

# Machine Learning for Physicists

OnlineTutorials

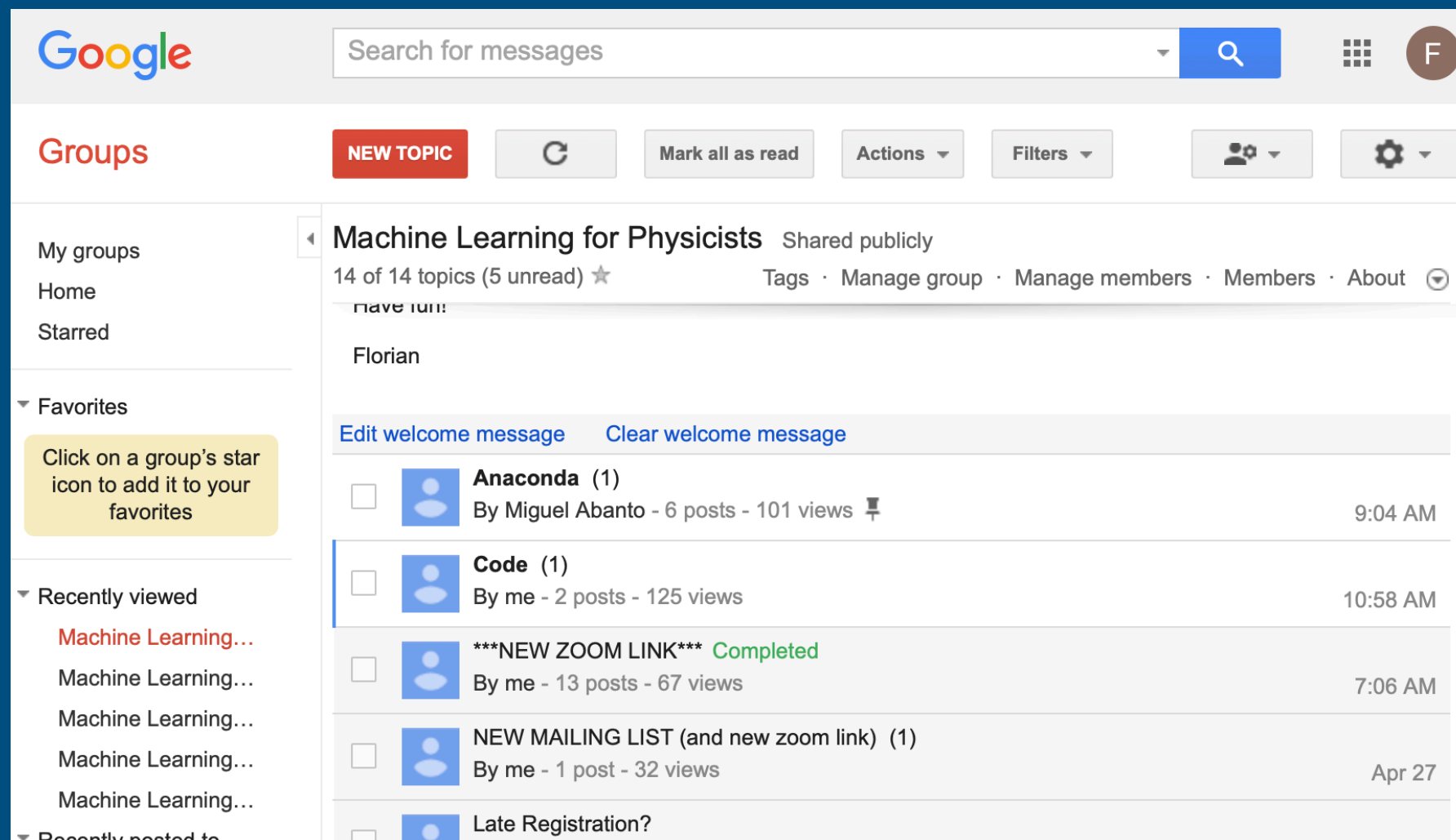
Florian Marquardt  
University of Erlangen-Nuremberg  
& Max Planck Institute for the  
Science of Light

[Florian.Marquardt@fau.de](mailto:Florian.Marquardt@fau.de)

<http://machine-learning-for-physicists.org>

(Image generated by a net with 20 hidden layers)

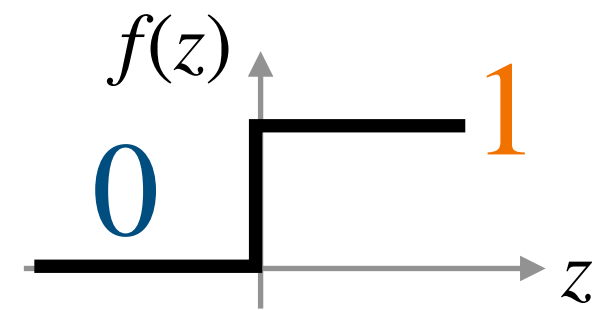
# Please use the forum for discussions & useful code & nice examples



## Machine Learning for Physicists

# Quiz: single-layer neural networks

$f(z)$  is the step function:

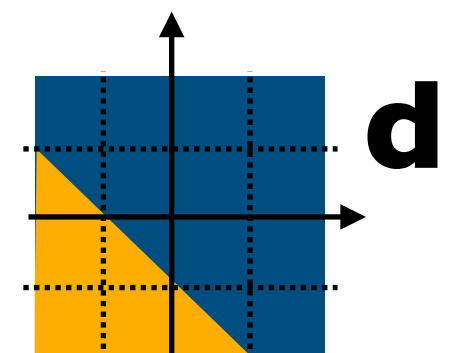
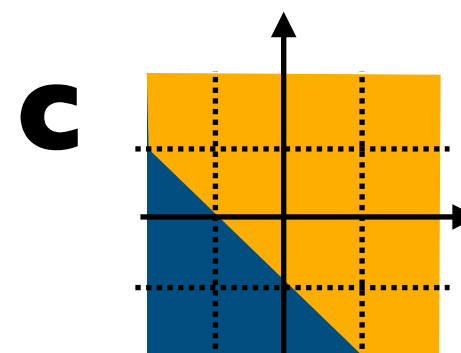
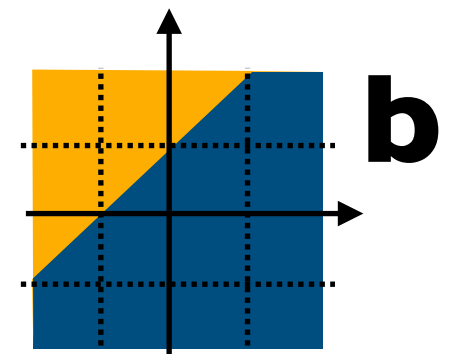
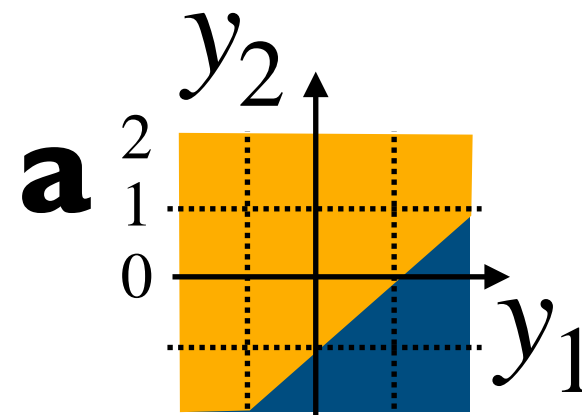
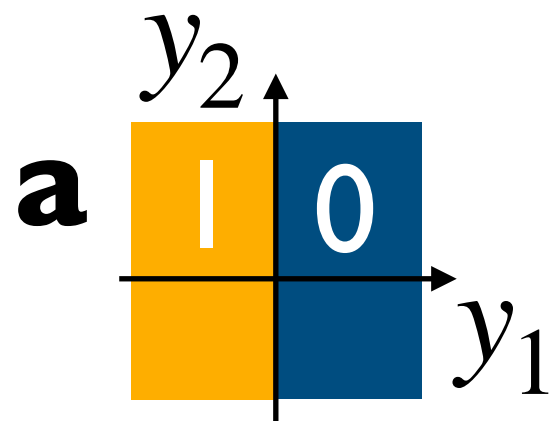


1

2

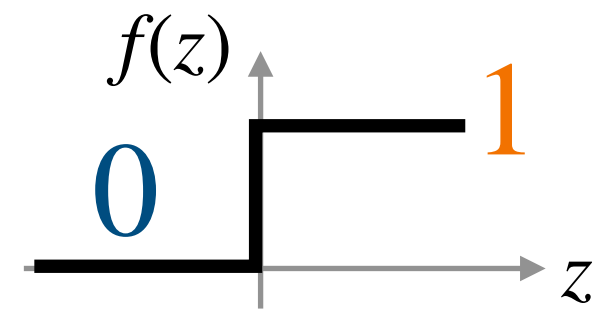
Which of these outputs  
is generated by  
 $y^{\text{out}} = f(-y_2)$  ?

...and for  
 $y^{\text{out}} = f(1 - y_1 + y_2)$  ?





$f(z)$  is the step function:

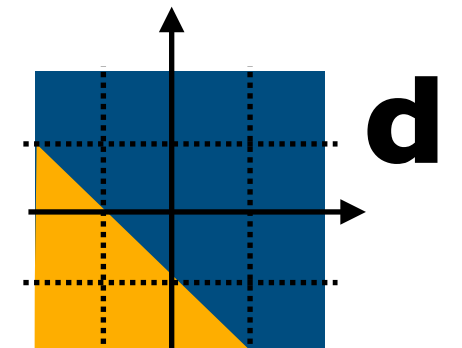
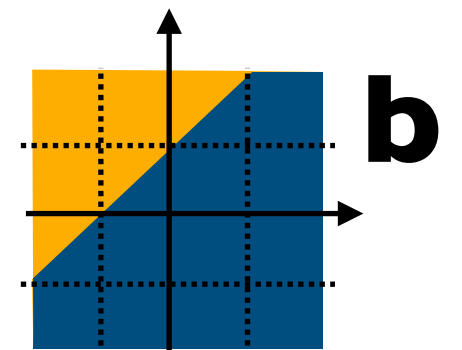
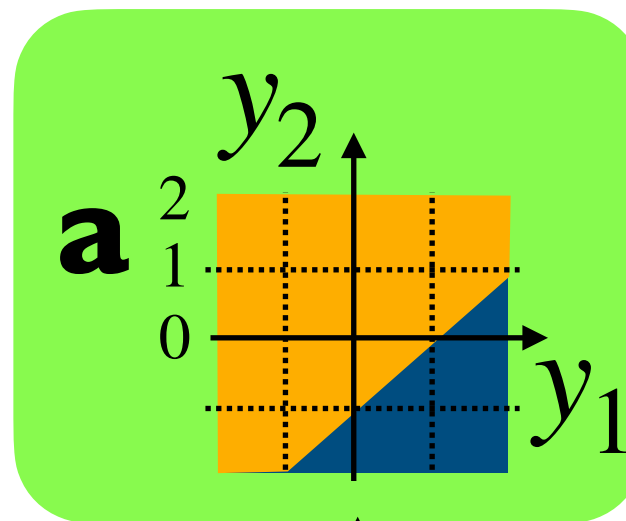
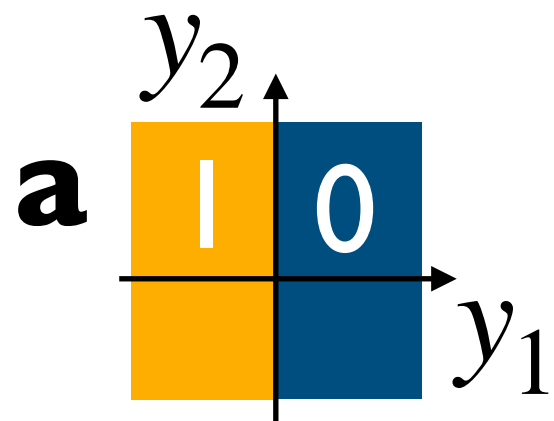


1

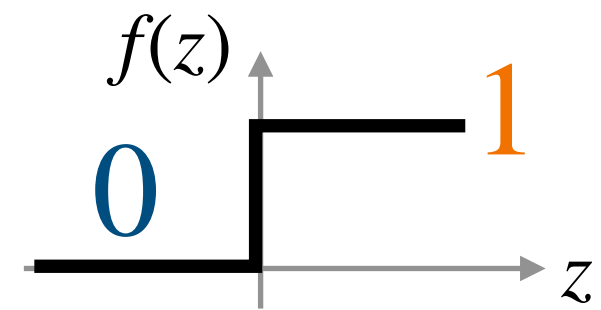
2

Which of these outputs  
is generated by  
 $y^{\text{out}} = f(-y_2)$  ?

