76	232	124	0	0	0
0	0	67	83	194	202	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	123	94	2	4	0	0	0
0	0	11	3	22	192	0	0	0
0	0	12	4	23	34	0	0	0
0	0	194	83	12	94	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	123	94	83	2	0	0	0
0	0	34	44	37	30	0	0	0
0	0	34	114	234	124	0	0	0
0	0	49	18	204	142	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

ONE CONV LAYER OF CNN



Different 3D filters (kernels) are applied to the 3D input image and the result matrices are stacked to form a 3D output volume.

After the convolution operation the result is passed through an activation function (e.g. ReLU, sigmoid, linear, etc.).

The outputs of the conv layers are known as feature maps.

Play conv kiank

POOLING (POOL)

Average Pool

2	3	1	9
4	7	3	5
8	2	2	2
1	3	4	5

→

4	4.5
3.25	3.25

Average Pool with
a 2 by 2 filter and
stride 2.

Max Pool

2	3	1	9
4	7	3	5
8	2	2	2
1	3	4	5

→

7	9
8	5

Max-Pool with a
2 by 2 filter and
stride 2.

Pooling operation reduces the size of the representation to speed up the computation and make the features more robust.

Ex. Divide the input in regions (e.g. 2×2 filter), choose stride (e.g. $s=2$), each output will be the max (**max pooling**) or the average (**average pooling**) from the corresponding regions.

Some intuition:

Large number means there is some strong feature (edge, eye) detected in this part of the image, which is not present in another part.

Max Pool: whenever this feature is detected it remains preserved in the output.

Softmax Layer

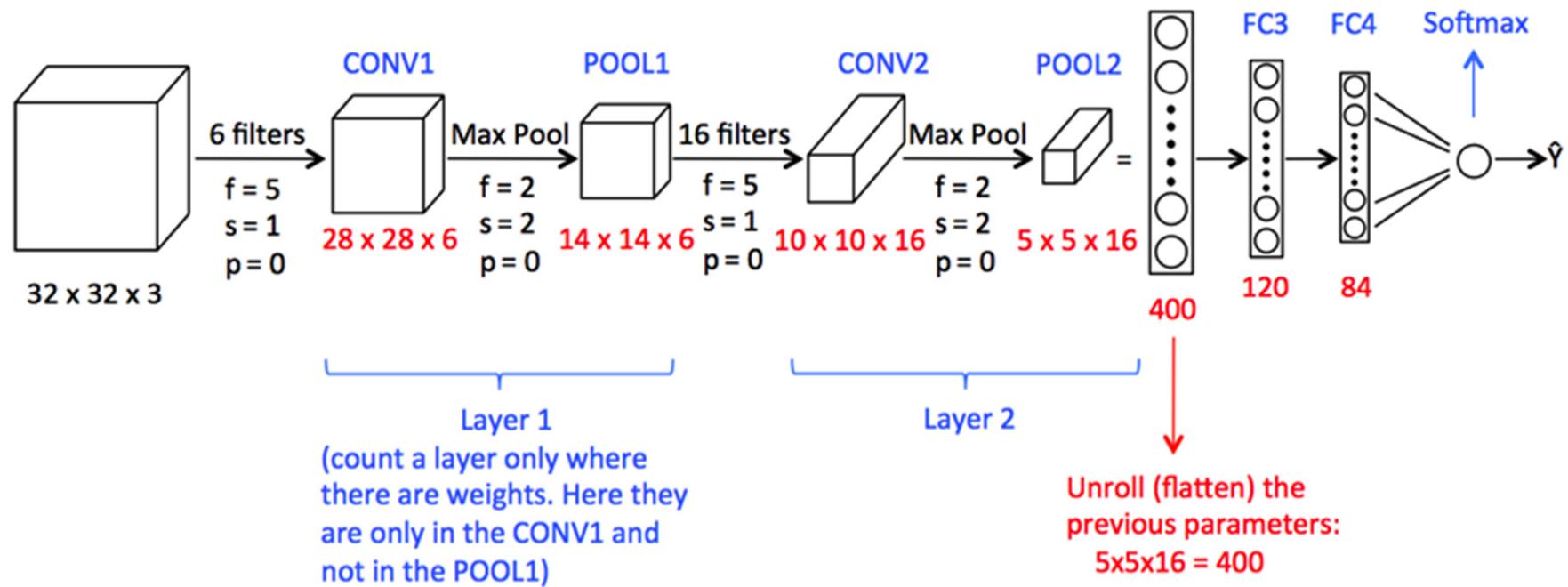
Softmax Layer (SL) estimates the probability that an example $x^{(i)}$ belongs to each of the k classes ($j=1,2,\dots,k$).

$$p(y^{(i)} = j | x^{(i)}; \theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}}$$

SL outputs k dimensional vector with estimated probability for each class k :

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}; \theta) \\ p(y^{(i)} = 2 | x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k | x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^k e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

CNN - LeNet5*



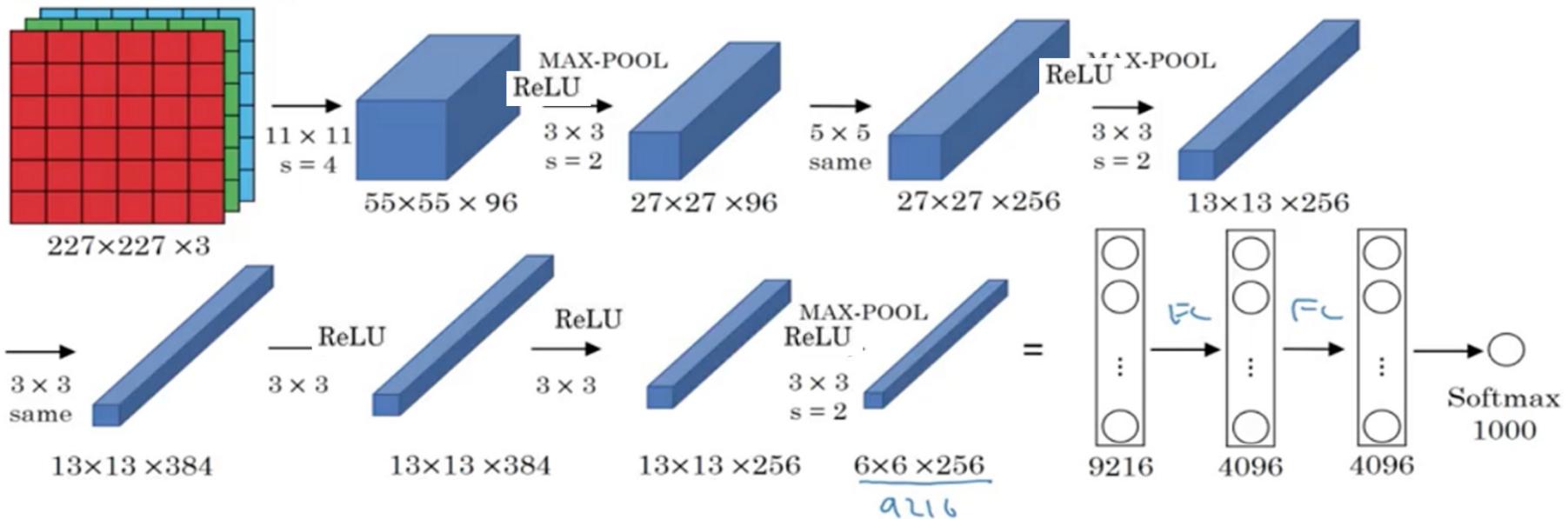
*LeCun et al., 1998, “**Gradient-based learning applied to document recognition**”. Original LeNet5 applied to handwritten digit recognition (grey scale images). Avg pooling, no padding, softmax classifier; ReLU and sigmoid/tanh neurons in the Fully Connected (FC) layers.

The activation function is always present after the convolution, even if it is not drawn on the CNN diagram.

General trend: CNNs start with large image, then height and width gradually decrease as it goes deeper in the network, whereas the number of channels increase.

