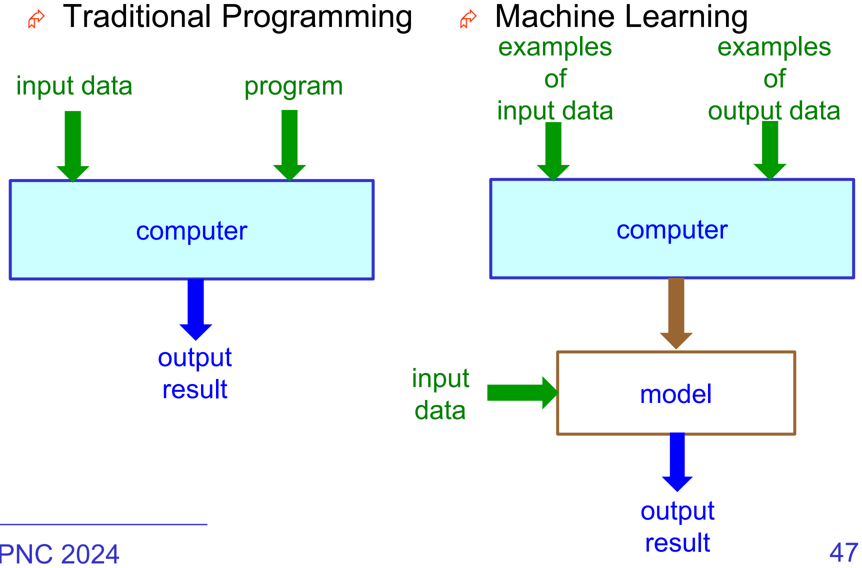
Short Course in Artificial Intelligence

19th June 2024

Lecture 2: AI Applications

Traditional Programming and Machine Learning

- Traditional Programming
- Machine Learning



Traditional Programming and Machine Learning

◆ Traditional Programming ◆ Machine Learning

Find the square of a number

Write Python code:

```
>>> x = int(input())
2
>>> print('the square is', x*x)
the square is 4
```

Provide examples:

[[2,4],[3,9],[4,16],[5,25]…]

Train a ML model

Apply the model to new data

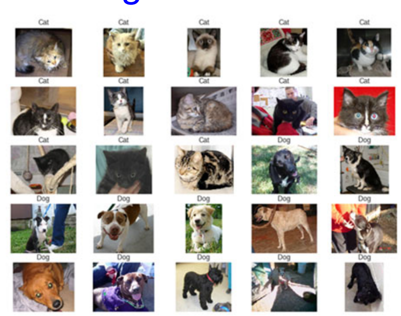
Traditional Programming and Machine Learning

Why is Machine Learning so useful ?

Because we often have problems for which we do not know a formula

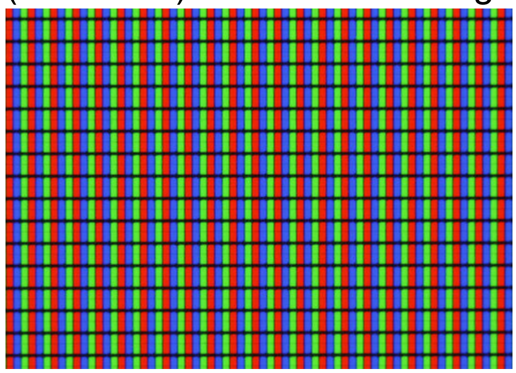
Example: Is an image a cat or dog ?

cat or dog ?



Data as Vectors

- Every type of data can be represented as a vector
- Image: $(\mathrm{w} \times h \times 3)$ for RGB encoding



Data as Vectors

- Every type of data can be represented as a vector
- Words and text:

Vocabulary

Word	w
"the"	1
"and"	2
"dog"	3
"."	4
"OOV"	5

"the"	1
"cat"	5
"and"	2
"the"	1
"dog"	3
"play"	5
"."	4

Data as Vectors

- Every type of data can be represented as a vector
- Data:
  - Yes – No (1) – (0)
  - User profile (Age, Income, family…)
  - …

Artificial Intelligence – Machine Learning

- Learning from data
  - Data = set of objects $\{x\}$ $\{images\}$

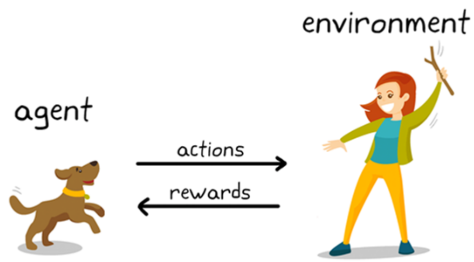
1. Unsupervised learning
   - Groups the objects by similarity
2. Supervised learning $\{(image_1, cat), (image_2, dog)\}$
   - Objects have labels y: $\{(x, y)\}$
   - From training data, build a model to predict the label

$$y = f(x)$$

3. Learning by reinforcement
   - Learn based on reward *(like Tic-Tac-Toe)*

Reinforcement Learning

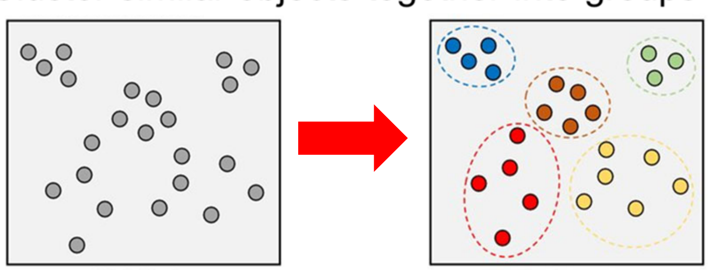
- Data is action, action get reward



- Learning by adjusting rewards
- Typical example: games, moves are rewarded based on winning or losing game

