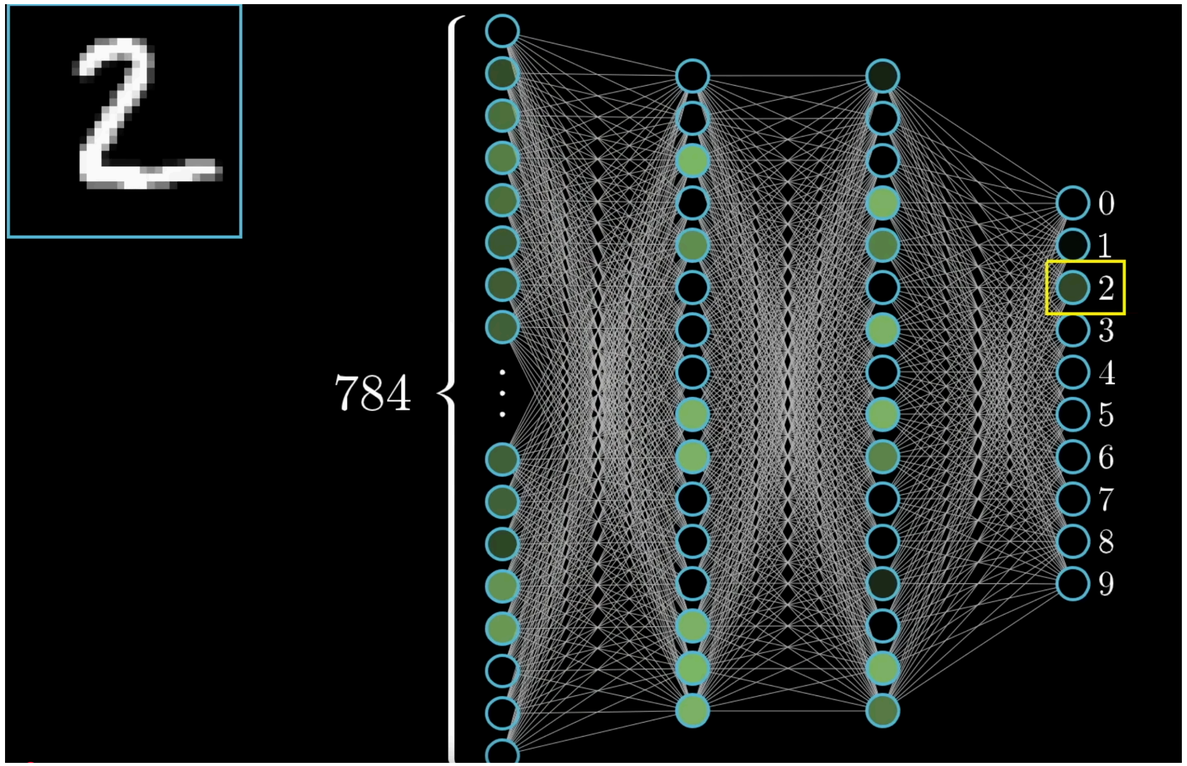
Unit 10 SML

Gradient Descent and k-fold cross-validation

Dominic Bunnett, TU Berlin + Open University

Recap

- What is Supervised Machine Learning?
- We want to write a program which will perform some sort of task for us.
- E.g. recognise handwriting and input it into text
- We start with a neural network, a shell
- Supervised means, for example here, that we start with a bunch of pictures of numbers with the data of what they are. Then using this, we tweak the network so that it can do it for random input!



