Introduction to Deep Learning (I2DL)

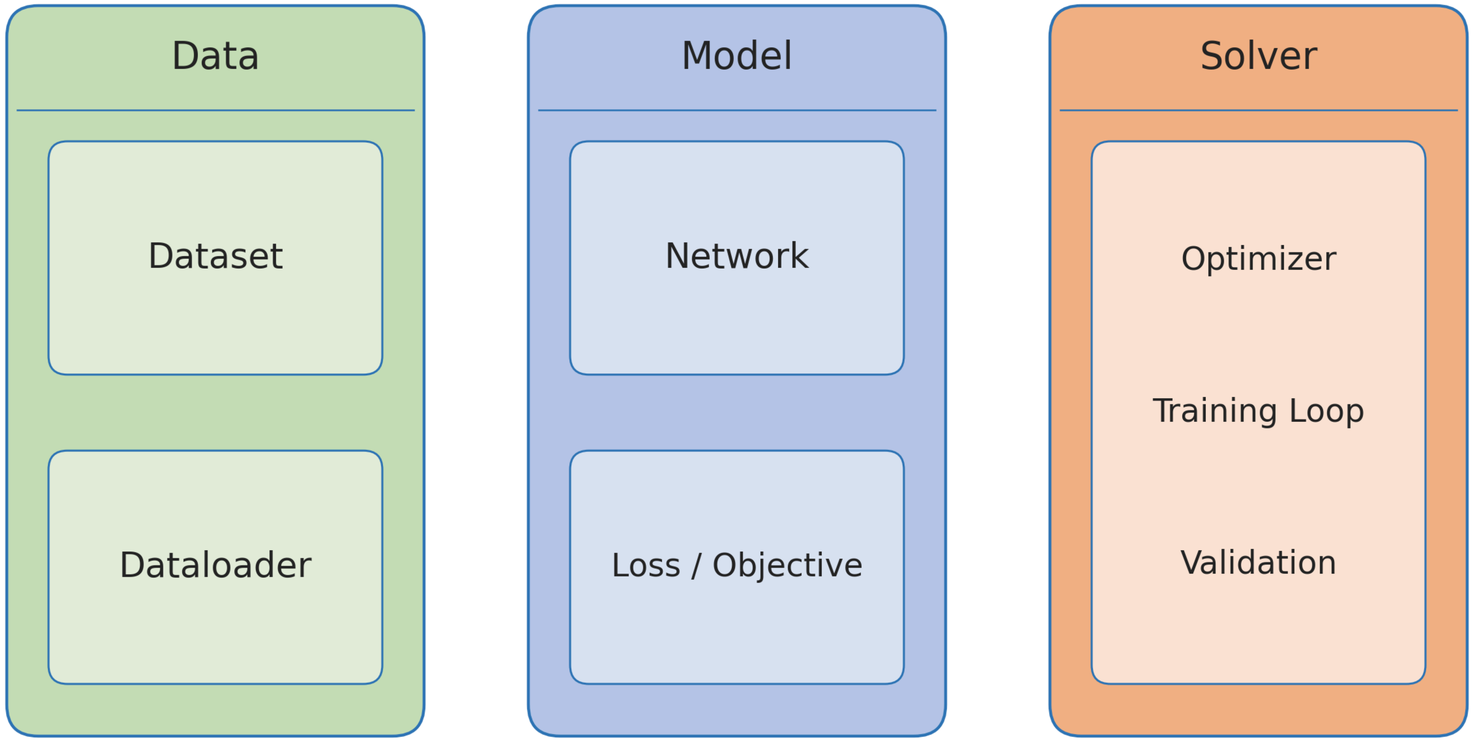
Tutorial 5: Neural Networks

Today's Outline

- **Recap of Exercise 4** — the pillars of deep learning
- **Universal Approximation Theorem** — why a single hidden layer is already powerful in theory
- **Exercise 5: Neural Networks — more NumPy, this time structured**
  - Modularized forward and backward blocks
  - Stack them into deeper networks for CIFAR ten