Unsupervised Machine Learning

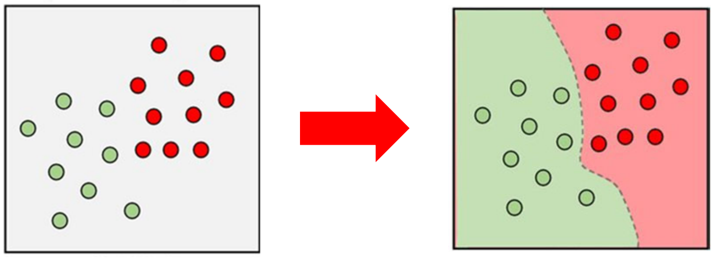
- Data = set of objects $\{x\}$
- Cluster similar objects together into groups



- Typical application: marketing, group customers who are likely to buy the same products

Supervised Machine Learning

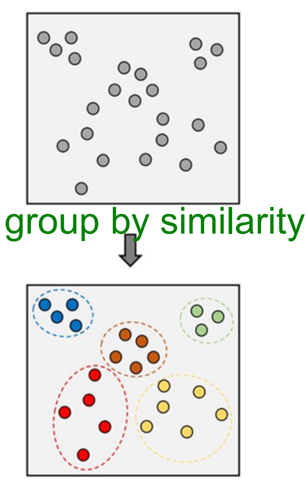
- Data = set of objects with label $\{(x, y)\}$
- Find a model to guess the label y from the value x

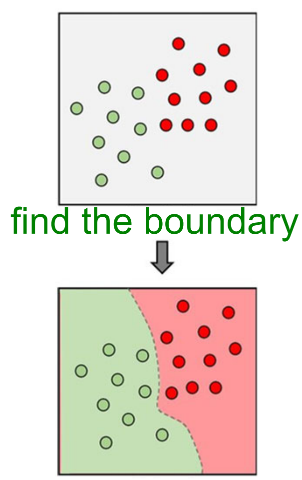


- Typical application: prediction of a value

Machine Learning

- Unsupervised
- Supervised





Supervised Machine Learning

- Which data ?
  - Image → Computer Vision CV
  - Text → Natural Language Processing NLP
  - Data → Example: shopping user data
- Which applications ?
  - Retail: predict what customers will buy
  - Finance: predict future stock value
  - Health: automatic diagnostic
  - Spam detection: fraudulent emails or sms
  - Weather forecast: predict future rain, storm, typhoon
  - Vision: face recognition
  - Natural Language: chatbots
  - …

Supervised Machine Learning

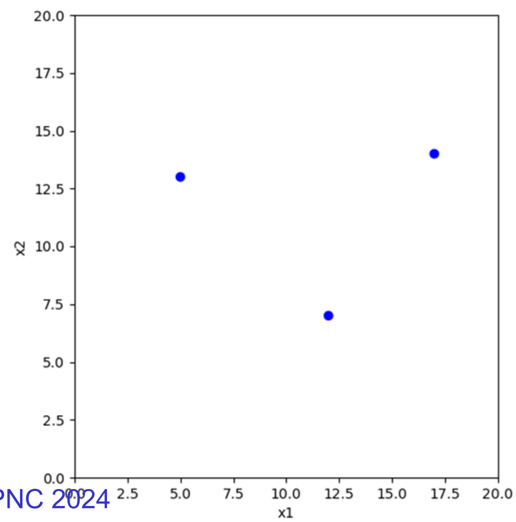
- Data:
  - Object: $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$
  - Training data: set $\{\boldsymbol{x_i}\} = \{(x_{i1}, x_{i2}, \dots, x_{in})\}$
  - Each object $\boldsymbol{x_i}$ has a label y_i :

    $\{(\boldsymbol{x_i}, y_i)\} = \{(x_{i1}, x_{i2}, \dots, x_{in}, y_i)\}$
- Example: student s has grade x at course C
  - We predict grade at C_3 from grades at (C_1, C_2)