CNN - AlexNet*

AlexNet



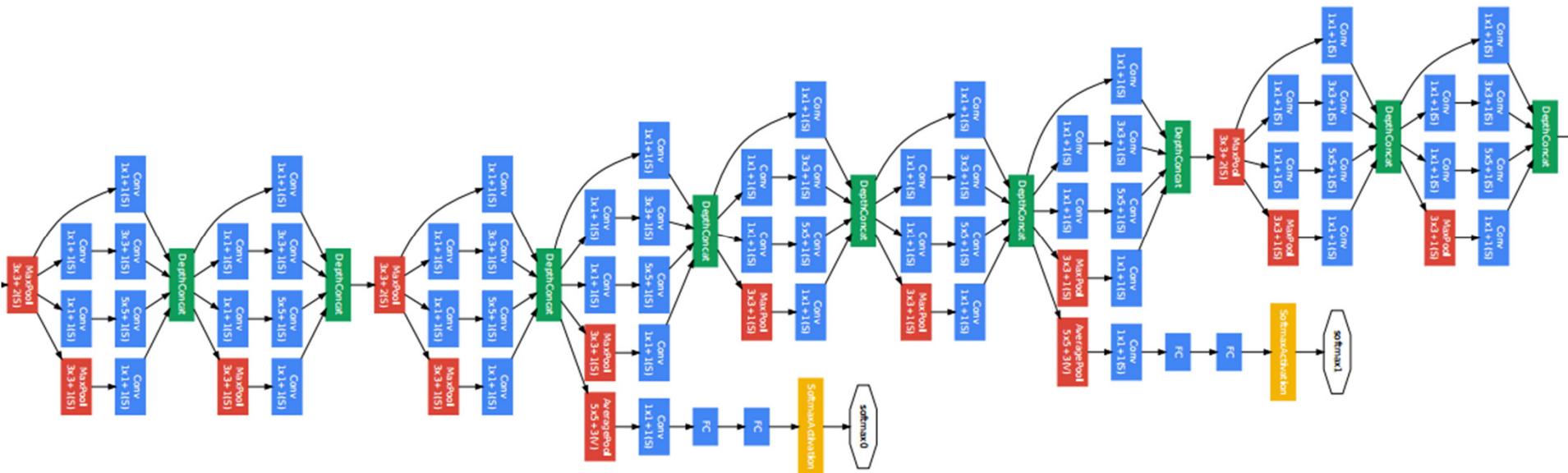
* **Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012, ImageNet classification with deep convolutional neural networks.**

5 conv layers, 3 FC layers with softmax output, 60 million parameters in total.

AlexNet applied to ImageNet LSVRC-2010 dataset to recognize 1000 different classes.

This paper convinced the Computer Vision (CV) community that DL really works and will have a huge impact not only in CV but also in speech/language processing. 22

INCEPTION NETWORK (GoogLeNet)



Ref. Szegedy et al 2014, “Going deeper with Convolutions”. Around 25 mln. Of parameters



You Only Look Once (YOLO) CNN Architecture

Image classification/localization/detection

Image classification	Classification & Localization	Detection
	 b_x, b_y, b_h, b_w	 multiple objects

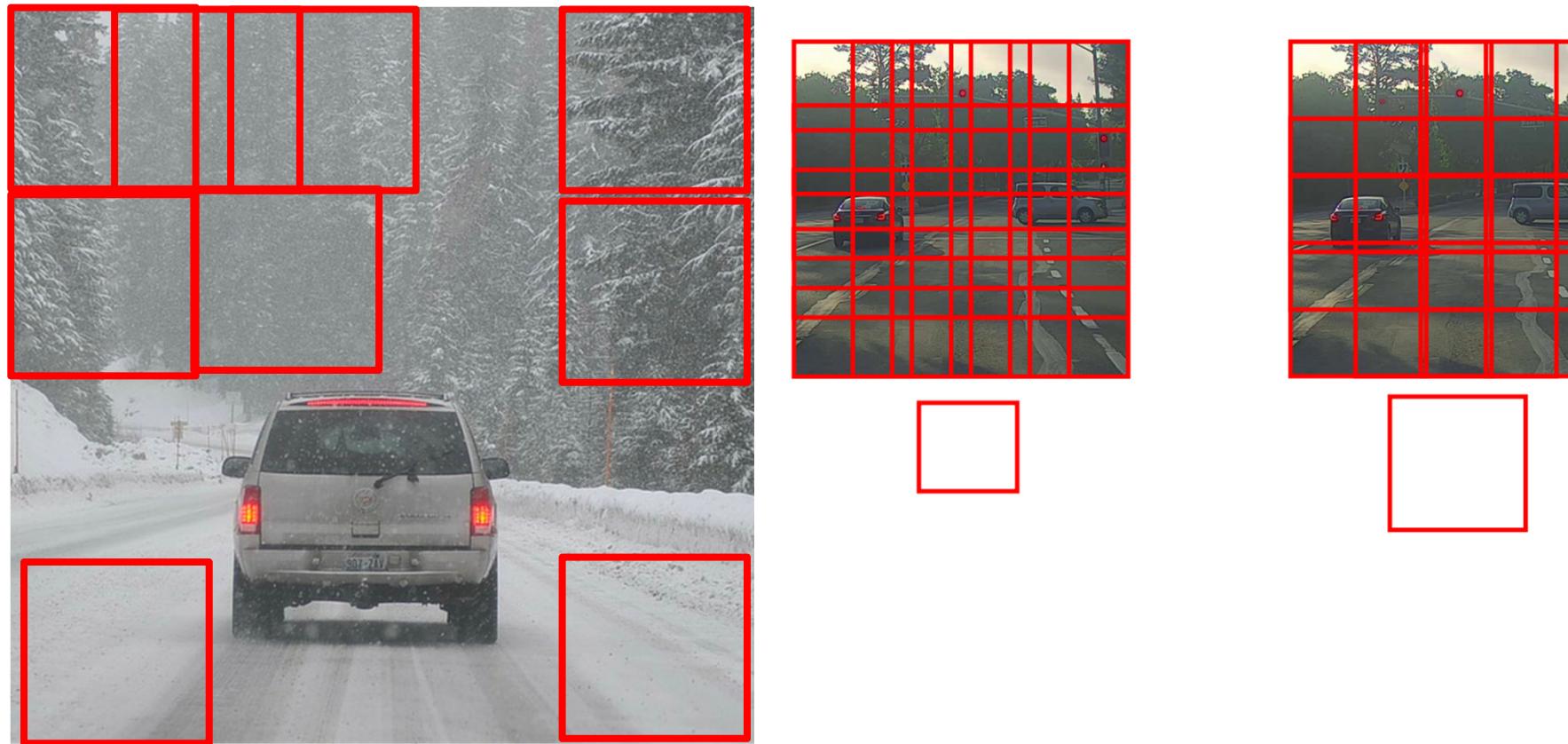
Image classification: input a picture to CNN and the output is a class label (e.g. person, bike, car, background, etc.)

Classification with localization: the algorithm gives not only the class label of the object but also draws a bounding box (the coordinates) of its position in the image.

Standard notation: (0,0) as the upper-left corner and (1,1) to be the lower-right corner.

(b_x, b_y, b_h, b_w) describes the bounding box.

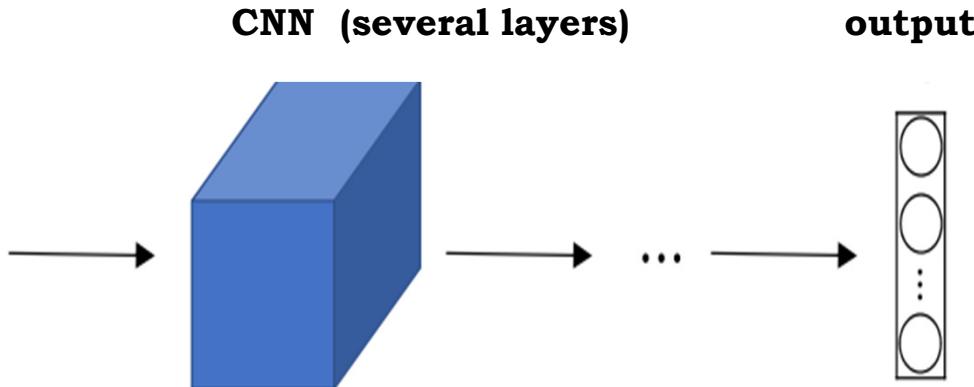
Sliding windows detection algorithm



Pick a certain window size, input this sub-picture into already trained CNN to detect objects. Then shift the detection window to the right with one step (stride) and feed the new sub-picture into ConvNet. Go through every region (the stride needs to be small for the detection algorithm to run smoothly). Repeat the same with different sizes of the detection window in order to detect objects with different sizes of the picture.

The Sliding Windows object detection algorithm has **infeasibly high computationally cost**.

