KENTECH
Korea Institute of Energy Technology

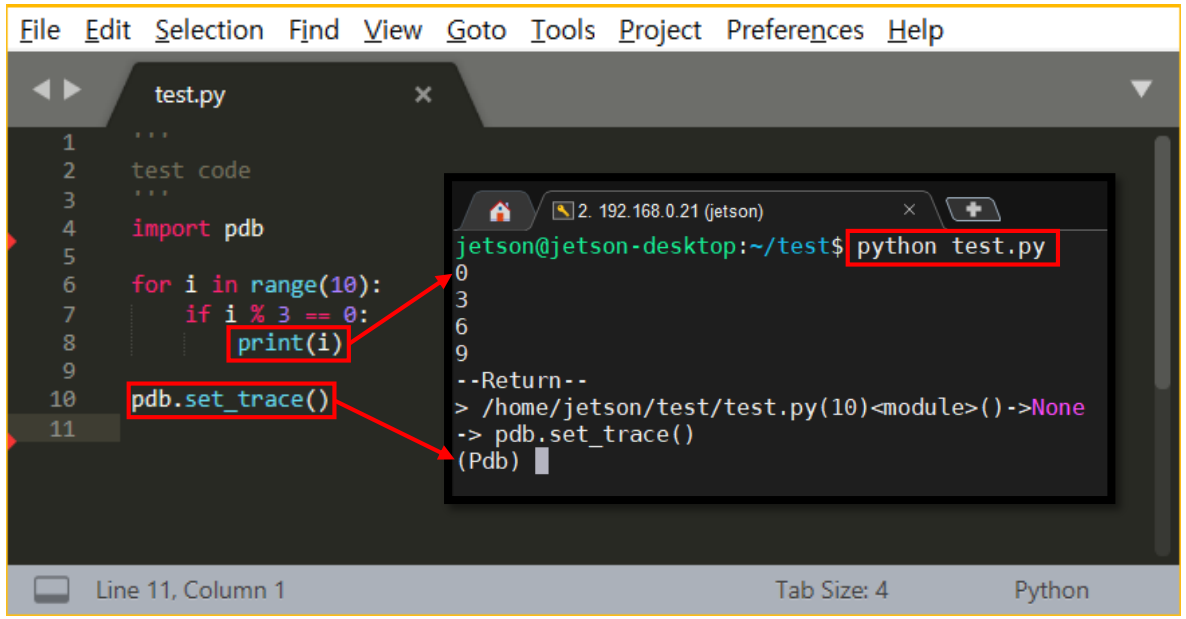
Visionary Course – Energy AI
Week 05

Apr. 1, 2022

Seokju Lee

Discussions: Python Debugging

Please copy the below python code into your server and execute it.



Q1. What happens if you type "c", "n", or "p [variable]" in (Pdb)? Please discuss each role. Type "ctrl+d" or "q" to exit (Pdb).

```
(Pdb) c
(Pdb) n
(Pdb) p i
```

Q2. Please insert break points into *line 7* (right after the loop begins), and into *line 8* (right after the condition is satisfied). What is the value of "i" after each loop? Please trace the value.

```
(Pdb) p i
(Pdb) c
```

Q3. What is the meaning of the operator "%"?

Q4. Please implement a code to determine whether an input is a prime number or not.

```
### script ###
n = int(input())
is_prime = True
for i in range(2, n):
    if n % i == 0:
        is_prime = False

print("{} is prime: {}".format(n, is_prime))
```

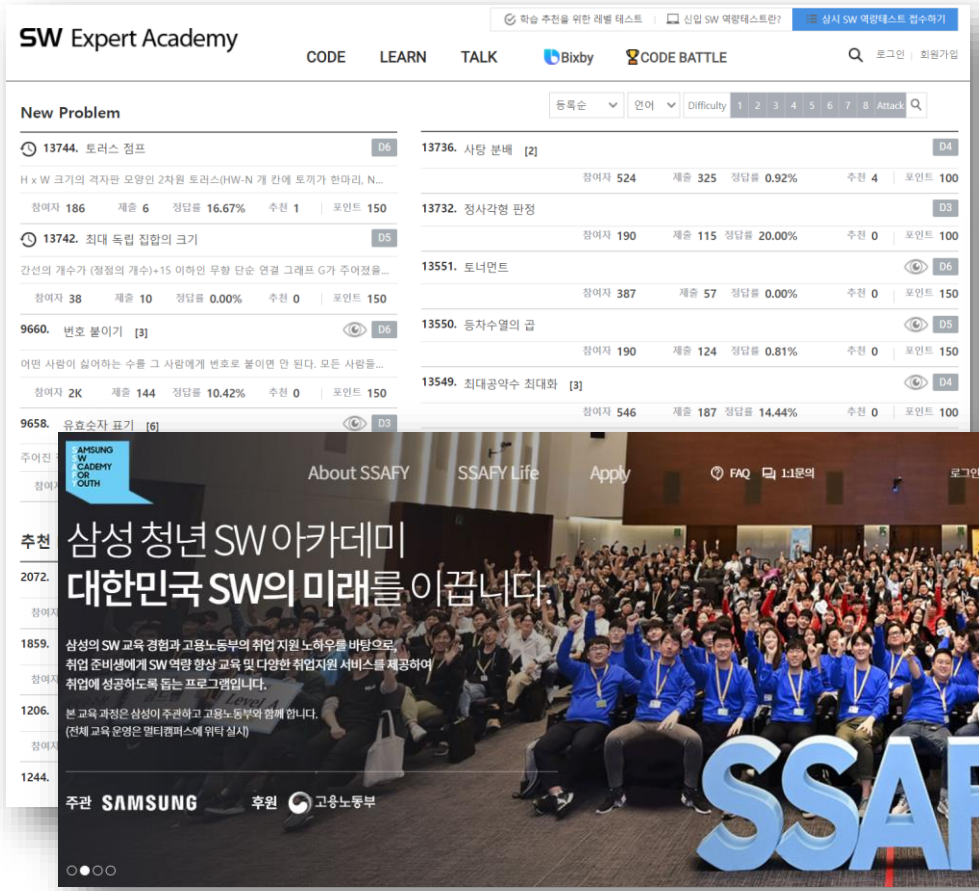
The number whose **only factors** are **1** and **itself**.

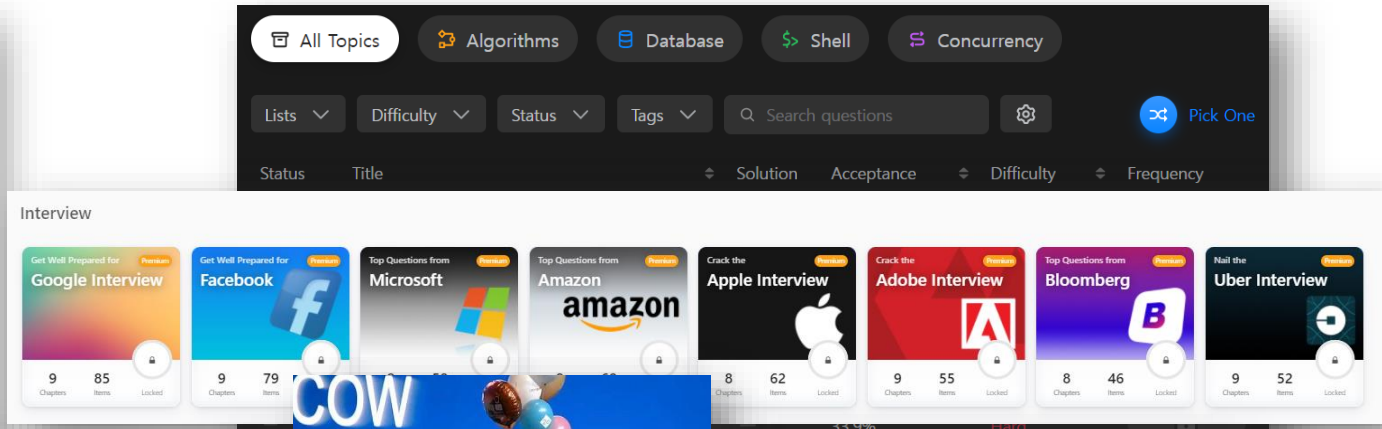
If You Want to Explore More about Programming,