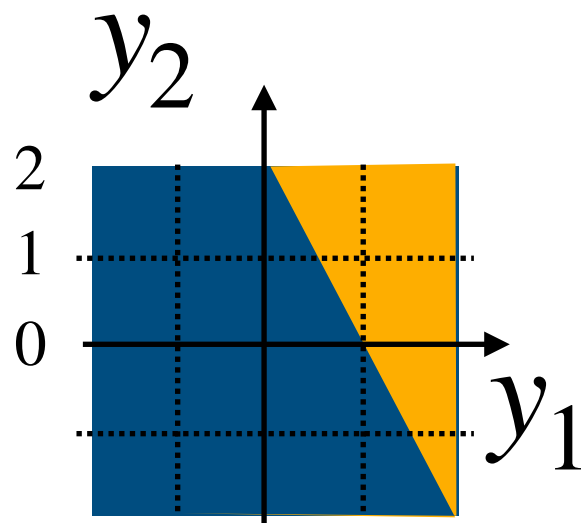
...and for  
 $y^{\text{out}} = f(1 - y_1 + y_2)$  ?



$f(z)$  is the step function:



**1** How can we obtain this output?



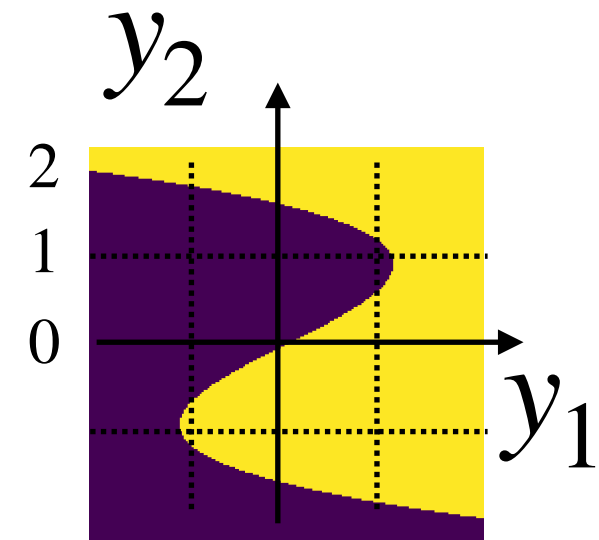
**a**  $y^{\text{out}} = f(y_1 + y_2 - 2)$

**b**  $y^{\text{out}} = f(2y_1 + y_2 - 2)$

**c**  $y^{\text{out}} = f(2 - 2y_1 + y_2)$

**d**  $y^{\text{out}} = f(y_1 - 2y_2)$

**2** ...and this one?



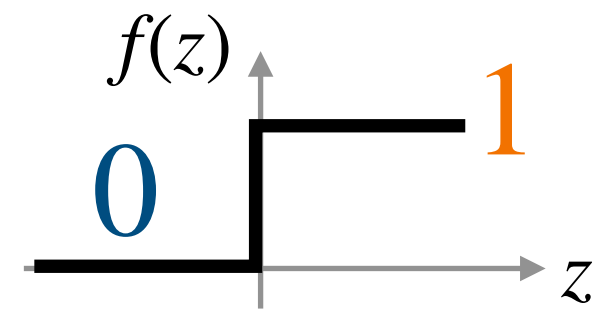
**a**  $y^{\text{out}} = f(-2y_2 + y_1)$

**b**  $y^{\text{out}} = f(y_2^3 - 2y_2 - y_1)$

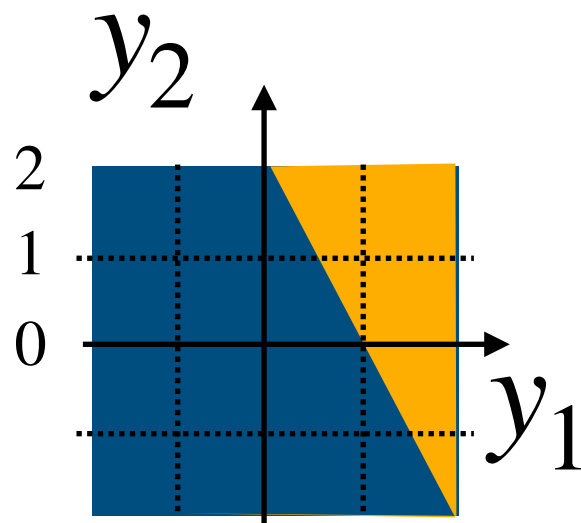
**c**  $y^{\text{out}} = f(y_2^3 - 2y_2 + y_1)$

**d**  $y^{\text{out}} = f(-y_2^3 - 2y_2 + y_1)$

$f(z)$  is the step function:



**1** How can we obtain this output?



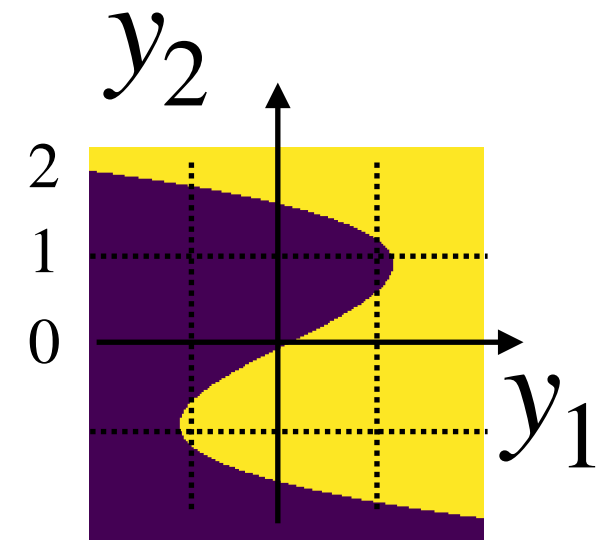
**a**  $y^{\text{out}} = f(y_1 + y_2 - 2)$

**b**  $y^{\text{out}} = f(2y_1 + y_2 - 2)$

**c**  $y^{\text{out}} = f(2 - 2y_1 + y_2)$

**d**  $y^{\text{out}} = f(y_1 - 2y_2)$

**2** ...and this one?



**a**  $y^{\text{out}} = f(-2y_2 + y_1)$

**b**  $y^{\text{out}} = f(y_2^3 - 2y_2 - y_1)$

**c**  $y^{\text{out}} = f(y_2^3 - 2y_2 + y_1)$

**d**  $y^{\text{out}} = f(-y_2^3 - 2y_2 + y_1)$

# Quiz: python

Machine Learning for Physicists



1

```
for j in range(2):  
    print(j)
```

**a**     0       1

**b**     1       2

**c**     0       1       2

2

```
def f(n):  
    if n<=2:  
        return(n+f(n+1))  
    else:  
        return(42)
```