Recap

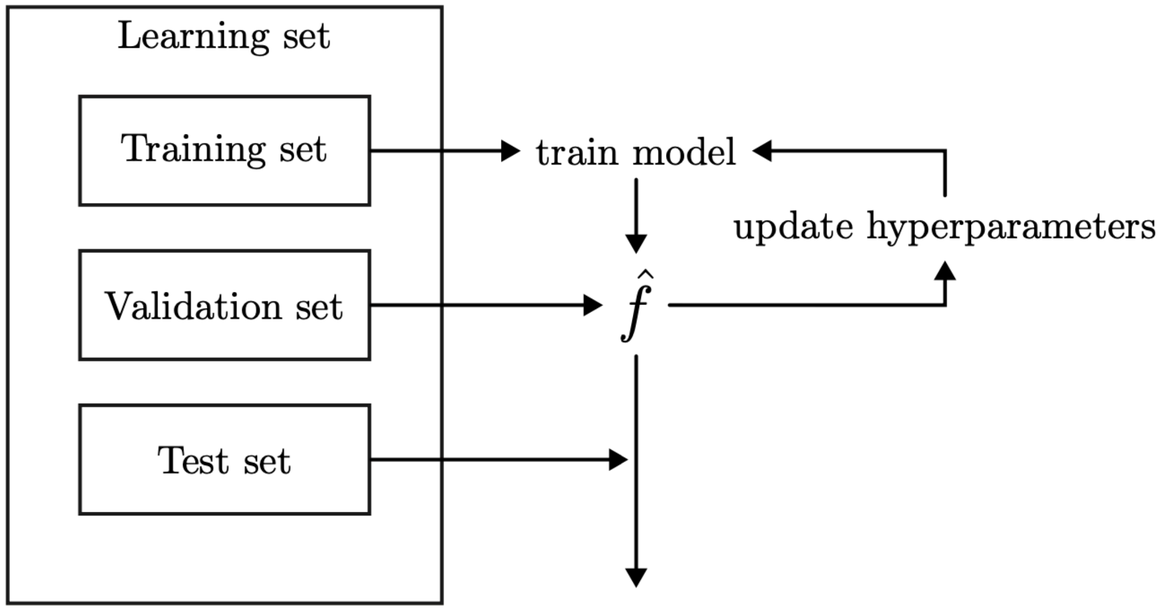
- We will see some examples in due course.
- The first important step is to pre-process the data, to make sure you're giving the network the best chance.
- To do this, you remove obviously ugly things. Such as missing entries (how?), data that's clearly out of bounds / typos
- Then using t-testing, you check what data actually matters (that is, to a certain statistical significance)
- If it is irrelevant, get rid of it!
- For example, if you want to know if it's going to rain, and in your data sample, an input is whether Mercury was in retrograde or not, hopefully, the data reflects that this is insignificant! Drop it!

Recap

Training a MLP

To train an MLP with weights and biases as parameters, using given training, validation and test datasets containing feature vectors $\mathbf{x}_i$ with corresponding expected outputs y_i, we proceed as follows.