Student	C3	C1	C2
s1	16	17	14
s2	9	12	7
s3	7	5	13
…			

Supervised Machine Learning

- Geometrical representation



Student	x1=C1	x2=C2
s1	17	14
s2	12	7
s3	5	13
…		

Supervised Machine Learning

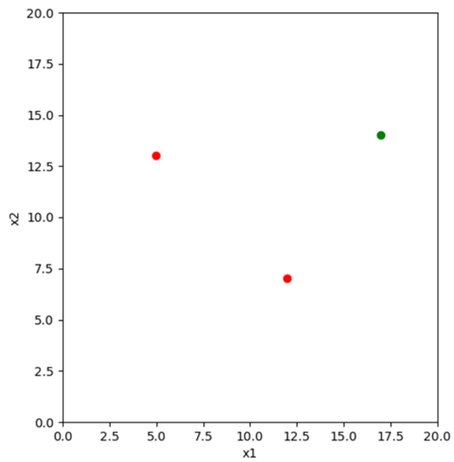
- To make it simpler, we only predict if the student succeeds or fails in C_3
- Data:

Student	C3	C3	C1	C2
s1	S	16	17	14
s2	F	9	12	7
s3	F	7	5	13
…				

- For easier mathematics, we note:
  - Success: S $y_1 = +1$
  - Failure : F $y_2 = -1$ $y_3 = -1$

Supervised Machine Learning

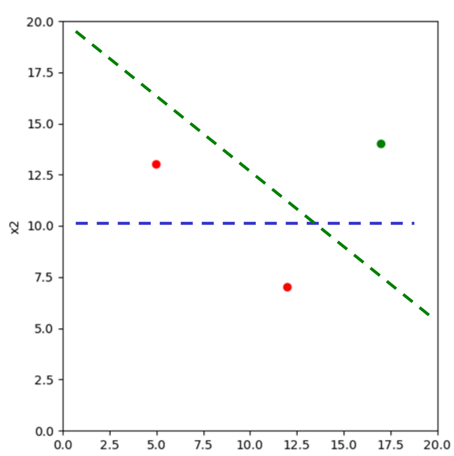
- Geometrical representation



Student	C3	x1=C1	x2=C2
s1	S	17	14
s2	F	12	7
s3	F	5	13
…			

Supervised Machine Learning

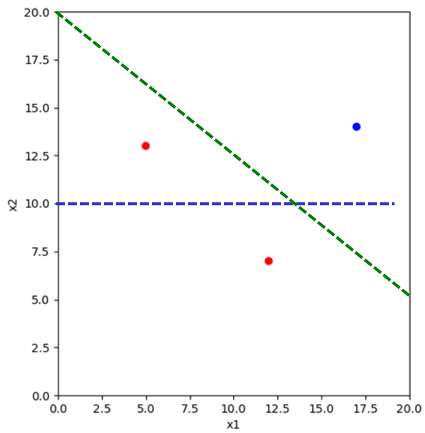
- Linear Classifier



Student	C3	x1=C1	x2=C2
s1	S	17	14
s2	F	12	7
s3	F	5	13
…			

Supervised Machine Learning

- Linear Classifier: $w_1 x_1 + w_2 x_2 + w_0 = 0$



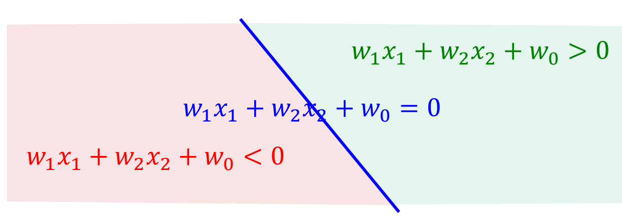
w_1	0
w_2	1
w_0	-10
$x_2 - 10 = 0$	

w_1	3
w_2	4
w_0	-80
$3x_1 + 4x_2 - 80 = 0$	

Supervised Machine Learning

- The boundary of the linear classifier is the line:

  $w_1 x_1 + w_2 x_2 + w_0 = 0$
- The line splits the space into two half-spaces
  - $w_1 x_1 + w_2 x_2 + w_0 > 0$
  - $w_1 x_1 + w_2 x_2 + w_0 < 0$

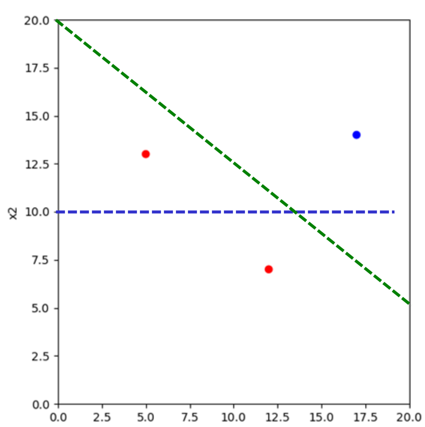


Supervised Machine Learning

- How to use a linear classifier ?
- Given a set of points, we define an error function and we find the values of (w_1, w_2, w_0) that minimize this function
  - The common minimization technique is called “Gradient Descent”
- With the values of (w_1, w_2, w_0) we can classify any point (x_1, x_2):
  - we compute $w_1x_1 + w_2x_2 + w_0$
  - if $w_1x_1 + w_2x_2 + w_0 > 0$, the classifier predicts $y = +1$
  - if $w_1x_1 + w_2x_2 + w_0 < 0$, the classifier predicts $y = -1$