$f(1)=?$

**a**     85

**b**     45

**c**     44

**d**     42

**e**     3

1

```
for j in range(2):  
    print(j)
```

<b>a</b>	0	1
----------	---	---

<b>b</b>	1	2
----------	---	---

<b>c</b>	0	1	2
----------	---	---	---

2

```
def f(n):  
    if n<=2:  
        return(n+f(n+1))  
    else:  
        return(42)
```

$f(1)=?$

<b>a</b>	85
----------	----

<b>b</b>	45
----------	----

<b>c</b>	44
----------	----

<b>d</b>	42
----------	----

<b>e</b>	3
----------	---

$f(1)=1+f(2)=1+2+f(3)=1+2+42$

```
a=np.array([[11,22],[33,44]])
```

**1** `a[0,1]=?`

**a** 11      **c** 33

**b** 22      **d** 44

**2** `a[1]=?`

**a** 22

**b** [11,22]

**c** [33,44]

**d** [22,44]

**3** `a[:,1]=?`

**a** [22,44]

**b** [11,33]

**c** [33,44]

**d** [11,22]

```
a=np.array([[11,22],[33,44]])
```

1  $a[0,1]=?$

**a** 11      **c** 33

**b** 22      **d** 44

2  $a[1]=?$

**a** 22

**b** [11,22]

**c** [33,44]

**d** [22,44]

3  $a[:,1]=?$

**a** [22,44]

**b** [11,33]

**c** [33,44]

**d** [11,22]

$$\begin{pmatrix} a_{0,0} & a_{0,1} \\ 11 & 22 \\ a_{1,0} & a_{1,1} \\ 33 & 44 \end{pmatrix}$$

```
a=np.array([[11,22],[33,44]])
```

**1** `a.flatten()`=?

**a** `[11,33,22,44]`

**b** `[11,22,33,44]`

**c** `[[11],[33],[22],[44]]`

**2**

```
j0=np.array([1,0,1])
```

```
j1=np.array([0,0,1])
```

```
a[j0,j1]=?
```

**a** `[11,33,44]`    **c** `[22,11,44]`

**b** error        **d** `[33,11,44]`

**3**

```
b=np.array([19,50])
```

```
a[:,0]=b[:]
```

```
a=?
```

**a** `[[11,22],[19,50]]`

**b** `[[19,50],[33,44]]`

**c** `[[19,22],[50,44]]`

**d** `[[11,19],[33,50]]`

```
a=np.array([[11,22],[33,44]])
```

1 `a.flatten()`=?

**a** `[11,33,22,44]`

**b** `[11,22,33,44]`

**c** `[[11],[33],[22],[44]]`

2

```
j0=np.array([1,0,1])
```

```
j1=np.array([0,0,1])
```

```
a[j0,j1]=?
```