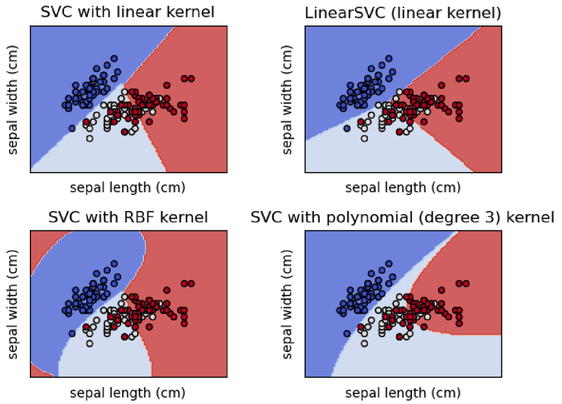
Recap: The Pillars of Deep Learning

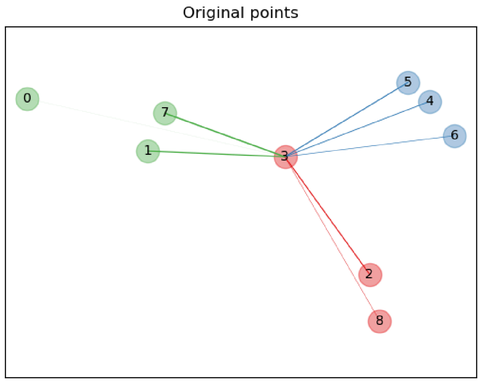
Data: done Model: today + Ex 6 Solver: more today



Recap: Back to the Roots

- **Before neural networks, classical ML did the job**
  - Support Vector Machines: project data into a higher-dimensional space, then separate with a hyperplane
  - Nearest Neighbors: classify by the closest labelled point
- **What neural networks add**
  - Stacked non-linearities: learn far more complex decision boundaries
  - Features are learned end to end, not hand-crafted



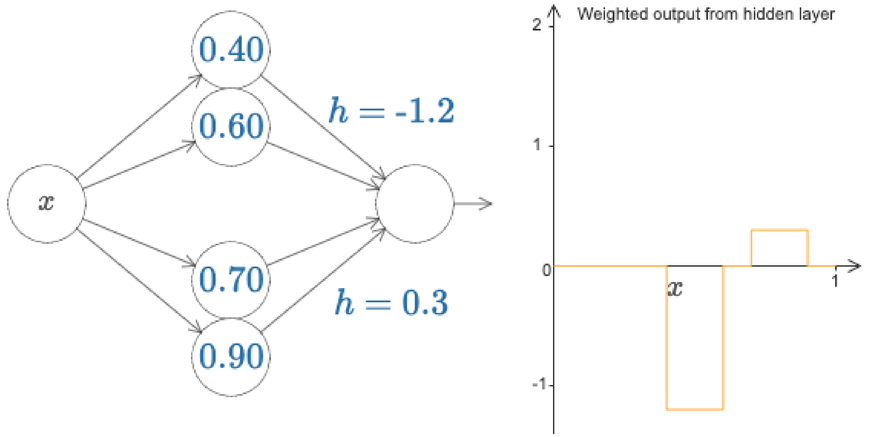


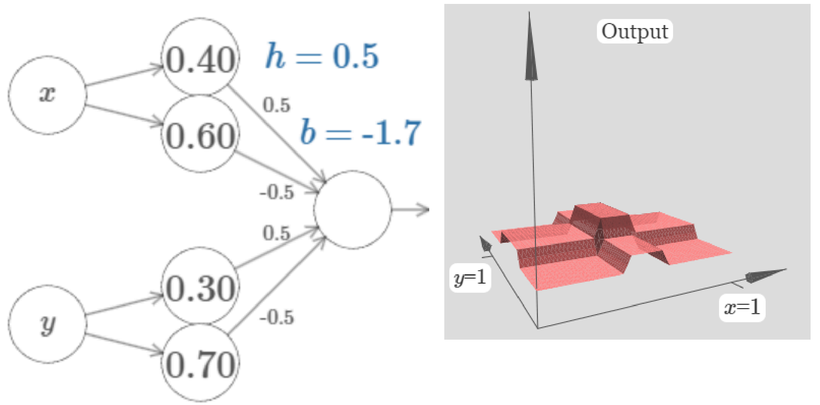
Universal Approximation Theorem

The theorem

- A feed-forward network with **one hidden layer** and a non-linear activation can approximate any continuous function on a compact set.
- **Needs:** enough hidden units, plus a non-linearity (sigmoid, ReLU).
- **Doesn't say:** that the weights are easy to find, or that one layer is best in practice.