Supervised Machine Learning

- Linear Classifier: $w_1x_1 + w_2x_2 + w_0 = 0$



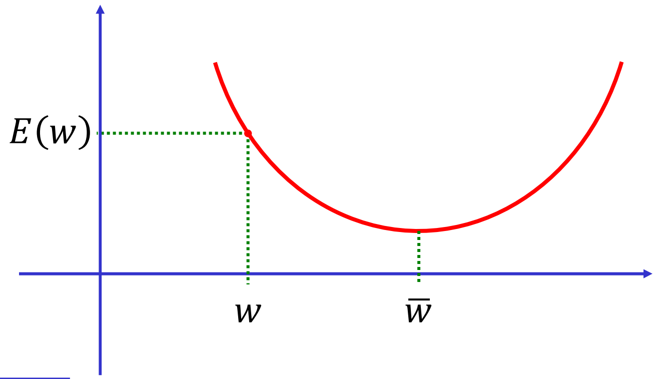
$sign(x_2 - 10)$	
(17,14)	$+4$
(12,7)	-3
(5,13)	$+3$
1 error	

$sign(3x_1 + 4x_2 - 80)$	
(17,14)	27
(12,7)	-16
(5,13)	-13
0 error	

Intuition for Gradient Descent

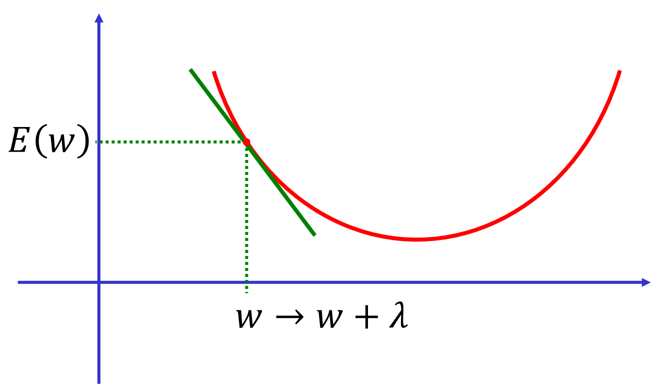
- We want to minimize an Error function $E(w)$

Should we increase or decrease w to reduce the value of $E(w)$?



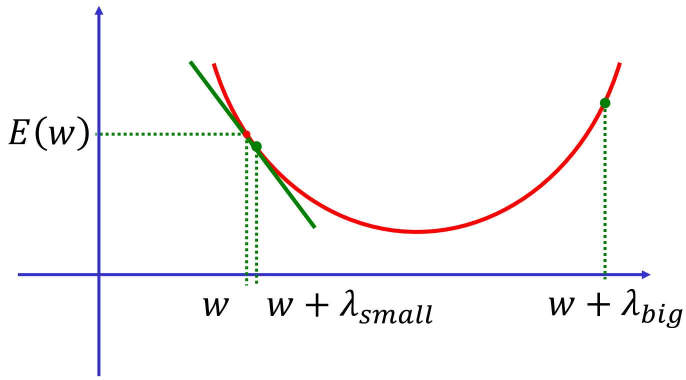
Intuition for Gradient Descent

- We use the derivative to reduce the value $E(w)$
  - if the derivative is negative, we increase $w \to w + \lambda$
  - if the derivative is positive, we decrease $w \to w - \lambda$



Intuition for Gradient Descent

- Problems:
  - if λ is too small, it takes very long to reach $\overline{w}$
  - if λ is too big, we miss $\overline{w}$ and we need to come back

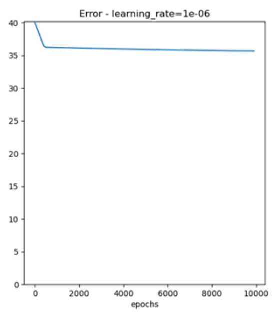


Intuition for Gradient Descent

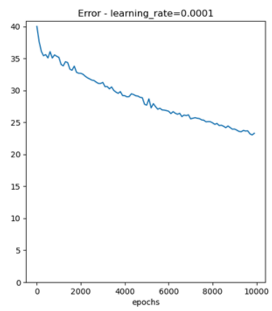
- Gradient Descent procedure:
  - start with an initial value of w
  - compute the derivative
  - use it to modify the value of w by an amount that depends on the learning rate λ
  - check the error to make sure that λ is not too big
  - iterate until convergence
    - the iterations are called “epoch”

Intuition for Gradient Descent

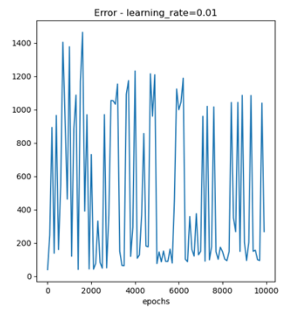
- Gradient Descent procedure:
  - to find a good learning rate, watch how the value of the error changes with the number of epochs