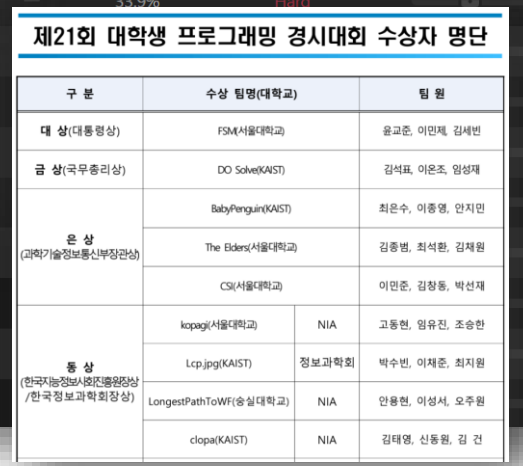
Recommendations for your self-study

→ Samsung SW Expert Academy (link)

→ LeetCode (link)







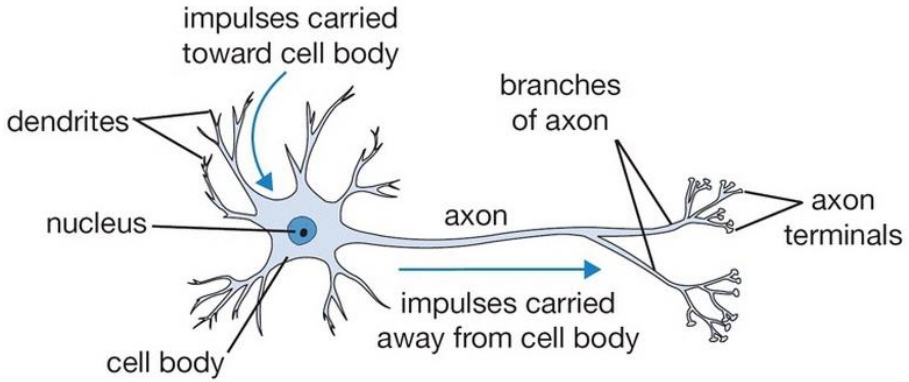
제21회 대학생 프로그래밍 경시대회 수상자 명단

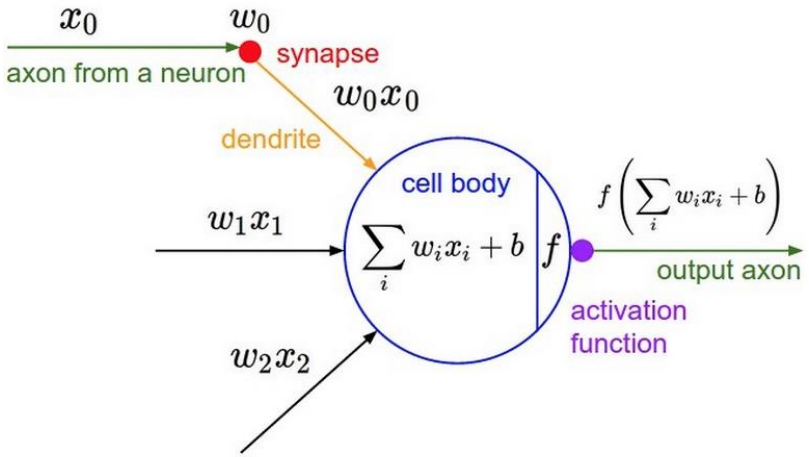
구 분	수상 팀명(대학교)		팀 원
대 상(대통령상)	FSM(서울대학교)		윤교준, 이민제, 김세빈
금 상(국무총리상)	DO Solve(KAIST)		김석표, 이온조, 임성재
은 상 (과학기술정보통신부장관상)	BabyPenguin(KAIST)		최은수, 이종영, 안지민
	The Elders(서울대학교)		김종범, 최석환, 김채원
	CSI(서울대학교)		이민준, 김창동, 박선재
동 상 (한국지능정보사회진흥원장상 /한국정보과학회장상)	kopag(서울대학교)	NIA	고동현, 임유진, 조승한
	Lcp.jpg(KAIST)	정보과학회	박수빈, 이채준, 최지원
	LongestPathToWF(숭실대학교)	NIA	안용현, 이성서, 오주원
	clopa(KAIST)	NIA	김태영, 신동원, 김 건

KENTECH
Korea Institute of Energy Technology

Week 05a – Brief Overview of Neural Networks

Neuron: Biological Inspiration for Computation

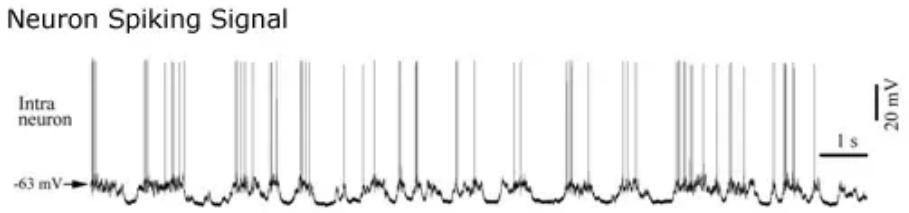




(Biological) neuron: computational building block for the brain (20W)

→ Analog signals: smooth and continuous

→ Human brain: ~100-1,000 trillion synapses,



(Artificial) neuron: computational building block for the neural network (20kW)