Approximation examples

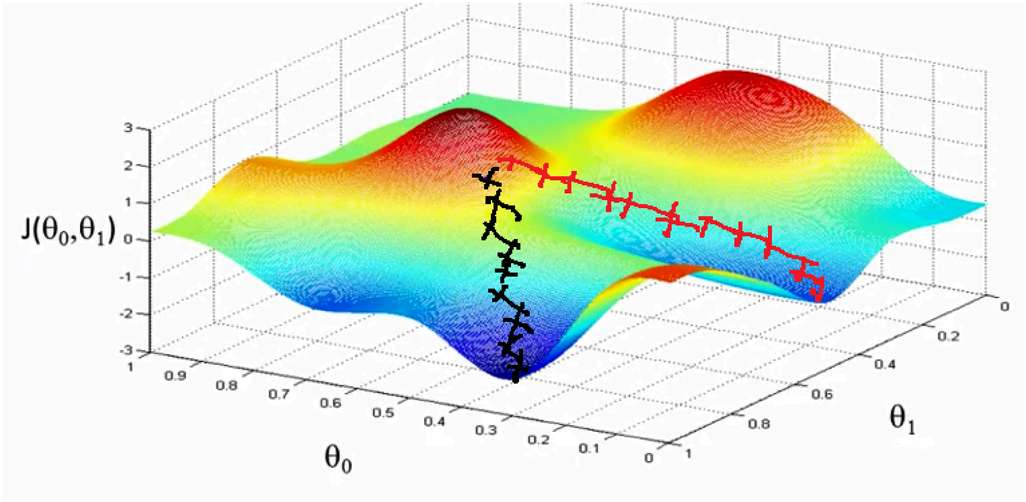




UAT — Further Reading

- **Readable proof — accessible to a third-semester math student**
  - mcneela.github.io — Universal Approximation Theorem
- **Visual proof — interactive web page; change parameters and watch the approximation update**
  - neuralnetworksanddeeplearning.com — Chapter 4

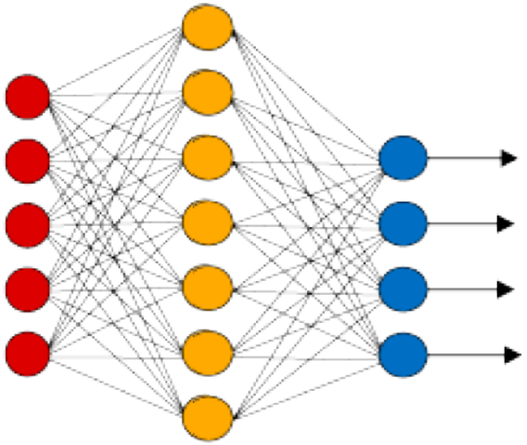
A Word of Warning



- **The theorem promises existence, not findability.**
  - We still have to *find* the weights by gradient descent from a random initialisation.
  - The loss surface is full of local minima; one hidden layer rarely generalises well in practice.

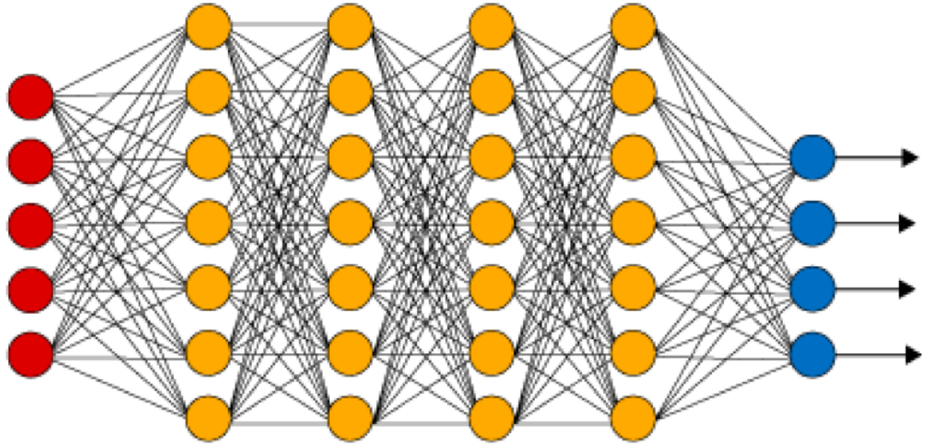
How Deep is Your Love?