λ too small



good λ



λ too big

Intuition for Gradient Descent

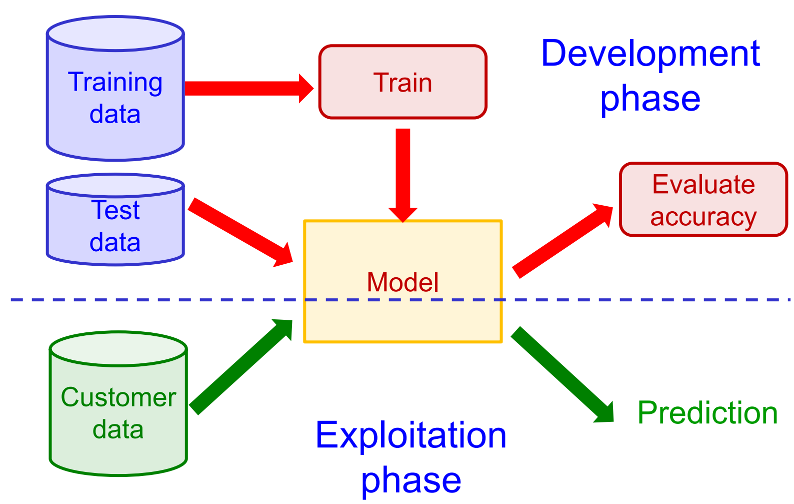
- A good value for the learning rate may be different for each function
- There are many algorithms to change the value of the learning rate during the epochs:
  - Start with a high learning rate $\rightarrow$ move fast
  - Later reduce the learning rate $\rightarrow$ be accurate

AI Applications

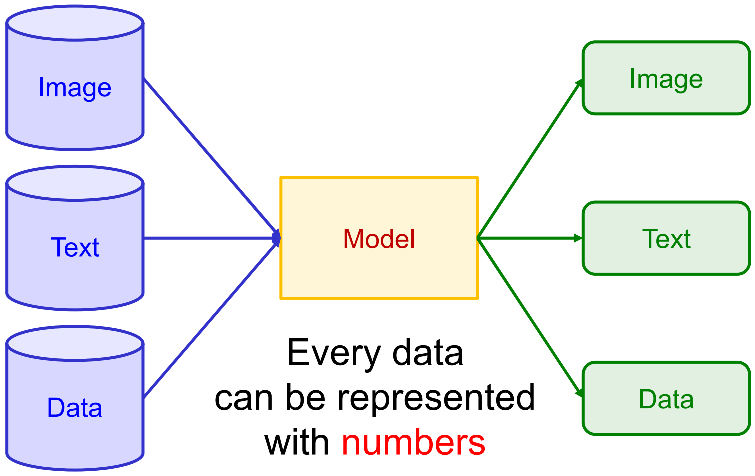
- How to build a AI application ?
- Steps:

1. Collect data
2. Prepare the data (into suitable format)
   - cleaning, labelling, …
3. Choose a model
4. Train the model
   - find the best values of the parameters
5. Evaluate the model
   - on data that was not used to train
6. Make predictions
   - on new data

AI Applications



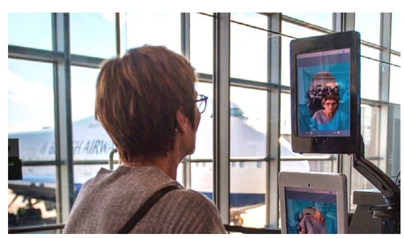
AI Applications



Every data
can be represented
with numbers

AI Applications

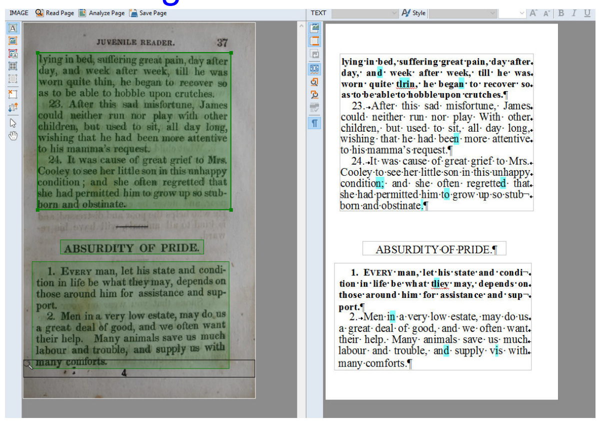
- Image → Data
- Face Recognition
- Emotion detection



- Self-driving cars

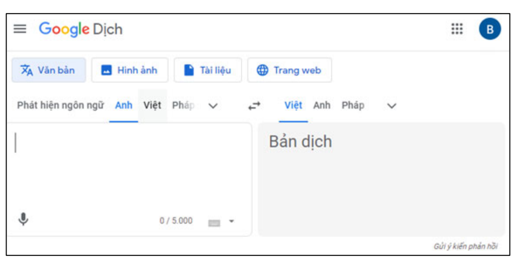
AI Applications

- Image → Text
- Character Recognition



AI Applications

- Text → Text
- Automatic Translation

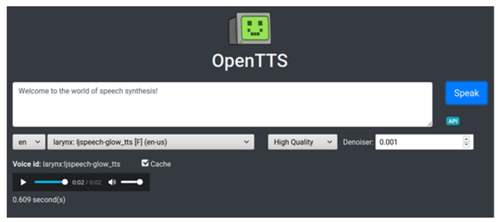


- Summarization
- Information search

AI Applications

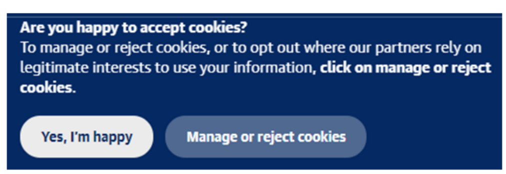
- Speech Data → Text
- Speech Recognition
- Language Learning

