

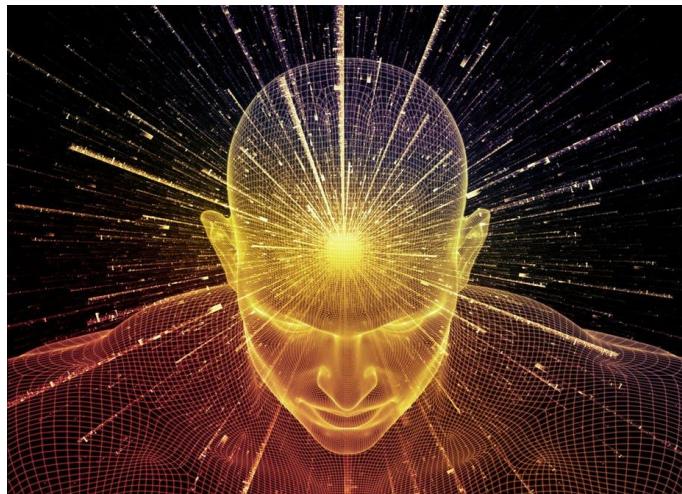
# Python for Data Science Machine Learning

Francesco Pugliese

DIRM-DCME-MEC  
[frpuglie@istat.it](mailto:frpuglie@istat.it)

# **WHAT IS CONSCIOUSNESS? WHAT IS THE SUBJECTIVE EXPERIENCE? NOBODY KNOWS**

**Cogito Ergo Sum (René Descartes)**



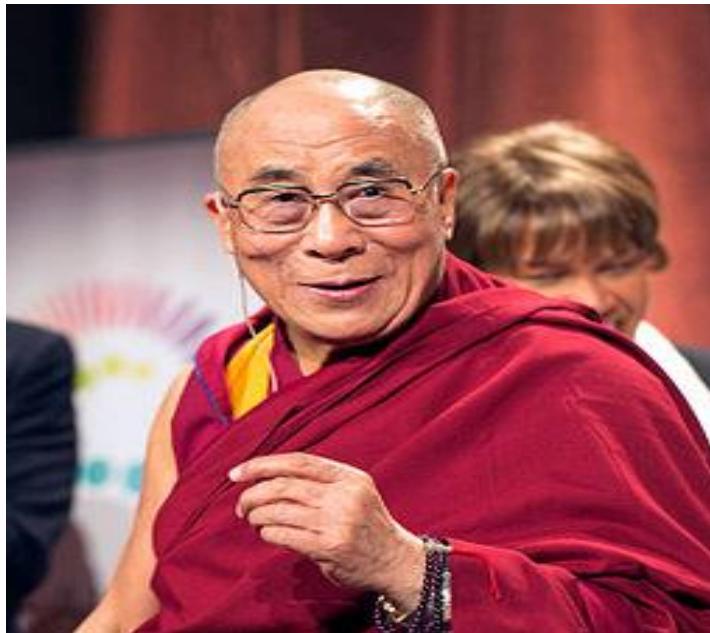
## **DISSOCIATION BETWEEN CORTICAL ACTIVITY AND CONSCIOUSNESS**

- During episodes of unconsciousness, the child's brain is by no means inactive; it is actually hyperactive.
- The observation of similar firing rates between waking and slow-wave sleep represents both the most striking dissociation between cortical activity and consciousness.
- It indicates that the presence of “normal” firing levels in the cerebral cortex is not a sufficient criterion for consciousness.

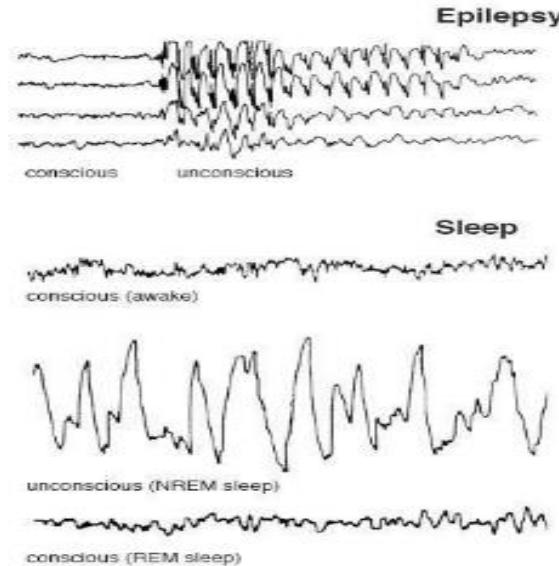
**Edelman, Gerald. A Universe Of Consciousness:  
How Matter Becomes Imagination**

# WHAT IS CONSCIOUSNESS? WHAT IS THE SUBJECTIVE EXPERIENCE? NOBODY KNOWS

How Thoughts can change the Brain



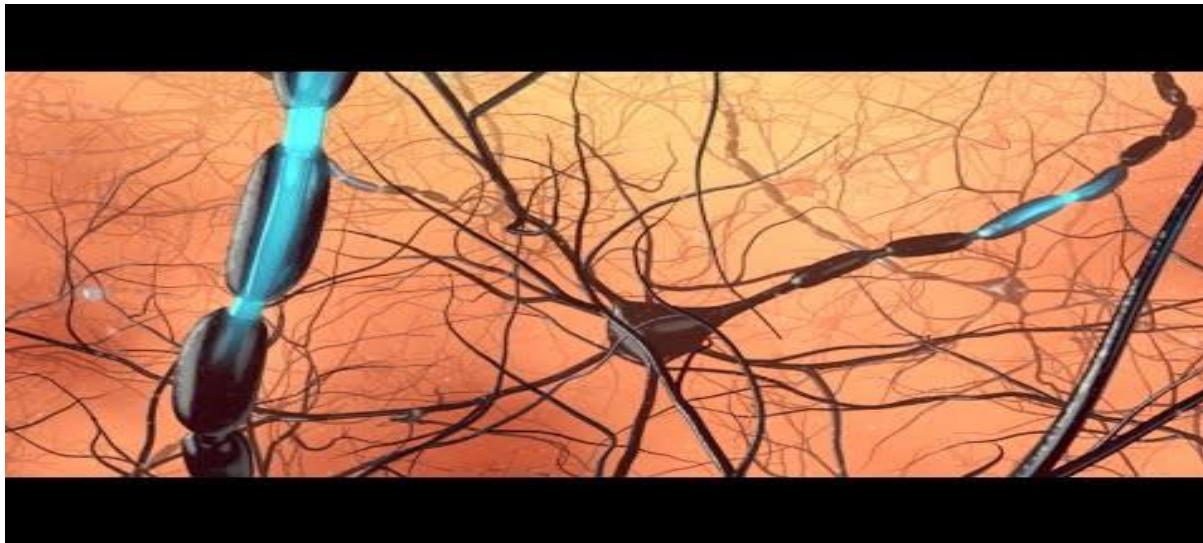
Conscious Experience requires PATTERNS OF NEURAL ACTIVITY THAT ARE HIGHLY DIFFERENTIATED (Gerald Edelman



Tenzin Gyatso: “What if the brain comes from consciousness instead of consciousness coming from the brain?”

# HUMAN BRAIN VS CONSCIOUSNESS

An High Dimensional and Complex Neural Network



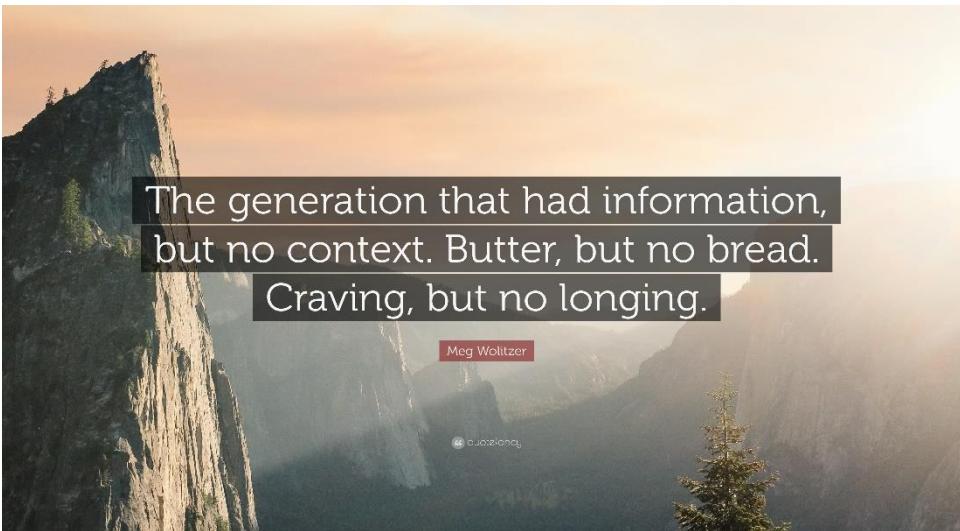
- The lack of a formal link between neural network structure and its emergent function has hampered our understanding of how the brain processes information.

Scientists find evidence of a Multidimensional Universe within the Brain



Reimann, M. W., Nolte, M., Scolamiero, M., Turner, K., Perin, R., Chindemi, G., ... & Markram, H. (2017). Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in computational neuroscience*, 11, 48.

# MACHINE LEARNING VS CONSCIOUSNESS



**Is this a Glass or a Pot?**



## Where is the Human-Like Thoughts Flow?

**Why and How Lea Seadol beats Alpha  
Go Zero  
once?**

**Even though all metrics say it is  
impossible.**



# MACHINE LEARNING LIMITS

Will ever Self Driving Car cope with Mumbai Traffic?



Will ever AI have context information ?

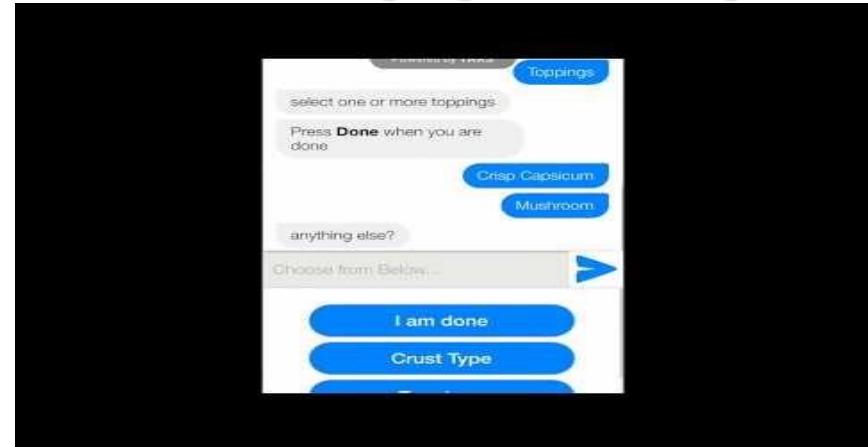


# MACHINE LEARNING BREAKTHROUGHTS

## Computer Vision



## Natural Language Processing



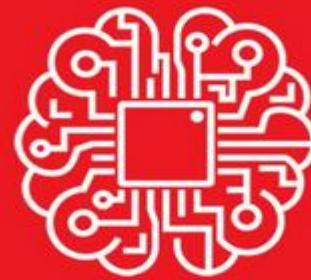
## A truly general Artificial Intelligence



## A Generative Artificial Intelligence



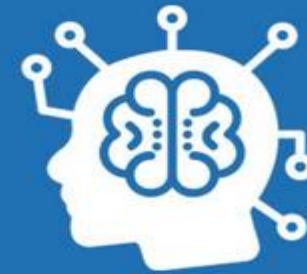
# Difference between AI and ML



## ARTIFICIAL INTELLIGENCE

---

If it is written  
in **PowerPoint**,  
It's probably  
**Artificial**  
**Intelligence**



## MACHINE LEARNING

---

If it is written  
in **Python**;  
It's probably  
**Machine**  
**Learning**

# MACHINE LEARNING ENGINEER



**What My Parent Think I do**



**What My Friend Think I do**



**What Society Think I do**



**What Media Think I do**



**What I Think I do**

```
1 import tensorflow as tf  
2 import torch  
3
```

**What I really do**

# MACHINE LEARNING IN OFFICIAL STATISTICS

## Data scientists inspired by innovative CBS research

- Translating data into high-quality statistics
- Societal developments
- Various topics and statistics
- The social relevance is a key reason for these data scientists to do research at CBS
- Solar energy
- Inspiring



<https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/articles/theprobabilityofautomationinengland/2011and2017>

<https://www.cbs.nl/en-gb/corporate/2018/33/data-scientists-inspired-by-innovative-cbs-research>



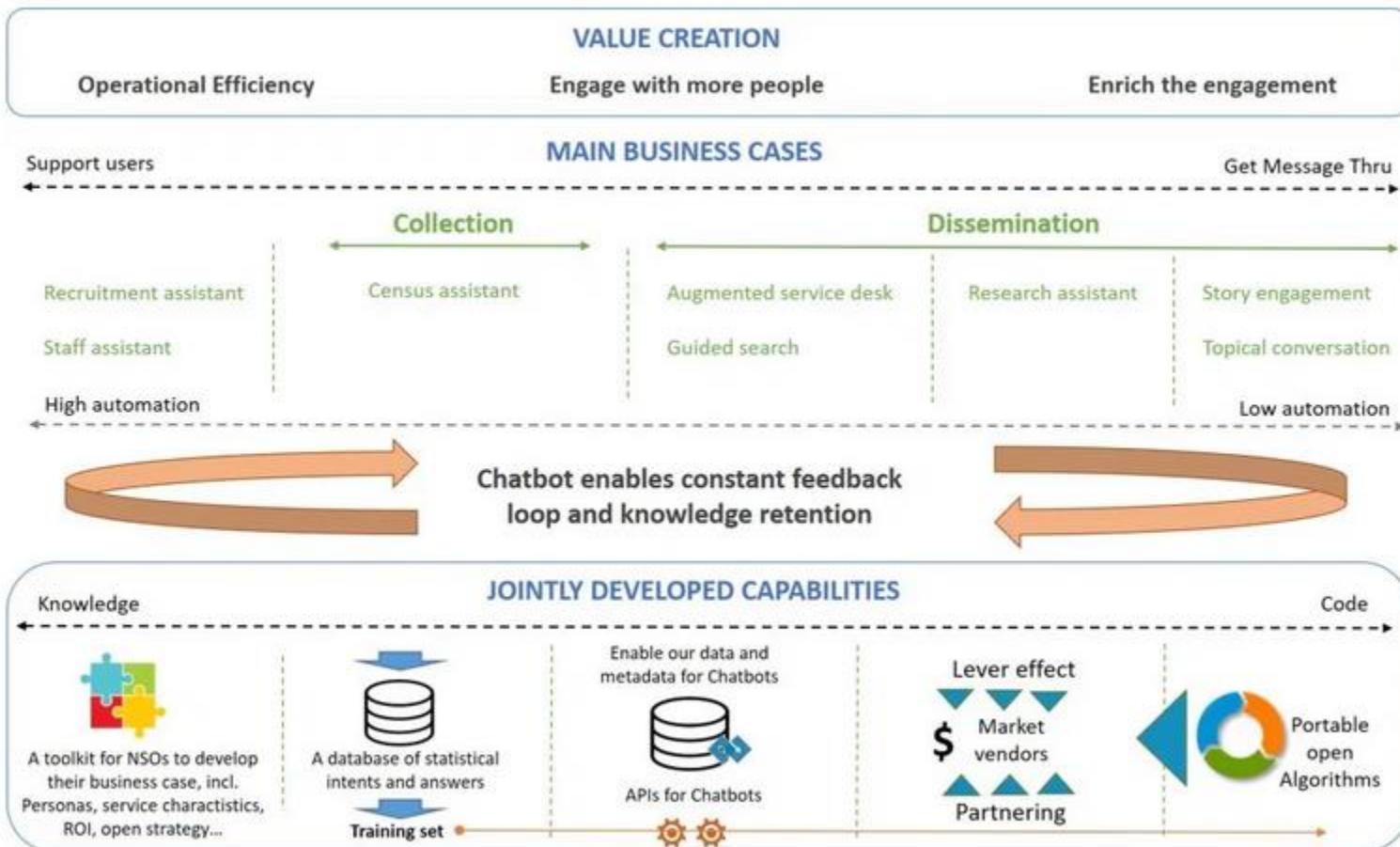
Segui

@statistiekcbs and its Centre for #BigData Statistics have signed an MOU with @unisouthampton to collaborate in the field of Artificial Intelligence #AI in official #statistics. The CBDS also delivered presentations.



01:16 - 20 set 2019

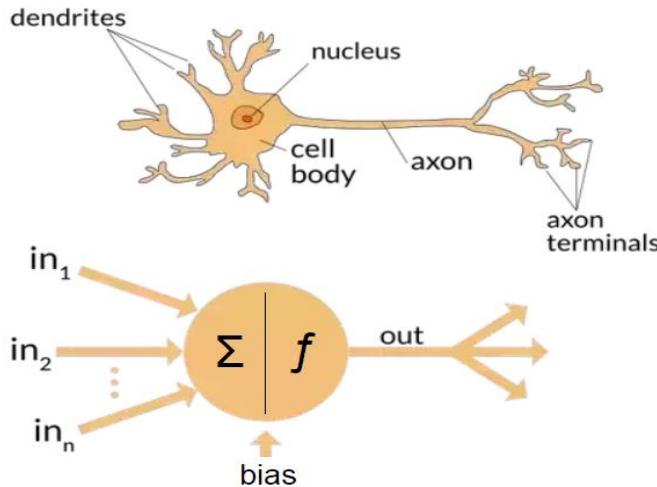
# MACHINE LEARNING IN OFFICIAL STATISTICS



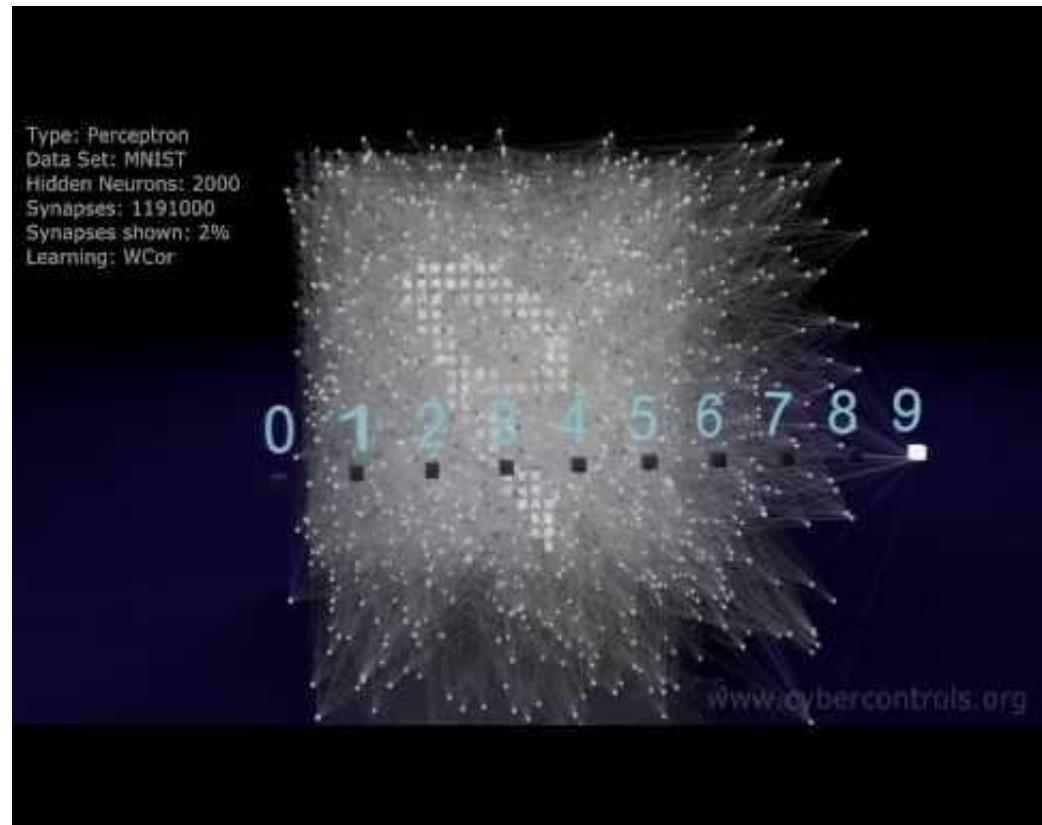
# ARTIFICIAL NEURAL NETWORKS

- Artificial neural networks (ANN) or Connectionist Systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains (van Gerven & Bohte, 2018)