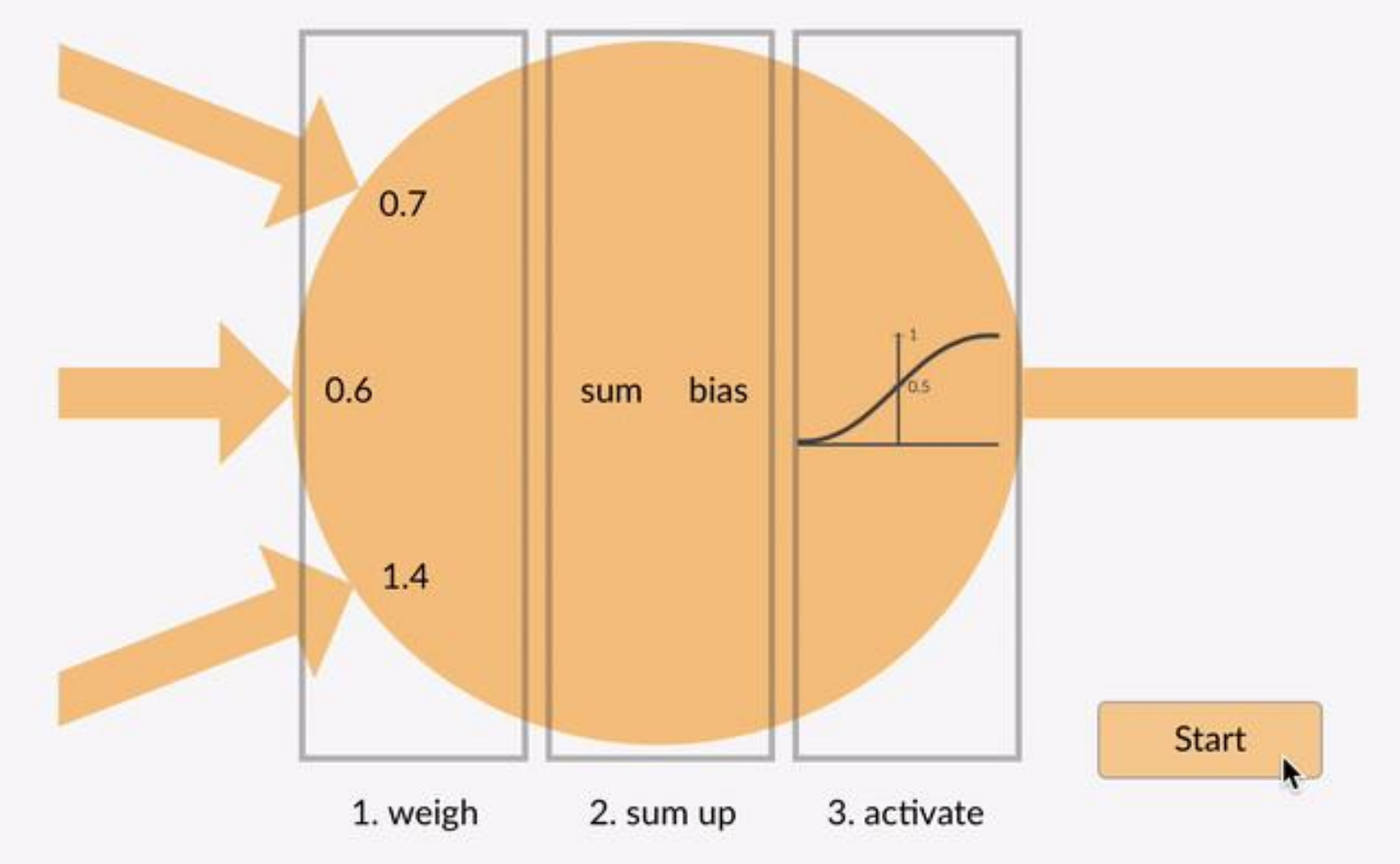
→ Digital signals: discrete and discontinuous

→ Neural network: ~1-10 billion synapses,

Compared to neural networks, the human brain has ×**10k computational power**, and consumes only **0.1% of the power**.

[1] https://www.datacamp.com/community/tutorials/deep-learning-python

Perceptron: Forward Pass



Output of activation: $f(input \times weight + bias)$

[1] https://medium.com/@b.terryjack/introduction-to-deep-learning-feed-forward-neural-networks-ffnns-a-k-a-c688d83a309d

Activation Functions

Why does it require activation?

→ Nonlinearity ↑, complexity ↑ to represent high dimensional information

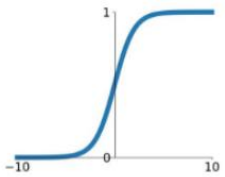
Properties of activation functions

→ Differentiable (for backpropagation)

→ Monotonic (one-to-one correspondence for input & output)

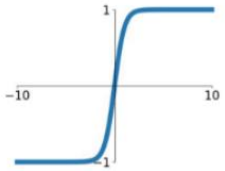
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



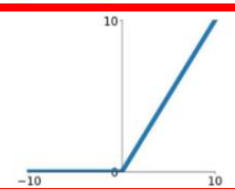
tanh

$$\tanh(x)$$



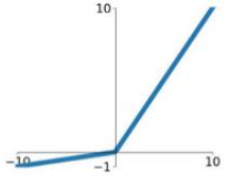
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

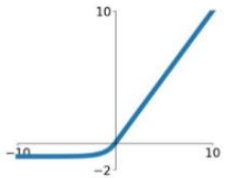


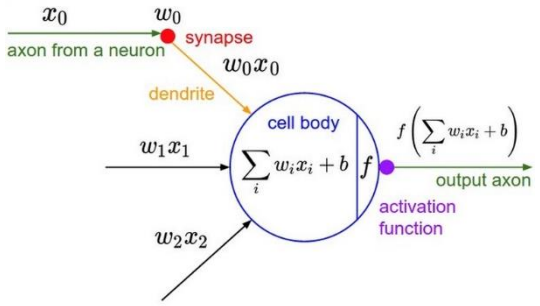
Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





For more details, please check
*vanishing gradient problem

Experiment: Perceptron Algorithm

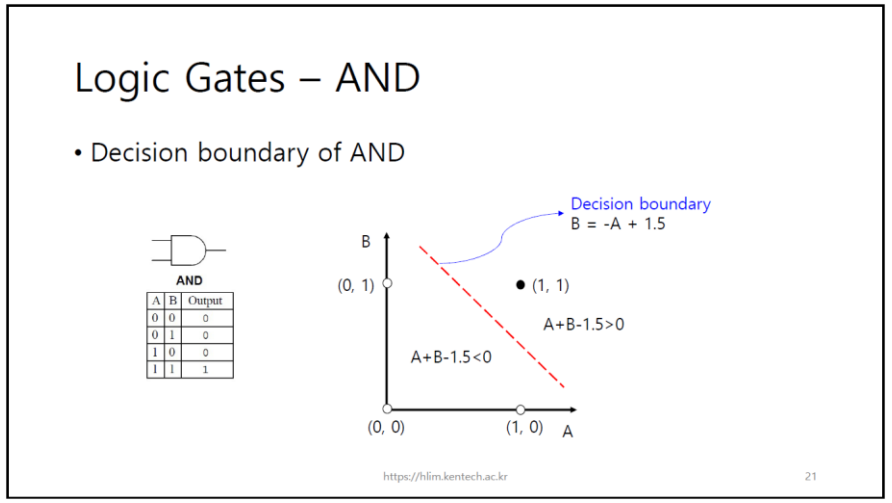
Algorithm: Perceptron Learning Algorithm

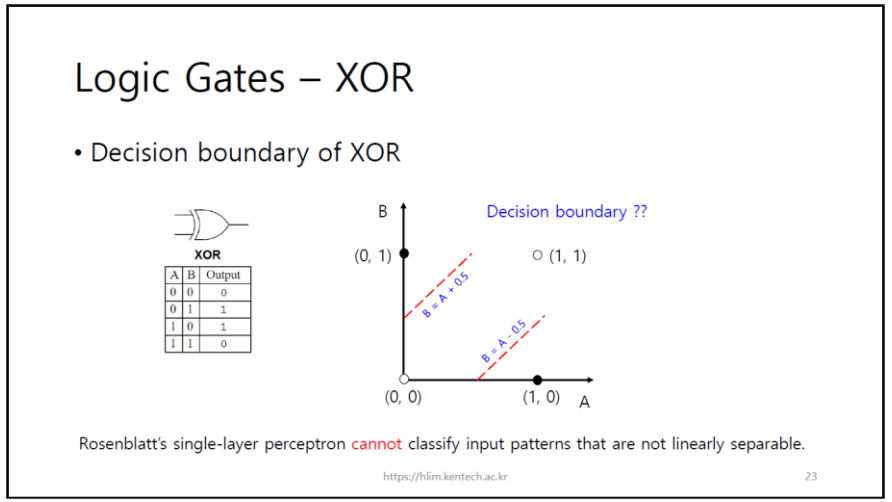
$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize **w** randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

(should be in upper)

(should be in lower)

Try to implement OR, AND, NAND, XOR gates with Perceptron and `MLPRegressor` in `sklearn`!