Object classification with localization



Classes (e.g.):

1. person
2. Bikes
3. Car
4. Background (no object)



b_x, b_y, b_h, b_w

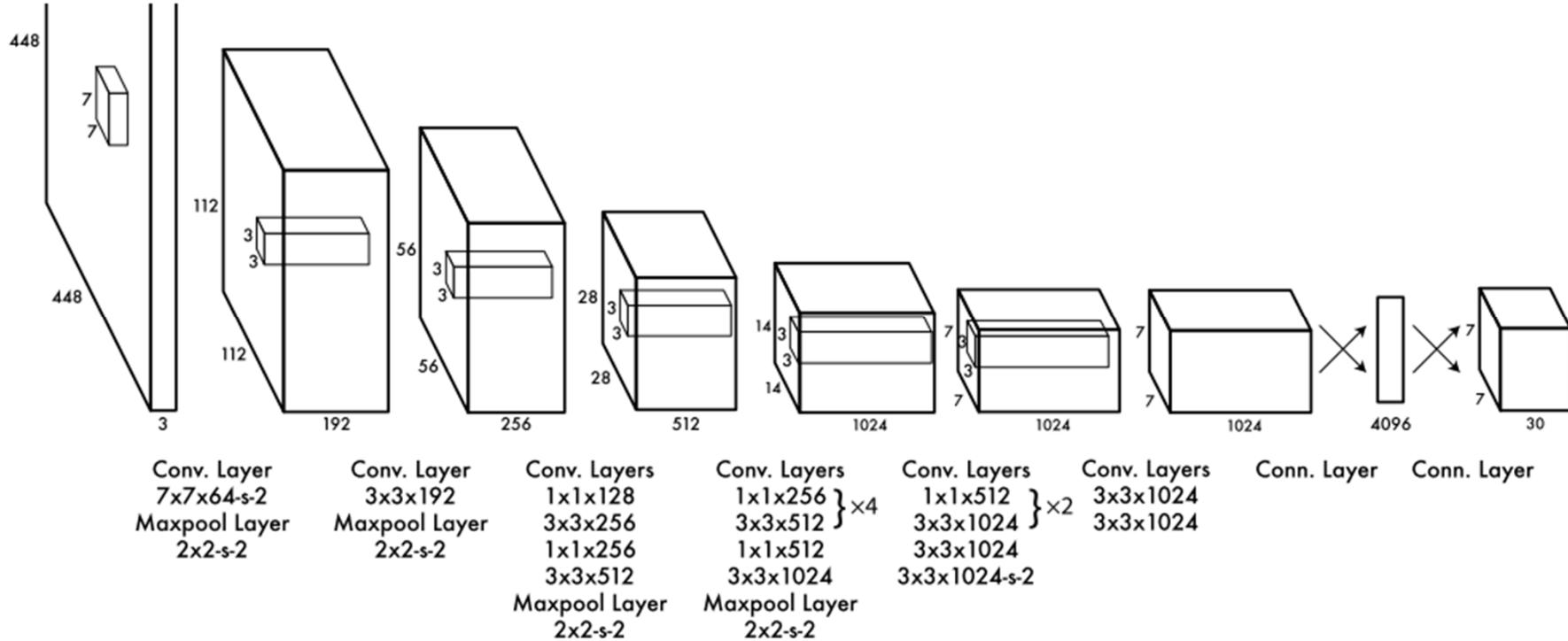
Output: $(p_c, b_x, b_y, b_h, b_w, c = [c_1, c_2, \dots, c_{end}])$
 p_c – is there an object or not (1/0)

<= Image label: [1, $b_x, b_y, b_h, b_w, 0, 0, 1]$

<= Image label: [0, ?, ?, ?, ?, ?, ?, ?, ?, ?]
? – “don’t care”



YOLO (You Only Look Once)



YOLO - CNN network for both classification and localising the object using bounding boxes.

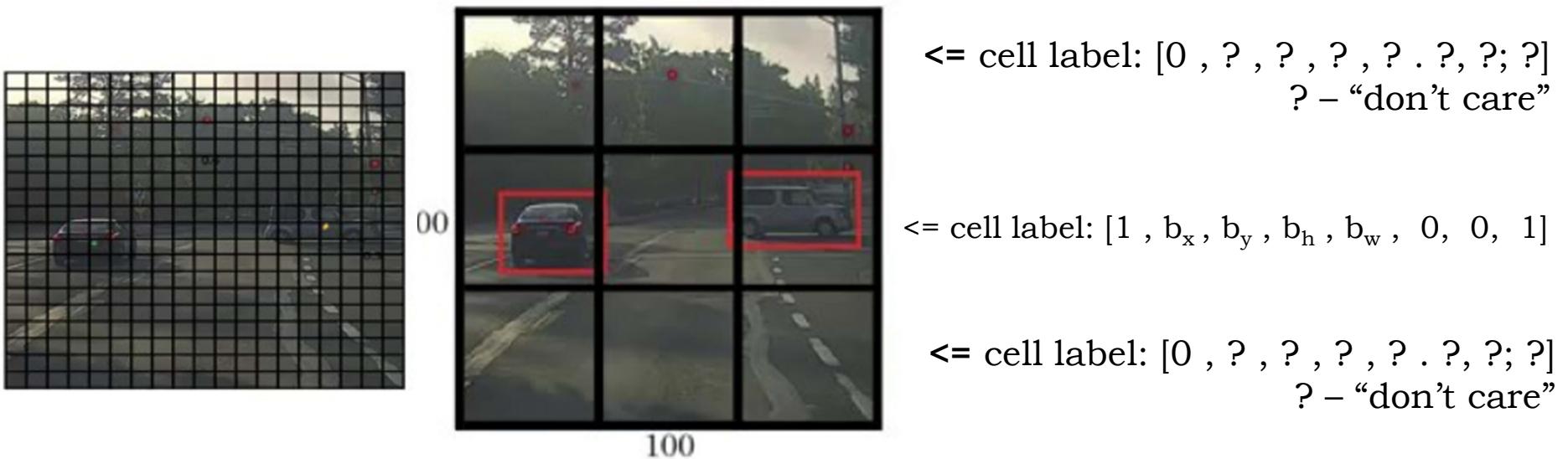
24 convolutional layers + 2 fully connected layers.

Conv layers pretrained on ImageNet dataset.

*Redmon et al, 2015, “You Only Look Once: Unified, Real-Time Object Detection” (<https://arxiv.org/abs/1506.02640>)

Redmon & Farhadi, 2016 (<https://arxiv.org/abs/1612.08242>).

YOLO (You Only Look Once) algorithm



Let we have 100x100 pixels input image. We place a grid on this image.
In the original paper the grid is 19x19, here for illustration is 3x3 grid (9 cells).
Apply object classification and localization to each cell.

How to define the labels used for training:

For each cell, the label is 8 dimensional vector (if we have 3 classes)

$y = [p_c, b_x, b_y, b_h, b_w, c1, c2, c3]$, where

- p_c (0 or 1) specifies if there is or not an object in that cell.
- $[b_x, b_y, b_h, b_w]$ specify the bounding box if there is an object in that cell.
- $c1; c2; c3$ (0 or 1) - to assign the class (e.g. person, bicycle, car)

YOLO assigns the object only to the cell containing the midpoint of the object.

Since we have 3x3 grid, the total volume of the target output Y is 3x3x8.

Sequence Models - Long-Short Term Memory (LSTM)

Examples of sequence data

Speech recognition



y (output)

==> “The quick brown fox jumped over the lazy dog.”

Sentiment classification

“There is nothing to like in this movie.”

==>



DNA sequence analysis

AGCCCCTGTGAGGAACTAG

==> AGCCCCTGTGAGGAACTAG

Machine translation

Voulez-vous chanter avec moi?

==> Do you want to sing with me?



==> Running

Video activity recognition

Name entity recognition

Yesterday, Harry Potter met Hermione Granger.

==> Yesterday, Harry Potter met Hermione Granger.

x and y are both sequences, or only x is a sequence, or only y is a sequence.

Sequence Model Notation

Name Entity Recognition application is to find people's names, companies names, locations, countries names, currency names, etc. in text.

Given an input sequence of words (X), the sequence model has to automatically tell where are the proper names in this sentence.

x : Harry Potter and Hermione Granger invented a new spell.