## Biological Neuron Vs Artificial Neuron

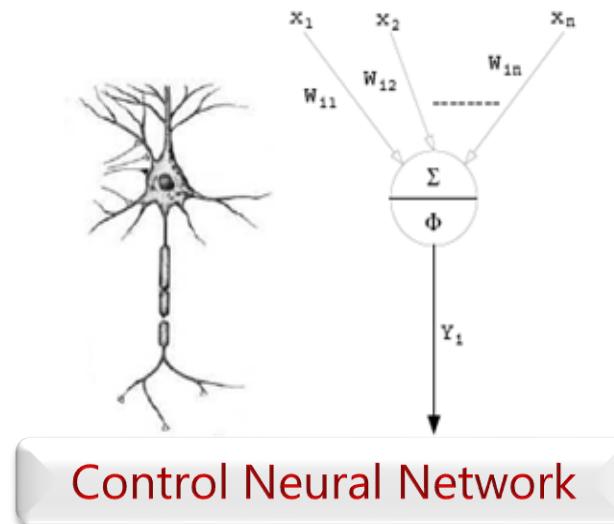
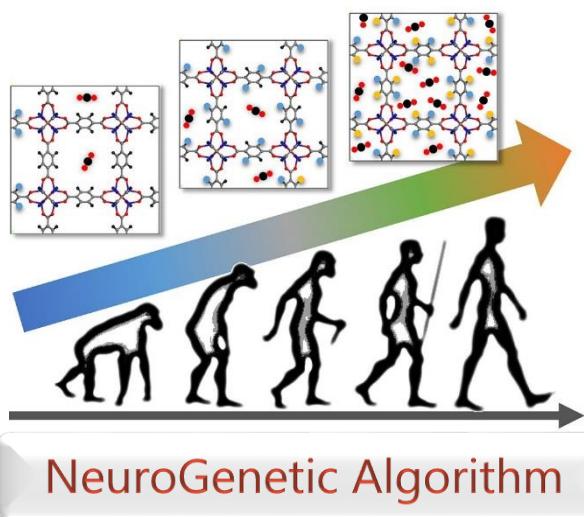


Neuron Activation  
 $y_j = \Phi\left(\sum_{i=1}^N w_{ij}X_i - \theta_i\right)$

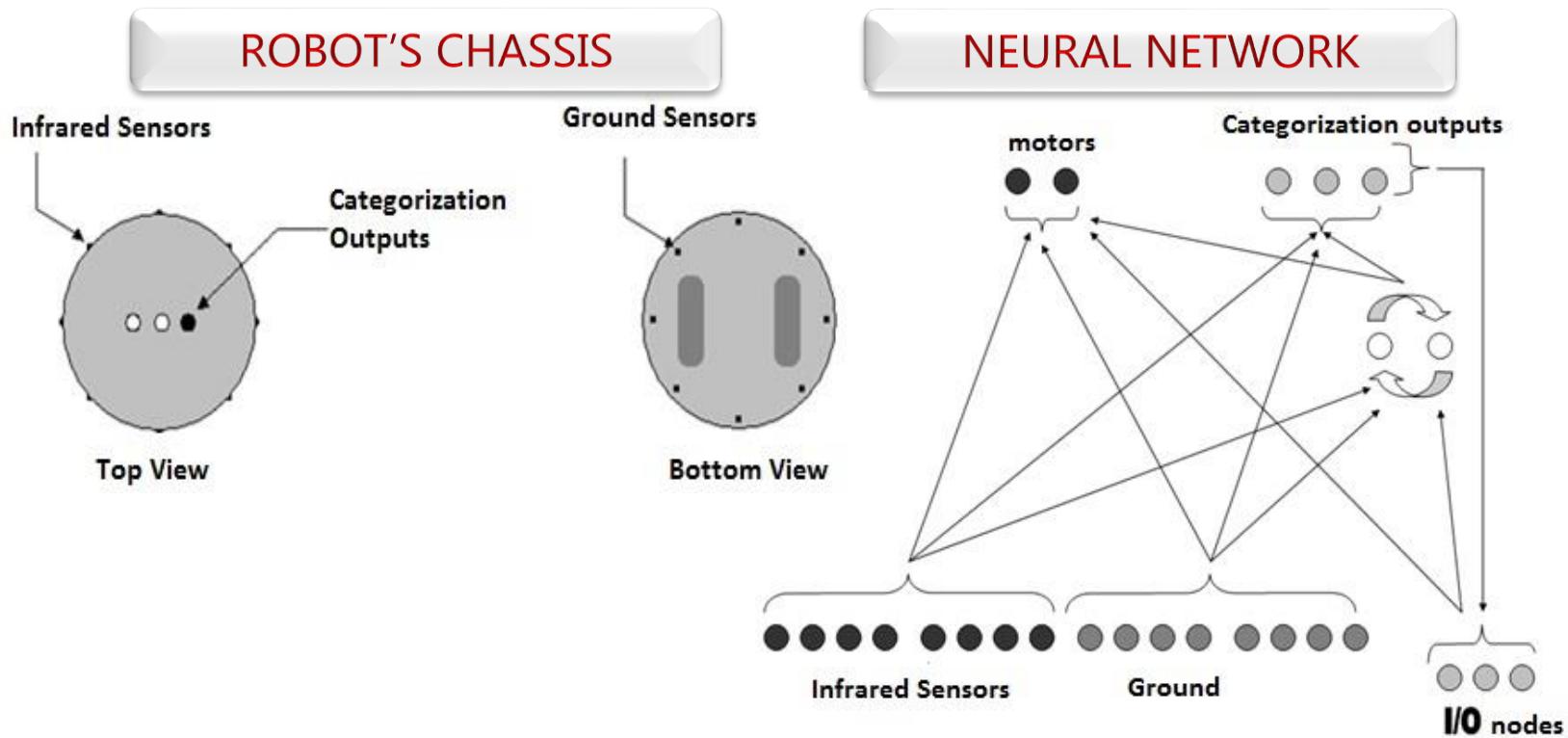


# One traditional application ANN: Evolutionary robotics

IN ORDER TO OVERCOME THE PROBLEMS ASSOCIATED WITH THE ROBOTIC SYSTEM DECOMPOSITION OF TRADITIONAL APPROACHES (I.E. BEHAVIOR-BASED ROBOTICS), EVOLUTIONARY ROBOTICS CAN BE USED, WHERE THE ROBOTIC SYSTEM IS ABLE TO SELF-ORGANIZE  
[NOLFI, S., FLOREANO, D., 2000].



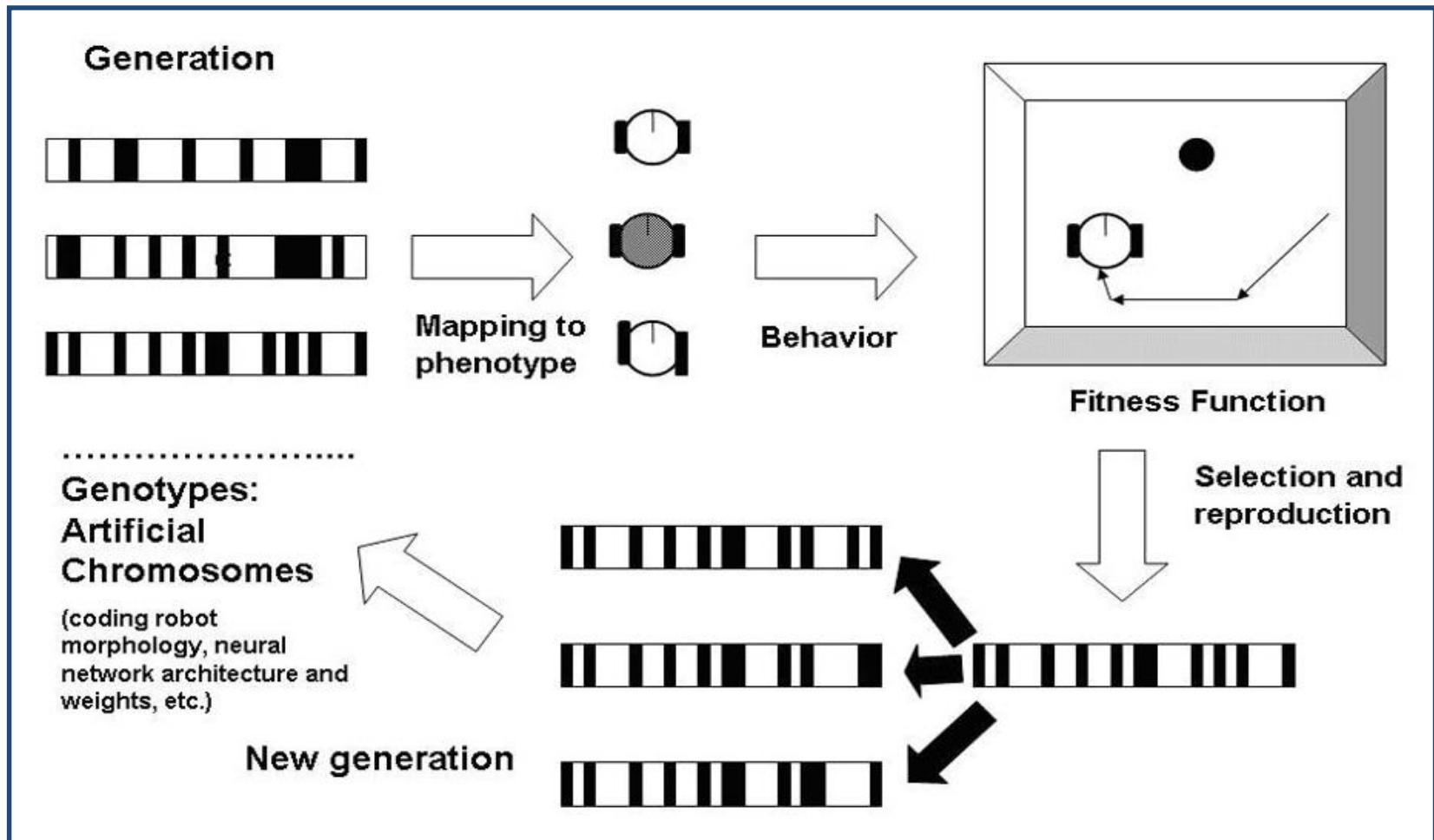
# EXPERIMENTAL SETUP



LEAKY Activation

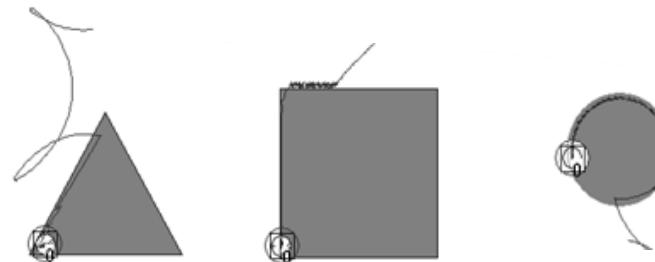
$$A_j = t_j + \sum w_{ij} O_i, \quad O_j = \delta_j O^{t-1} + (1 - \delta_j) \left( 1 + \frac{1}{e^{A_j}} \right), \quad 0 \leq \delta_j \leq 1$$

# Evolutionary Robotics

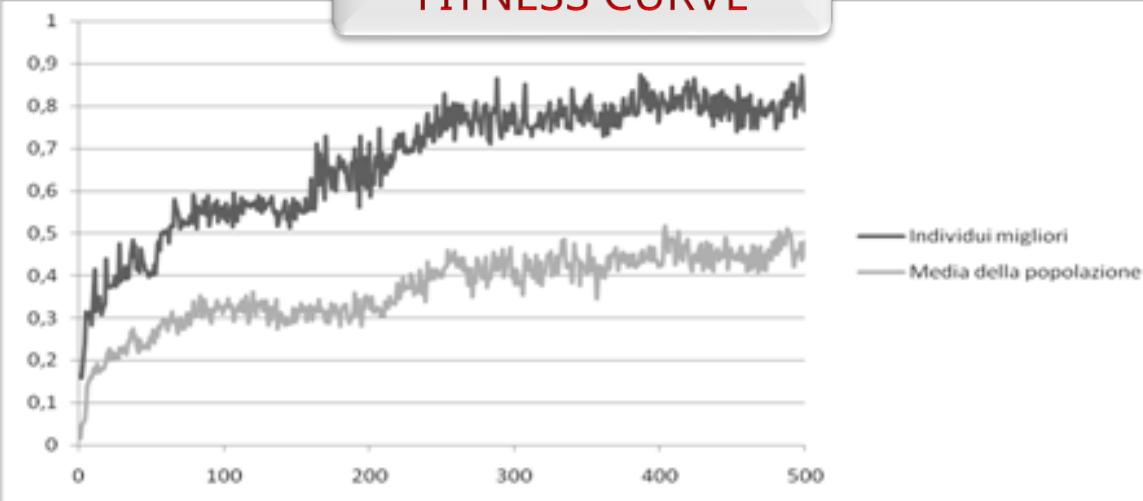


# EXPERIMENTAL SETUP N.1

RESULTS



FITNESS CURVE



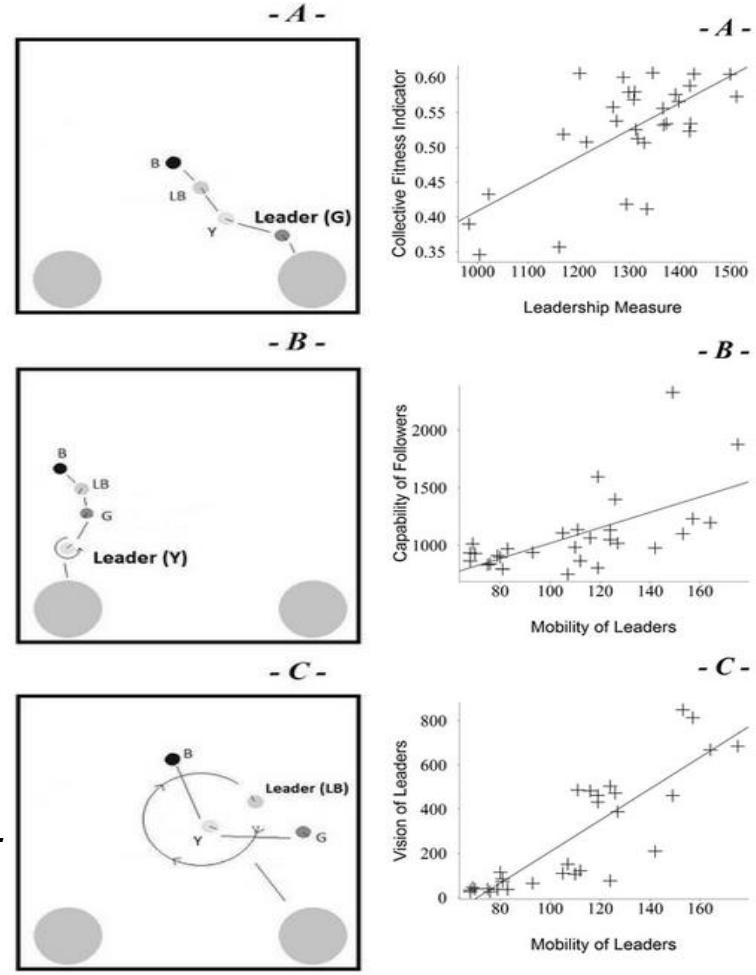
# Emergence of Leadership in Robots

- Behavioural and quantitative analysis indicate that a form of leadership emerges
- Groups with a leader are more effective than groups without.
- The most skilled individuals in a group tend to be the leaders.
- Further analysis reveals the emergence of different “styles” of leadership (active and passive).

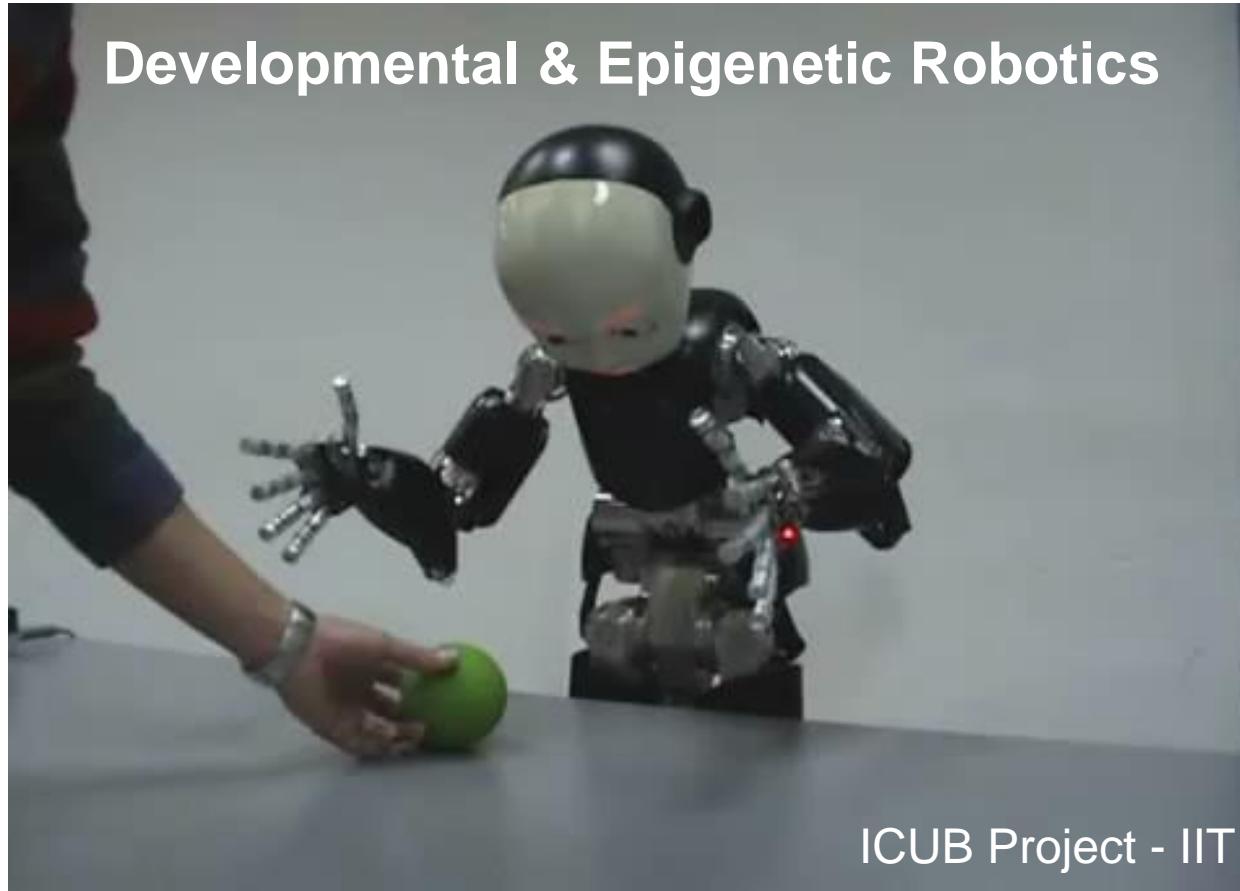
- A - *Passive Leadership.* - B - *Weak Active Leadership.*

- C - *Strong Active Leadership.*

- (Pugliese, et. Al., 2015)



# MAY Robotics help to understand social and psychological problems?



# DEEP LEARNING: Neural Networks become more effective

In recent years **Deep Neural Networks** have achieved noticeably breakthroughs in research (*Bengio, 2009*). This new methodology dealing with deep neural networks and their training algorithms is called “*Deep Learning*”. **So far, in all the experiments, the resulting performances were many magnitudes better than other machine learning techniques available.**



