LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Deep Learning for Physicists

Tutorial #0

Kosmas Kepesidis

Some helpful links

- Introduction to Python language: https://docs.python.org/3.7/tutorial/index.html

Some helpful links

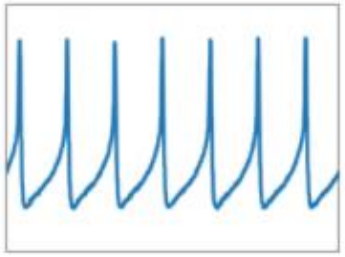
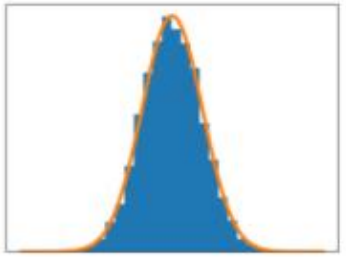
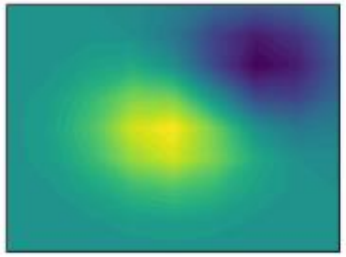
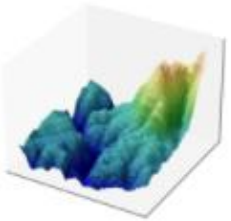
- **Numpy**: https://numpy.org/doc/stable/user/quickstart.html

  ➢ widely used library for numerical operations in Python

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<class 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<class 'numpy.ndarray'>
```

Some helpful links

- **Matplotlib**: https://matplotlib.org/
  - ➢ Basic visualization with Python
  - ➢ Easy to use library for creating static, animated, and interactive visualizations

Some helpful links

- **Pandas**: https://pandas.pydata.org/

  ➢ Fast, powerful, flexible and easy to use data analysis and manipulation tool

Take a Quick Look at the Data Structure

```
In [5]: housing = load_housing_data()
        housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Some helpful links

- **Keras**: https://keras.io/
  - beginner-friendly Deep Learning library
  - Main package to be used in the computational assignments
  - **TensorFlow** will be used as backend: https://www.tensorflow.org/

```
from tensorflow import keras
import numpy as np
```

Some helpful links

- **SciKit-Learn**: https://scikit-learn.org/stable/
  - Helpful machine-learning library
  - Widely used tool in data science

Some helpful links

- **Anaconda** distribution: https://www.anaconda.com/products/individual
  - Python distribution for local installation
  - **Anaconda** includes the so-called **Jupyter Notebooks** or **JupyterLab**: https://jupyter.org/
  - Computational assignments will be of the **Jupyter Notebook** format