Discussions: Multi-Layer Perceptron

1. **Install libraries:** please type the below commands into your terminal.

*When installing the library, please do **not** enter **multiple** commands for each team at the same time!

*Run only one student per team!

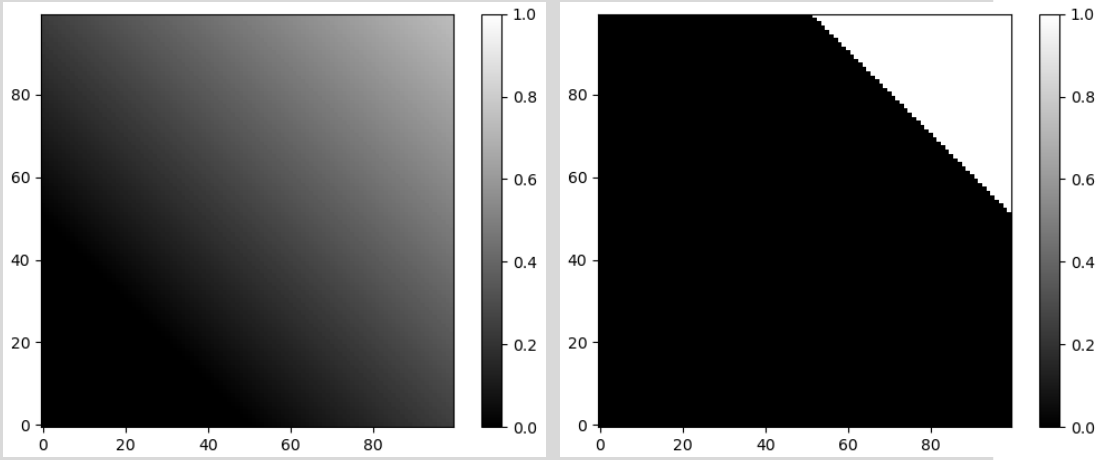
```
$ python -V                                                    // version check for Python
Python 2.7.17
$ sudo ln -sf /usr/bin/python3 /usr/bin/python                 // make a symbolic link for python3
$ python -V
python 3.6.9
$ pip install scikit-learn                                     // install python package
$ python -c "import scipy; print(scipy.__version__)"           // version check for scipy pkg
0.19.1
$ pip install scipy==1.5.4                                     // upgrade scipy pkg
$ python -c "import scipy; print(scipy.__version__)"
1.5.4
```

Discussions: Multi-Layer Perceptron

2.1. Please copy the below python code and run it.

```
from sklearn.neural_network import MLPRegressor
import numpy as np
import matplotlib.pyplot as plt
import pdb

X = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 0, 0, 1]

regr = MLPRegressor(hidden_layer_sizes=(1), activation='identity', solver='lbfgs')
model = regr.fit(X, y)

res = 100
output = [None] * res

for i in range(res):
    output[i] = [None] * res
    for j in range(res):
        x = np.array([i/res, j/res]).reshape(1, -1)
        output[i][j] = model.predict(x)[0]

output = np.array(output)          # output = np.array(output).round()
pdb.set_trace()

plt.close('all'); plt.imshow(output, vmin=0, vmax=1, cmap='gray'); plt.gca().invert_yaxis(); plt.colorbar(); plt.ion(); plt.show();
plt.savefig('output.png');
```

Reference: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
Code: https://view.kentech.ac.kr/lecture/2022s/supp → Codes for MLP

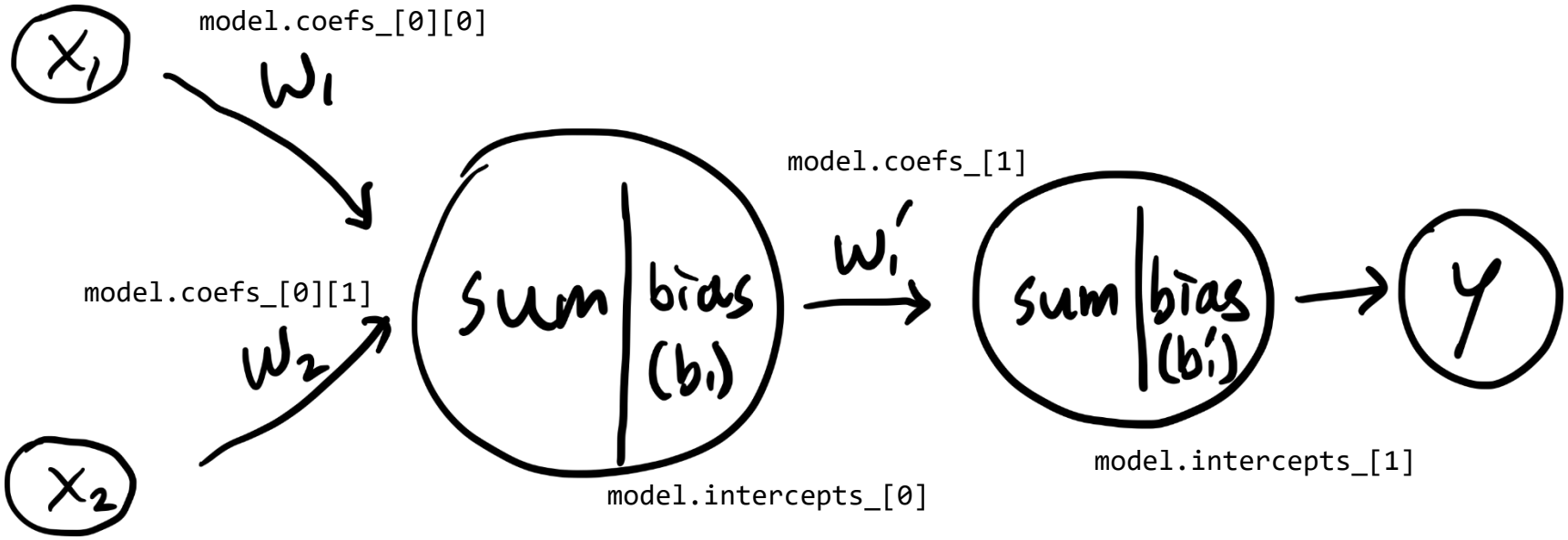


Q1. Which gate is this model designed to learn?
Q2. Please draw the decision boundaries.
Q3. Please discuss each line using comments.

Discussions: Multi-Layer Perceptron

2.2. Please draw the structure (with the weight and bias) of the perceptron.

<Hints>
- Please go to the scikit-learn library.
- Please find the attributes to access the values of the weight and bias.
- How are they designed?