- Text → Speech
- Speech Synthesis



AI Applications

- Marketing: targeted advertising
- Is the customer X willing to buy product Y
- Data:
  - customer age
  - salary
  - family
  - address
  - products already bought
  - web pages visited
  - …
- Prediction: the customer will buy Y ? (yes/no)



AI Applications

- Common question to AI engineers from users:
  - how much data do you need ?
- Common answer:
  - as much as possible
- But quality of data is important

DATA = $$

Personal Data = $$$

Quick Introduction to Deep Learning and Demo

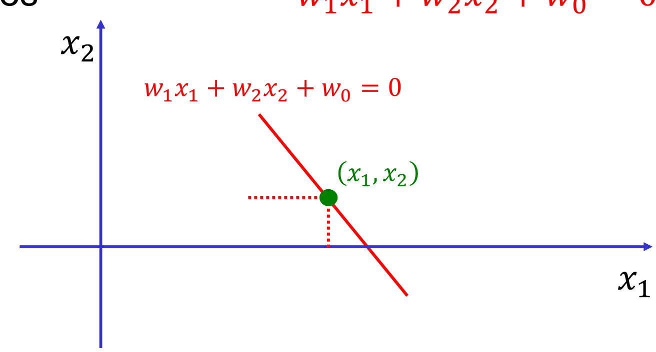
if you are afraid of formulas, don't look at them, just look at the figures

A Quick Course On Deep Networks

- Deep Learning is a particular type of models based on the combination of linear classifiers
- Those models are the basis for most of the recent progress in AI/ML
- They can be applied to (almost) any type of data (text, image, speech, data…), and can be extremely efficient for very diverse tasks.
- The models can be huge (billions of parameters) and be trained on huge training sets.

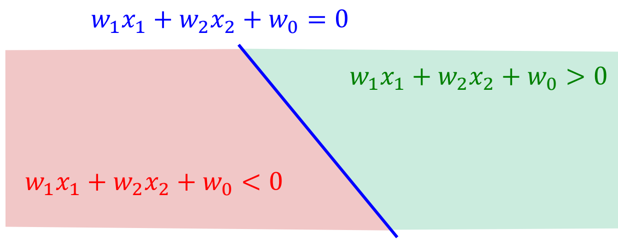
A Quick Course On Deep Networks

- The real plane $\mathbb{R}^2$
- Points (vectors) (x_1, x_2)
- Lines $w_1 x_1 + w_2 x_2 + w_0 = 0$



A Quick Course On Deep Networks

- Lines $w_1x_1 + w_2x_2 + w_0 = 0$
- Half-spaces
  - $w_1x_1 + w_2x_2 + w_0 > 0$
  - $w_1x_1 + w_2x_2 + w_0 < 0$



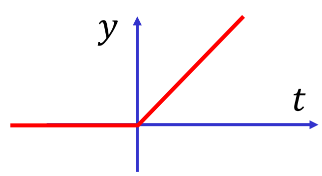
PNC 2024

A Quick Course On Deep Networks

- The ReLU Function (Rectified Linear Unit)

$$ReLU(t) = max(t, 0)$$

- $ReLU(t) = \begin{cases} t & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$



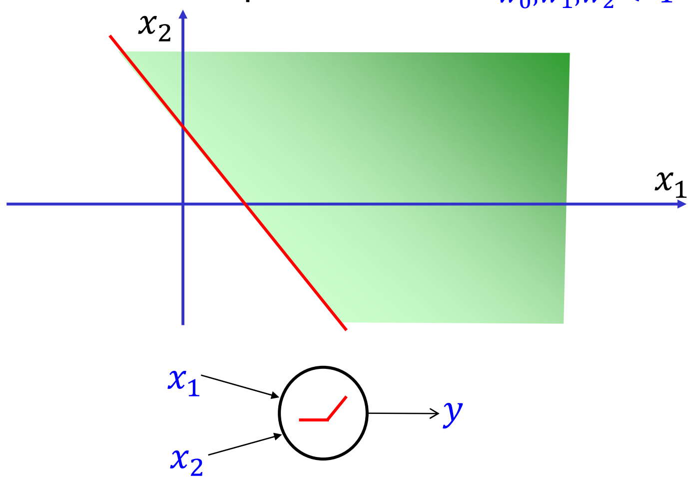
- ReLU on the real plane $\mathbb{R}^2$

$$ReLU(x_1, x_2) = ReLU(w_1x_1 + w_2x_2 + w_0)$$
$$= max(w_1x_1 + w_2x_2 + w_0, 0)$$