Working with **Jupyter Notebooks**

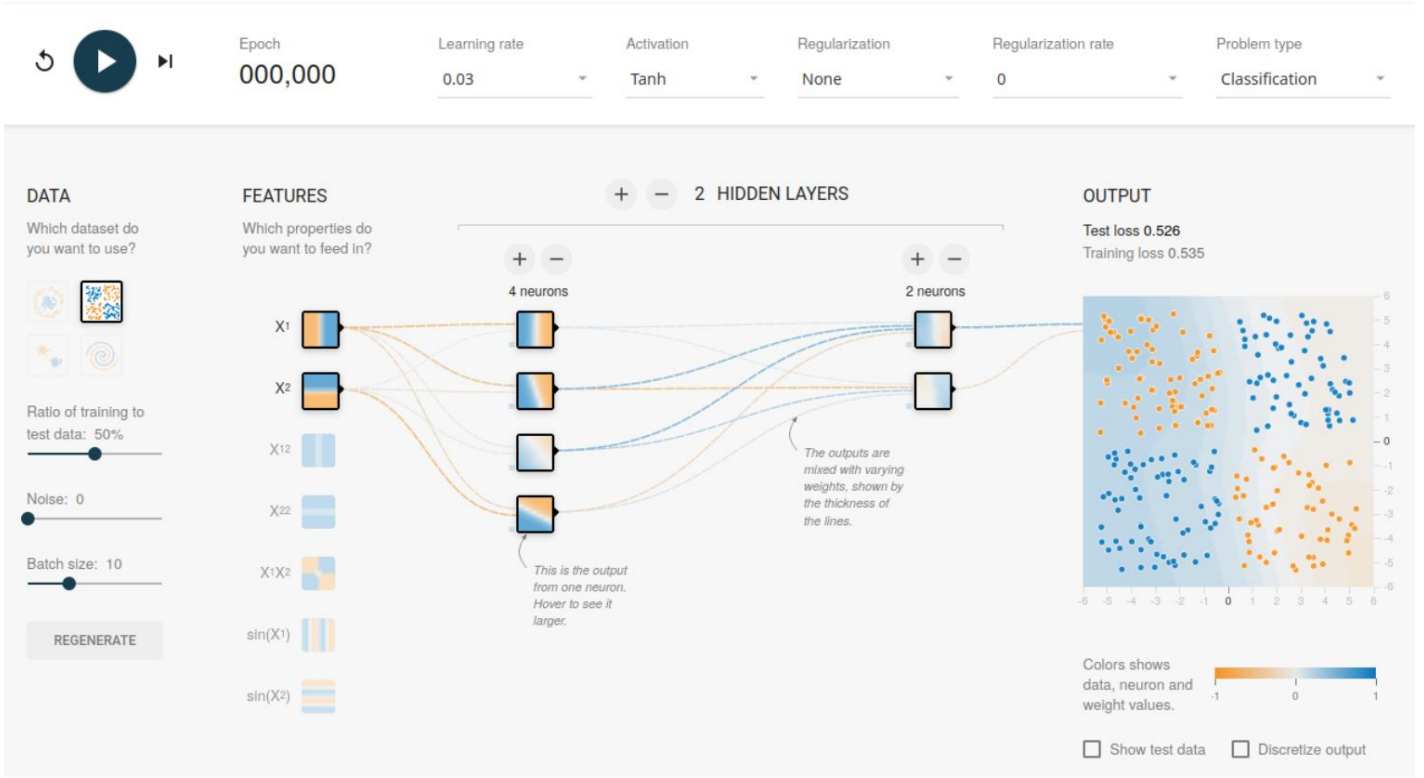
- See notebook *Tutorial_0.ipynbk* in folder *Tutorial 0*
- Analysis of the *California Housing* data set (Luís Torgo's page): https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html
  - Reference*:*
    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions, Statistics and Probability Letters, 33 (1997) 291-297
  - Description:
    - Collected information on the variables using all the block groups in California from the 1990 Concensus
    - Includes 1425.5 individuals living in a geographically compact areas
    - Final data contained 20,640 observations on 9 variables

Some helpful links

- Google **Colab**: https://colab.research.google.com/
  - Similar to Jupyter notebooks but in your web-browser
  - Files stored on Google Drive
  - Computational resources provided by Google, including virtual CPUs and GPUs
  - Tutorial: https://www.youtube.com/watch?v=inN8seMm7UI
  - Jupyter Notebooks can be uploaded to Google Drive and be used in Colab: https://medium.com/swlh/migrating-from-jupyter-to-colaboratory-2888332d57a7

Some helpful links

- **TensorFlow** Playground: www.playground.tensorflow.org



Some helpful links

- **TensorFlow** Playground: www.playground.tensorflow.org
  - Description:
    - Data corresponds to a 2D probability distribution and is represented by the value pairs (x1, x2)
    - Second data set: regions $x1, x2 > 0$ and $x1, x2 < 0$ are shown by one color, value pairs with $x1 > 0, x2 < 0$ and $x1 < 0, x2 > 0$, the regions are indicated by a different color.
  - Questions:
    1. Using ReLU activation function, what is the smallest network that gives a good fit result?
    2. What do you observe when training networks with the same settings multiple times?
    3. Which of the features is most helpful?

Some helpful links

- **TensorFlow** Playground: www.playground.tensorflow.org
  - Answers:
    1. A network with a single layer having 3 nodes seems to work but this configuration is not stable. A single layer with 4 nodes is more stable
    2. Due to the random initialization of weights, the network training development is always a bit different, leading to different results
    3. Obviously, $x1 \cdot x2$!

For next time…

- See folder *Tutorial 1*
  - Try to work on the problems in Jupyter notebook *Tutorial_1*