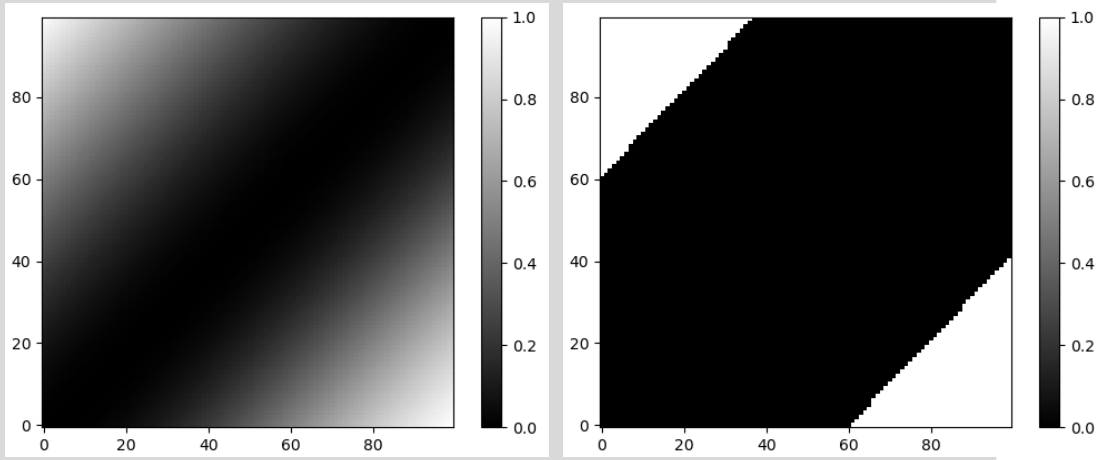
model.predict([1,1])
= (model.coefs_[0][0] * 1 + model.coefs_[0][1] * 1 + model.intercepts_[0]) * model.coefs_[1] + model.intercepts_[1]

Discussions: Multi-Layer Perceptron

3.1. Please copy the below python code and run it.

```
from sklearn.neural_network import MLPRegressor
import numpy as np
import matplotlib.pyplot as plt
import pdb

X = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 1, 1, 0]

regr = MLPRegressor(hidden_layer_sizes=(2), activation='tanh', solver='lbfgs')
model = regr.fit(X, y)

res = 100
output = [None] * res

for i in range(res):
    output[i] = [None] * res
    for j in range(res):
        x = np.array([i/res, j/res]).reshape(1, -1)
        output[i][j] = model.predict(x)[0]

output = np.array(output)          # output = np.array(output).round()
pdb.set_trace()

plt.close('all'); plt.imshow(output, vmin=0, vmax=1, cmap='gray'); plt.gca().invert_yaxis(); plt.colorbar(); plt.ion(); plt.show();
plt.savefig('output.png');
```

Reference: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
Code: https://view.kentech.ac.kr/lecture/2022s/supp → Codes for MLP



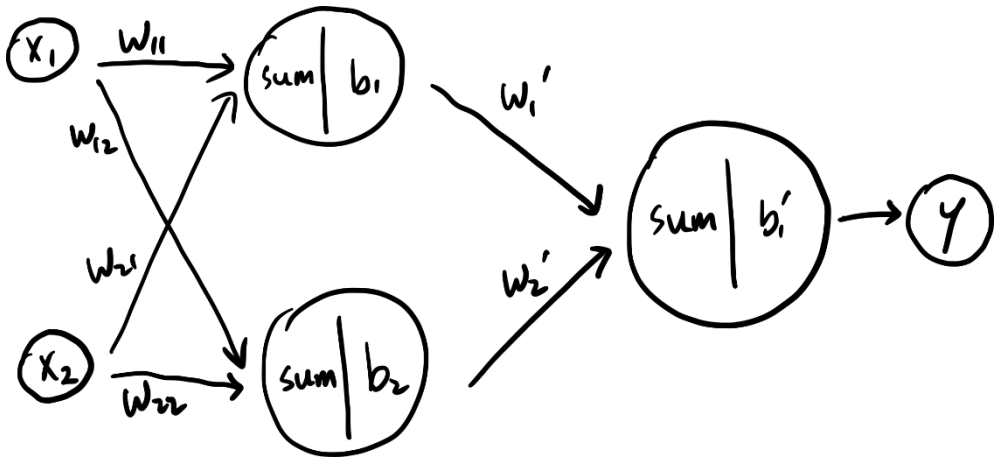
Q1. Which gate is this model designed to learn?
Q2. Please draw the decision boundaries.
Q3. Please discuss each line using comments.

Discussions: Multi-Layer Perceptron

3.2. Please draw the structure (with the weight and bias) of the perceptron.

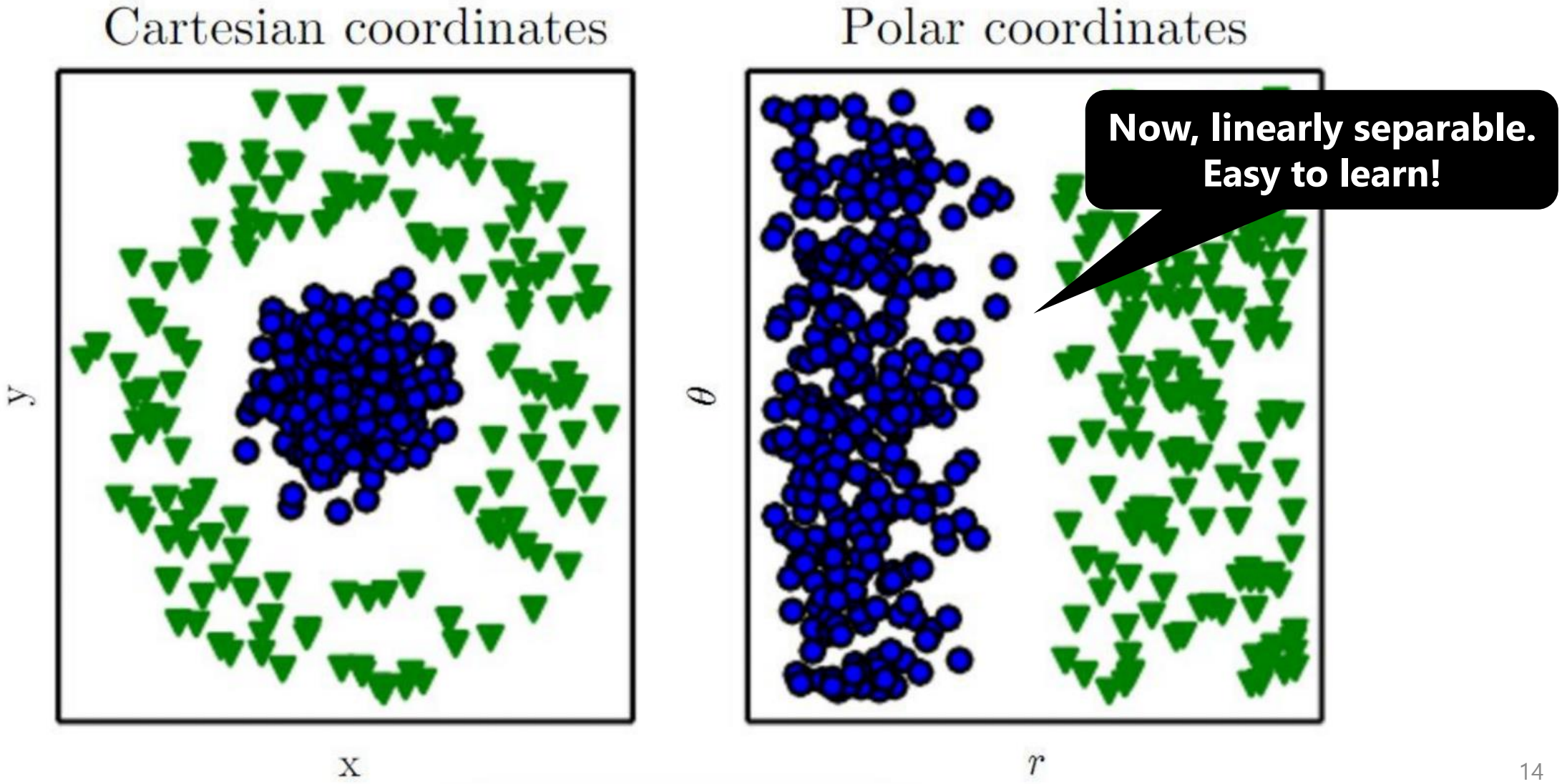
<Hints>
- Please go to the scikit-learn library.
- Please find the attributes to access the values of the weight and bias.
- How are they designed?

```
model.predict([1,1])
= model.coefs_[1][0]*x1_ + model.coefs_[1][1]*x2_ + model.intercepts_[1]
, where
x1_ = (model.coefs_[0][0][0]*1 + model.coefs_[0][1][0]*1 + model.intercepts_[0][0])
x2_ = (model.coefs_[0][0][1]*1 + model.coefs_[0][1][1]*1 + model.intercepts_[0][1])
```




```
w11 = coefs_[0][0][0]
w21 = coefs_[0][1][0]
w12 = coefs_[0][0][1]
w22 = coefs_[0][1][1]

b1 = intercepts_[0][0]
b2 = intercepts_[0][1]
w1' = coefs_[1][0]
w2' = coefs_[1][1]
b1' = intercepts_[1]
```