**a** `[11,33,44]` **c** `[22,11,44]`

**b** error

**d** `[33,11,44]`

3

```
b=np.array([19,50])
```

```
a[:,0]=b[:]
```

```
a=?
```

**a** `[[11,22],[19,50]]`

**b** `[[19,50],[33,44]]`

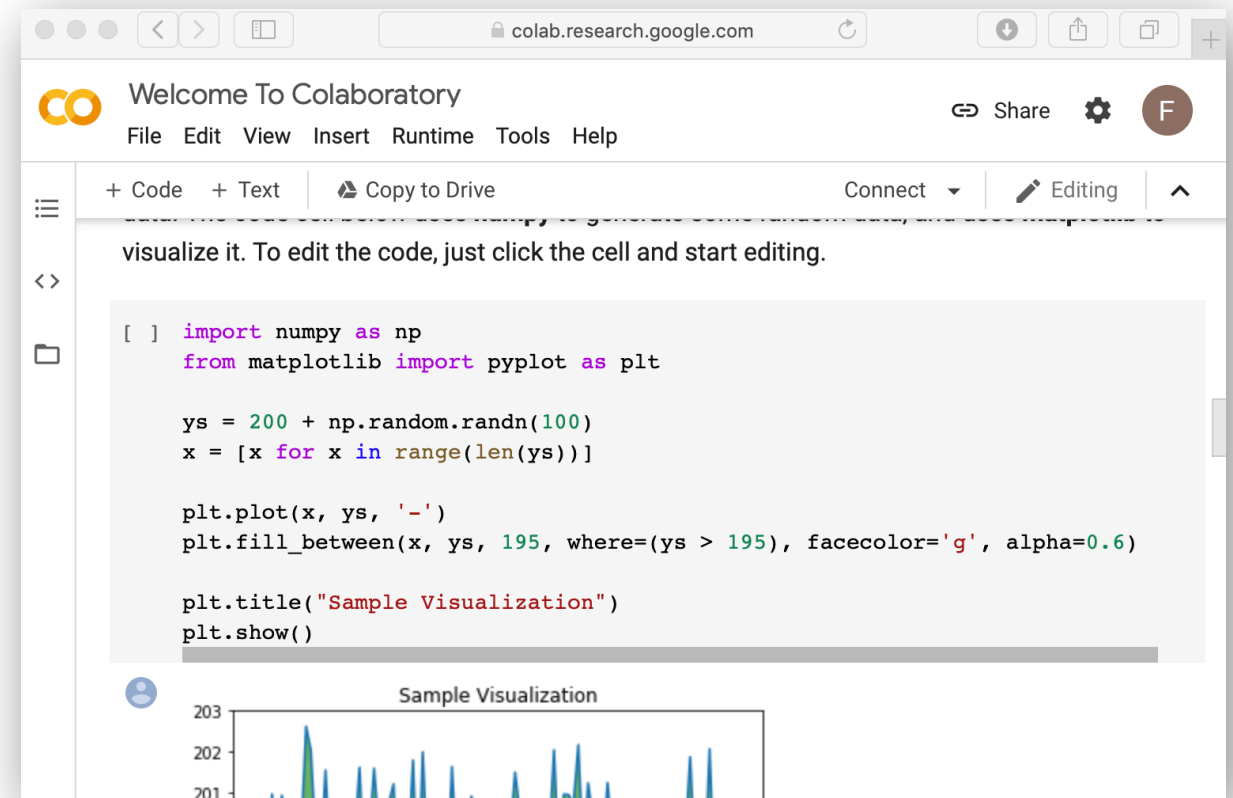
**c** `[[19,22],[50,44]]`

**d** `[[11,19],[33,50]]`

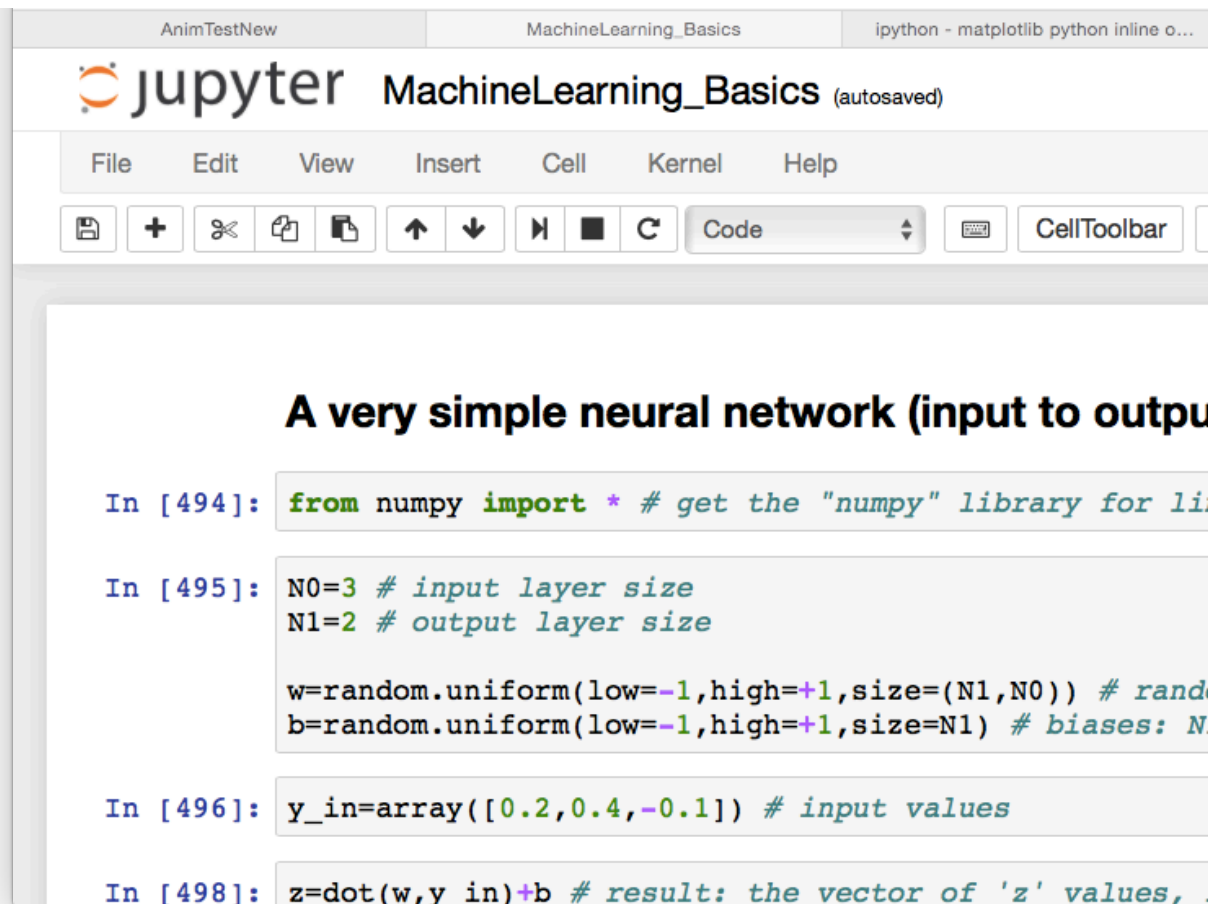


# Jupyter

# Colaboratory



# Jupyter



The screenshot shows the Jupyter web interface with a notebook titled "MachineLearning\_Basics". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for saving, adding cells, and running code. The notebook content is as follows:

```
In [494]: from numpy import * # get the "numpy" library for li

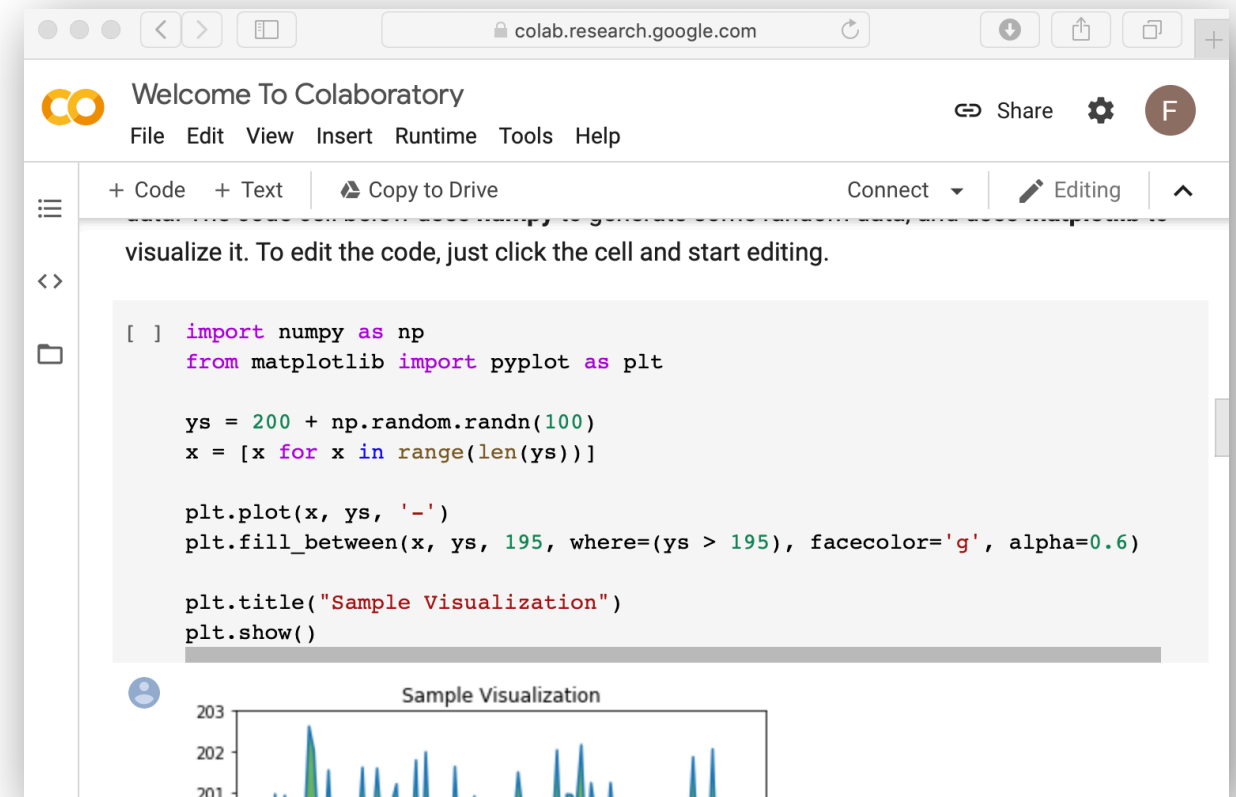
In [495]: N0=3 # input layer size
          N1=2 # output layer size

          w=random.uniform(low=-1,high=+1,size=(N1,N0)) # rand
          b=random.uniform(low=-1,high=+1,size=N1) # biases: N

In [496]: y_in=array([0.2,0.4,-0.1]) # input values

In [498]: z=dot(w,y_in)+b # result: the vector of 'z' values,
```