Shallow — 1 hidden layer



- What the UAT covers. Powerful in theory; brittle in practice.

Deep — more than 1 hidden layer



- Two, three, or many more layers stacked together.

Obvious Questions

- **Q: Do we even need deep networks?**
  - **A: Yes.** Multiple layers give more abstraction power for the same compute, and learn hierarchical features — local edges, then parts, then whole objects.
- **Q: So we just build hundred-layer networks?**
  - **A: Not trivially.** Memory blows up. Gradients vanish through long chains.
  - Deeper is not automatically better — data, hyperparameters, and architecture all matter.

Recap: Exercise 4 — Doesn't Scale

Exercise 4 — what you built

- Small dataset, simple objective (house price: cheap vs expensive)
- One classifier class — single weight matrix
- Full-batch gradient descent, whole forward pass in memory

Exercise 5 — where we are headed

- **CIFAR-10** — a real image-classification benchmark, ten classes
- **Modularized network** — chain-rule blocks you stack at will
- **Stochastic gradient descent** — process one mini-batch at a time

New: Modularization

- **Each layer is its own block:**
  - **forward** does the computation and caches what it needs
  - **backward** consumes the cache and the upstream gradient, returns the downstream gradient
- The chain rule glues the blocks together — stack as many as you want.

Chain Rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial y}$$

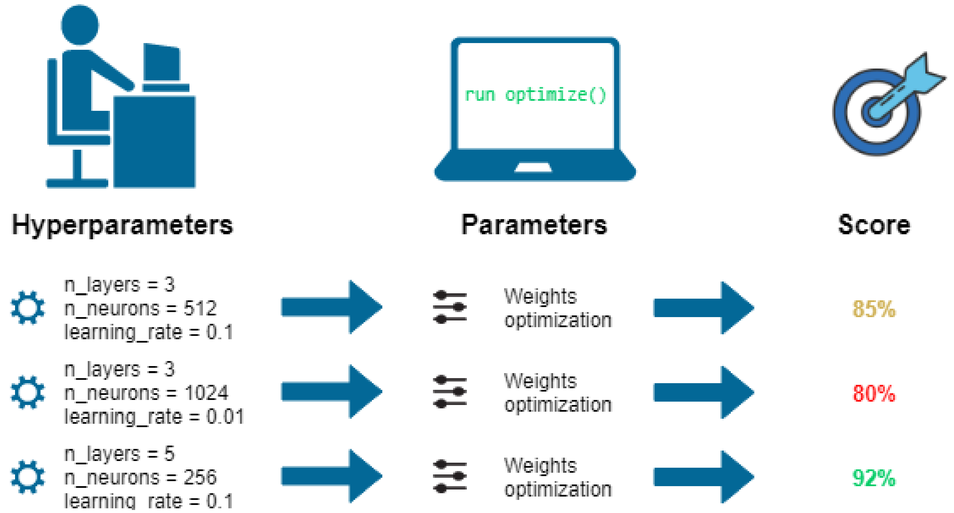
```
class Sigmoid:
    def __init__(self):
        pass

    def forward(self, x):
        """
        :param x: Inputs, of any shape

        :return out: Output, of the same shape as x
        :return cache: Cache, for backward computation, of the same shape as x
        """
```

Overview: Exercise 5

- **One notebook** — but a long one
- **Multiple smaller implementation objectives stacked inside it**
  - Layer building blocks: Sigmoid, ReLU, Affine
  - An N-layer classification network
  - Cross-entropy loss for multi-class output
  - Optimizers: SGD, SGD with momentum, Adam
- You won't write each block from scratch — the notebook walks you through with detailed instructions; follow them step by step.

Outlook: Exercise 6 — CIFAR-10 Again

- **You'll reuse the Exercise 5 network and tune its hyperparameters**
  - Number of layers and units per layer
  - Learning rate, batch size, optimizer choice
- Watch training and validation curves shift as you change one knob at a time — try to find the best setting for the same task.



See you next week!