$x^{<1>} \quad x^{<2>} \quad x^{<3>} \quad \dots \quad x^{<9>}$

For this example the sequence length is 9 words (features) => $T_x=9$

Target output y is binary vector with same length as the input
(1 if the word is name; 0 if not)

$y = [\begin{matrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}]$
 $y^{<1>} \quad y^{<2>} \quad y^{<3>} \quad \dots \quad y^{<t>} \quad y^{<T_y>}$
 $T_y=9$

In this problem every input $x^{<i>}$ has an output $y^{<i>} \Rightarrow \text{length } T_y=T_x$
Each sentence (example) can have different sequence length.

NLP - representing individual words

Natural Language Processing (NLP). Create vocabulary or download existing dictionary.
For modern NLP, 10000 words is a small dictionary, 30-50 thousand is more common.
Large internet companies use 1 million words dictionary.

Each word is represented by a binary vector with # elements = dimension of dictionary
where only the index of the word in the dictionary = 1, all others are 0 (one-hot vector).

word index (in dictionary)

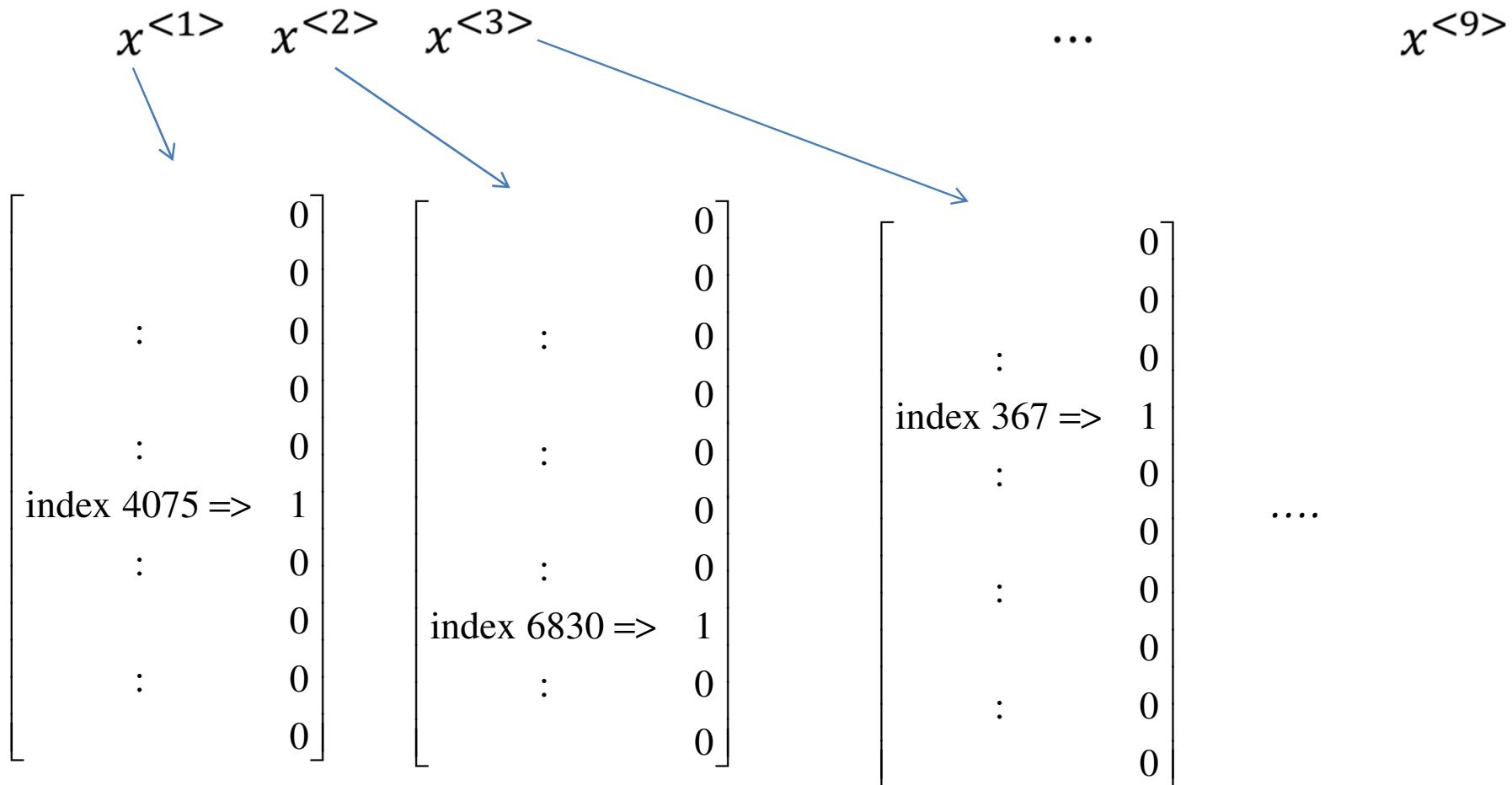
$a \Rightarrow$	1
$aaron \Rightarrow$	2
:	
$and \Rightarrow$	367
:	
$Harry \Rightarrow$	4075
:	
$Potter \Rightarrow$	6830
:	
$zulu \Rightarrow$	10000

one-hot encoding

$Harry =$	[0
		0
	:	:
		0
	:	:
		1
	:	:
		0
	:	:
		0

NLP – one hot encoding

x: Harry Potter and Hermione Granger invented a new spell.



<unk> - notation for words not in the dictionary. It can be added to encode all missing words that may appear in sentences.

The problem is formulated as supervised learning with labelled data (x, y) .³⁴

Recurrent Neural Networks (RNN)

Read the sentence word by word (one time step per word).

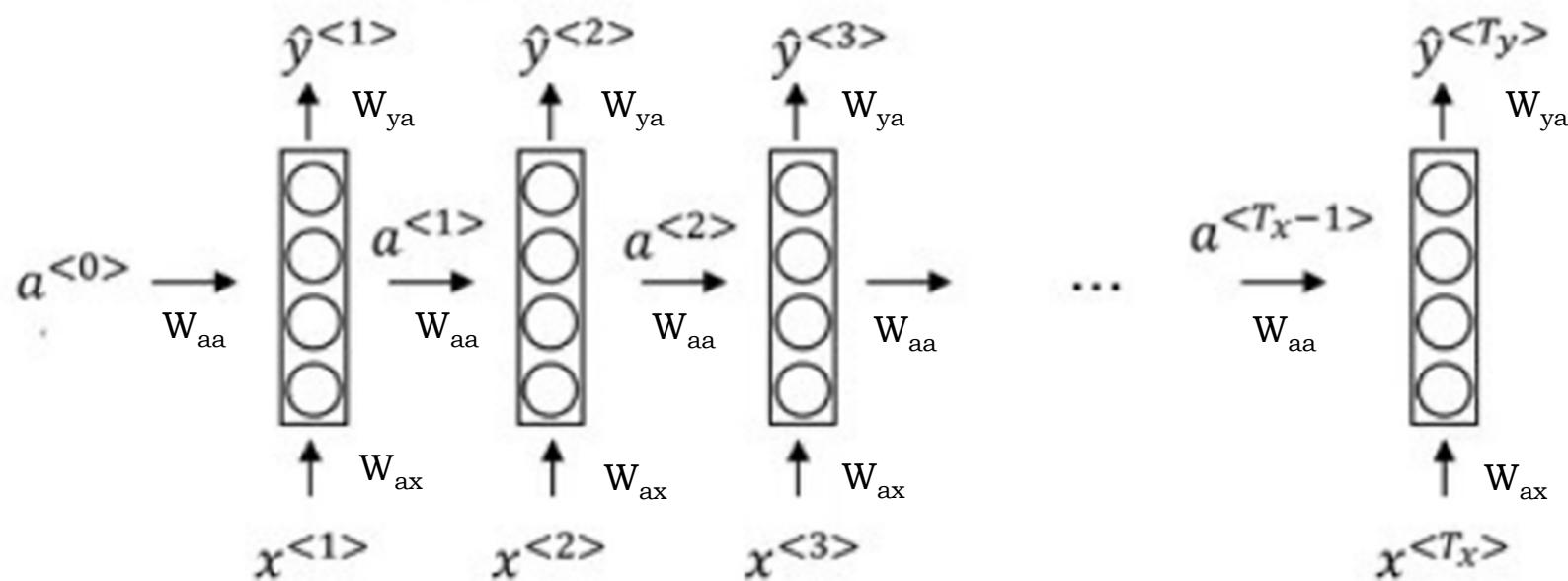
Activation produced by 1st word is taken into account when read 2nd word.