Representation Matters

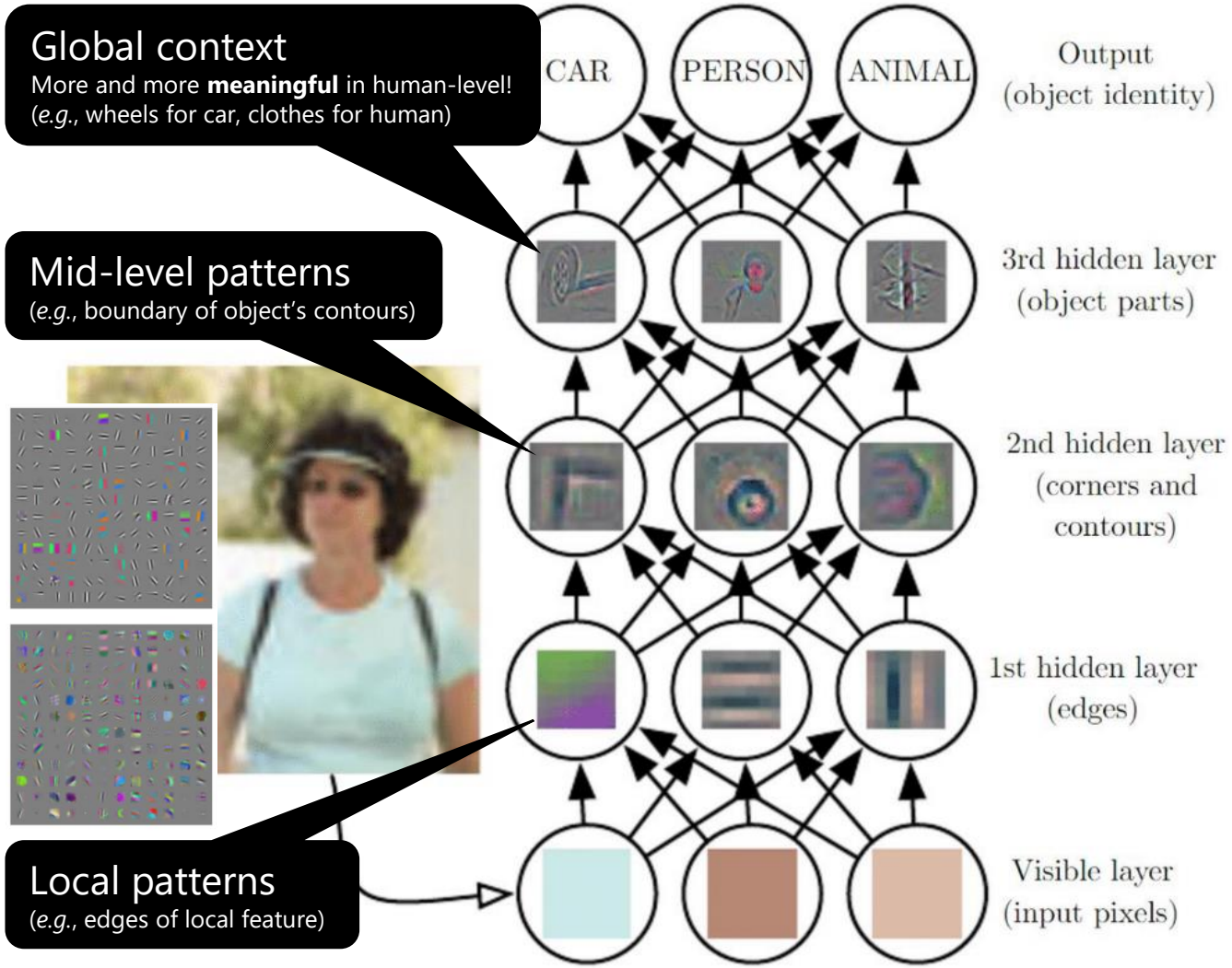


Deep Learning is Representation Learning

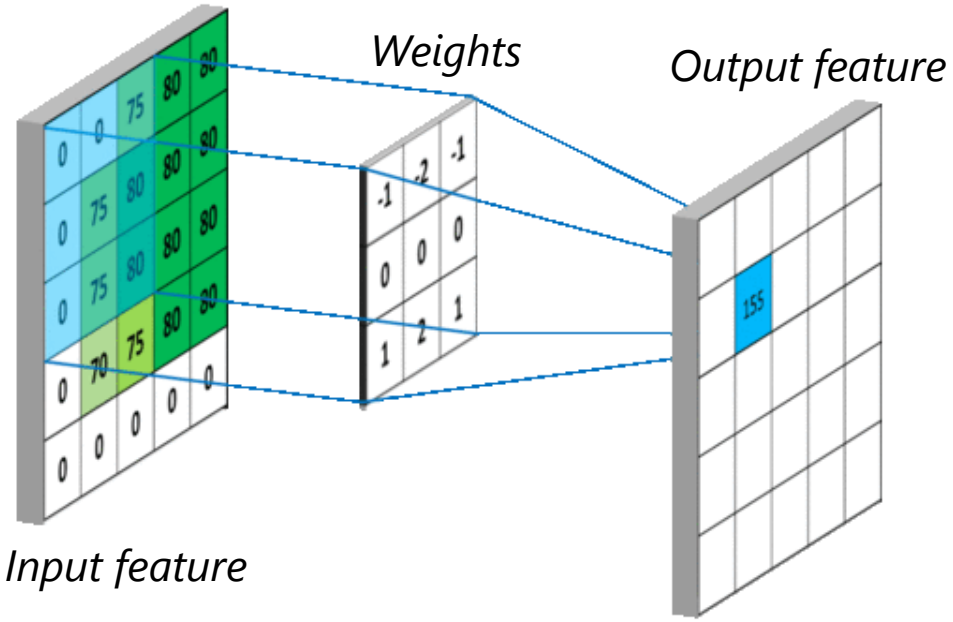
Global context
More and more **meaningful** in human-level!
(*e.g.*, wheels for car, clothes for human)