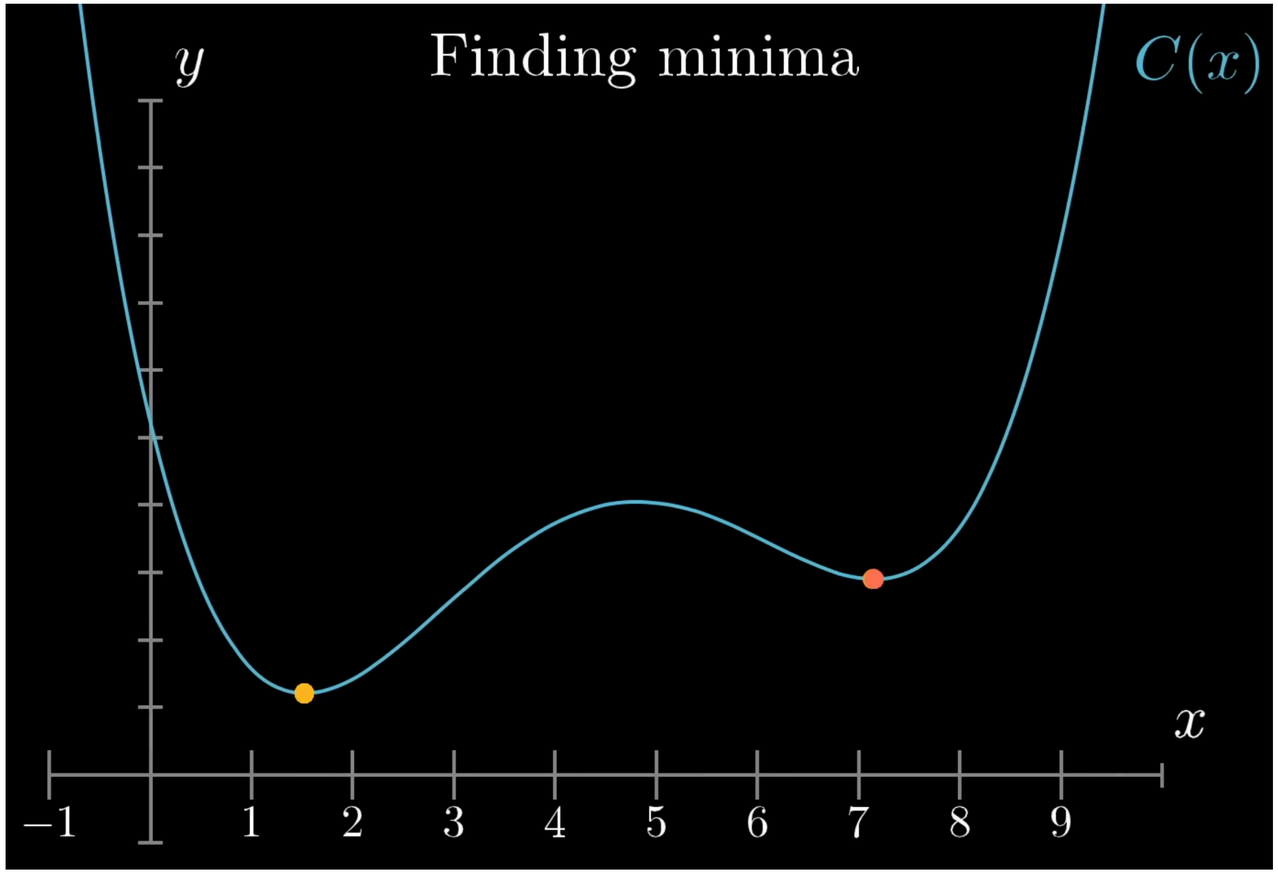
1. Assign random values to the weights and biases.
2. Select one or more cases from the training set at random, without replacement.
3. (*Forward pass*) Calculate the output of the MLP for each selected case (using the current values of the weights and biases), and use a cost function to measure the error between the output and the expected output.
4. (*Error back-propagation*) Use the calculated error to update the weights and biases.
5. Repeat from step 2 until all cases have been selected.
6. Repeat steps 2–5 a specified number of times, or until the calculated error is sufficiently small.
7. Evaluate the performance of the model using the validation dataset. If the performance is not sufficient, adjust the hyperparameters (for example the number of hidden layers or nodes), and repeat steps 1–6.
8. Assess the generalisability of the trained network using the test dataset.



Gradient descent

You've seen this before

- Gradient descent is very similar Newton-Raphson and other types of optimisation you've seen before
- It is about finding local minima of a certain function
- It's just calculus!