Notation: inputs - $x^{<t>}$; outputs - $y^{<t>}$; hidden states - $a^{<t>}$

$a^{<0>}$ - initial state at time step 0, usually vector of zeros.

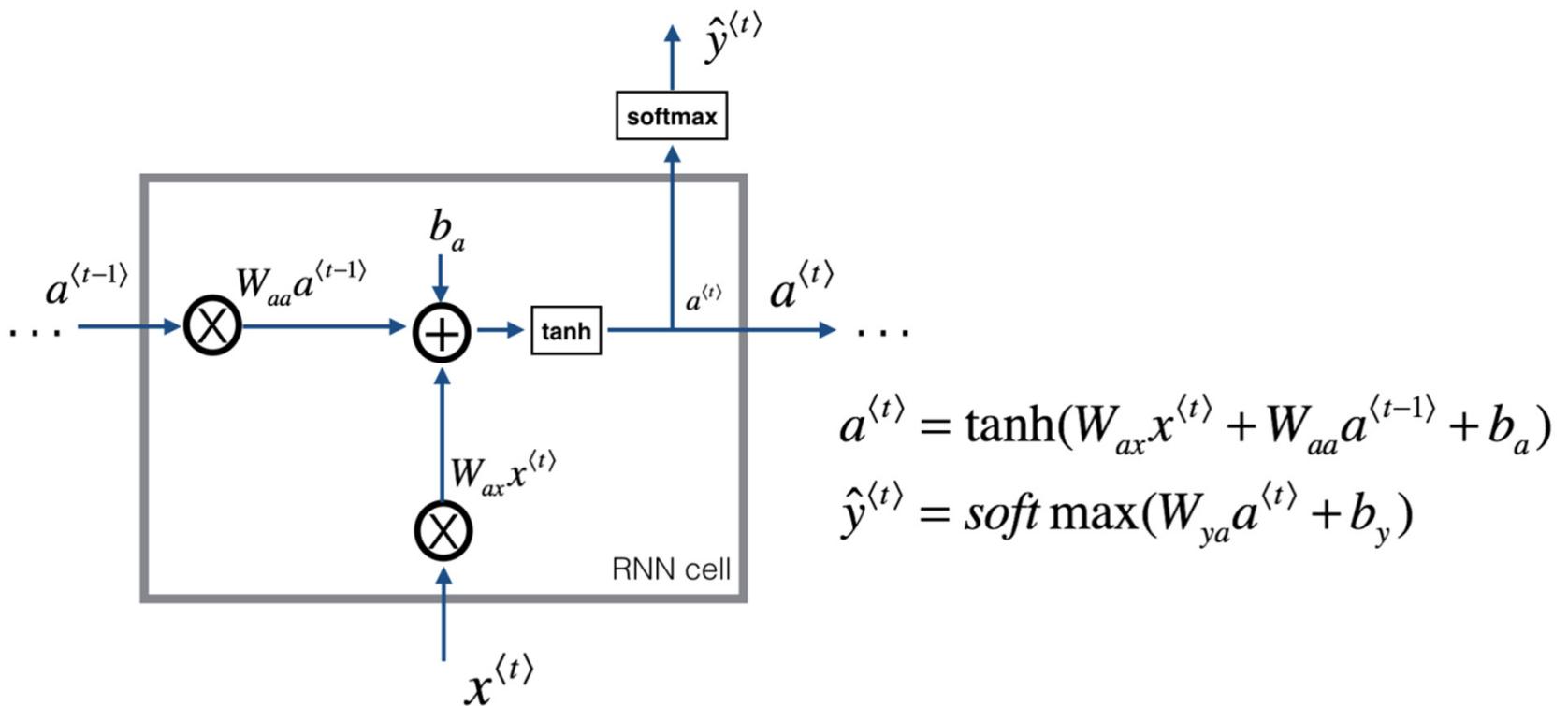
Previous results are passed as inputs => we get context !

Fig. Flow chart of RNN representation



Basic RNN unit

Takes $x^{(t)}$ (current input) and $a^{(t-1)}$ (activation from previous step) as inputs, and computes $a^{(t)}$ (current activation). $a^{(t)}$ is then used to predict $y^{(t)}$ (current output) and passed forward to the next RNN unit (next time step). W_{aa} , W_{ax} , W_{ya} , b_a , b_y – the same RNN weights used in all time steps.

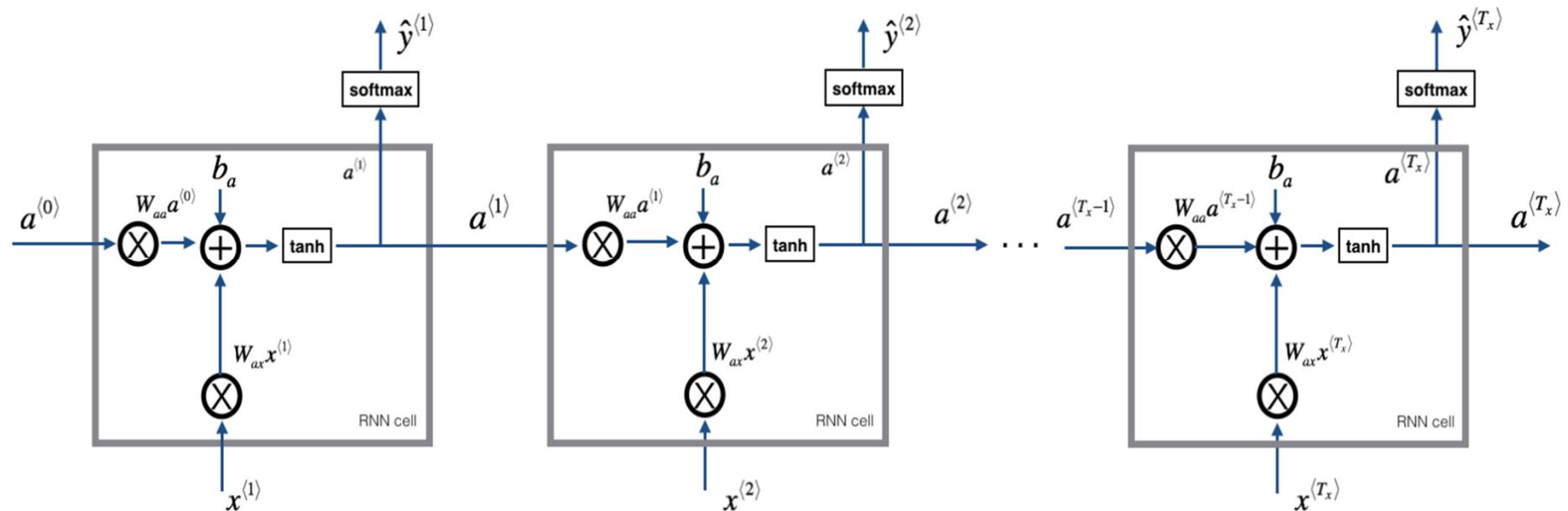


RNN forward pass

RNN is a sequence of basic RNN cells.

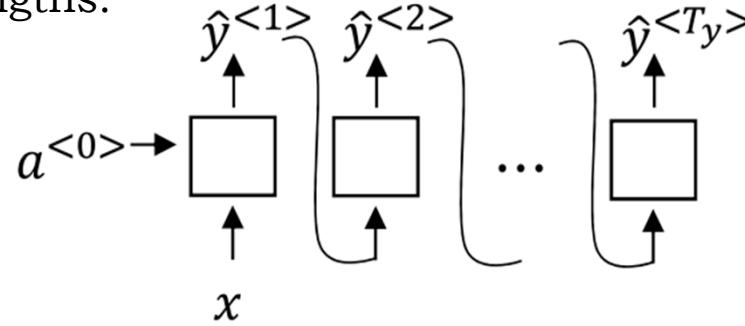
If input sequence is $x = [x^{<1>} , x^{<2>} , \dots x^{<T_x>}]$ => RNN cell is copied T_x times.