PNC 2024

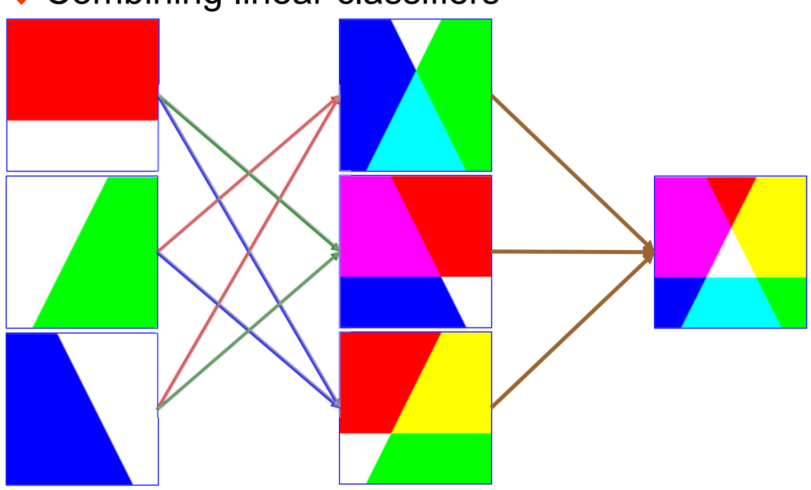
A Quick Course On Deep Networks

- ReLU on the real plane $\mathbb{R}^2$: $ReLU_{w_0,w_1,w_2}(x_1,x_2)$



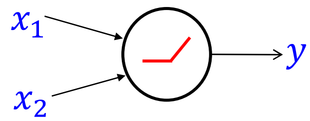
A Quick Course On Deep Networks

- Combining linear classifiers

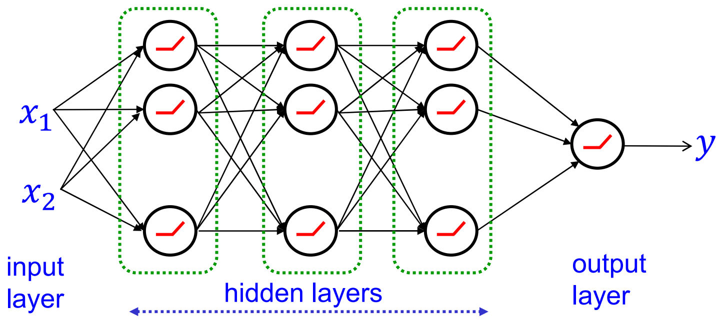


A Quick Course On Deep Networks

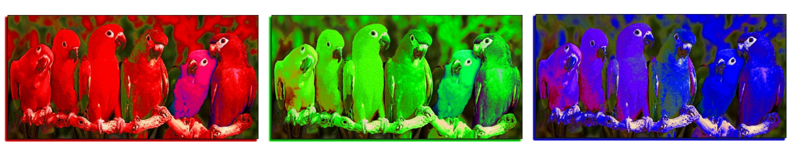
- From Linear Classifier



- To Deep Networks

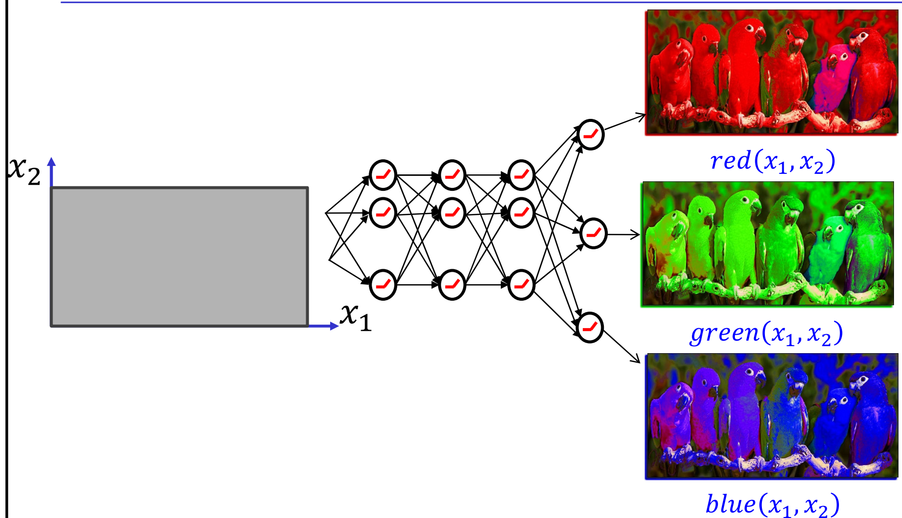


A Quick Deep Network Demo



$red(x_1, x_2)$ $green(x_1, x_2)$ $blue(x_1, x_2)$

A Quick Deep Network



Demo: Learning Deep Networks

- You are given a Javascript code that trains a Deep Network to represent a color image
- You can see how the network evolves during training
- You can modify the number of hidden layers (just cut and paste one line) to see how deeper networks can better represent the image

