

Welcome to Deep Learning

Goals

At the end of this class, you should be able to:

Design, implement, and train your own deep network on any data you wish.

Plan

- Introduction / background
- First example (1-layer "deep" network)
- Deep networks
- Residuals and Normalizations (deeper networks)
- Convolution
- Transformers
- Making it work

At the end of this class, you should be able to:

Design, implement, and train your own deep network on any data you wish.

Content

Slides

Linear regression

Regression model:

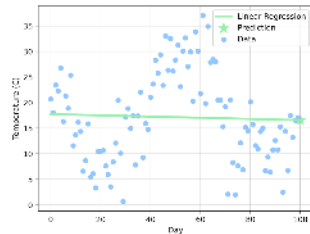
$$f_{\theta} : \mathbb{R}^n \rightarrow \mathbb{R}^d$$

Linear regression:

$$f_{\theta}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Parameter of linear regression model:

$$\theta = (\mathbf{W}, \mathbf{b})$$



Hands-on notebooks

The screenshot shows a Jupyter Notebook interface with two tabs: 'distributions.ipynb' and 'model.ipynb'. The active cell in 'distributions.ipynb' contains the following Python code:

```
import torch
import torch.distributions as dist

## Define a Bernoulli distribution
p = torch.tensor([0.3])
bernoulli = dist.Bernoulli(p)
print(bernoulli.sample([10]))
```

The output of the cell is a list of 10 Bernoulli samples, each represented as a 1x1 tensor:

```
tensor([[1.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.]])
```

The interface includes a toolbar with icons for search, run, and other actions. The status bar at the bottom indicates 'SSH: solo', '7' connections, '0' errors, '6' warnings, and '0' info messages. The current cell is 'Cell 1 of 5', and the time is '09:53'.

2 / 23

Have fun!!