RNN outputs $y = [y^{<1>} , y^{<2>} , \dots y^{<T_x>}]$

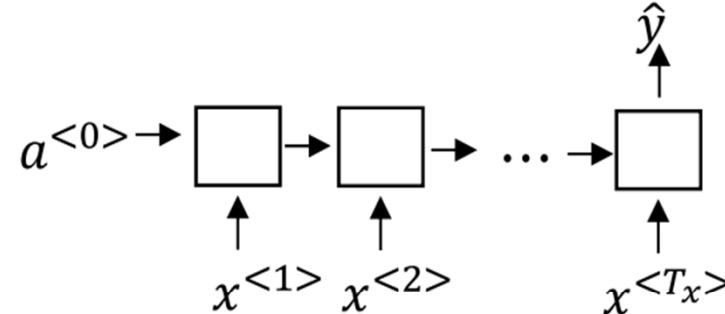


Different Types of RNN

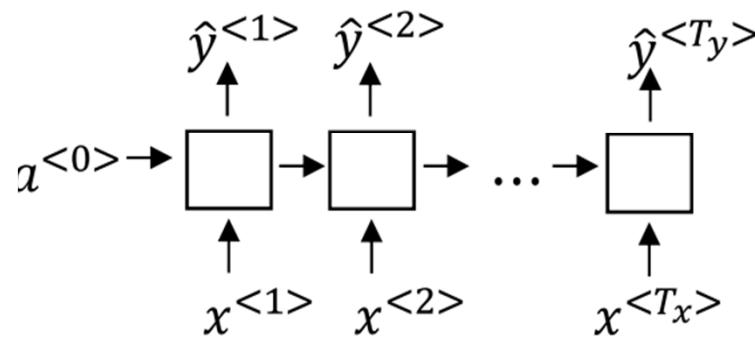
The input X and Y can be of many types and they do not have to be of the same lengths.



One to many (e.g. Music generation)

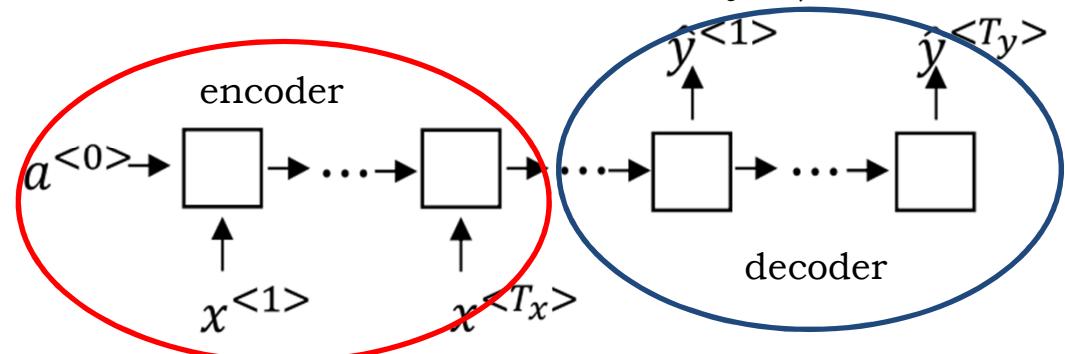


Many to one (e.g. Sentiment analysis)



Many to many

$T_x = T_y$ (e.g. Name entity rec.)

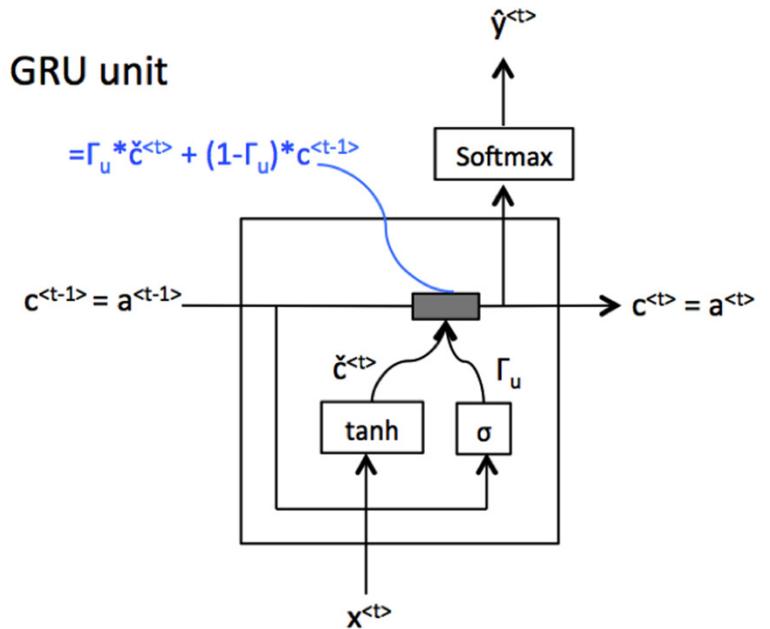


Many to many

T_x different from T_y (e.g. machine translation)

Note: Time series forecasting (many to one, or many to many)

Gated Recurrent Unit (GRU) vs basic RNN unit



c - memory cell

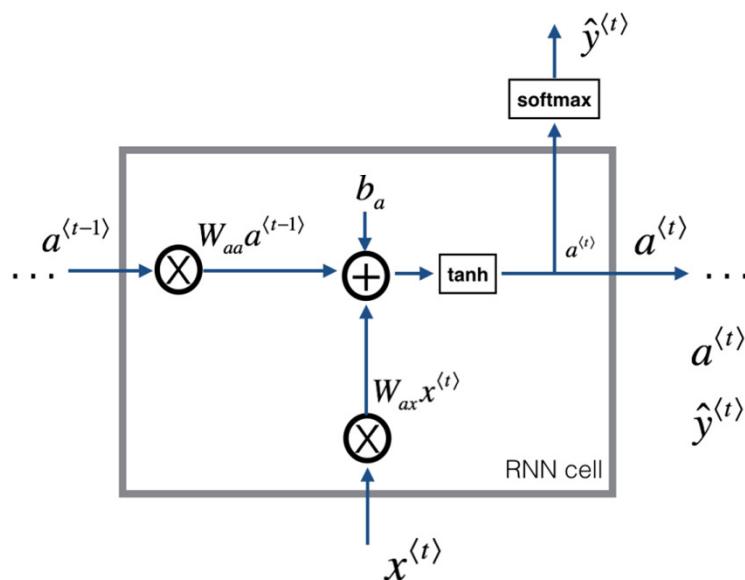
$c^{<t>} = a^{<t>} \quad$ (for LSTM they are different)

GRU outputs activation value = memory cell

$\tilde{c}^{<t>} = \tanh(w_c [c^{<t-1>}, x^{<t>}] + b_c)$ – candidate

$\Gamma_u = \sigma(w_u [c^{<t-1>}, x^{<t>}] + b_u)$ – update gate

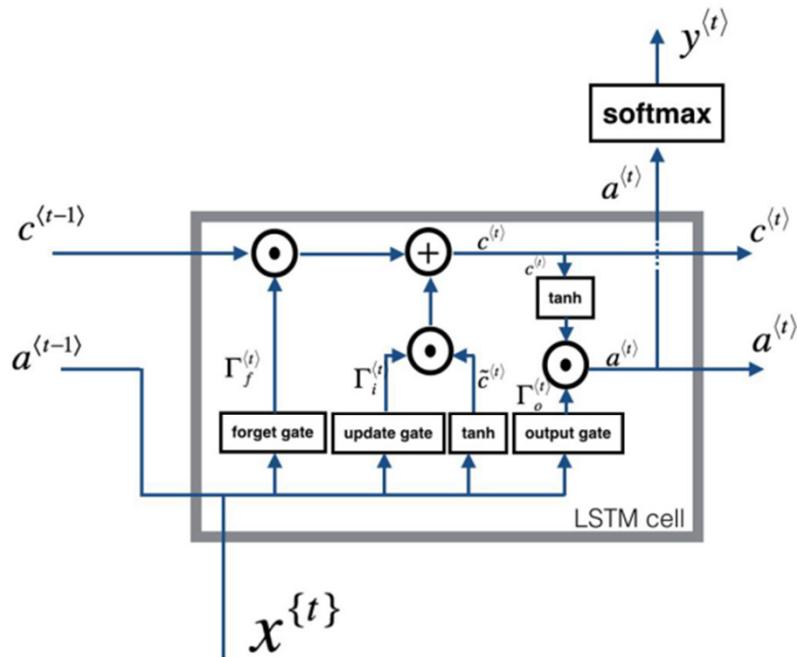
$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>} \quad$ – update memory cell