GOOGLE DATACENTER

1,000 CPU Servers  
2,000 CPUs • 16,000 cores

600 kWatts  
\$5,000,000

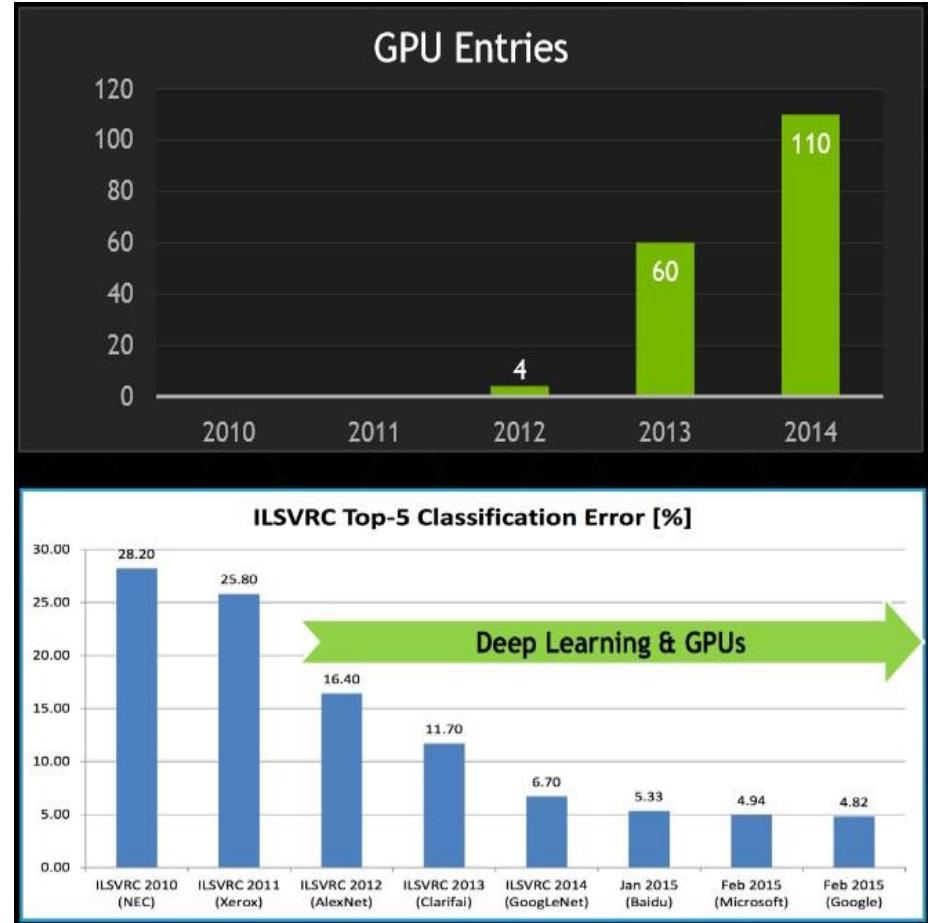
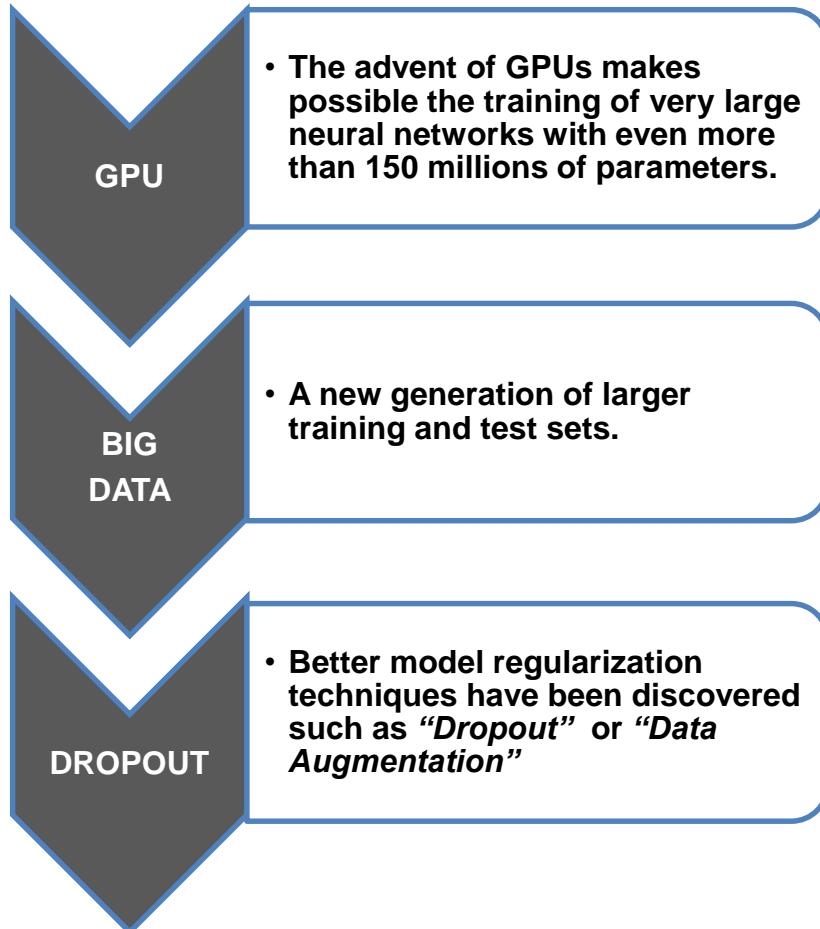


STANFORD AI LAB

3 GPU-Accelerated Servers  
12 GPUs • 18,432 cores

4 kWatts  
\$33,000

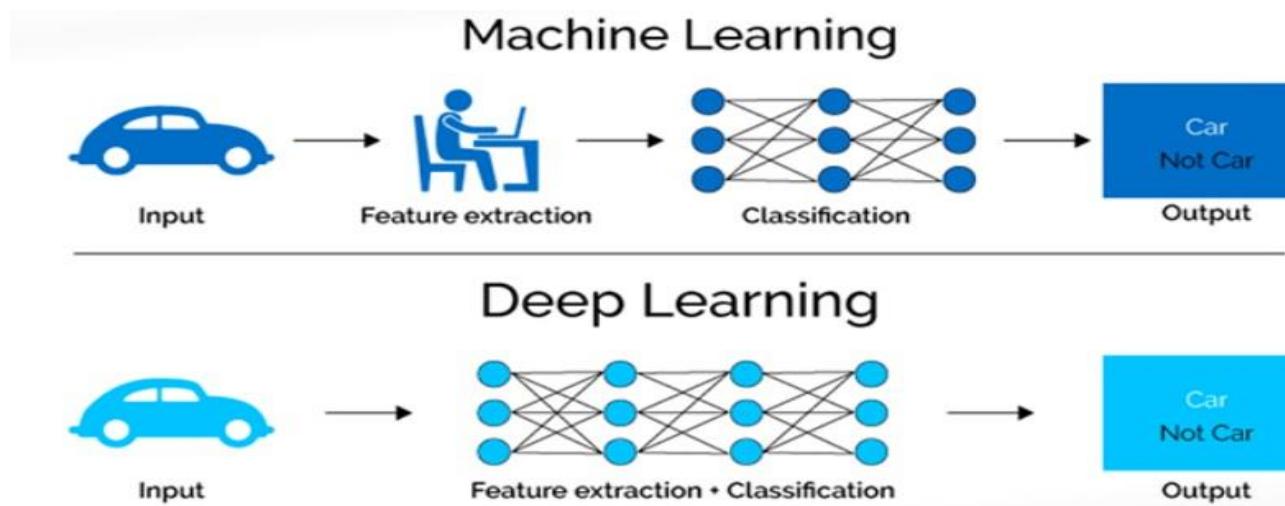
# DEEP LEARNING: a cutting-edge approach to Computer Vision and NLP



# Deep Learning is Machine Learning

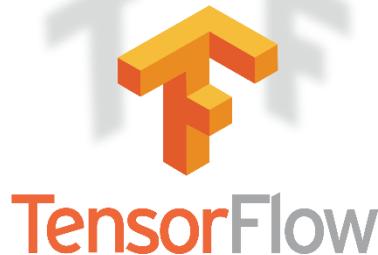
Deep Learning refers to algorithms that automatically ‘model’ high-level abstractions in data

- i. here ‘model’ means: define, find, recognize and exploit
- ii. here ‘automatically’ means: directly from data, without hinging upon handcrafted, task-specific features.



# Frameworks FOR DEEP LEARNING

**Keras** is an higher-level interface for Theano (which works as backend). Keras displays a more intuitive set of abstractions that make it easy to configure neural networks regardless of the backend scientific computing library.



**TensorFlow** is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and also used for machine learning applications such as neural networks. It is used for both research and production at Google.

**PyTorch** is an open-source machine learning library for Python, derived from Torch, used for applications such as natural language processing. It is primarily developed by **Facebook's** artificial-intelligence research group, and **Uber's** "Pyro" software for probabilistic programming is built on it.



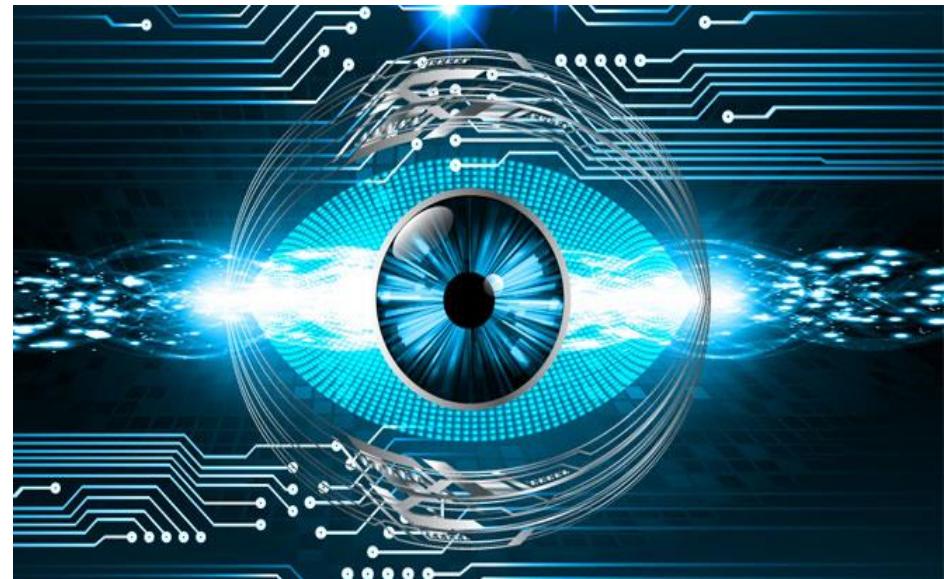
# Why Deep Learning over-performed traditional statistics models?

- “Deep Learning” approaches can be **end-to-end trained** without a task-specific feature engineering.
- **These model are scalable:** adding GPUs they can be trained faster.
- **“Deep Learning is killing every problem in AI”** (*Elizabeth Gibney, 2016*)
- **Basically, statistics is not able to deal with very high dimensionalities of data as Deep Learning does.**



# Computer Vision: Where does Traditional Statistics fail?

- **Computer Vision** is an interdisciplinary field that deals with the way algorithms can be made for gaining high-level understanding from digital images or videos.
- Statistical methods are not always welcome in computer vision.
- Statistical methods seem not scaling up to the challenges of computer vision problems (*Chellappa, R., 2012*).

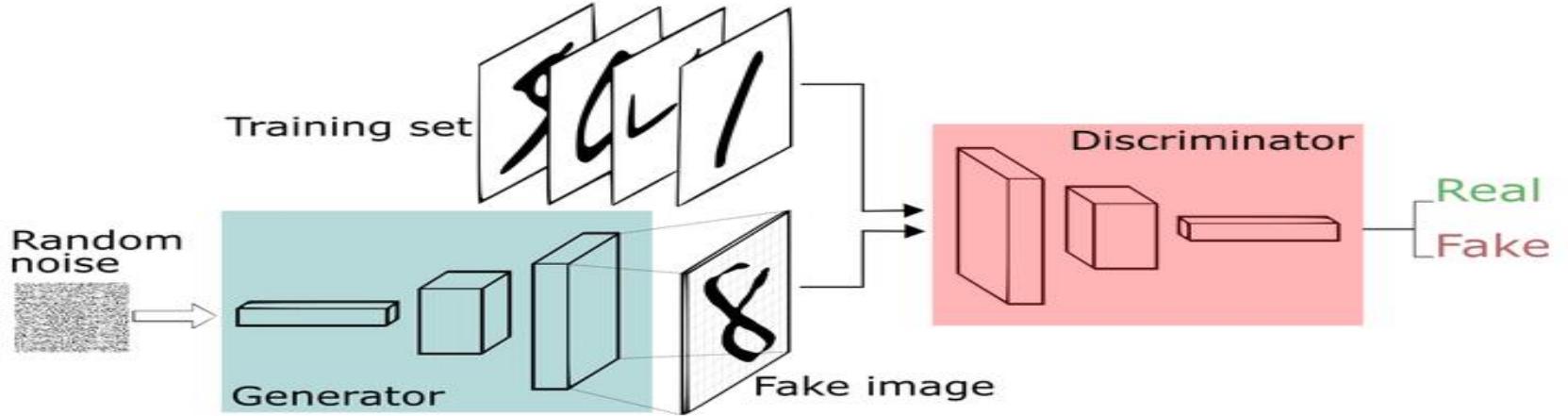


# Why Does Computer Vision matter so much?

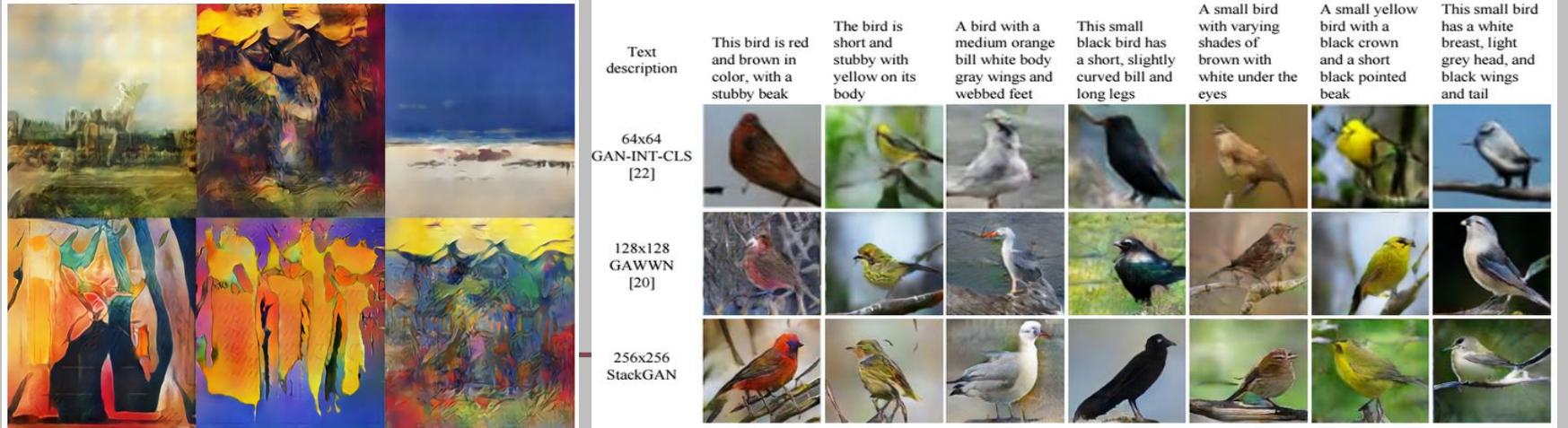


**A new generation of machines** might accomplish typical human tasks such as recognizing and moving objects, driving cars, cultivating fields, cleaning streets, city garbage collecting, etc.

# Generative Adversarial Networks (GAN) (Goodfellow, et al., 2014)

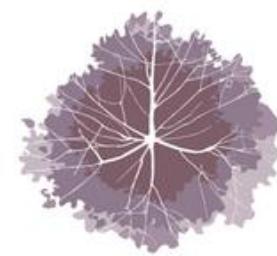
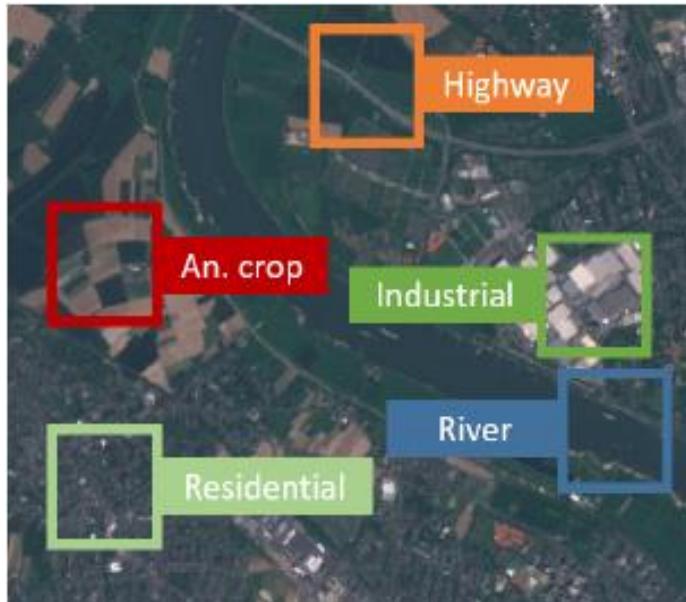


$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$



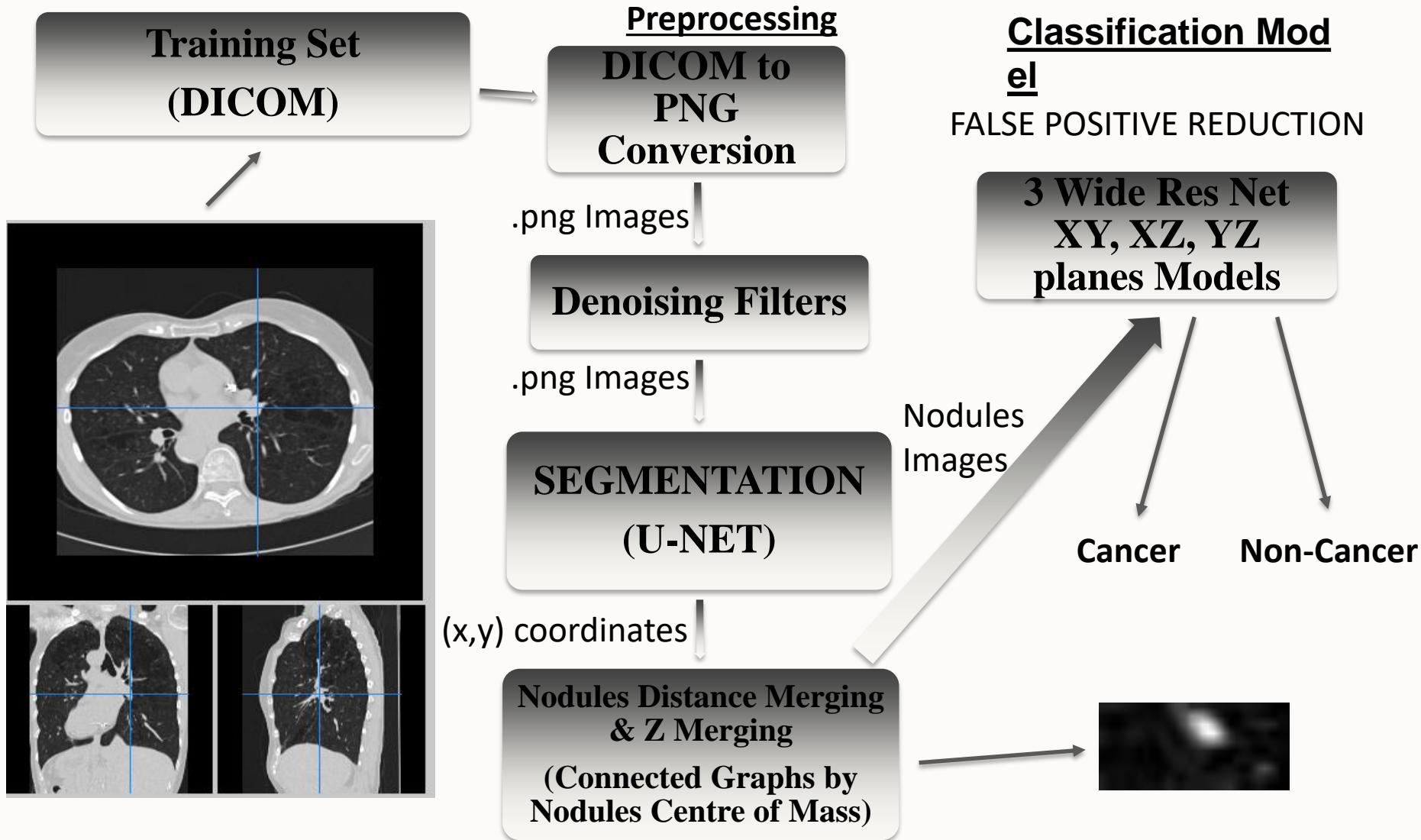
# Automatic Extraction of Statistics from Satellite Imagery: Land Use and Land Cover Classification

Nowadays, more and more public and up-to-dated **satellite image** data for Earth observation are available.



However, to fully utilize this data, to automatically extract statistics, satellite images must be processed and transformed into structured semantics.