Mid-level patterns
(*e.g.*, boundary of object's contours)

Local patterns
(*e.g.*, edges of local feature)

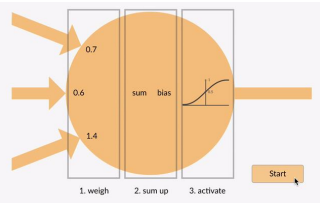


Convolutional Neural Network (CNN)



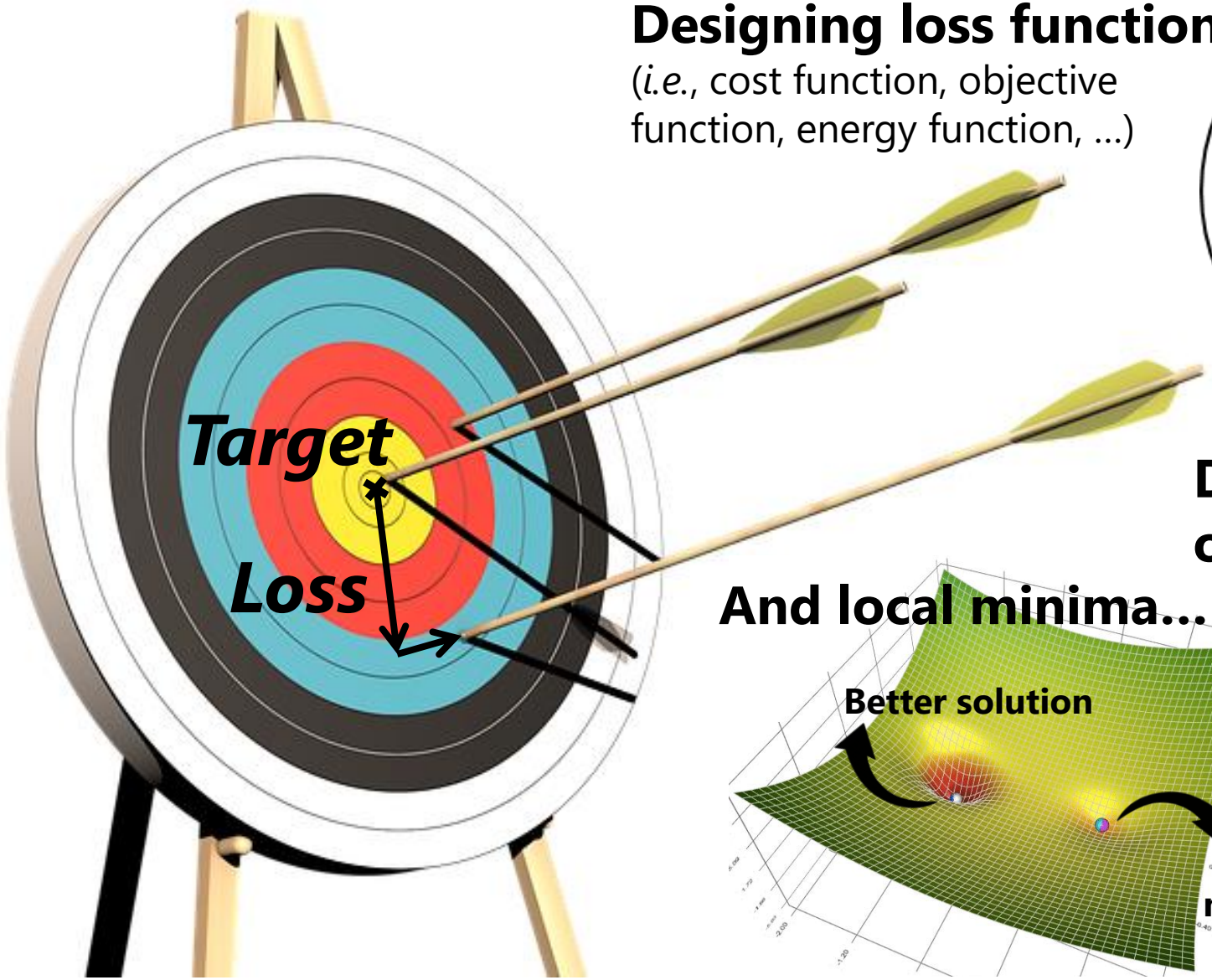
→ **Convolution** works on **spatial** features!

cf., **perceptron** works on **vector**:

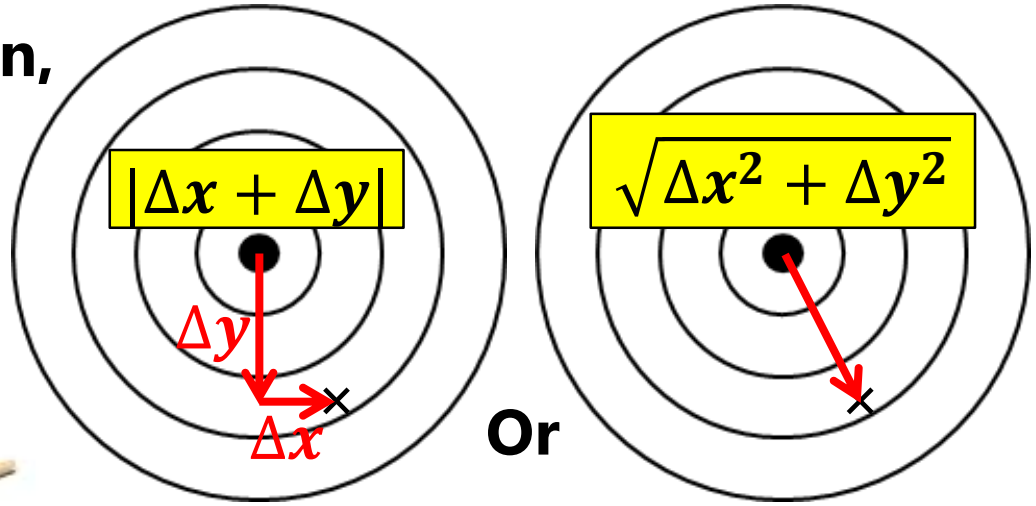


[1] Goodfellow, et al., "Deep Learning", 2017

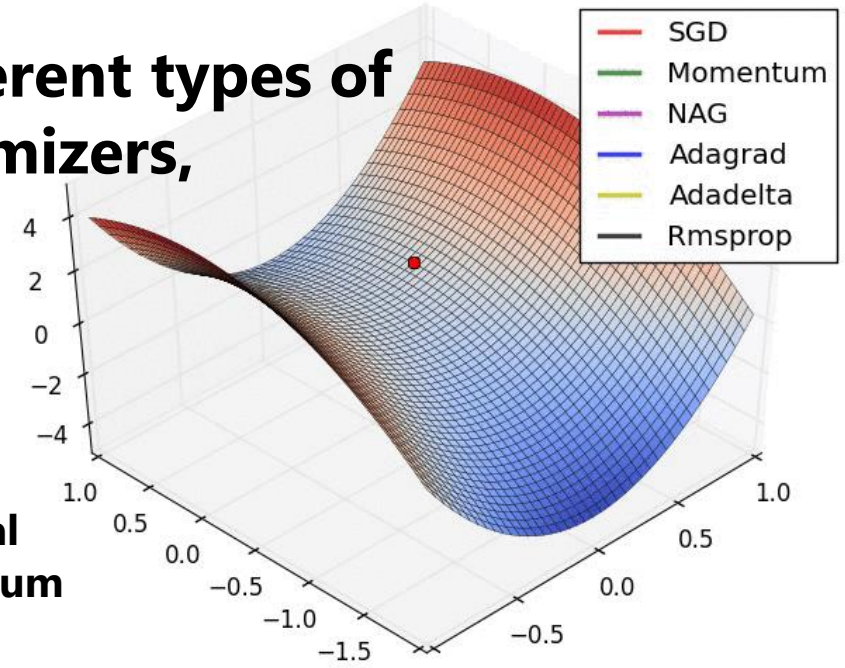
How to Optimize Deep Neural Networks?