$$a^{(t)} = \tanh(W_{ax}x^{(t)} + W_{aa}a^{(t-1)} + b_a)$$

$$\hat{y}^{(t)} = \text{softmax}(W_{ya}a^{(t)} + b_y)$$

Long Short-Term Memory (LSTM) unit vs RNN



$$\tilde{c}^{(t)} = \tanh(W_c [a^{(t-1)}, x^{(t)}] + b_c) - \text{candidate}$$

$$\Gamma_u = \sigma(W_u [a^{(t-1)}, x^{(t)}] + b_u) - \text{update gate}$$

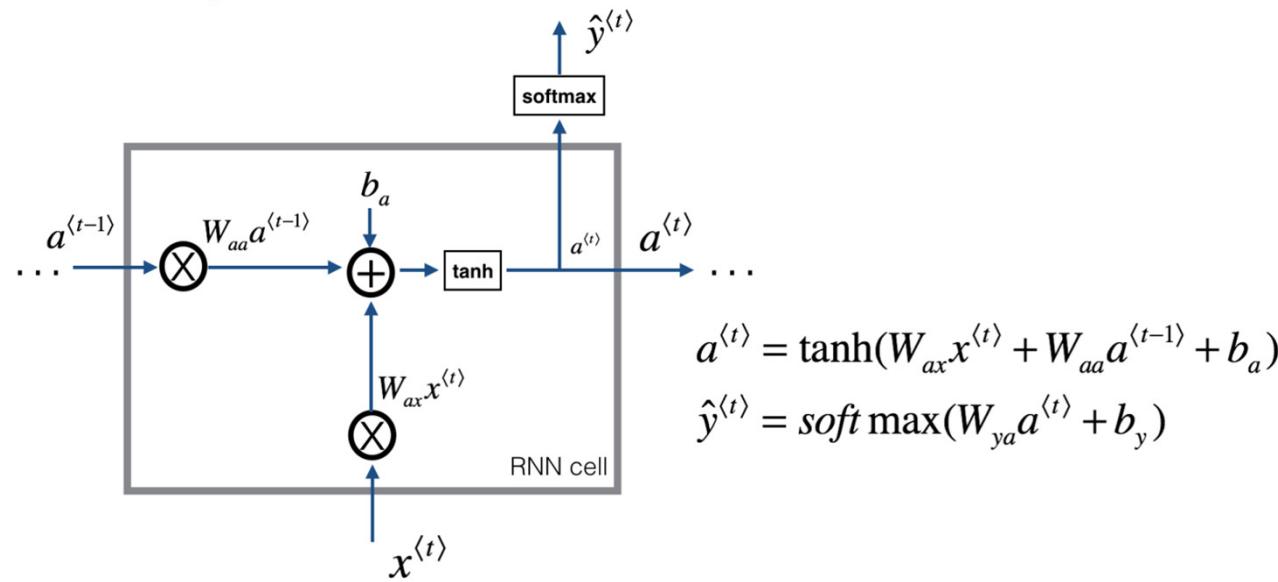
$$\Gamma_f = \sigma(W_f [a^{(t-1)}, x^{(t)}] + b_f) - \text{forget gate}$$

$$\Gamma_o = \sigma(W_o [a^{(t-1)}, x^{(t)}] + b_o) - \text{output gate}$$

$$c^{(t)} = \Gamma_u * \tilde{c}^{(t)} + \Gamma_f * c^{(t-1)} - \text{update memory cell}$$

$$a^{(t)} = \Gamma_o * \tanh(c^{(t)}) - \text{activation}$$

LSTM outputs both activation value and memory cell



$$a^{(t)} = \tanh(W_{ax}x^{(t)} + W_{aa}a^{(t-1)} + b_a)$$

$$\hat{y}^{(t)} = \text{softmax}(W_{ya}a^{(t)} + b_y)$$

GRU & LSTM

LSTM and GRU are two variations of RNNs to capture better long range dependencies (connections) in sequences.

They mitigate short-term memory using mechanisms called memory cell and gates.

Gates remember (keep saved) bits of info for many time steps.

Ex.: Read text, and use LSTM/GRU to keep track of grammatical structures
=> if the subject is singular or plural. If the subject changes from a singular word to a plural word, forget the previously stored memory value of the singular state.

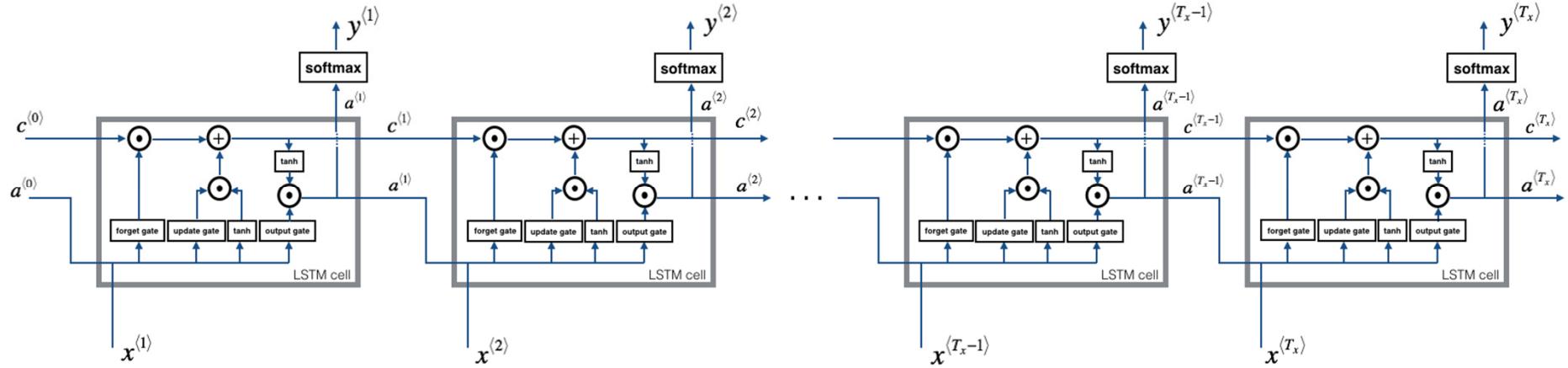
LSTM's and GRU's are successfully applied in speech recognition, speech synthesis, natural language understanding, time series forecasting, etc.

When to use LSTM or GRU ? - there is not a consensus.

LSTM appeared first (1997), GRU (2014) is a simplification of LSTM.
LSTM is more powerful (3 gates in LSTM, 1 gate in GPU).

* ref. Hochreiter and Schmidhuber 1997, "Long short-term Memory".

LSTM network



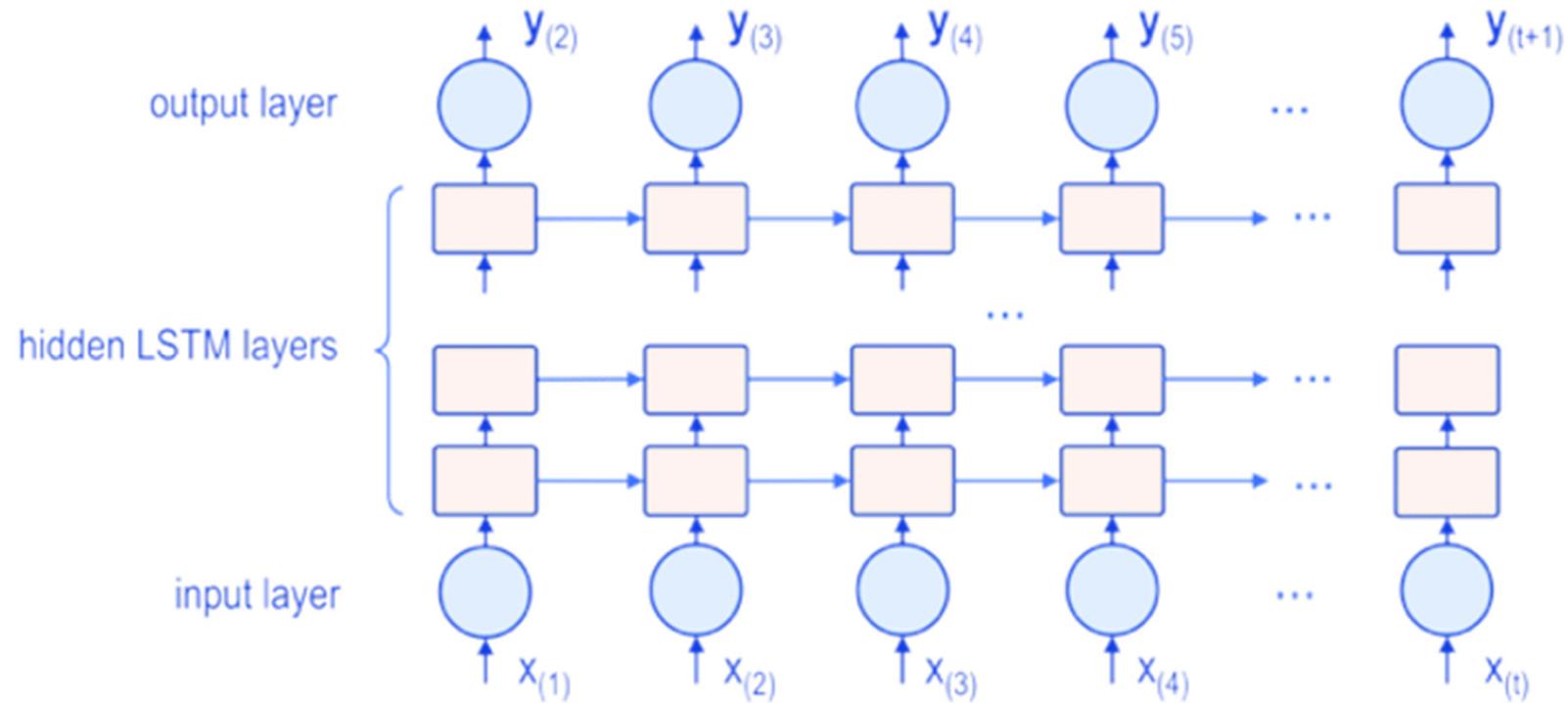
LSTM network is a temporal sequence of LSTM units.