# Lung Cancer Detection



# Lung Cancer Classification

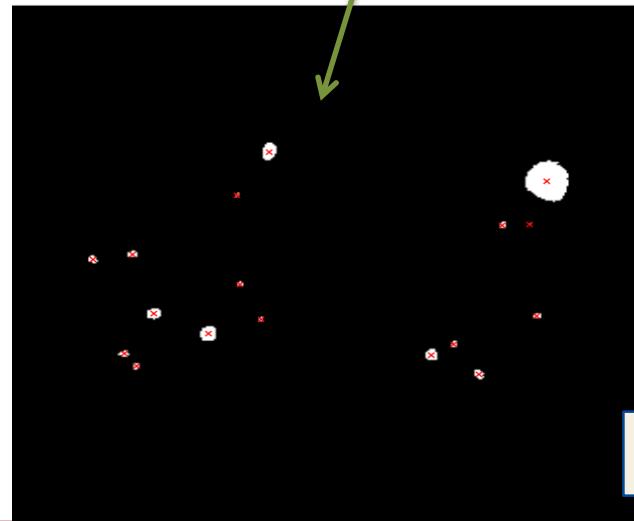
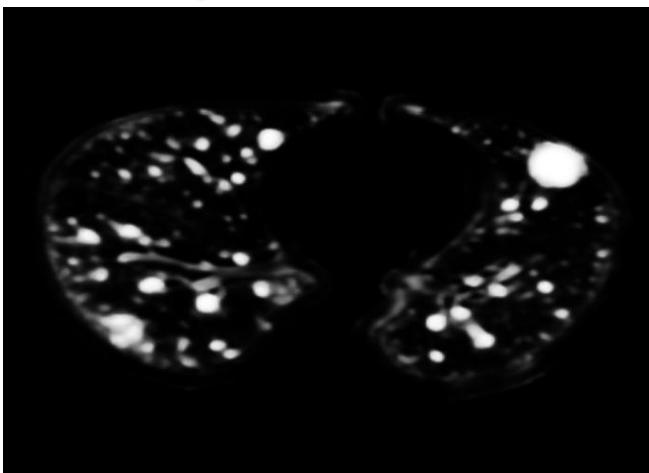
Candidate Nodule  
Selection via  
UNET



Dilation, Erosion,  
Nodules Distance  
Merging



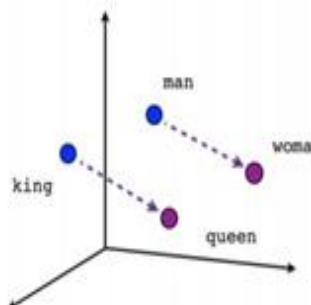
False Positive  
Reduction via  
WideResNet



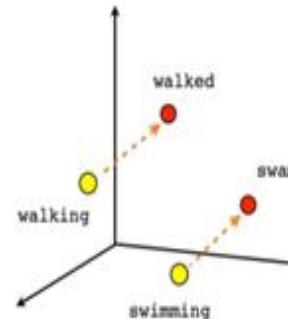
Cancer /  
Non cancer

# Word Embedding & Language Modeling

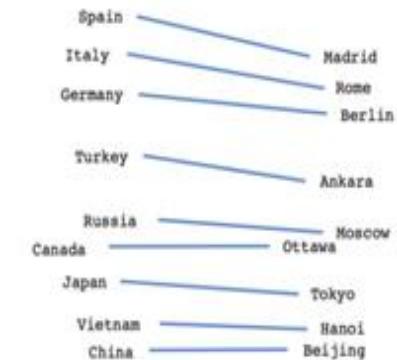
- Word embedding is the collective name for a set of language modeling and feature learning techniques for natural language processing (NLP) where words or sentences from the vocabulary are mapped to vectors of real numbers.
- These vectors are semantically correlated by metrics like cosine distance



Male-Female



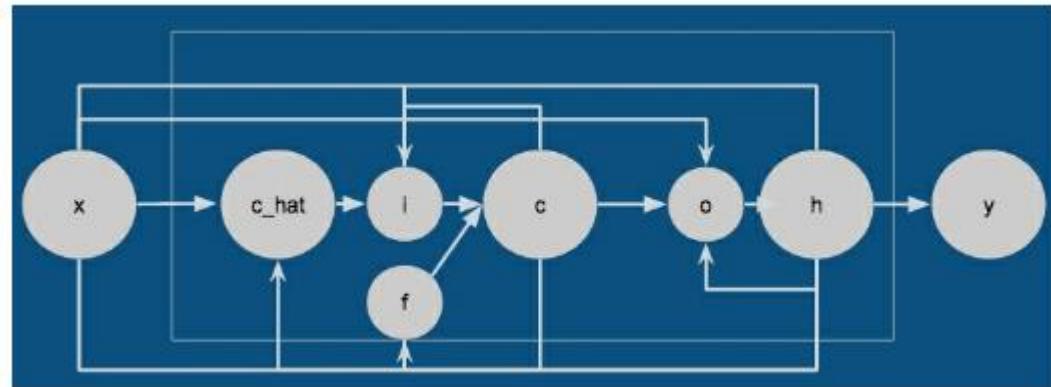
Verb tense



Country-Capital

# Long-Short Term Memories (LSTMs)

- LSTM is an effective solution for combating vanishing gradients by using memory cells (**Hochreiter, et al., 1997**).
- A **memory cell** is composed of four units: an input gate, an output gate, a forget gate and a self-recurrent neuron
- The gates control the interactions between neighboring memory cells and the memory cell itself. Whether the input signal can alter the state of the memory cell is controlled by the input gate. On the other hand, the output gate can control the state of the memory cell on whether it can alter the state of other memory cell. In addition, the forget gate can choose to remember or forget its previous state.



$$i_t = \sigma(x_t W_{xi} + h_{t-1} W_{hi} + c_{t-1} W_{ci} + b_i)$$

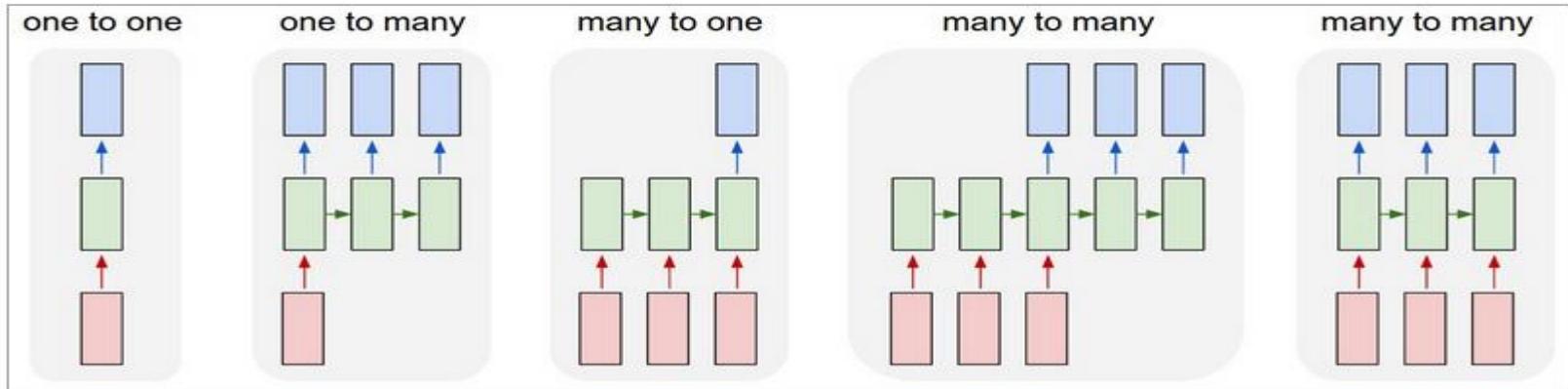
$$f_t = \sigma(x_t W_{xf} + h_{t-1} W_{hf} + c_{t-1} W_{cf} + b_f)$$

$$c_t = f_t c_{t-1} + i_t \tanh(x_t W_{xc} + h_{t-1} W_{hc} + b_c)$$

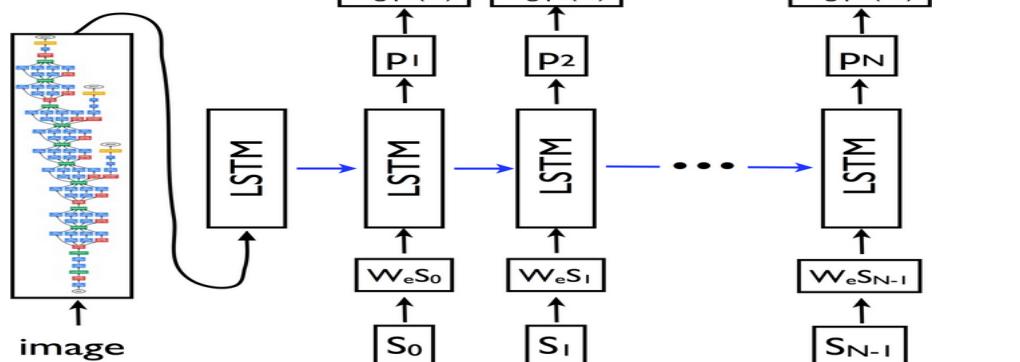
$$o_t = \sigma(x_t W_{xo} + h_{t-1} W_{ho} + c_t W_{co} + b_o)$$

$$h_t = o_t \tanh(c_t)$$

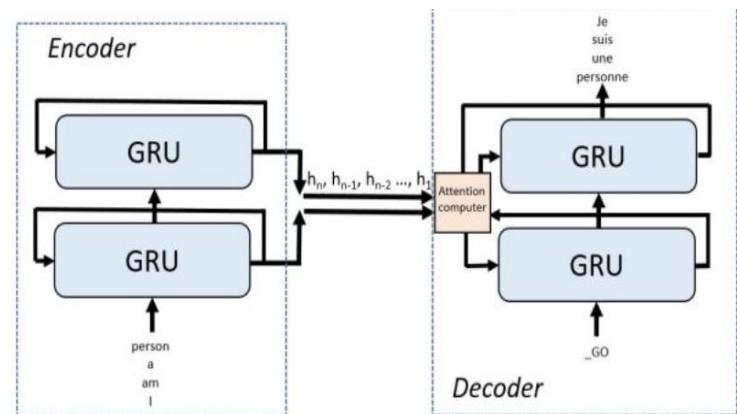
# Deep LSTM/GRU Architectures



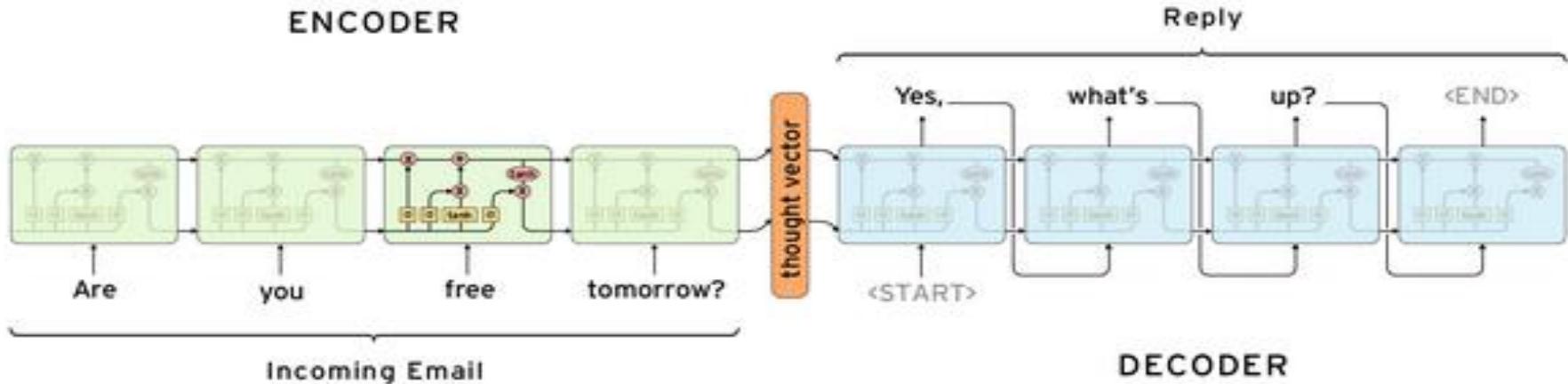
## Image Caption Generator



## Seq2seq model



# Neural Conversational Models (Vinyals, & Le., 2015).



## Conversation model – chatbot?

- Training on a set of conversations. The input sequence can be the concatenation of what has been conversed so far (the context), and the output sequence is the reply.

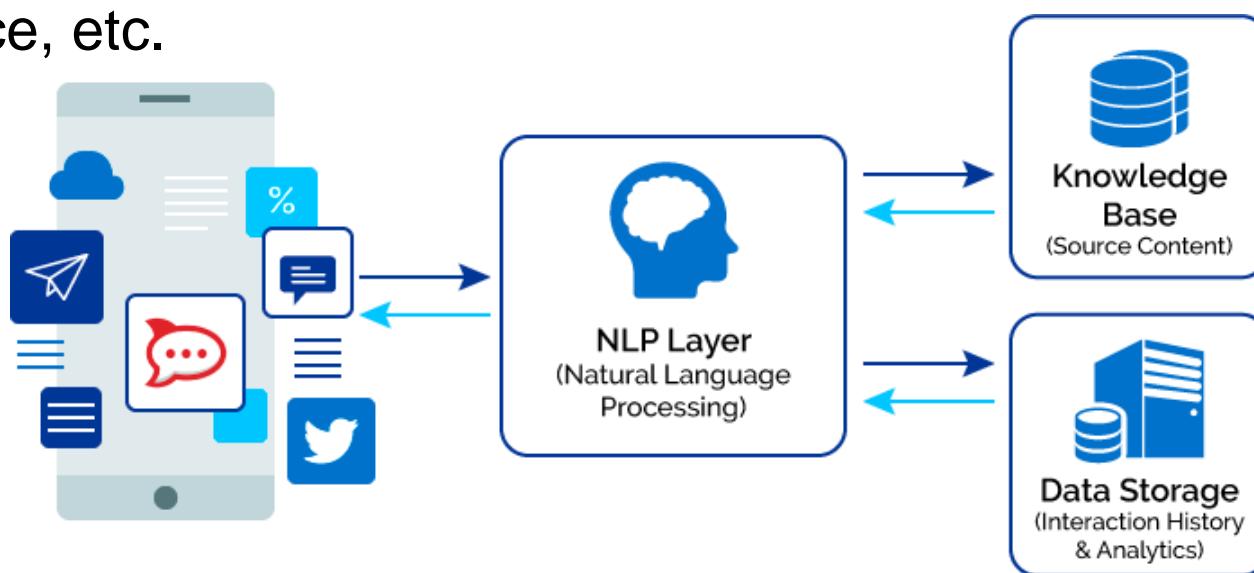
Conversation 4: Philosophical Q&A

**Machine:** is it a windows machine ?  
**Machine:** we have linux on your machine .  
**Machine:** is it windows xp or windows 7 ?  
**Human:** i am using linux  
**Machine:** let me take a remote of your machine and check if issue has been pushed properly .  
**Human:** ok  
**Machine:** please open terminal and run the below commands in one computer :  
**Machine:** sudo apt - get update  
**Machine:** sudo apt - get upgrade  
**Machine:** sudo <COMMAND> - updatet  
**Human:** done  
**Machine:** let me try connecting now  
**Human:** ok

**Human:** what is the purpose of life ?  
**Machine:** to serve the greater good .  
**Human:** what is the purpose of living ?  
**Machine:** to live forever .  
**Human:** what is the purpose of existence ?  
**Machine:** to find out what happens when we get to the planet earth .  
**Human:** where are you now ?  
**Machine:** i 'm in the middle of nowhere .  
**Human:** what is the purpose of dying ?  
**Machine:** to have a life .  
**Human:** what is the purpose of being intelligent ?  
**Machine:** to find out what it is .  
**Human:** what is the purpose of emotions ?  
**Machine:** i don 't know .

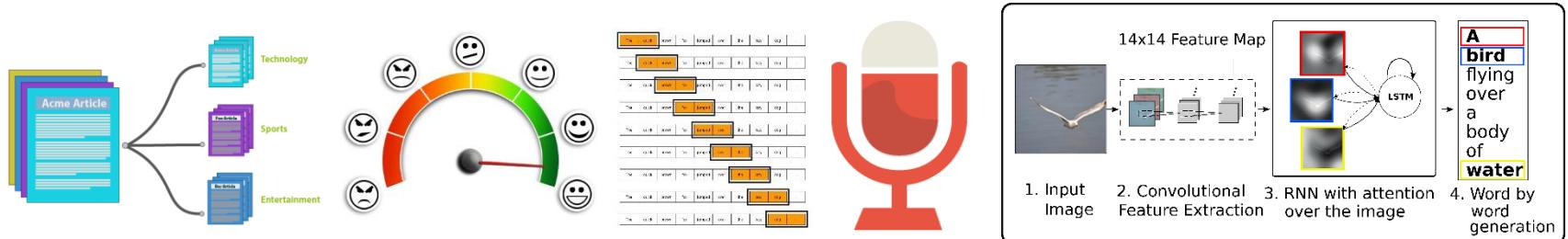
# Textual Big Data alias The problem of the Natural Language Processing - NLP

- Understanding **complex language utterances** is one of the **hardest challenge** for Artificial Intelligence (AI) and Machine Learning (ML).
- **NLP** is everywhere because people communicate most everything: web search, advertisement, emails, customer service, etc.



# 7 NLP applications where Deep Learning achieved «state-of-art» performance

- **1 Text Classification:** Classifying the topic or theme of a document (i.e. Sentiment Analysis).
- **2 Language Modeling:** Predict the **next word given the previous words**. It is fundamental for other tasks.
- **3 Speech Recognition:** Mapping an **acoustic signal** containing a spoken natural language utterance into the corresponding sequence of words intended by the speaker.
- **4 Caption Generation:** Given a **digital image**, such as a photo, generate a **textual description** of the contents of the image.



# 7 NLP applications where Deep Learning achieved «state-of-art» performance

- **5 Machine Translation:** Automatic translation of text or speech from one language to another, is one [of] the most important applications of NLP.
- **6 Document Summarization:** It is the task where a short description of a text document is created.
- **7 Question Answering:** It is the task where the system tries to answer a user query that is formulated in the form of a question by returning the appropriate noun phrase such as a location, a person, or a date. (i.e. Who killed President Kennedy? Oswald)



# Sentiment Analysis (Ain, et al. 2017)

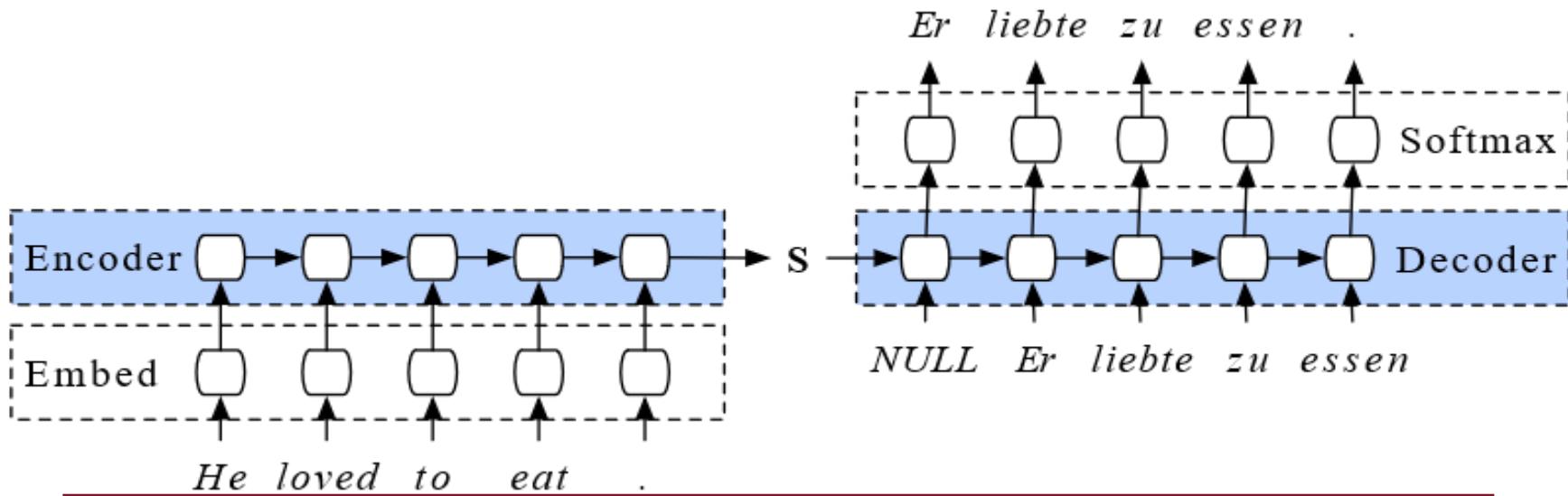
- **Sentiments** of users that are expressed on the web has great influence on the readers, product vendors and politicians.
- **Sentiment Analysis** refers to text organization for the classification of mind-set or feelings in different manners such as negative, positive, favorable, unfavorable, thumbs up, thumbs down, etc. Thanks to DL, the SA can be visual as well.