# Colaboratory



The screenshot shows the Google Colaboratory web interface. The notebook contains the following code:

```
[ ] import numpy as np
    from matplotlib import pyplot as plt

    ys = 200 + np.random.randn(100)
    x = [x for x in range(len(ys))]

    plt.plot(x, ys, '-')
    plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

    plt.title("Sample Visualization")
    plt.show()
```

Below the code, a plot titled "Sample Visualization" is displayed. The plot shows a line graph with blue data points and a green shaded area representing the region where the values are greater than 195. The y-axis ranges from 201 to 203.

# Practice session: Visualizing neural networks

Machine Learning for Physicists

Code (jupyter notebook): 01\_MachineLearning\_Basics\_NeuralNetworksPython.ipynb  
(or download code as pure python script)

see: website/course overview/lecture I

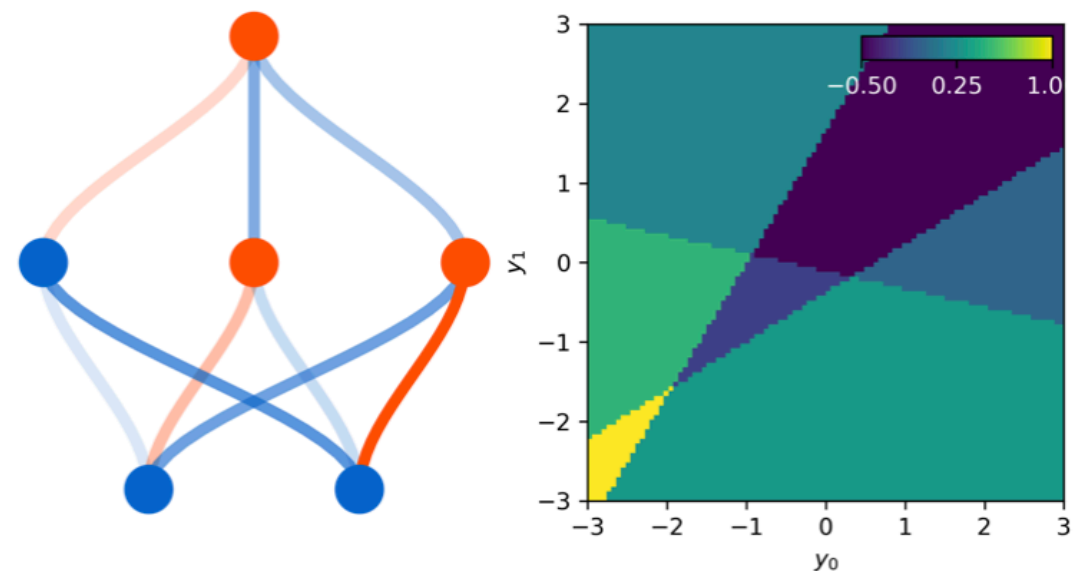
This notebook shows how to calculate the forward-pass through a neural network in pure python, and how to illustrate the results for randomly initialized deep neural networks (as shown in the lecture).

#### Notebooks for tutorials:

[Tutorial: Network Visualization](#)

[Tutorial: Curve Fitting](#)

```
0.5], # biases of 3 hi  
# biases for output neuron  
],  
activations=[ 'jump', # activation fo  
               'linear' # activation for  
],  
y0range=[-3,3],y1range=[-3,3])
```



(1) Try to construct a square!

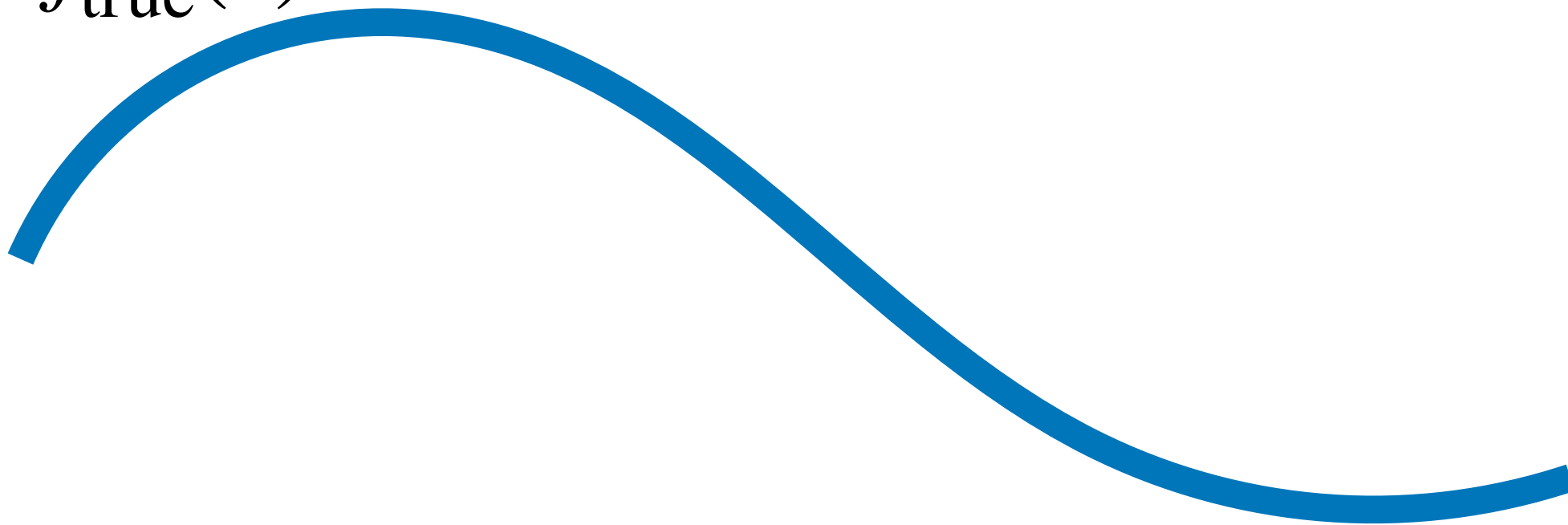
(2) What happens when you take a sigmoid hidden layer and a 'jump' (stepfunction) output layer? Which shapes can you construct?