There is a line at the top that shows how LSTM can memorise and pass certain values $c^{<t>}$ through several temporal steps.

Other versions : the gate's values depend also on $c^{<t-1>}$. (peephole connection)

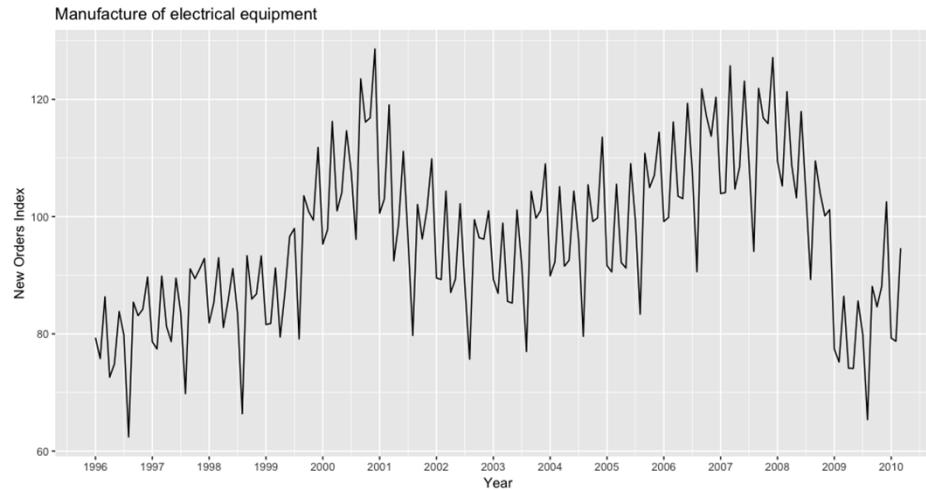
The gates are vectors (e.g. if 100 dimensional hidden memory cell units, the gates will have 100 elements).

Deep LSTM



Time Series Data

Sequence Models for Time Series Data



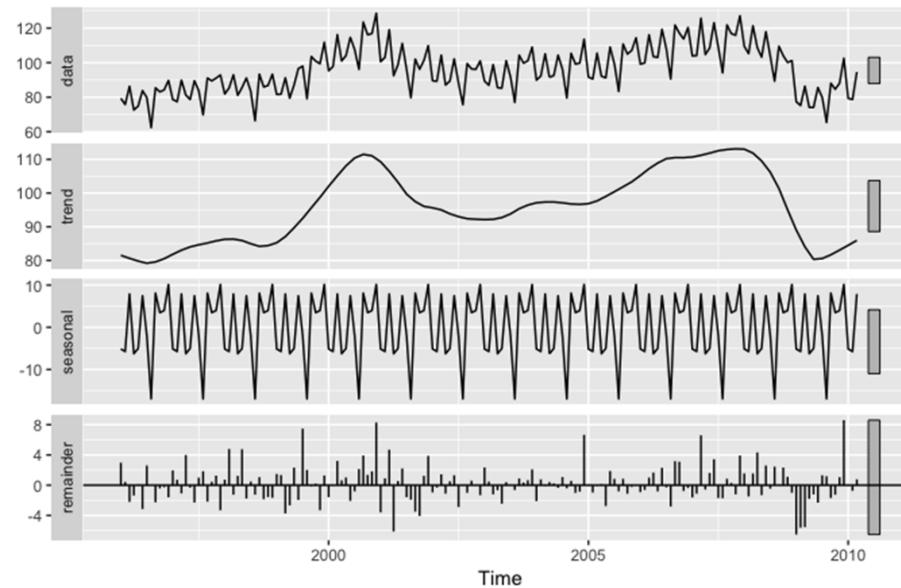
Time Series (TS) - collection of data points indexed based on the time they were collected => TS models are naturally sequence models.
Most often, data is recorded at regular time intervals.

TS forecasting (prediction) is a hot topic with many applications in practice:

- Stock prices forecasting / Weather forecasting
- Bitcoin price forecasting
- Biological sensors (ECG, EEG) – predict health problems
- Network traffic prediction

Time Series Decomposition

$Y(t) \Rightarrow$



$T(t) \Rightarrow$

$S(t) \Rightarrow$

$R(t) \Rightarrow$

If TS shows some seasonality (e.g. daily, weekly, yearly) it may be useful to decompose the original TS $Y(t)$ into sum of 3 components:

$$Y(t) = S(t) + T(t) + R(t)$$

$T(t)$ - trend component, estimate $T(t)$ through the mean of the last n samples (rolling mean)

$S(t)$ - seasonal component : $S(t) = Y(t) - T(t)$

$R(t)$ - remaining component: $R(t) = Y(t) - T(t) - S(t)$

Trend component in TS

Trend is a long-term increase or decrease in the level of the time series (TS). Systematic change in TS that does not appear to be periodic is known as a trend.

Identify a Trend => Plot TS data to see if a trend is obvious or not.

Remove a Trend => TS with a trend is called non-stationary. Identified trend can be removed (TS de-trending).

If TS does not have a trend or the trend is successfully removed, the dataset is said to be trend stationary.

To get stationary TS, subtract the trend => $S(t) = Y(t) - T(t)$

To get the trend, subtract the seasonality => $T(t) = Y(t) - S(t)$

$Y(t)$ – original time series

Before forecasting, it is helpful to remove both the trend and seasonality.

Time Series Forecasting – classical methods

1) Exponential smoothing

The forecasts are equal to a weighted average of past observations and the corresponding weights decrease exponentially as we go back in time.

$$\hat{Y}(t+h | t) = \alpha Y(t) + \alpha(1-\alpha)Y(t-1) + \alpha(1-\alpha)^2Y(t-2) + \dots, \quad 0 < \alpha < 1 \quad (\text{basic model})$$

2) ARIMA (popular method for TS forecasting)

AutoRegressive model - linear combination of past values of the TS.

Moving Average model - linear combination of past forecasting errors.

ARIMA (AutoRegressive Integrated Moving Average) models combine the two models.

TS has to be stationary => instead of the original values use differences : $\hat{Y}(t) = Y(t) - Y(t-T)$

3) SARIMA model (Seasonal ARIMA) extends the ARIMA by adding a linear combination of seasonal past values and/or forecast errors.⁴⁸

Time Series Forecasting – with LSTM/GRU

Forecasting with classical methods (ARIMA, etc.) => work better with features (e.g. extract trend, seasonality, etc.) than with raw TS data.

Forecasting with Sequence models (LSTM, GRU, etc.) => no need to extract features, work with raw data.

Cut TS into smaller sequences, and use Sequence models to forecast it.

LSTM/GRU inputs: $Y(t), Y(t-1), \dots, Y(t-p)$

LSTM /GRU output (predict the next value): $Y(t+1), Y(t+2), \dots, Y(t+k)$

Further Reading

- Ian Goodfellow, and Yoshua Bengio, Deep Learning, MIT Press, 2016
- Andrew Ng, Machine Learning Yearning, 2018
[\(https://www.deeplearning.ai/machine-learning-yearning/\)](https://www.deeplearning.ai/machine-learning-yearning/)
- Understanding LSTM Networks – colah's blog :
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>