Designing loss function,
(*i.e.*, cost function, objective function, energy function, …)



Different types of optimizers,



And local minima…

Other Drawbacks (including but not limited to)

- Works for specifically defined tasks (**weak AI**)
- Requires large amount of **data**
- Requires human **annotation** for real world data (supervised learning)
- **Domain** issues (virtual ↔ real, daytime ↔ night)
- **Manually** select neural architecture
- Performance is varied over different **loss** functions
- **Hyperparameter** tuning
  - Learning rate, loss function
  - Batch size, number of iteration, number of kernels
  - Optimizer, momentum

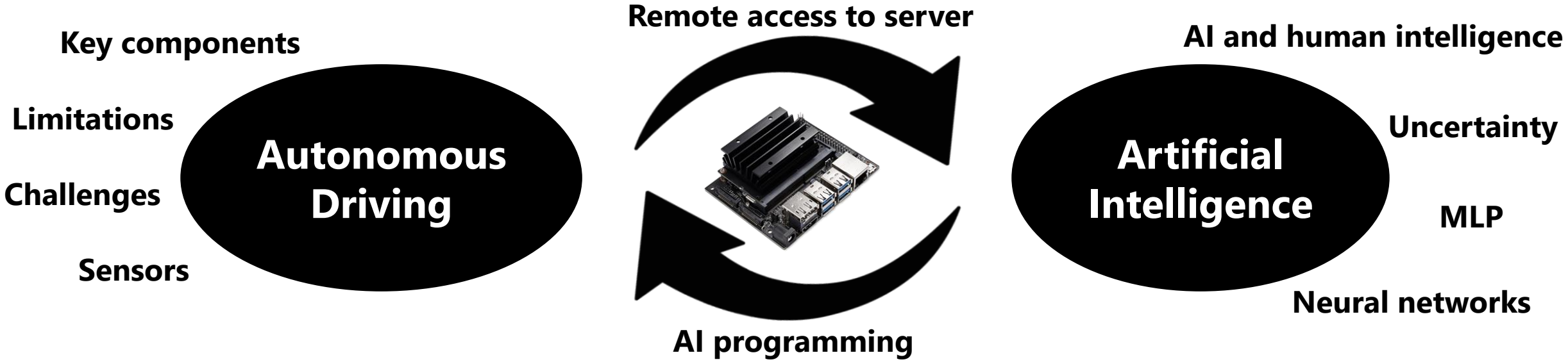
GANcraft (*ICCV'21*)

Making **virtual** worlds more **realistic**!

Useful Deep Learning Terms (including but not limited to)

- Deep learning = **Deep neural network (DNN)**
- Deep learning ⊂ Machine learning ⊂ AI
- **MLP**: Multi-layer perceptron
- **CNN**: Convolutional neural network
- GAN, GNN, RNN, LSTM, autoencoder
- Spiking neural network
- **Neural network operations**:
  - Convolution, pooling, activation function
  - Feed forward, backpropagation
  - Batch normalization, KL divergence
  - Data augmentation, regularization
- **AlexNet, Inception, VGG, ResNet**
- Others:
  - ViT, Transformer, attention, cost volume
  - PyTorch, Caffe, TensorFlow
- **Supervised** learning, **self-supervised** learning, **unsupervised** learning, **reinforcement** Learning
- Few-shot (one-shot) learning, **adversarial** learning, **domain adaptation**, meta learning, active learning, **multimodal** learning, **contrastive** learning
- **Visual learning tasks:**
  - Image classification, object detection
  - Semantic/instance/panoptic/video segmentation
  - Optical flow, depth, neural rendering
  - Stereo matching, SfM, MVS
  - Image enhancement, super resolution
  - Stylization, image-to-image, VQA, VLN

Take Home Messages



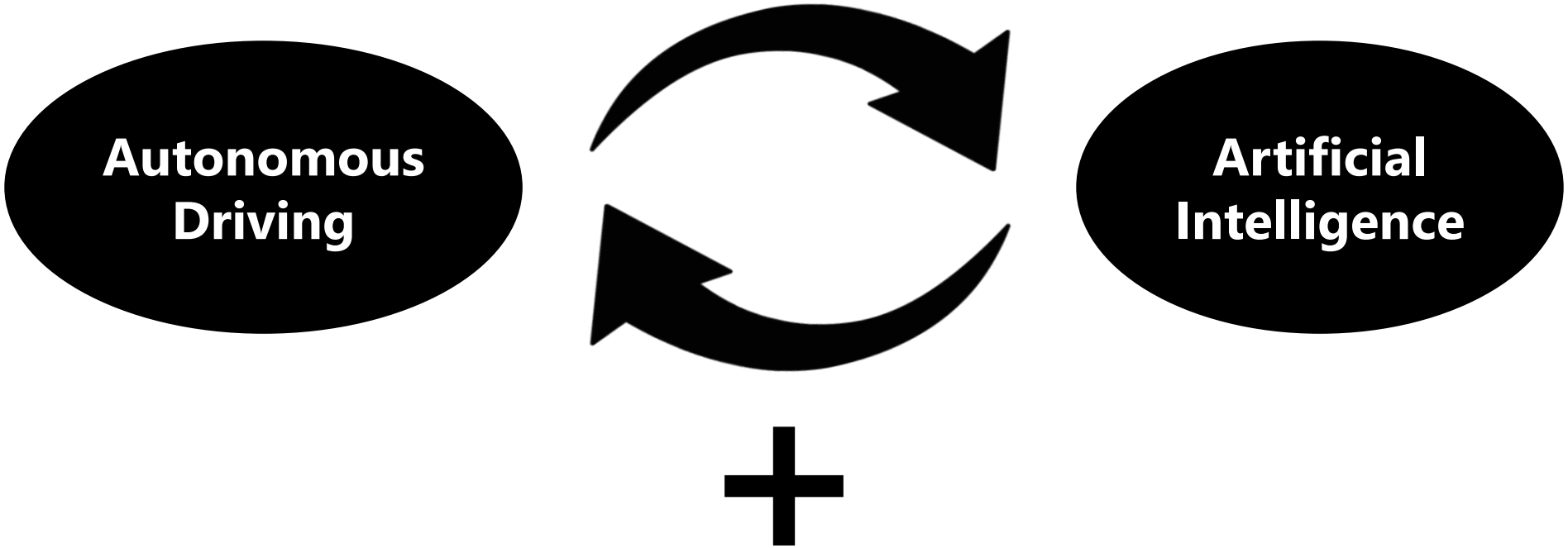
AI researchers want to build a **real human**...

Everyday, we meet **new exciting** works.

Please check the below link!

https://arxiv.org/list/cs.CV/recent

Next Contents



- Basic camera model
- Image Processing Puzzle

- Computer vision tasks
  - Image classification
  - Object detection
  - Segmentation, etc.