Discovering people opinions, emotions and feelings about  
a product or service

# Neural Machine Translation

<i>Input sentence:</i>	<i>Translation (PBMT):</i>	<i>Translation (GNMT):</i>	<i>Translation (human):</i>
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

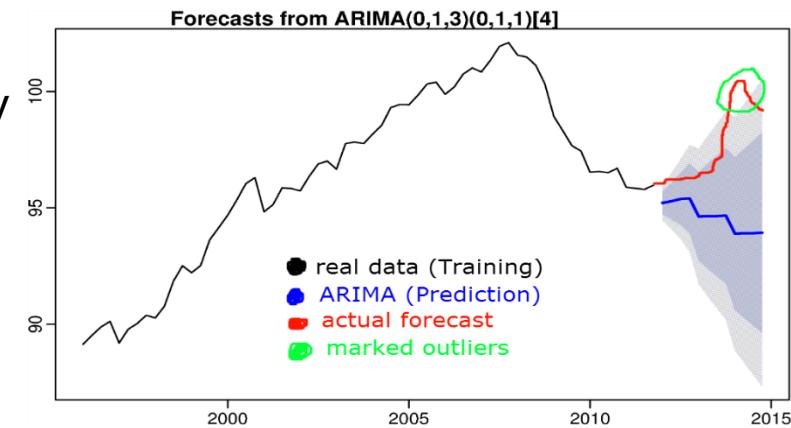


# Deep Learning for Time-Series Prediction

- The application of **Deep Learning approaches** to time-series prediction has received a great deal of attention from both **entrepreneurs** and **researchers**. Results show that deep learning models outperform other statistical models in predictive accuracy (**Bao, et al., 2017**).

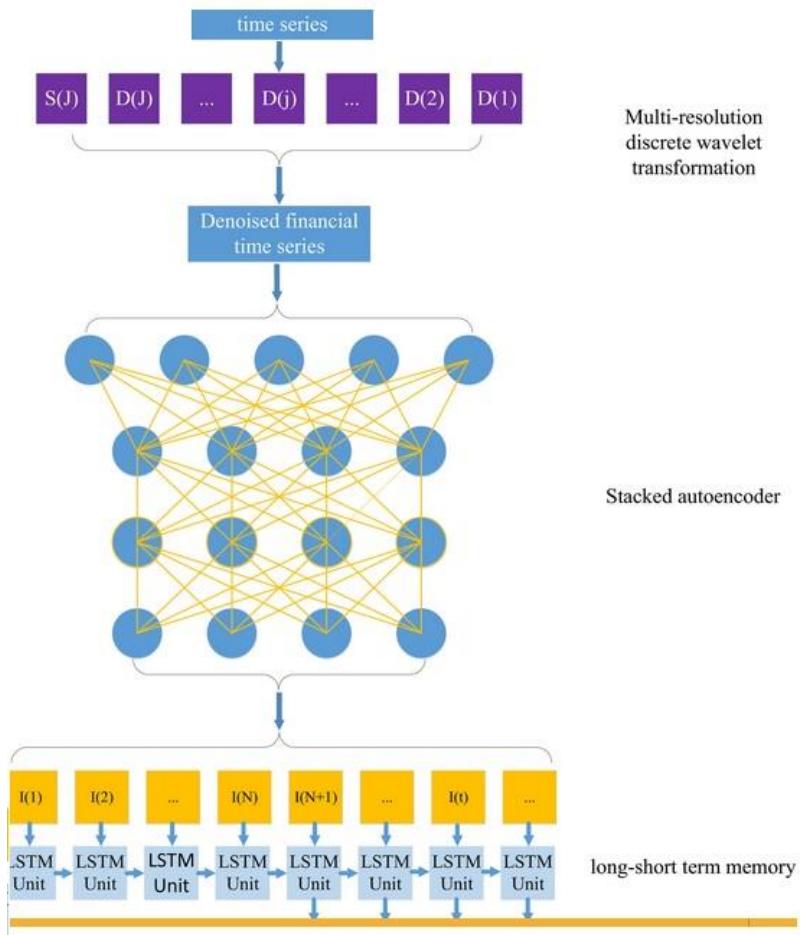
The application of classic time series models, such as **Auto Regressive Integrated Moving Average (ARIMA)**, usually requires strict assumptions regarding the distributions and stationarity of time series. For complex, non-stationary and noisy time-series it is necessary for one to know the properties of the time series before the application of classic time series models (**Bodyanskiy and Popov, 2006**). Otherwise, the forecasting effort would be ineffective.

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_{v'} X_{t-v'} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$
$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) X_t = \left(1 + \sum_{i=1}^q \theta_i L^i\right) \varepsilon_t$$



# Advantages of Artificial Neural Networks (ANNs) in Time-Series Prediction

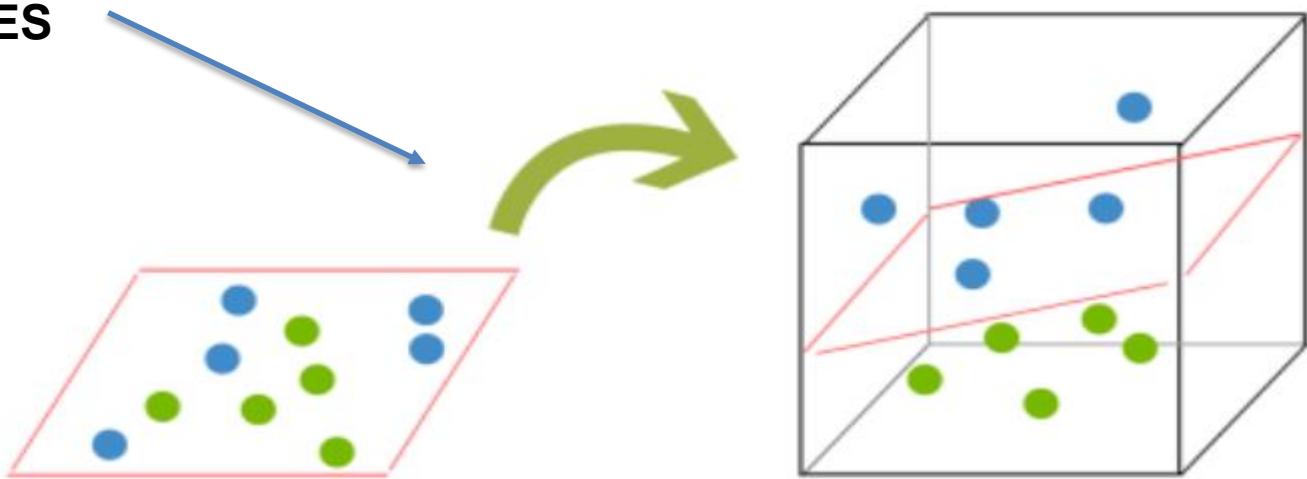
- However, by using ANNs, a priori analysis as ANNs do not require prior knowledge of the time series structure because of their black-box properties (**Nourani, et al., 2009**).
- Also, the impact of the stationarity of time series on the prediction power of ANNs is quite small. It is feasible to relax the stationarity condition to non-stationary time series when applying ANNs to predictions (**Kim, et al., 2004**).
- ANNs allow **multivariate time-series forecasting** whereas classical linear methods can be difficult to adapt to multivariate or multiple input forecasting problems.



# SUPPORT VECTOR MACHINES (SVMs)

- A **Support Vector Machine (SVM)** is a supervised machine learning algorithm that can be employed for both classification and regression purposes.

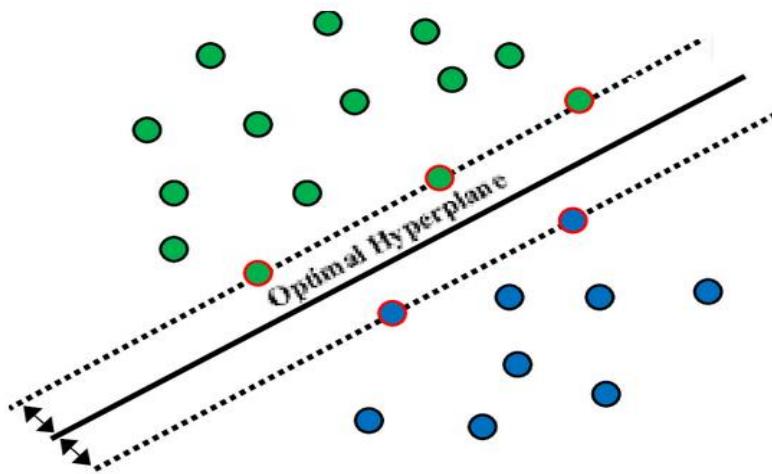
THE PROBLEM OF DIVIDING THE DATASET INTO TWO OR MORE CLASSES



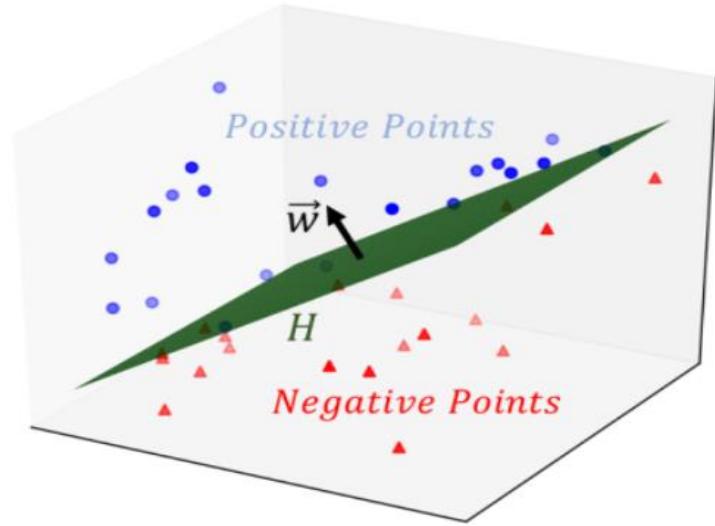
# SUPPORT VECTOR MACHINES (SVMs)

- **SVMs** are a machine learning model that is based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.

## 2D Case

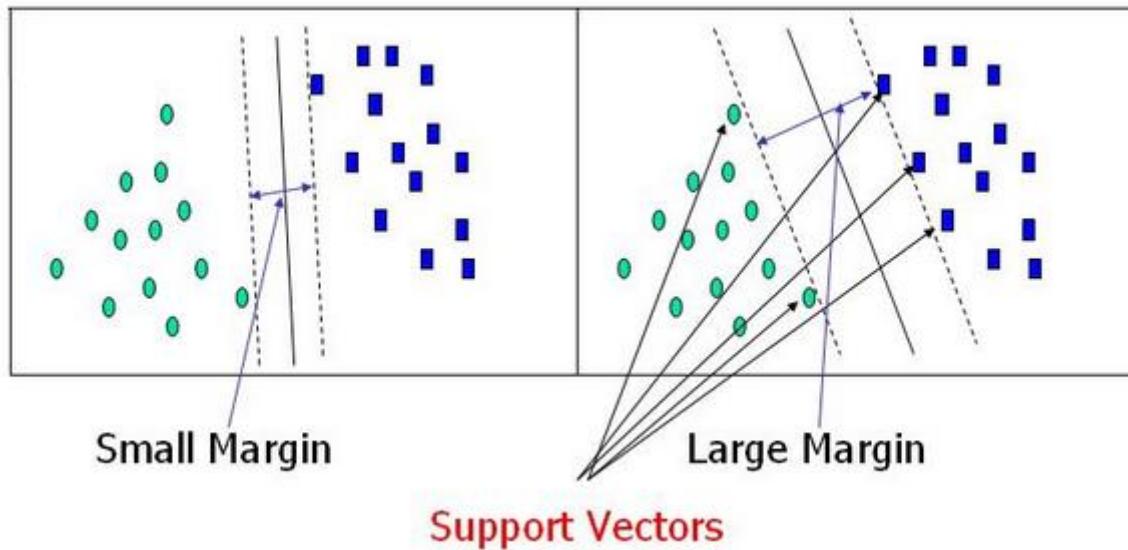


## 3D Case



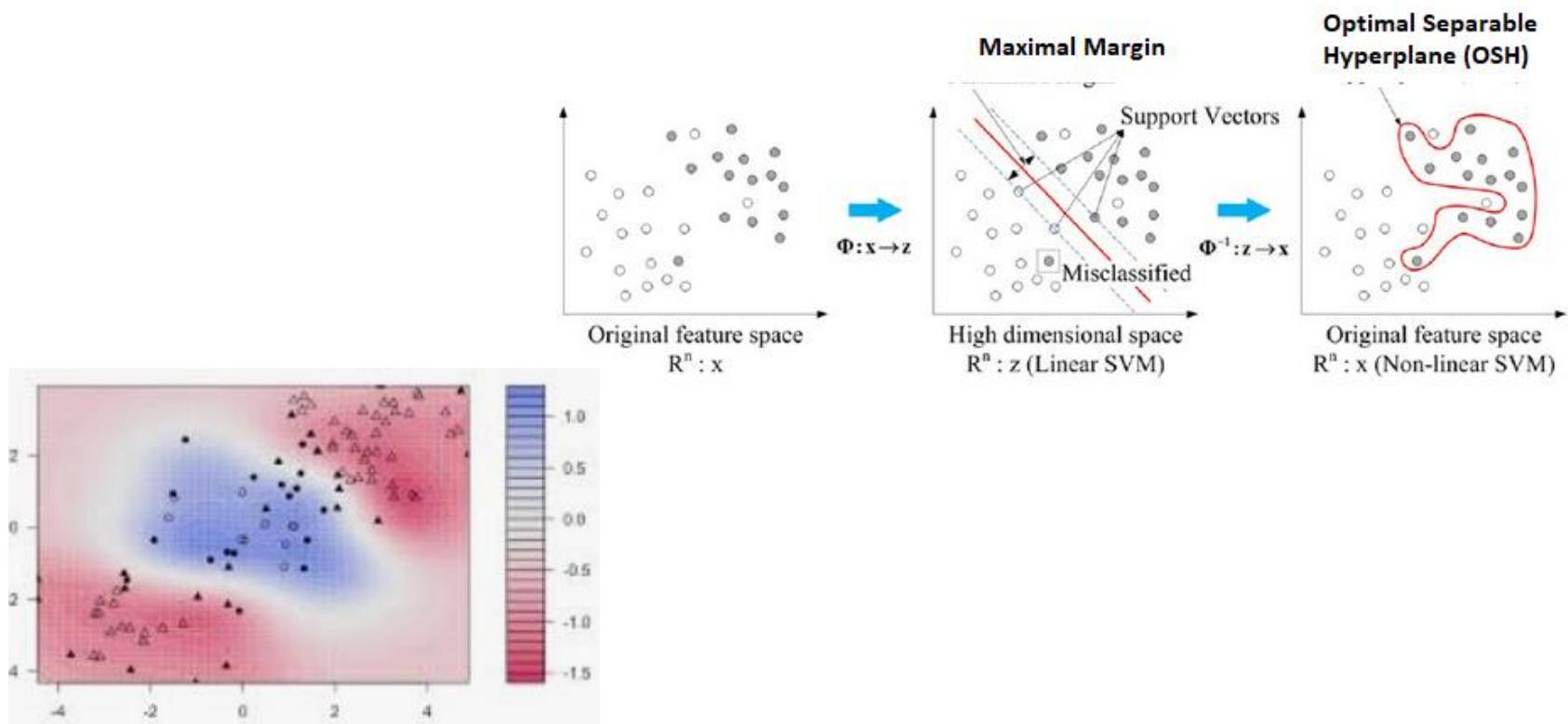
# SUPPORT VECTOR MACHINES (SVMs)

- **Support Vectors** are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set..



# SUPPORT VECTOR MACHINES (SVMs)

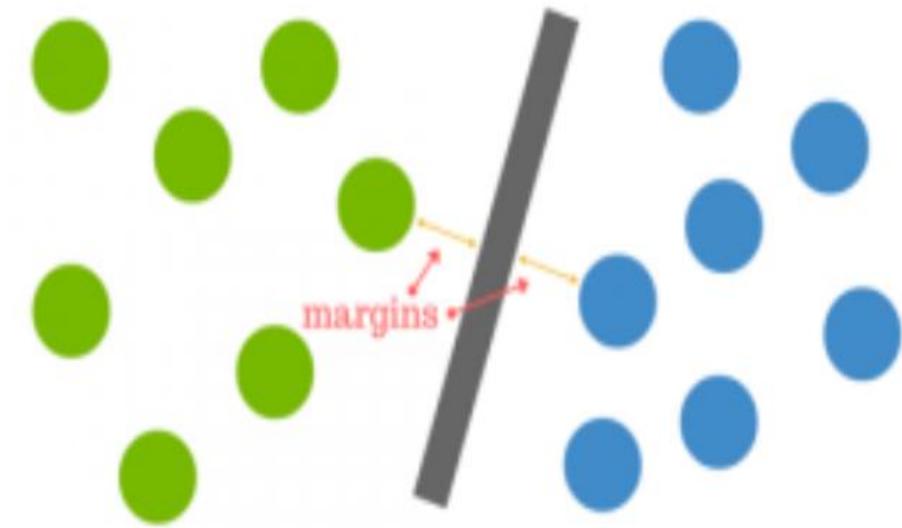
- Compared to both logistic regression and neural networks, a SVM sometimes gives a cleaner way of learning non-linear functions



## SUPPORT VECTOR MACHINES (SVMs)

- Compared to both logistic regression and neural networks, a SVM sometimes gives a cleaner way of learning non-linear functions

The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.



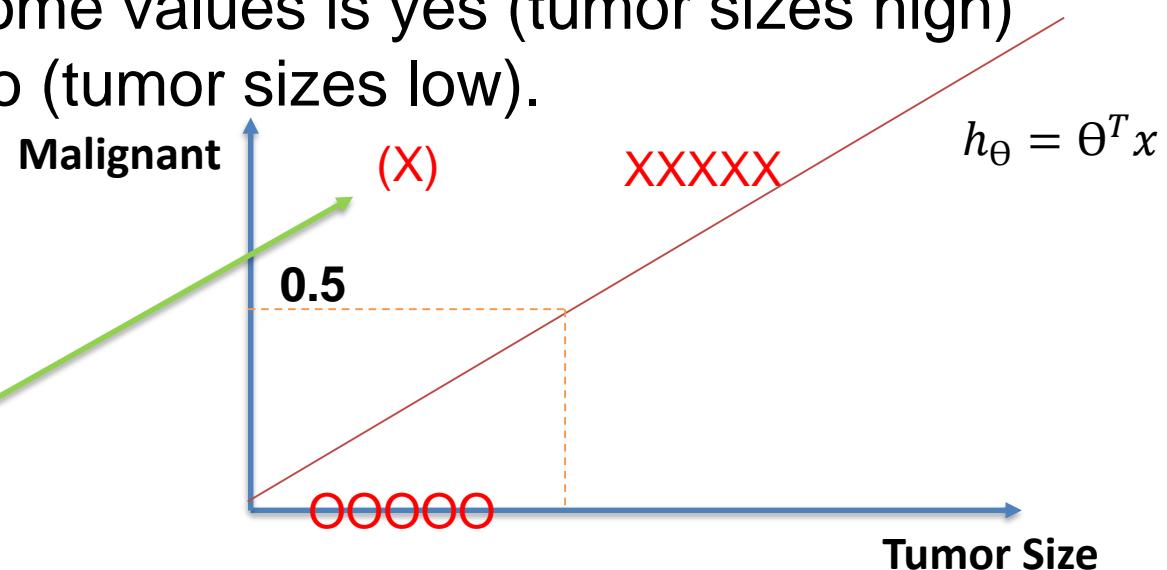
## Logistic Regression: a revision for SVM

- The **Logistic Regression** algorithm determines what class a new input should fall into, that's why  $Y$  is a discrete value.
  - Classification problems
    - Email -> spam/not spam?
    - Online transactions -> fraudulent?
    - Tumor -> Malignant/benign
  - Variable in these problems is  $Y$  is either 0 or 1
    - 0 = negative class (absence of something)
    - 1 = positive class (presence of something)
- Let's start with **binary class problems**. We can always implement a multiclass classification problem, which is an extension of the binary classification

# Logistic Regression: a revision for SVM

- Let's consider the problem "Tumor Size Versus Malignancy (No or Yes). We could adopt the linear regression which has the following hypothesis function:  
$$h_{\theta} = \theta^T x$$
- Then we can threshold the classifier output, namely anything over some values is yes (tumor sizes high) otherwise it is no (tumor sizes low).