Demo: Learning Deep Networks

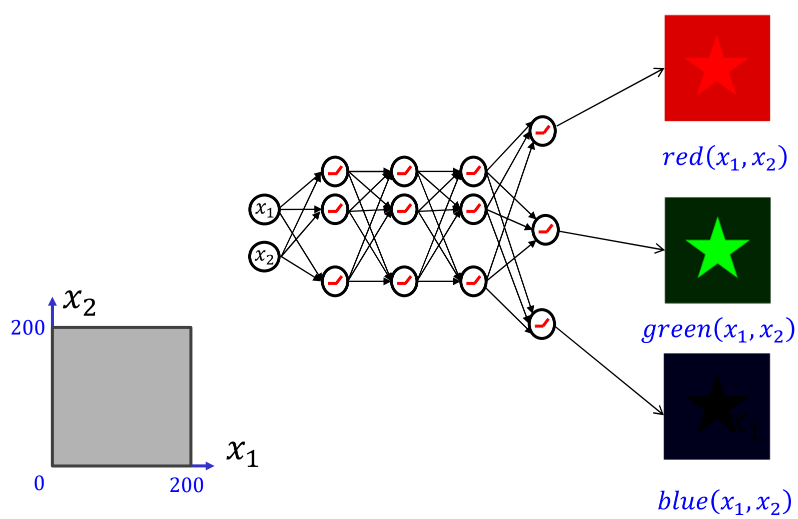
- Data :
  - One image 200x200

- Find models of the color functions:
  - $red(x_1, x_2)$

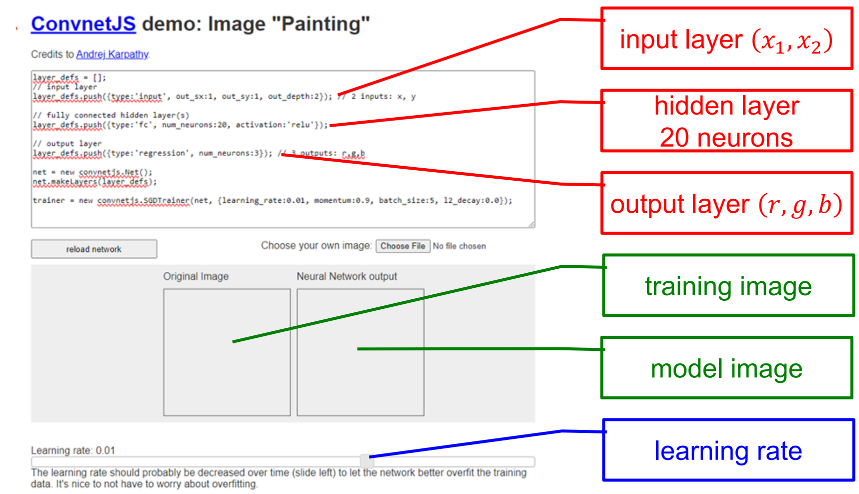
  - $\mathrm{green}(x_1, x_2)$

  - $blue(x_1, x_2)$

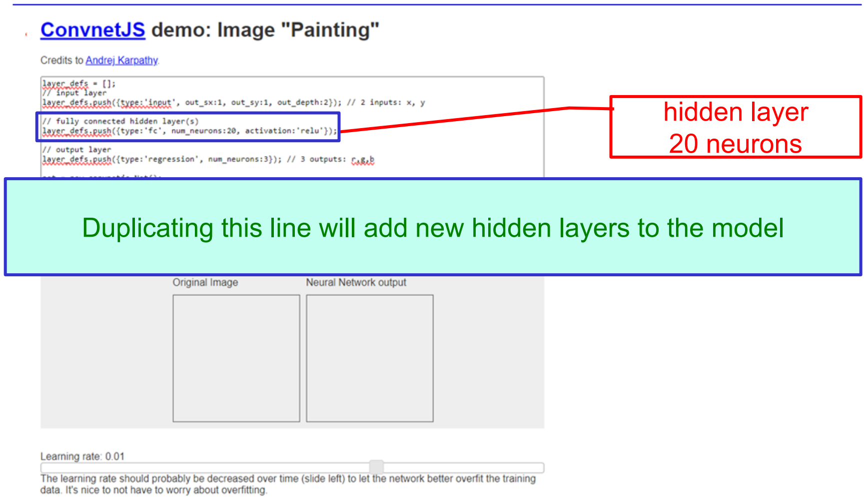
Demo: Learning Deep Networks



Demo: Learning Deep Networks



Demo: Learning Deep Networks



Practice 2: Learning Linear Classifier

- You are provided with a Python notebook that already contains the code to train and visualize
- There are 3 datasets, from easy to more complex
- You should experiment with several learning rates and the number of epochs to achieve a good classifier

- Note: not all values for the learning_rate will lead to a successful training

Practice 2: Learning Linear Classifier

- Log on https://colab.research.google.com/
- File -> upload notebook
- upload practice2.ipynb
- Files -> upload
- upload practice2.data1.data
- Runtime -> Run All

- you can try different values of the learning_rate
- repeat with practice2.data2.data and practice2.data3.data

Practice 2: Learning Linear Classifier

- Data : student and grades

Student	Grade1	Grade2	Success
S1	17	14	1
S2	12	7	-1
S3	5	13	-1

- Goal: build linear classifier to predict success