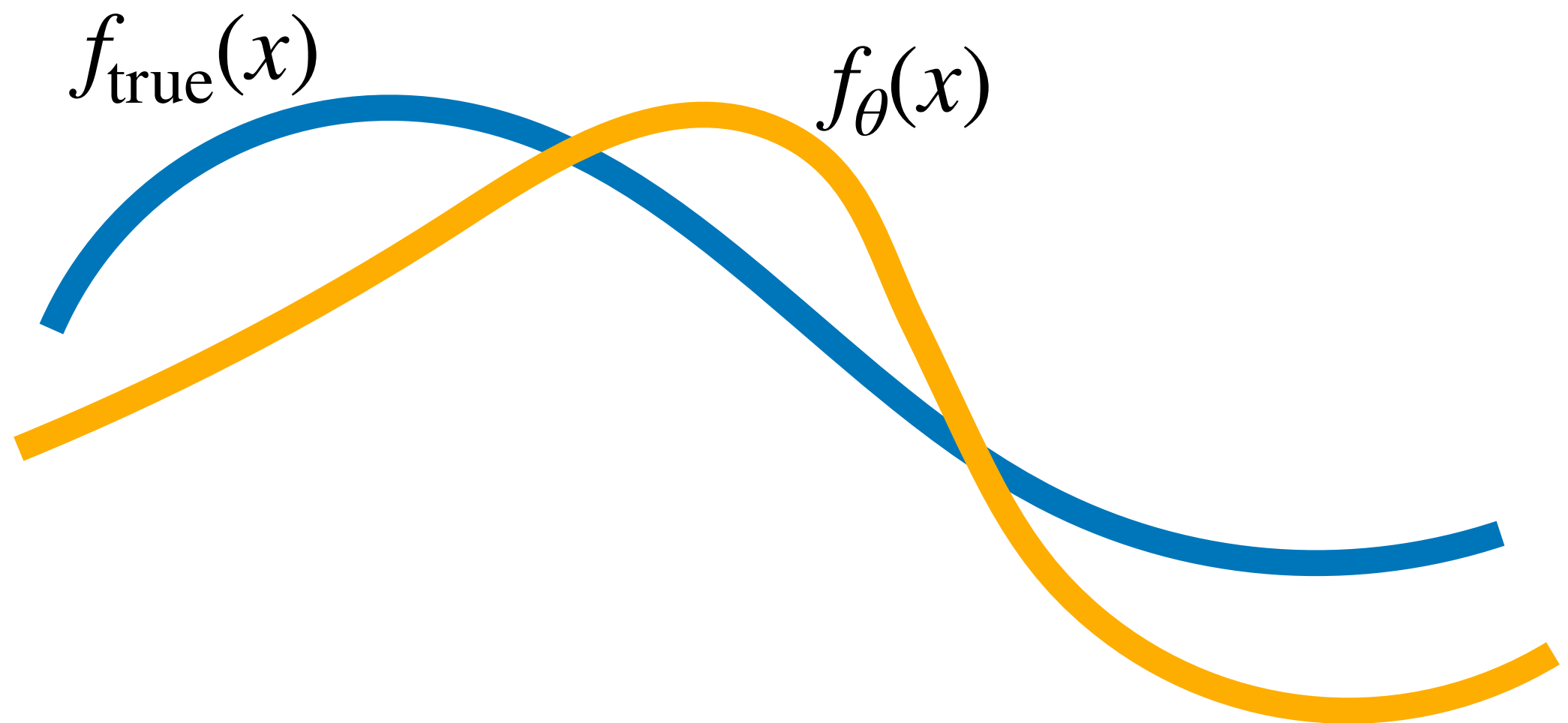
# Practice session: Nonlinear curve fitting

Machine Learning for Physicists



$f_{\text{true}}(x)$





e.g. 
$$f_{\theta}(x) = \frac{\theta_0}{(x - \theta_1)^2 + 1}$$

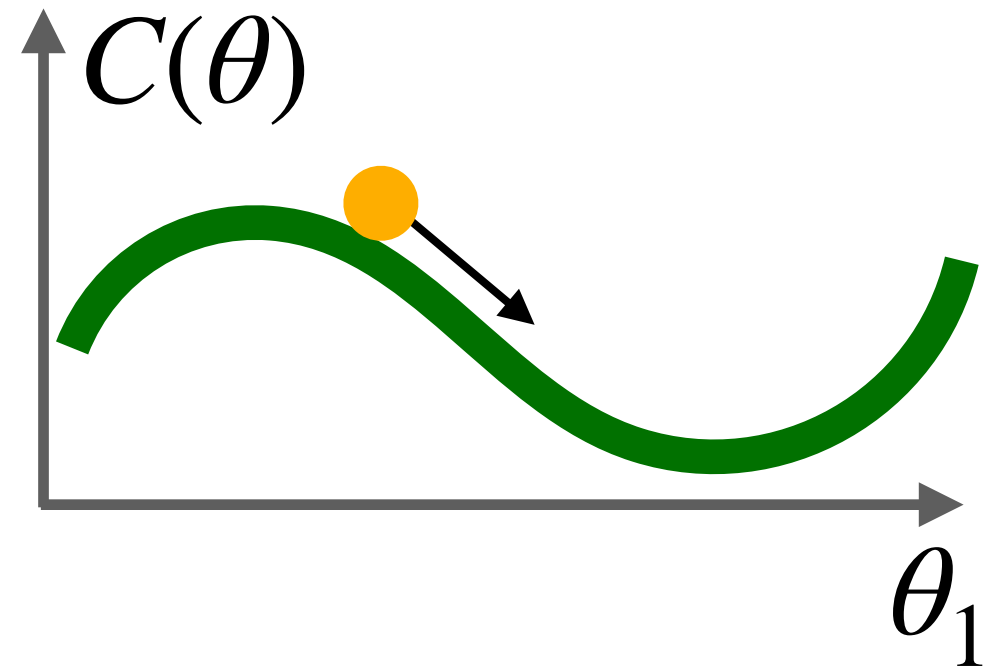
"least squares fitting":  
consider quadratic deviation

$$C(\theta) = \frac{1}{2} \left\langle (f_{\theta}(x) - f_{\text{true}}(x))^2 \right\rangle$$