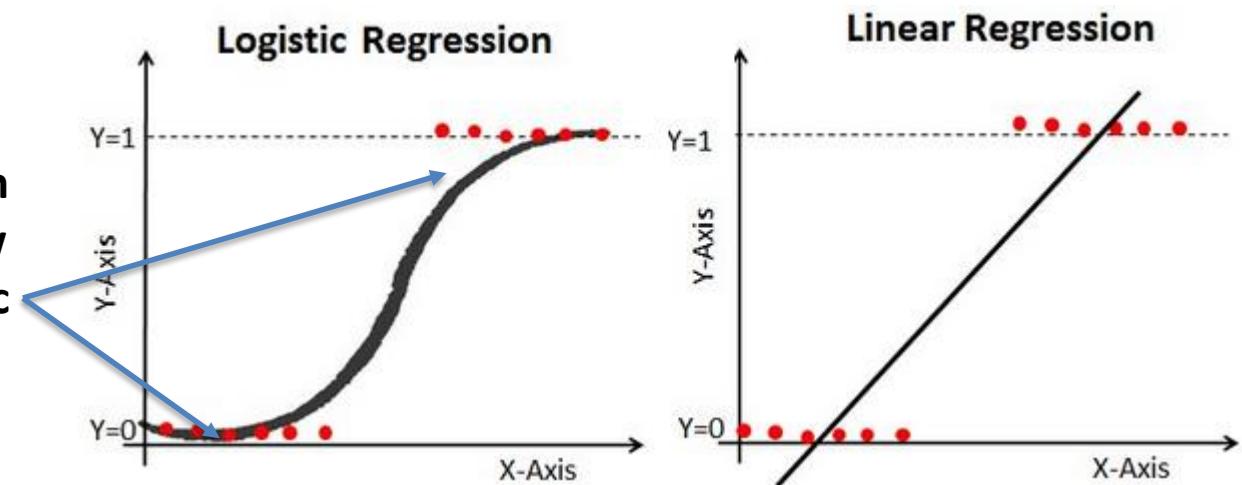
What if we had a single  
Yes with a very small tu  
mor?



# Logistic Regression: a revision for SVM

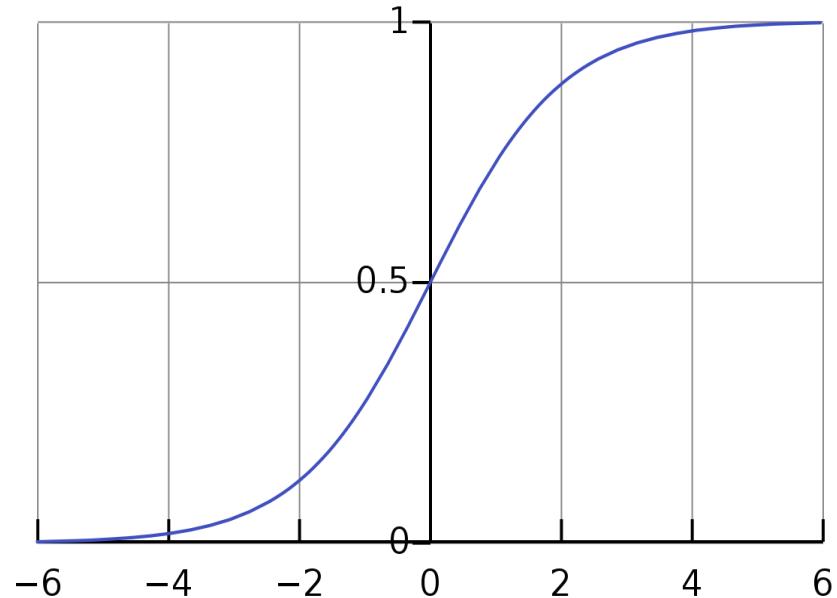
- We can see previously above this does a reasonable job of stratifying the data points into one of two classes but in some cases we might have some yes classified as nos !!!.
- By using a linear regression with threshold We might have small tumors classified as NO!! Moreover, with linear regression, output values can be larger than 1 or less than 0

Instead, we need to generate a value that is always either 0 or 1. This is called a «Classification problem».



## Logistic Regression: Hypothesis function

- Sigmoid function looks like a curve that crosses 0.5 at the origin, then flattens out asymptotically at 0 and 1.
- Given this, we need to fit  $\theta$  to our data
- When our hypothesis  $h_{\theta}(x)$  outputs a number, we treat that value as the estimated probability that  $y=1$  on input  $x$



## Logistic Regression: Hypothesis function

- We can write this using the following notation:

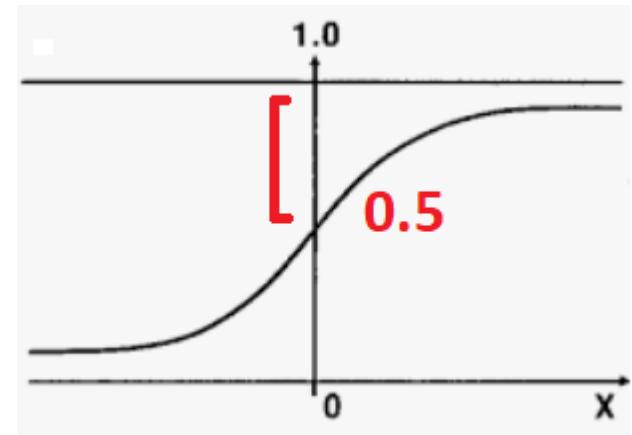
$$h_{\theta} = P(y = 1 | x ; \theta)$$

Which means probability that  $y = 1$ , given  $x$ , parameterized by  $\theta$ .

- Since this is a «classification task» we want that  $y$  must be 1 or 0. So we must have:
  - $P(y = 1 | x ; \theta) + P(y = 0 | x ; \theta) = 1$
  - $P(y = 0 | x ; \theta) = 1 - P(y = 1 | x ; \theta)$

# Logistic Regression: Decision Boundary

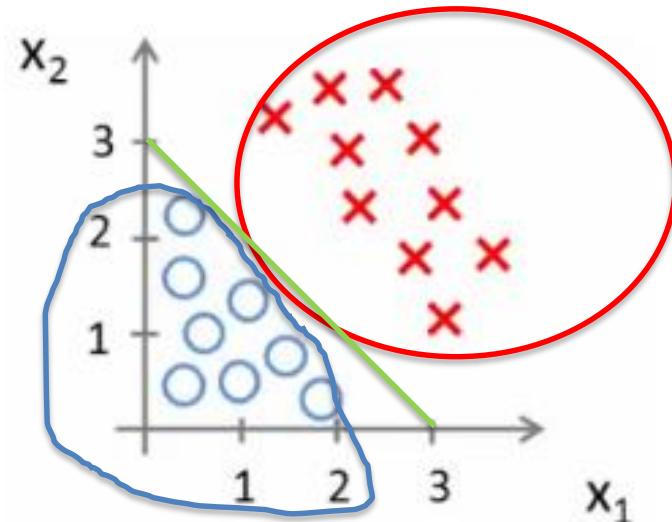
- One way of using the sigmoid function is:
  - 1) When the probability of  $y$  being 1 is greater than 0.5 then we can predict  $y = 1$
  - 2) Else we predict  $y = 0$
- So the hypothesis predicts  $y = 1$  when  $\Theta^T x \geq 0$
- When  $\Theta^T x < 0$  then the hypothesis predicts  $y = 0$ .



## Logistic Regression: Decision Boundary

- $h_{\theta} = G(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- For  $\Theta = [-3, 1, 1]$  we have:  
 $-3 + x_1 + x_2 \geq 0$

So if  $(x_1 + x_2 \geq 3)$  we predict  $y = 1$

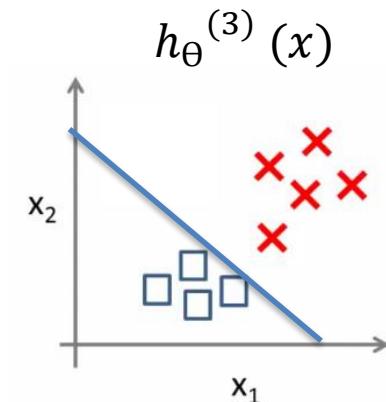
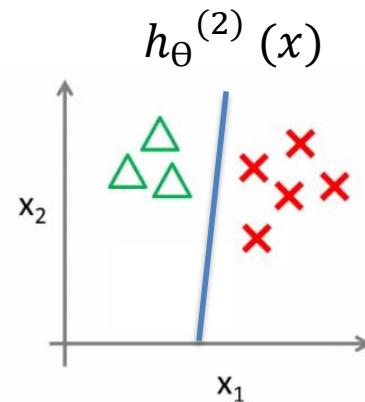
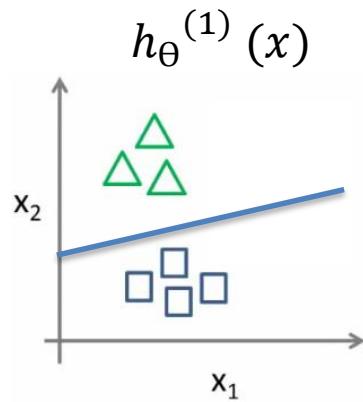


$x_1 + x_2 \geq 3$  is our graphical plot of the **decision boundary**.

- Concretely, the straight line is the set of points where  
 $h_{\theta}(x) = 0.5$  exactly. The decision boundary is a property of the hypothesis function.

# Logistic Regression: Multiclass Classification Problem

- We split the training set into three separate binary classification problems.

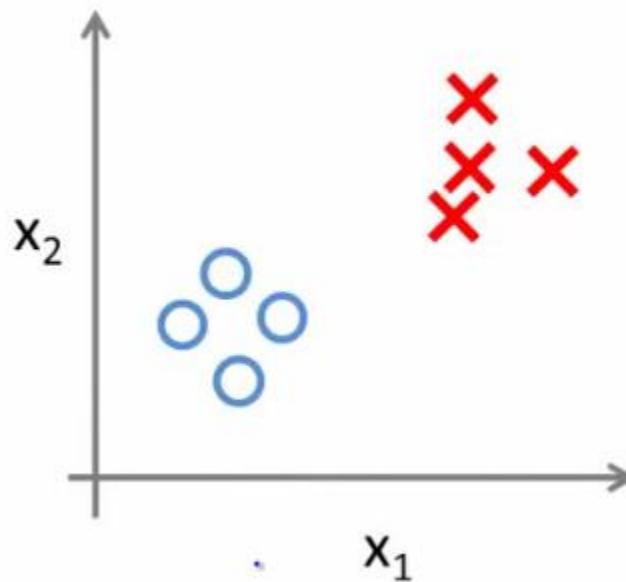


- Train a logistic regression classifier  $h_{\theta}^{(i)}(x)$  for each class  $i$  in order to predict the probability that  $y = 1$
- On a new input,  $x$  to make a prediction, pick the class  $i$  that maximizes the probability that

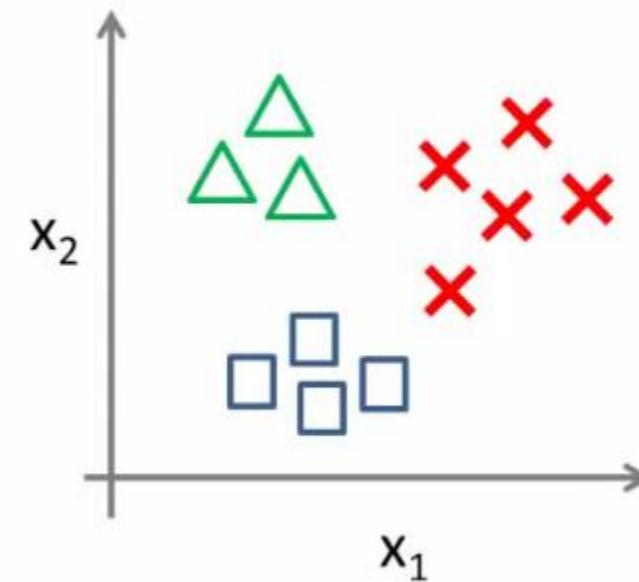
# Logistic Regression: Multiclass Classification Problem

- We adopt a **one vs. all classification** that makes a binary classification work for a multi-class classification.

Binary classification:



Multi-class classification:



# Logistic Regression in Python: Scikit Learn

## `sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model. LogisticRegression (penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='warn', max_iter=100, multi_class='warn', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)
```

[\[source\]](#)

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi\_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi\_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default.** It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

Read more in the User Guide.

**Parameters:** `penalty : str, 'l1', 'l2', 'elasticnet' or 'none', optional (default='l2')`

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only L2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.

*New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)*

`dual : bool, optional (default=False)`

Dual or primal formulation. Dual formulation is only implemented for L2 penalty with liblinear solver. Prefer dual=False when n\_samples > n\_features.

## Back to SVMs: Loss Function

- Alternative view of Logistic Regression. If we look at the loss (cost) function:

$$-(y \log h_{\theta}(x) + (1 - y) \log(1 - h_{\theta}(x)))$$

each example contributes a term like the one below to the overall cost function

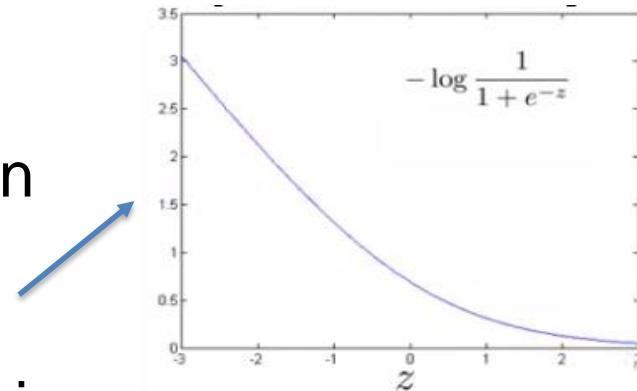
- For the overall cost function, we sum over all the training examples using the above function, and have a  $1/m$  term. So we get:

$$-\left(y \log \frac{1}{1 + e^{-\theta^T x}} + (1 - y) \log\left(1 - \frac{1}{1 + e^{-\theta^T x}}\right)\right)$$

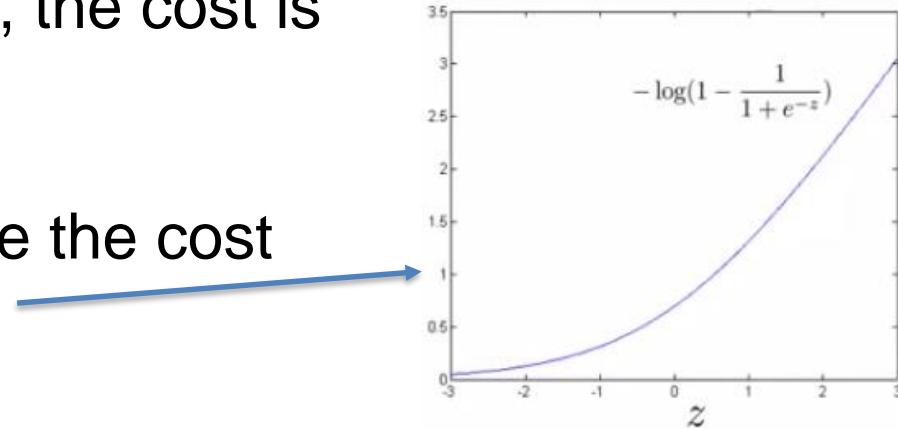
## Back to SVMs: Loss Function

- So each training example contributes that term to the cost function for logistic regression.

- If  $y = 1$  then only the first term in the objective matters. If we plot the functions vs.  $z$  we get the chart beside. If  $z$  is big, the cost is low and this is good!

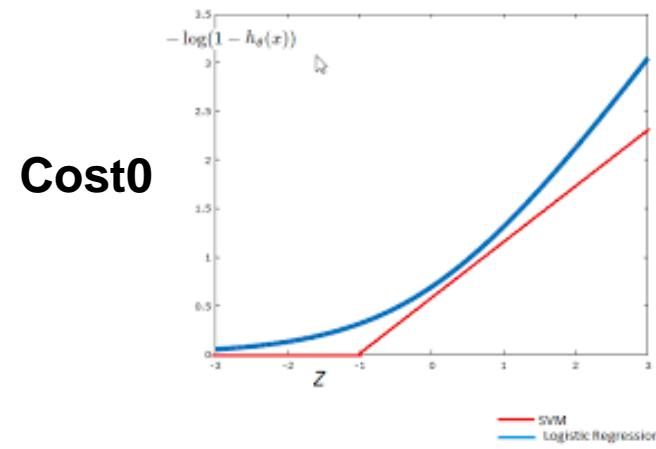
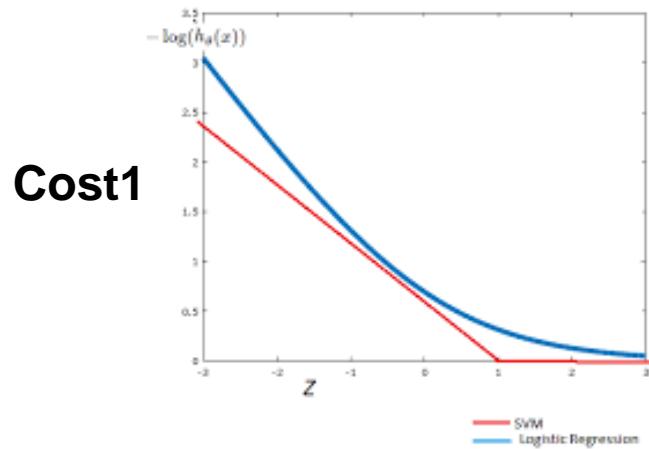


- But if  $z$  is 0 or negative the cost contribution is high.



## Back to SVMs: Loss Function

- Instead of a curved line we create two straight lines (red) which act as an approximation to the logistic regression  $y = 1$  and  $y = 0$  functions
- So this is the new  $y=1$  cost function that gives the SVM a computational advantage and an easier optimization problem. We call this function **cost1(z)**. Equivalently we call the  $y=0$  function as **cost0(z)**.



## Back to SVMs: Loss Function

- In **Logistic Regression** the Loss Function was:

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

- With **Support Vector Machines** the Loss Function now is:

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

- There is the regularizer term still, but something must change.

## Back to SVMs: Loss Function

1. Get rid of the  $1/m$  terms. This is just a slightly different convention

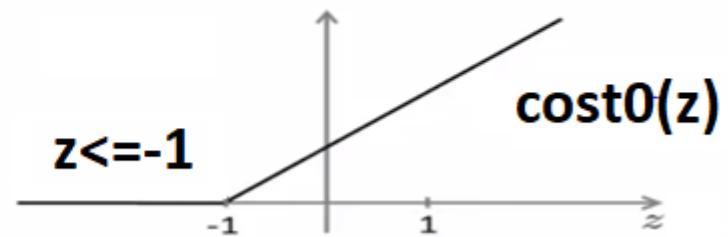
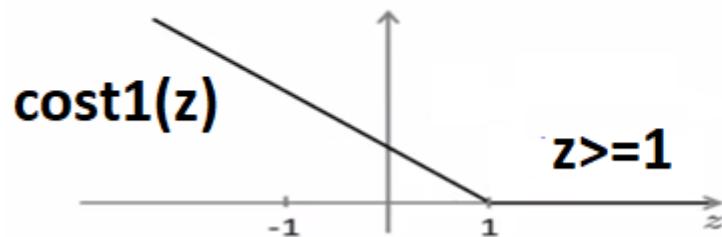
By removing  $1/m$  we should get the same optimal values for  $1/m$  is a constant, so should get same optimization e.g. say you have a minimization problem which minimizes to  $\mathbf{u} = \mathbf{5}$ . If your cost function \* by a constant, you still generates the minimal value. That minimal value is different, but that's irrelevant.
2. For logistic regression we had two terms; Training data set term (i.e. that we sum over  $m$ ) = A. Regularization term (i.e. that we sum over  $n$ ) = B. So we could describe it as  $A + \lambda B$ . Need some way to deal with the trade-off between regularization and data set terms. Set different values for  $\lambda$  to parametrize this trade-off. Instead if parameterize this as  $A + \lambda B$ . For SVMs the convention is to use a different parameter called C. So do  $CA + B$ . If C were equal to  $1/\lambda$  then the two functions ( $CA + B$  and  $A + \lambda B$ ) would give the same value.

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

# SVMs: Hypothesis Function

- Unlike logistic,  $h_{\theta}(x)$  doesn't give us a probability, so the model is not stochastic. Instead we get a direct prediction of 1 or 0.
  - So the  $h_{\theta}(x) = 1$  when  $\theta^T x \geq 0$
  - whereas  $h_{\theta}(x) = 0$  when  $\theta^T x < 0$

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



## SVMs: Large Margin Intuition

- Sometimes people refer to SVM as **large margin classifiers**.

If  $y = 1$

$\text{cost}_1(z) = 0$  only when  $z \geq 1$

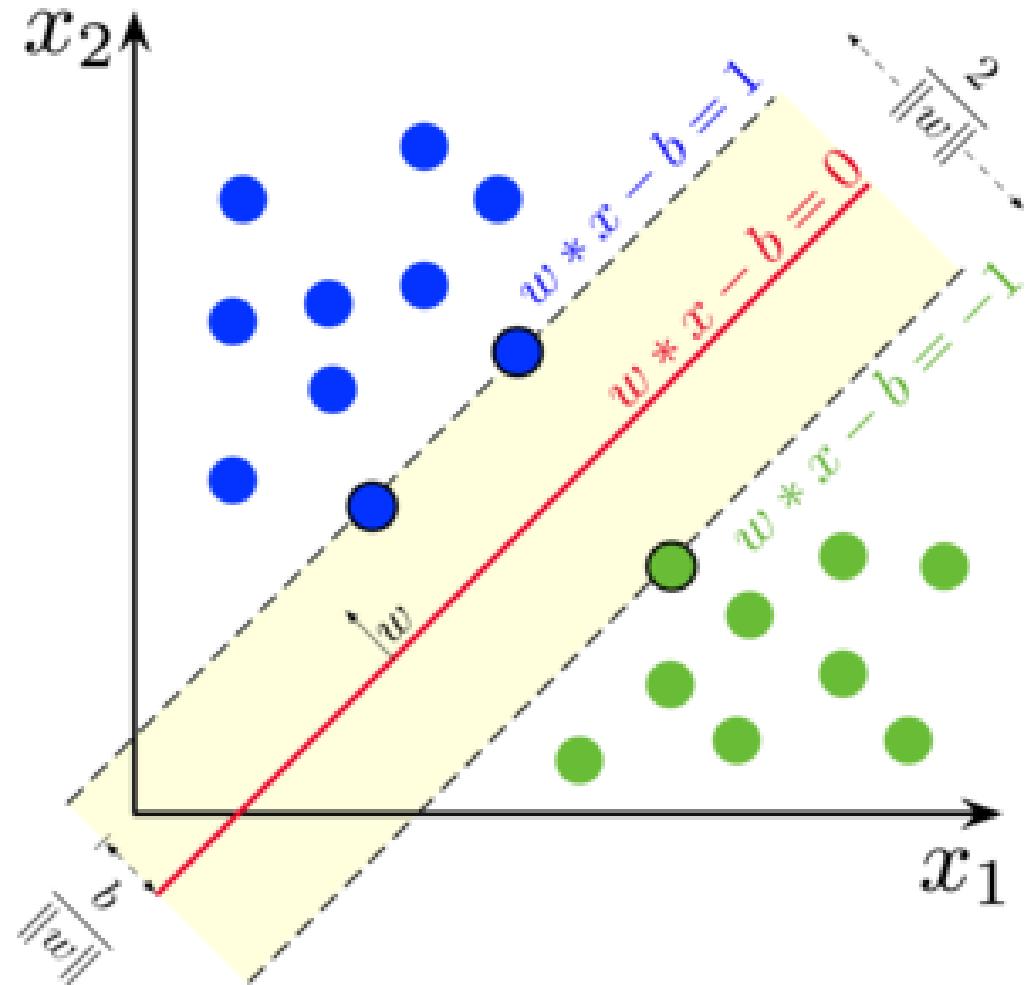
If  $y = 0$

$\text{cost}_0(z) = 0$  only when  $z \leq -1$

- **SVM** wants a bit more than that - doesn't want to \*just\* get it right, but have the value be quite a bit bigger than zero, this throws in an extra **safety margin factor**

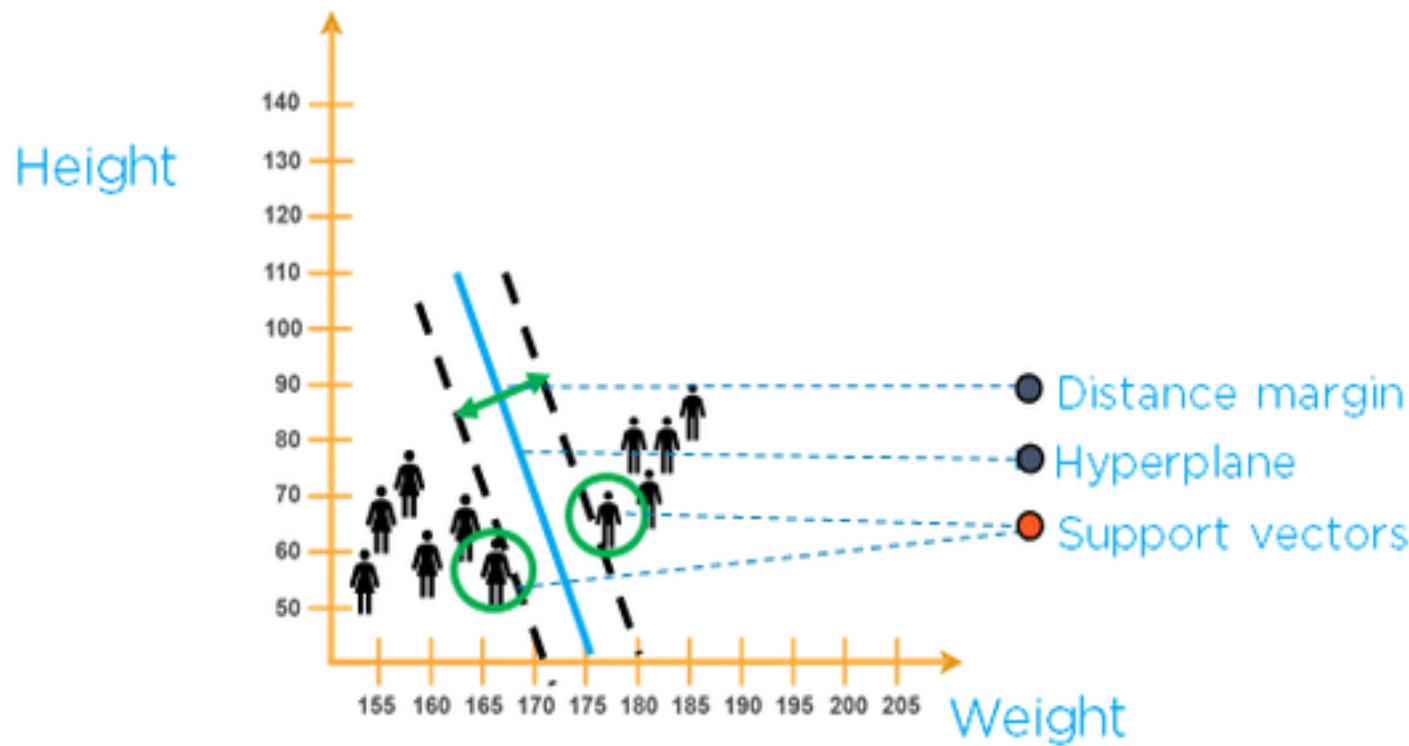
# SVMs: Large Margin Intuition

- A Support Vector Machine (**SVM**) performs classification by finding the hyperplane that maximizes the distance **margin** between the two classes.



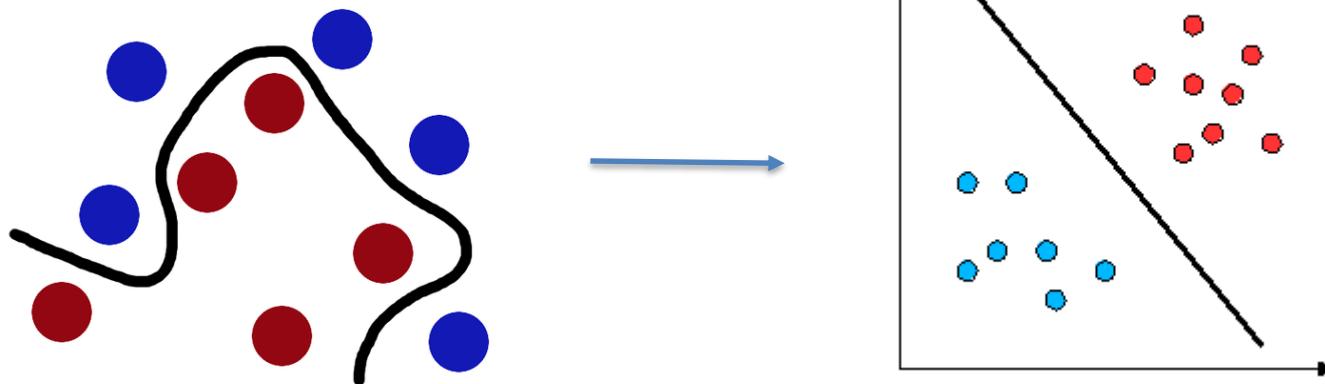
## SVMs: Support Vectors

- The vectors that define the hyperplane are the **support vectors**.



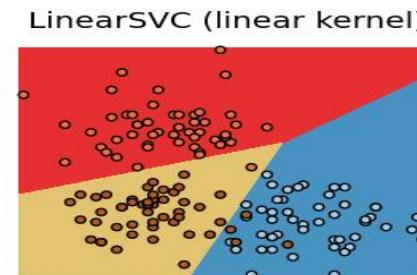
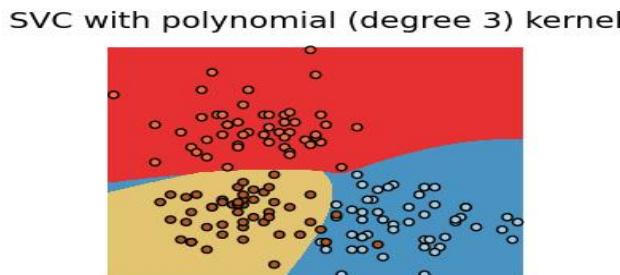
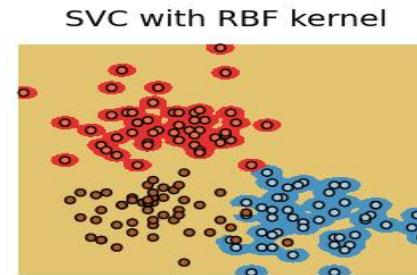
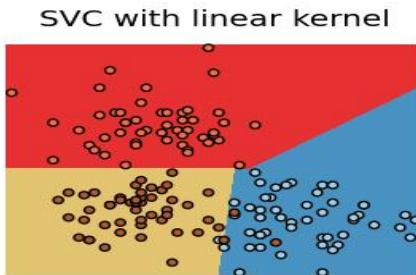
## SVMs: Kernel Functions

- In machine learning, a “**kernel**” is usually used to refer to the kernel trick, a method of using a linear classifier to solve a non-linear problem.
- It entails transforming linearly inseparable data like to linearly separable ones.



# SVMs: Kernel Functions

- **Mathematical definition:**  $K(x, y) = \langle f(x), f(y) \rangle$ .  
Here  $K$  is the kernel function,  $x, y$  are  $n$  dimensional inputs.  $f$  is a map from  $n$ -dimension to  $m$ -dimension space.  $\langle x, y \rangle$  denotes the dot product. usually  $m$  is much larger than  $n$ .



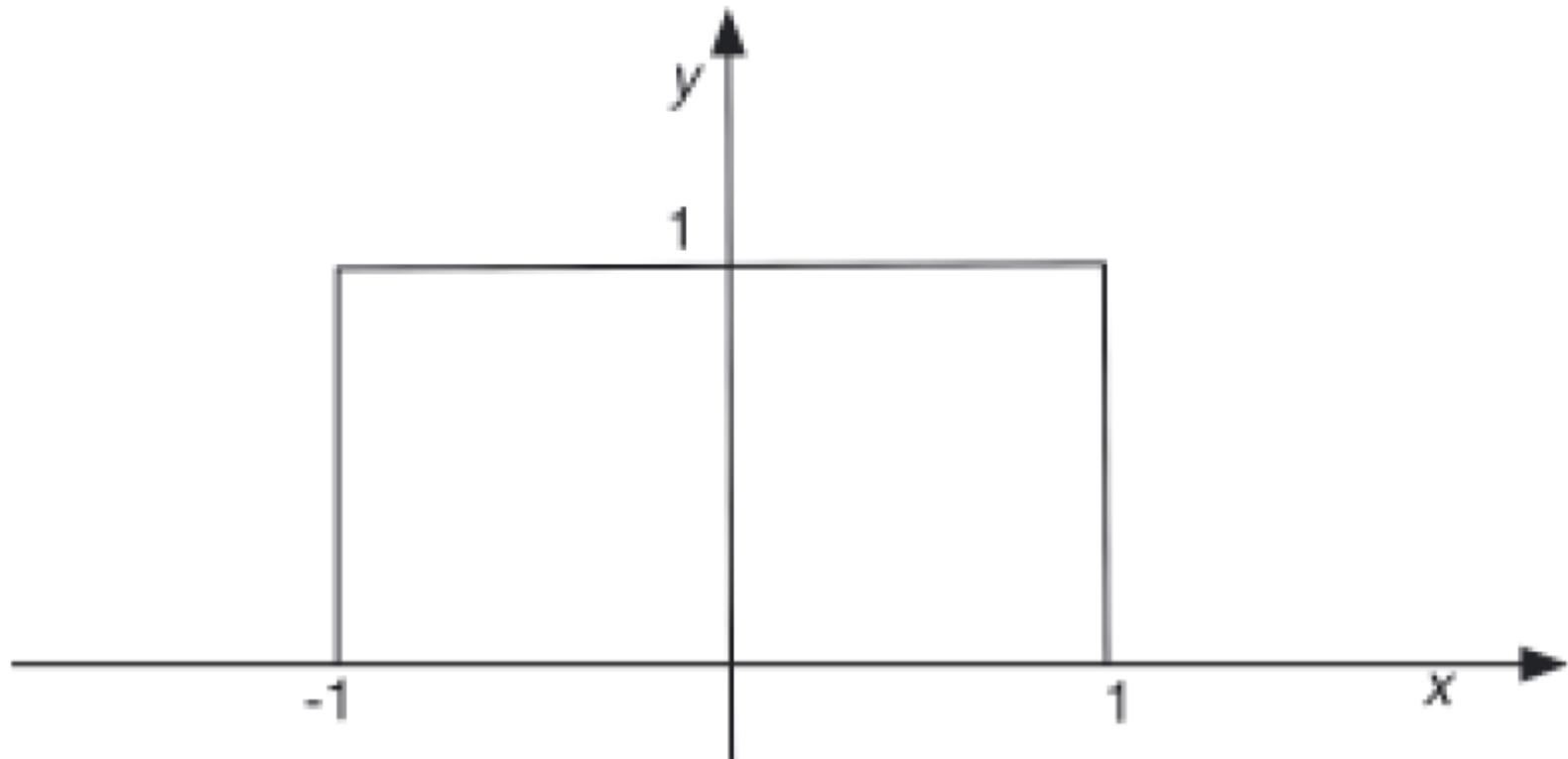
## SVMs: Kernel Functions

- The aim of **kernels** is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example ***linear, nonlinear, polynomial, radial basis function***
- The most used type of kernel function is **RBF**. Because it has localized and finite response along the entire x-axis.
- The kernel functions return the inner product between two points in a suitable feature space

## SVMs: Kernel Rules

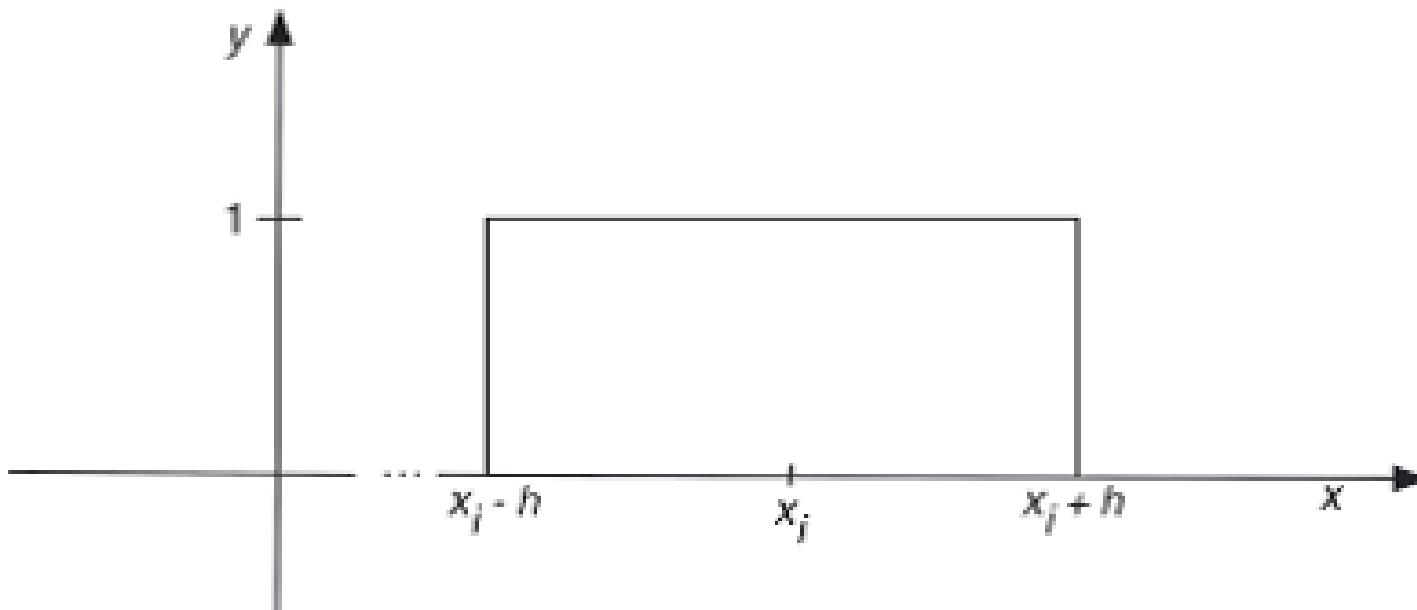
- Window Kernel

$$K(\bar{x}) = \begin{cases} 1 & \text{if } \|\bar{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



## SVMs: Kernel Rules

- For a fixed  $x_i$ , the function is  $K(z-x_i)/h = 1$  inside the closed ball of radius  $h$  centered at  $x_i$



# SVMs: Other Kernels

## 4.1. Polynomial kernel

It is popular in image processing.

Equation is:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$$

*Polynomial kernel equation*

where d is the degree of the polynomial.

## 4.2. Gaussian kernel

It is a general-purpose kernel; used when there is no prior knowledge about the data. Equation is:

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

*Gaussian kernel equation*

$\gamma > 0$

*Gaussian*

*radial*

*basis*

*function*

*(RBF)*

## 4.3. Gaussian radial basis function (RBF)

It is a general-purpose kernel; used when there is no prior knowledge about the data.

Equation is:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

*Gaussian radial basis function (RBF)*

# SVMs: Other Kernels

## Laplace RBF kernel

It is general-purpose kernel; used when there is no prior knowledge about the data.  
Equation is:

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{\sigma}\right)$$

*Laplace RBF kernel equation*

## Hyperbolic tangent kernel

We can use it in neural networks.

Equation is:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \cdot \mathbf{x}_j + c)$$

*Hyperbolic tangent kernel equation*

, for some (not every)  $\kappa > 0$  and  $c < 0$ .

## Sigmoid kernel

We can use it as the proxy for neural networks. Equation is

$$k(x, y) = \tanh(\alpha x^T y + c)$$

*Sigmoid kernel equation*

# SVM in Python: Scikit Learn

## `sklearn.svm.SVC`