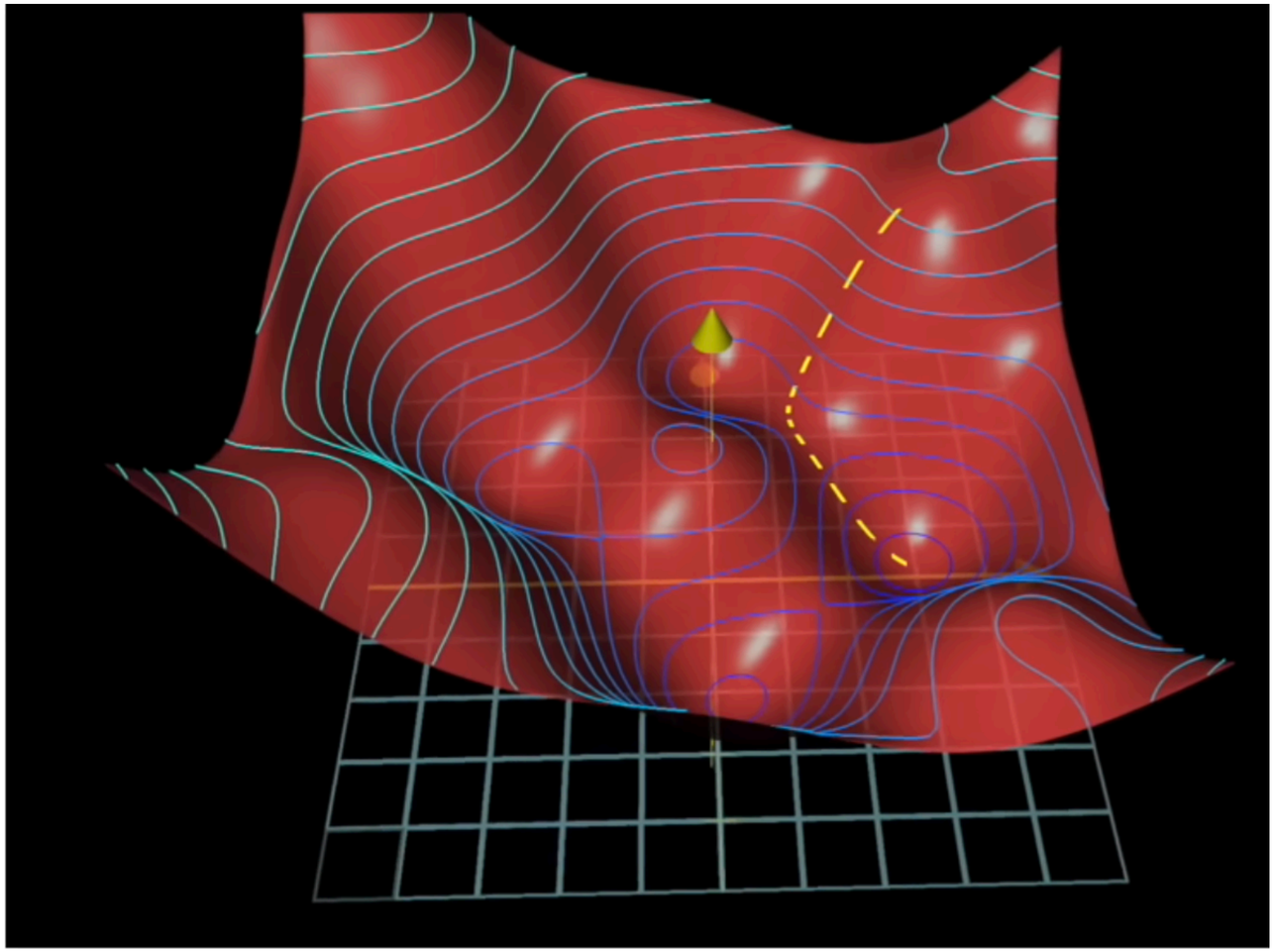
This is an example for one weight and no bias

Gradient descent

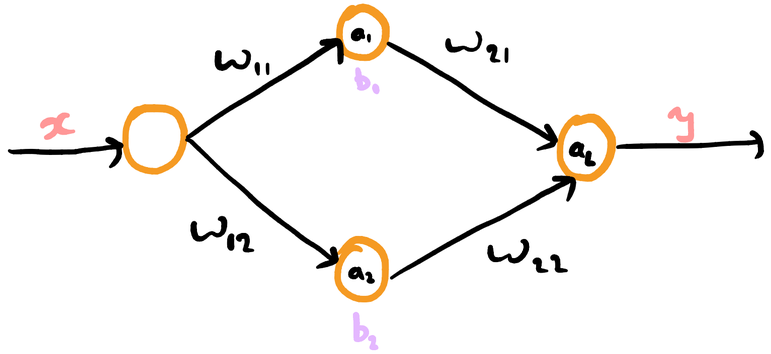
- The starting point of the yellow line is our random first choice of weights and biases
- Then we look for the steepest downward direction on the red surface, and we move amount η in that direction:

$$\omega_{i,new} = \omega_{i,old} - \eta \frac{\partial J}{\partial \omega_i}$$

- Let's see an example



Given a neural network



& activation function σ, then we can consider this whole setup as a function $f: \mathbb{R} \to \mathbb{R}$

$$f(x) = \sigma(a_L)$$
$$= \sigma(w_{21}a_1 + w_{22}a_2)$$
$$= \sigma\left(w_{21} \cdot \sigma(w_{11}x - b_1) + w_{22} \cdot \sigma(w_{12}x - b_2)\right)$$
$$= y.$$