(average over  $x$ )

gradient descent ("down the hill"):

$$\delta\theta_j = -\eta \frac{\partial C}{\partial \theta_j}$$



stochastic: sample  $x$

Code (jupyter notebook): 01\_MachineLearning\_Basics\_NeuralNetworksPython.ipynb  
(or download code as pure python script)

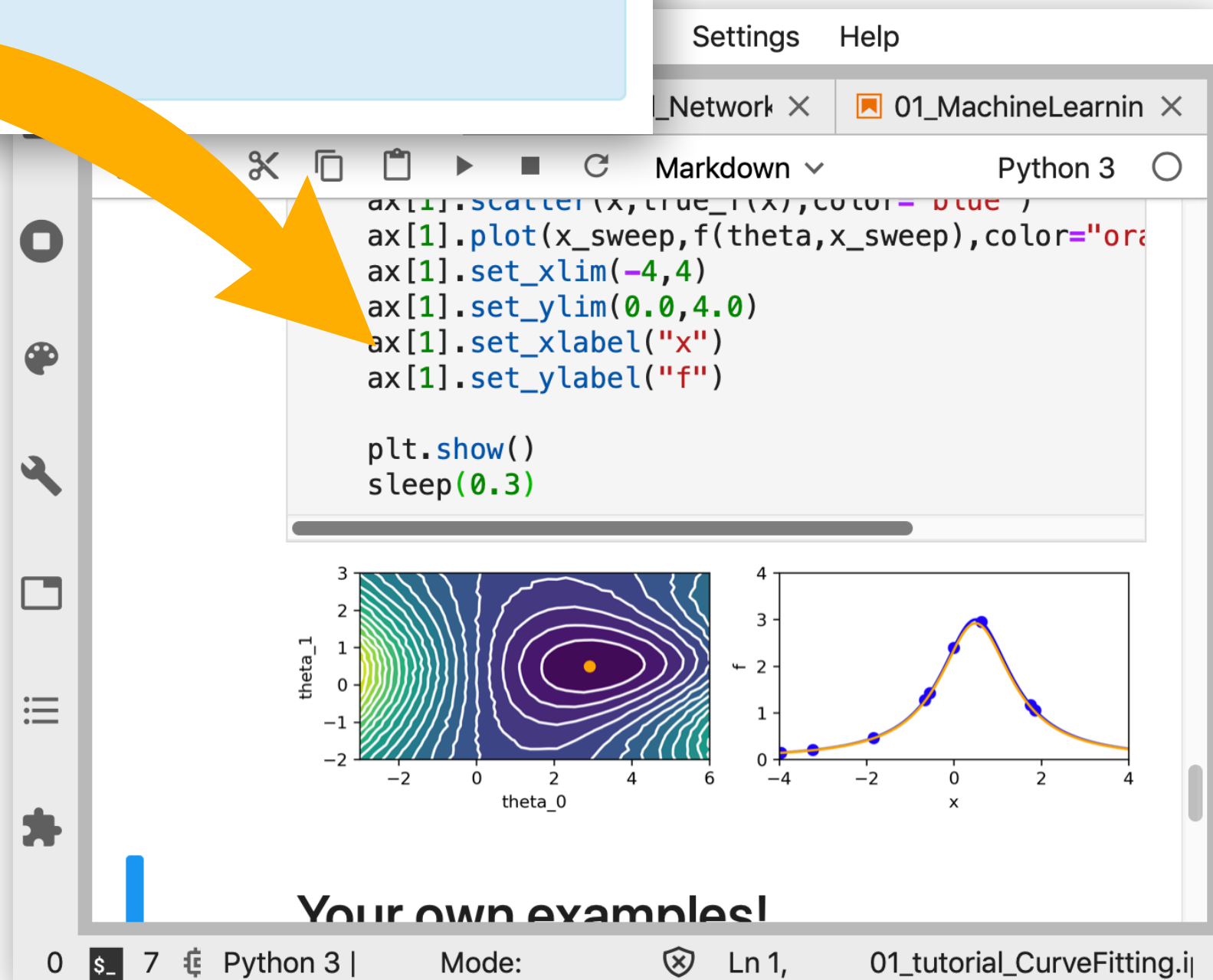
see: website/course overview/lecture I

This notebook shows how to calculate the forward-pass through a neural network in pure python, and how to illustrate the results for randomly initialized deep neural networks (as shown in the lecture).

### Notebooks for tutorials:

Tutorial: Network Visualization

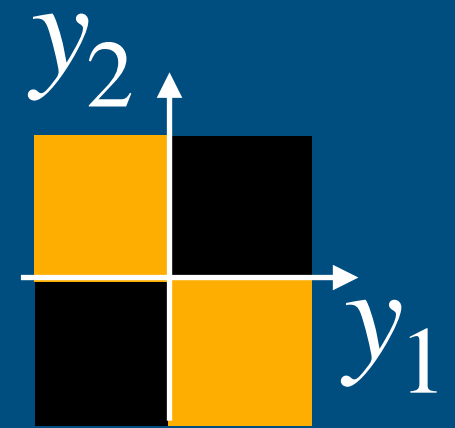
Tutorial: Curve Fitting



# Suggested Homework

Machine Learning for Physicists

- \* 1 Implement a network that computes XOR
- 2 Implement a network that approximately computes XOR, with 1 hidden layer(!)
- \* 3 Visualize the results of intermediate layers in a multi-layer randomly initialized NN
- 4 What happens when you change the spread of the random weights?
- 5 Explore cases of curve fitting where there are several (non-equivalent) local minima. Is sampling noise helpful?



\* minimum

Machine Learning for Physicists