```
class sklearn.svm. SVC (C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True,  
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1,  
decision_function_shape='ovr', random_state=None)
```

[source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using

`sklearn.linear_model.LinearSVC` or `sklearn.linear_model.SGDClassifier` instead, possibly after a `sklearn.kernel_approximation.Nystroem` transformer.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how `gamma`, `coef0` and `degree` affect each other, see the corresponding section in the narrative documentation: [Kernel functions](#).

Read more in the [User Guide](#).

**Parameters:** `C : float, optional (default=1.0)`

Penalty parameter C of the error term.

`kernel : string, optional (default='rbf')`

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

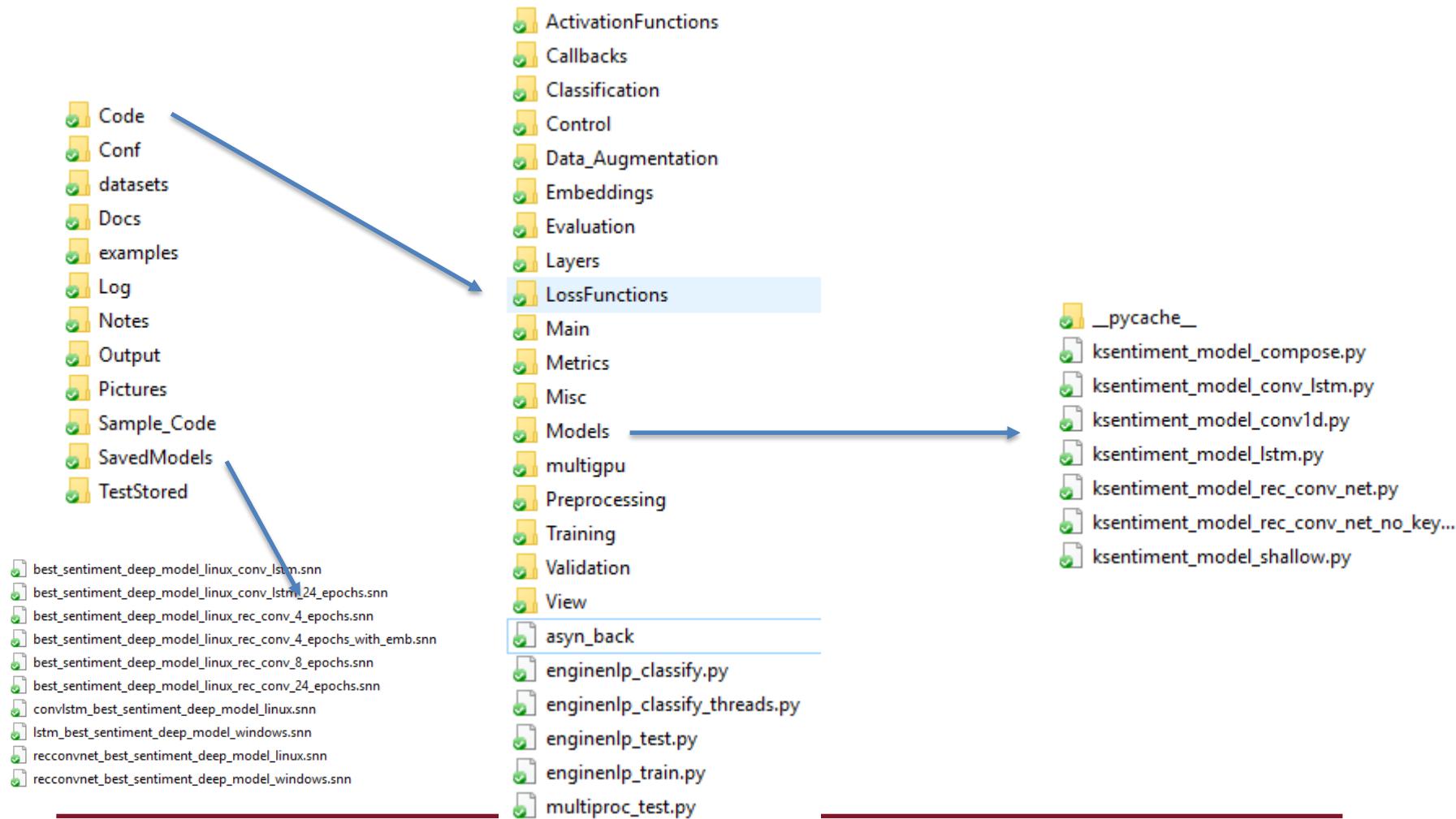
`degree : int, optional (default=3)`

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

`gamma : float, optional (default='auto')`

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

# Anatomy of an advanced Python Machine Learning Application



# Anatomy of a Machine Learning application in Keras

## Keras: The Python Deep Learning library



# Keras

### You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#). It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at [Keras.io](#).

Keras is compatible with: **Python 2.7-3.6**.

# Getting started: 30 seconds to Keras Library

The core data structure of Keras is a **model**, a way to organize layers. The simplest type of model is the [Sequential](#) model, a linear stack of layers. For more complex architectures, you should use the [Keras functional API](#), which allows to build arbitrary graphs of layers.

Here is the [Sequential](#) model:

```
from keras.models import Sequential  
model = Sequential()
```

Stacking layers is as easy as [.add\(\)](#):

```
from keras.layers import Dense  
  
model.add(Dense(units=64, activation='relu', input_dim=100))  
model.add(Dense(units=10, activation='softmax'))
```

# Getting started: 30 seconds to Keras Library

Once your model looks good, configure its learning process with `.compile()`:

```
model.compile(loss='categorical_crossentropy',
               optimizer='sgd',
               metrics=['accuracy'])
```

If you need to, you can further configure your optimizer. A core principle of Keras is to make things reasonably simple, while allowing the user to be fully in control when they need to (the ultimate control being the easy extensibility of the source code).

```
model.compile(loss=keras.losses.categorical_crossentropy,
               optimizer=keras.optimizers.SGD(lr=0.01, momentum=0.9, nesterov=True))
```

You can now iterate on your training data in batches:

```
# x_train and y_train are Numpy arrays --just like in the Scikit-Learn API.
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

# Getting started: 30 seconds to Keras Library

Alternatively, you can feed batches to your model manually:

```
model.train_on_batch(x_batch, y_batch)
```

Evaluate your performance in one line:

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

Or generate predictions on new data:

```
classes = model.predict(x_test, batch_size=128)
```

Building a question answering system, an image classification model, a Neural Turing Machine, or any other model is just as fast. The ideas behind deep learning are simple, so why should their implementation be painful?

# Keras Sequential model

The `Sequential` model is a linear stack of layers.

You can create a `Sequential` model by passing a list of layer instances to the constructor:

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

You can also simply add layers via the `.add()` method:

```
model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
```

# Keras Sequential model

## Compilation

Before training a model, you need to configure the learning process, which is done via the `compile` method. It receives three arguments:

- An optimizer. This could be the string identifier of an existing optimizer (such as `rmsprop` or `adagrad`), or an instance of the `Optimizer` class. See: [optimizers](#).
- A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as `categorical_crossentropy` or `mse`), or it can be an objective function. See: [losses](#).
- A list of metrics. For any classification problem you will want to set this to `metrics=['accuracy']`. A metric could be the string identifier of an existing metric or a custom metric function.

```
# For a multi-class classification problem
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# For a binary classification problem
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# For a mean squared error regression problem
model.compile(optimizer='rmsprop',
              loss='mse')

# For custom metrics
import keras.backend as K

def mean_pred(y_true, y_pred):
    return K.mean(y_pred)

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy', mean_pred])
```

# Keras Sequential model

## Training

Keras models are trained on Numpy arrays of input data and labels. For training a model, you will typically use the `fit` function. [Read its documentation here.](#)

```
# For a single-input model with 2 classes (binary classification):

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(2, size=(1000, 1))

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, labels, epochs=10, batch_size=32)
```

```
# For a single-input model with 10 classes (categorical classification):

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(10, size=(1000, 1))

# Convert labels to categorical one-hot encoding
one_hot_labels = keras.utils.to_categorical(labels, num_classes=10)

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, one_hot_labels, epochs=10, batch_size=32)
```

# Keras Sequential model

Multilayer Perceptron (MLP) for multi-class softmax classification:

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD

# Generate dummy data
import numpy as np
x_train = np.random.random((1000, 20))
y_train = keras.utils.to_categorical(np.random.randint(10, size=(1000, 1)), num_classes=10)
x_test = np.random.random((100, 20))
y_test = keras.utils.to_categorical(np.random.randint(10, size=(100, 1)), num_classes=10)

model = Sequential()
# Dense(64) is a fully-connected layer with 64 hidden units.
# in the first layer, you must specify the expected input data shape:
# here, 20-dimensional vectors.
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

model.fit(x_train, y_train,
          epochs=20,
          batch_size=128)
score = model.evaluate(x_test, y_test, batch_size=128)
```

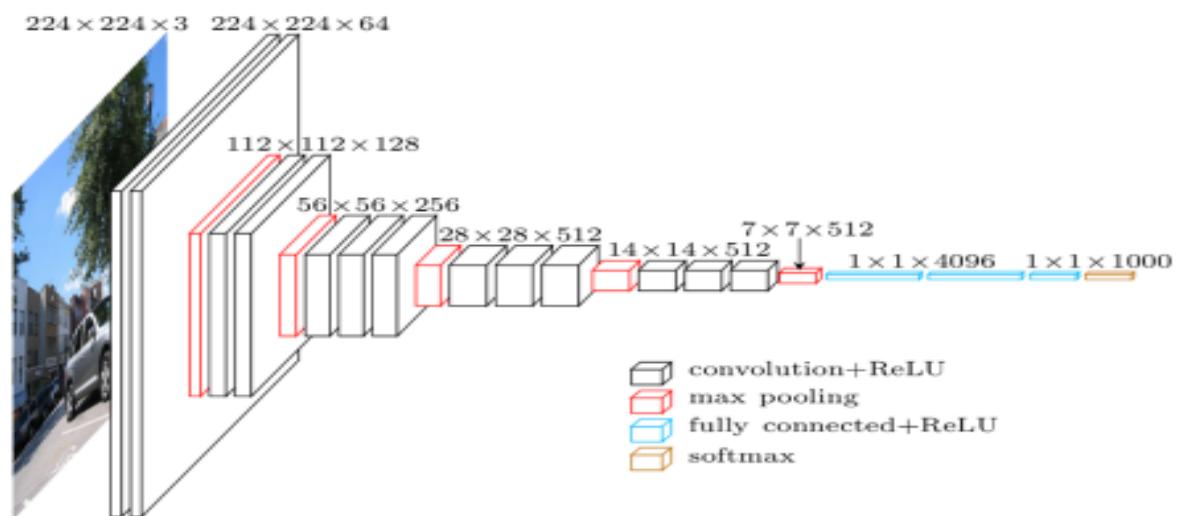
# VGG-Net – University of Oxford

## Critical Features (Simonyan, K., & Zisserman, A., 2014 ):

- **Kernels with small receptive fields:**  $3 \times 3$  which is the smallest size to capture the notion of left/right up/down, center. It is easy to see that a stack of two  $3 \times 3$  conv. layers (without spatial pooling in between) has an effective receptive field of  $5 \times 5$ , and so on.
- Small size **Receptive Field** is a way to increase the nonlinearity of the decision function fields of the conv. layers.
- **Increasing depth architectures:** **VGG-16** (2xConv3-64, 2xConv3-128, 3xConv3-256, 6xConv3-512, 3xFC), **VGG-19** (same as VGG-16 but with 8xConv3-512).
- **Upside:** less complex topology, outperforms GoogleNet on single-network classification accuracy
- **Downside: 138 million parameters for VGG-**

## Results:

- **Multi ConvNet model :** (D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop & dense eval: **23.7%** Top-1 Error, **6.8%** Top-5 Error.



# VggNet in Keras

```
class VGG_16:  
  
    @staticmethod  
    def build(width, height, depth, classes, mul_factor, summary, weightsPath=None):  
  
        model = Sequential()  
        model.add(ZeroPadding2D((1,1),input_shape=(depth, height, width)))  
        model.add(Convolution2D(64, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(64, 3, 3, activation='relu'))  
        model.add(MaxPooling2D((2,2), strides=(2,2)))  
  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(128, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(128, 3, 3, activation='relu'))  
        model.add(MaxPooling2D((2,2), strides=(2,2)))  
  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(256, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(256, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(256, 3, 3, activation='relu'))  
        model.add(MaxPooling2D((2,2), strides=(2,2)))
```

# VggNet in Keras

```
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(classes, activation='softmax'))

if summary==True:
    model.summary()

if weightsPath:
    model.load_weights(weightsPath)

return model
```

# Useful Python Libraries

`platform` — Access to underlying platform's identifying data

**Source code:** [Lib/platform.py](#)

**Note:** Specific platforms listed alphabetically, with Linux included in the Unix section.

`platform.system()`

Returns the system/OS name, e.g. `'Linux'`, `'Windows'`, or `'Java'`. An empty string is returned if the value cannot be determined.

# Useful Python Libraries

## 26.2. pdb – The Python Debugger

Source code: [Lib/pdb.py](#)

The module `pdb` defines an interactive source code debugger for Python programs. It supports setting (conditional) breakpoints and single stepping at the source line level, inspection of stack frames, source code listing, and evaluation of arbitrary Python code in the context of any stack frame. It also supports post-mortem debugging and can be called under program control.

The debugger is extensible — it is actually defined as the class `Pdb`. This is currently undocumented but easily understood by reading the source. The extension interface uses the modules `bdb` and `cmd`.

The debugger's prompt is `(Pdb)`. Typical usage to run a program under control of the debugger is:

```
>>> import pdb
>>> import mymodule
>>> pdb.run('mymodule.test()')
> <string>(0)?()
(Pdb) continue
> <string>(1)?()
(Pdb) continue
NameError: 'spam'
> <string>(1)?()
(Pdb)
```

`pdb.set_trace()`

Enter the debugger at the calling stack frame. This is useful to hard-code a breakpoint at a given point in a program, even if the code is not otherwise being debugged (e.g. when an assertion fails).

# Useful Python Libraries

## Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](#) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](#).

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

[Natural Language Processing with Python](#) provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The online version of the book has been updated for Python 3 and NLTK 3. (The original Python 2 version is still available at [http://nltk.org/book\\_1ed.](http://nltk.org/book_1ed.))

# Useful Python Libraries

## Some simple things you can do with NLTK

Tokenize and tag some text:

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

# NLP: A Textual Classifier with Keras – 20NewsGroup Dataset

## GloVe word embeddings

We will be using GloVe embeddings, which you can read about [here](#). GloVe stands for "Global Vectors for Word Representation". It's a somewhat popular embedding technique based on factorizing a matrix of word co-occurrence statistics.

Specifically, we will use the 100-dimensional GloVe embeddings of 400k words computed on a 2014 dump of English Wikipedia. You can download them [here](#) (warning: following this link will start a 822MB download).

## 20 Newsgroup dataset

The task we will try to solve will be to classify posts coming from 20 different newsgroup, into their original 20 categories --the infamous "20 Newsgroup dataset". You can read about the dataset and download the raw text data [here](#).

<b>comp.graphics</b> <b>comp.os.ms-windows.misc</b> <b>comp.sys.ibm.pc.hardware</b> <b>comp.sys.mac.hardware</b> <b>comp.windows.x</b>	<b>rec.autos</b> <b>rec.motorcycles</b> <b>rec.sport.baseball</b> <b>rec.sport.hockey</b>	<b>sci.crypt</b> <b>sci.electronics</b> <b>sci.med</b> <b>sci.space</b>
<b>misc.forsale</b>	<b>talk.politics.misc</b> <b>talk.politics.guns</b> <b>talk.politics.mideast</b>	<b>talk.religion.misc</b> <b>alt.atheism</b> <b>soc.religion.christian</b>

# NLP Preprocessing: Dataset Loading

```
texts = [] # list of text samples
labels_index = {} # dictionary mapping label name to numeric id
labels = [] # list of label ids
for name in sorted(os.listdir(TEXT_DATA_DIR)):
    path = os.path.join(TEXT_DATA_DIR, name)
    if os.path.isdir(path):
        label_id = len(labels_index)
        labels_index[name] = label_id
        for fname in sorted(os.listdir(path)):
            if fname.isdigit():
                fpath = os.path.join(path, fname)
                if sys.version_info < (3,):
                    f = open(fpath)
                else:
                    f = open(fpath, encoding='latin-1')
                t = f.read()
                i = t.find('\n\n') # skip header
                if 0 < i:
                    t = t[i:]
                texts.append(t)
                f.close()
                labels.append(label_id)

print('Found %s texts.' % len(texts))
```

## ARCHIVED: What is the Latin-1 (ISO-8859-1) character set?

This content has been [archived](#), and is no longer maintained by Indiana University. Resources linked from this page may no longer be available or reliable.

Latin-1, also called ISO-8859-1, is an 8-bit character set endorsed by the International Organization for Standardization (ISO) and represents the alphabets of Western European languages. As its name implies, it is a subset of ISO-8859, which includes several other related sets for writing systems like Cyrillic, Hebrew, and Arabic. It is used by most [Unix](#) systems as well as Windows, DOS and Mac OS, however, use their own sets.

## Building Texts List and Labels List

# NLP Preprocessing: Dataset Loading

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(nb_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
```

Stop-words Cleaning and tokenization

```
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))

data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
```

Building of the Word Index

```
labels = to_categorical(np.asarray(labels))
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)

# split the data into a training set and a validation set
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
nb_validation_samples = int(VALIDATION_SPLIT * data.shape[0])

x_train = data[:-nb_validation_samples]
y_train = labels[:-nb_validation_samples]
x_val = data[-nb_validation_samples:]
y_val = labels[-nb_validation_samples:]
```

Sequence 1D 0-Padding

To categorical variables conversion

Shuffle of Data and Labels

Data and Labels Splitting

# Practical Machine Learning Case Study

- The goal of this **Case Study** is to predict how much a house will sell for. This is for example similar to many problems Amazon has in online advertising and merchandising, where they try to predict what value a customer has for an item.
  - In order to reach this objective we have to produce a **Machine Learning** model and an evaluation metric.
  - You can find and download the data at:  
<http://bit.ly/1Fq0svx>
  - A description of the columns in the data is available at:  
<http://bit.ly/1hh97JI>

The file to download is named “**HM Land Registry Price Paid Data**”

# UK Government's Land Registry Dataset Description

## Explanations of column headers in the PPD

The data is published in columns in the order set out in the table, we do not supply column headers in the files.

Data item	Explanation (where appropriate)
Transaction unique identifier	A reference number which is generated automatically recording each published sale. The number is unique and will change each time a sale is recorded.
Price	Sale price stated on the transfer deed.
Date of Transfer	Date when the sale was completed, as stated on the transfer deed.
Postcode	This is the postcode used at the time of the original transaction. Note that postcodes can be reallocated and these changes are not reflected in the Price Paid Dataset.
Property Type	D = Detached, S = Semi-Detached, T = Terraced, F = Flats/Maisonettes, O = Other Note that: - we only record the above categories to describe property type, we do not separately identify bungalows. - end-of-terrace properties are included in the Terraced category above. - 'Other' is only valid where the transaction relates to a property type that is not covered by existing values.
Old/New	Indicates the age of the property and applies to all price paid transactions, residential and non-residential. Y = a newly built property, N = an established residential building

# UK Government's Land Registry Dataset Description

Duration	Relates to the tenure: F = Freehold, L= Leasehold etc. Note that HM Land Registry does not record leases of 7 years or less in the Price Paid Dataset.
PAON	Primary Addressable Object Name. Typically the house number or name.
SAON	Secondary Addressable Object Name. Where a property has been divided into separate units (for example, flats), the PAON (above) will identify the building and a SAON will be specified that identifies the separate unit/flat.
Street	
Locality	
Town/City	
District	
County	

# UK Government's Land Registry Dataset Description

PPD Category Type	Indicates the type of Price Paid transaction. A = Standard Price Paid entry, includes single residential property sold for full market value. B = Additional Price Paid entry including transfers under a power of sale/reposessions, buy-to-lets (where they can be identified by a Mortgage) and transfers to non-private individuals.
-------------------	--

Note that category B does not separately identify the transaction types stated.  
HM Land Registry has been collecting information on Category A transactions from January 1995. Category B transactions were identified from October 2013.

Record Status - monthly file only	Indicates additions, changes and deletions to the records.(see guide below). A = Addition C = Change D = Delete.
-----------------------------------	---

Note that where a transaction changes category type due to misallocation (as above) it will be deleted from the original category type and added to the correct category with a new transaction unique identifier.

# Practical Machine Learning Case Study

Columns of interest for our task are:

- **Column 1** contains a unique **ID** for the purchase and can be ignored.
- **Column 3** contains the date of the purchase
- **Column 5** contains the property type
- **Column 7** contains the lease duration
- **Column 12** contains the city or the town where the property was located. You can judge a property to London if the field contains the word “London”.



## Acknowledgements

**THANK YOU FOR YOUR ATTENTION**

**FRANCESCO PUGLIESE**