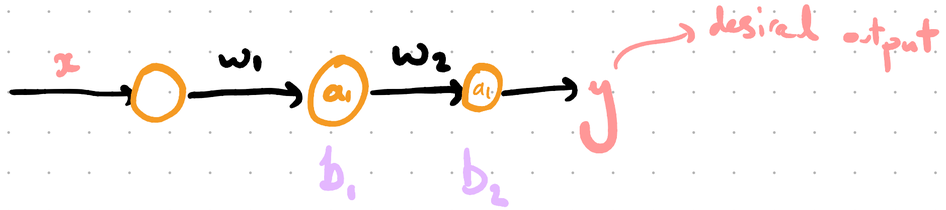
↳ by definition of y

Then given sampling data $SD := \{(x, y)\}$ ie data such that $f_{perfect}(x) = y$, we define an error function

$$J(SD, f) := \frac{1}{|SD|} \sum_{\substack{(x,y) \\ \in \\ SD}} (f(x) - y)^2$$

Careful! This is really a function in $\underline{w}$ & $\underline{b}$, that is, given our sampling data, we want to change f s.t. this error is as small as possible!

Let's do a really easy example:

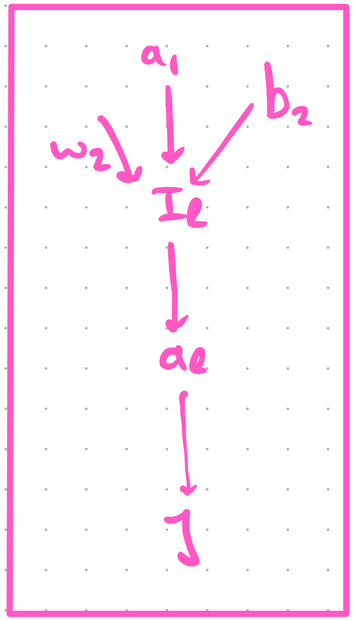


Thus $J = (a_2 - y)^2$. Now $a_2 = \sigma(\underbrace{w_2 a_1 + b_2}_{I_2})$

CHAIN RULE

$$\frac{\partial J}{\partial w_2} = \frac{\partial I_2}{\partial w_2} \cdot \frac{\partial a_2}{\partial I_2} \cdot \frac{\partial J}{\partial a_2}$$

DIAGRAM OF INFLUENCE



$$\frac{\partial J}{\partial a_L} = 2(a_L - y)$$

$$\frac{\partial a_L}{\partial I_L} = \sigma'(I_L)$$

$$\frac{\partial I_L}{\partial w_2} = a_{L-1} = a_1$$

Then we do the same with a_{L-1}!

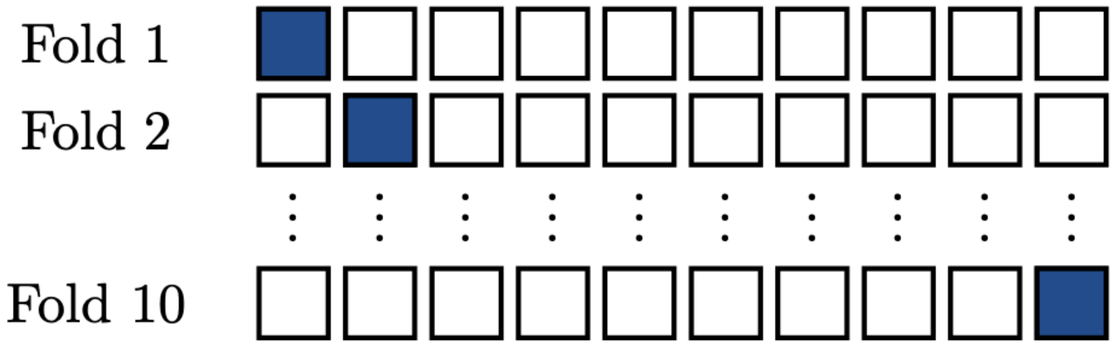
Finally: $$w_{2,new} = w_2 - \eta \frac{\partial J}{\partial w_2}$$

sub in values to compute this number!

k-fold cross-validation

Clever ways to get the most out of your training data

- Also called data resampling
- Useful for sample data
- Split up your data into k parts
- Remove one and train on what's left
- Use last one to test
- Simple!



Here is a visual of what 10-fold cross-validation looks like.
The blue is segment of the data we use for testing.

k-fold cross-validation

- Random sampling without replacement
- We get k performance value, the average of which is the k cross-validation performance
- Useful for hyperparameter / meta tuning
- That is, how many nodes, even how many hidden layers
- Let’s have a look at an example