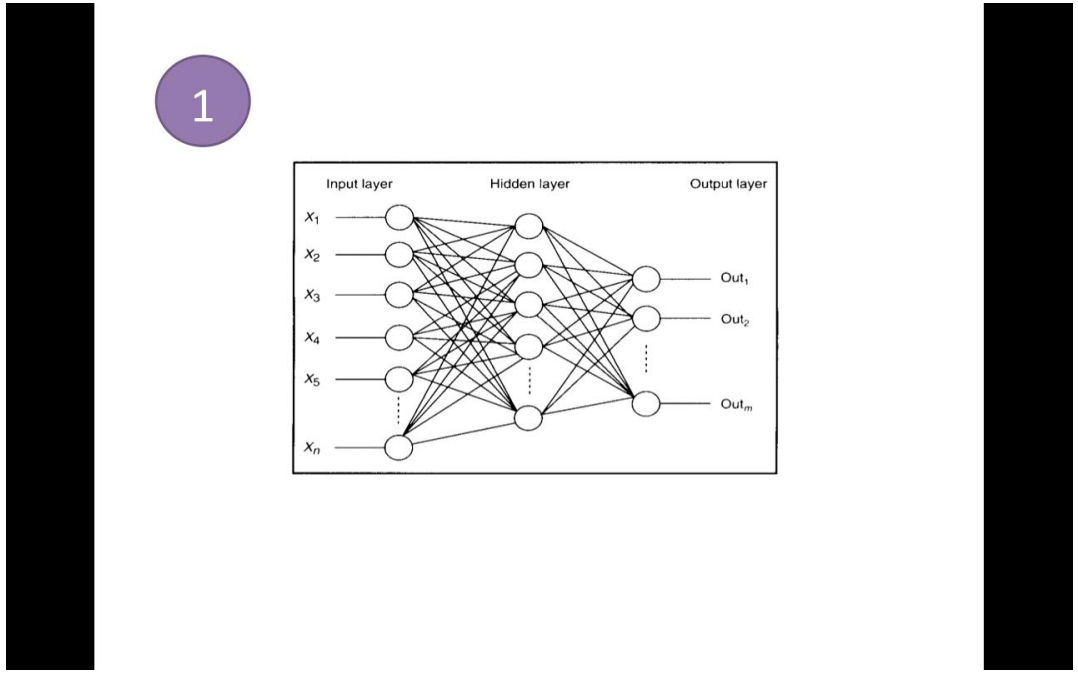
4 – Data Analysis in Python II

Recap and what this session is about

- You learnt how to read data, preprocess data, describe data using visualizations and summary statistics, and apply Decision Trees to data.
- In recent years, **artifical neural networks** have become particularly popular for supervised tasks, e.g., function approximation in general and text generation in NLP.
- Several advantages: **feature engineering** (e.g., „raw“ pixels from image), learns complex, non-linear functions in a flexible manner, scalable
- Disadvantages: large amounts of data needed to perform well, computationally intense

Neural networks - motivation

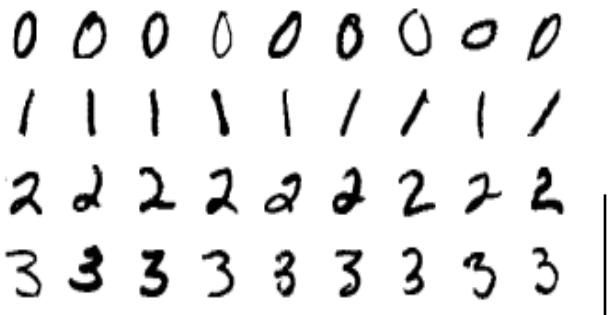
- Technology companies such as Facebook, Google and Amazon use Deep Learning models to, e.g., analyze customer data, provide recommendations, automatically tag videos or news, etc.
- Based on Rosenblatt (1958) Perceptron
- Example:



Source: https://giphy.com/gifs/neural-networks-sq7yhyvOg031S

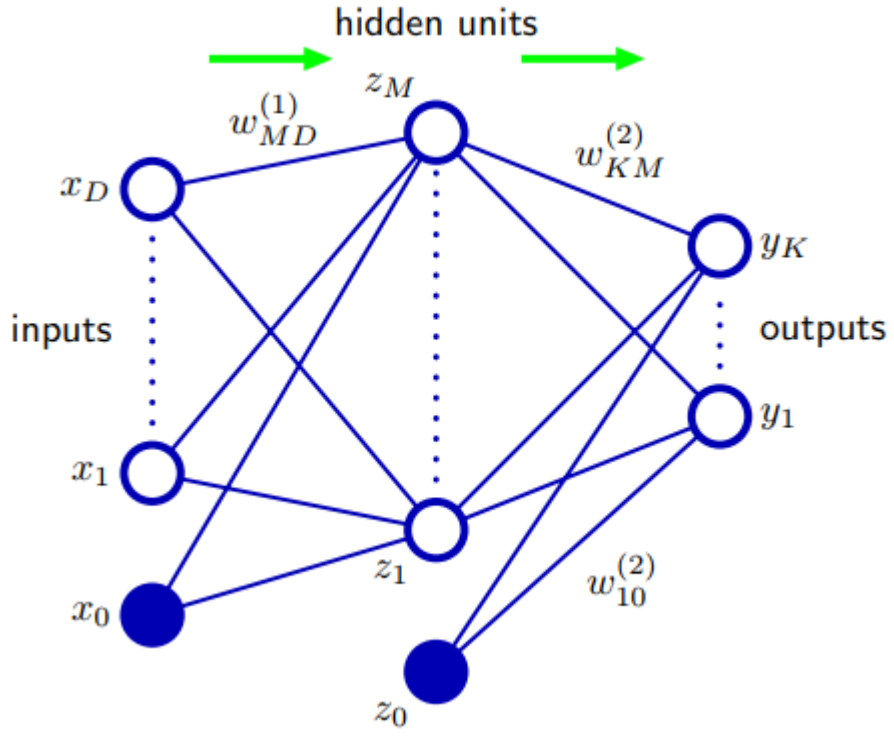
Neural networks – overview

- Origin in biology, i.e., modelling connections between neurons, their activation, and the flow of information between neurons
- Example: classification using MNIST data (http://yann.lecun.com/exdb/mnist/)



Feature "Generation"

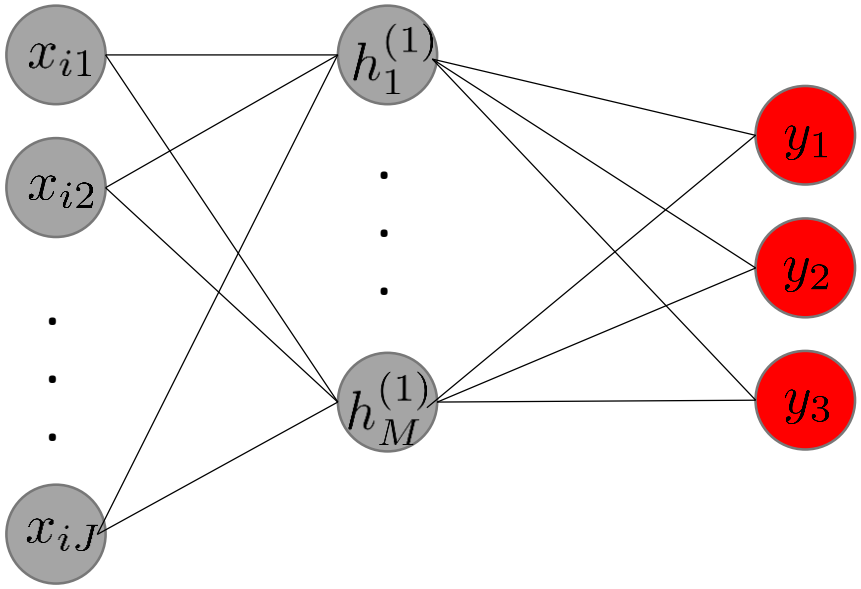
$$\begin{matrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{matrix}$$



Sources: https://en.wikipedia.org/wiki/MNIST_database
Bishop (2006)

Our today's goal with neural networks

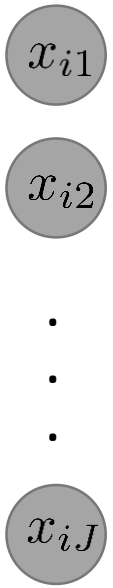
- Given a set of a credit applicant's features, e.g., employment length, we want to predict his/her credit grade.
- Therefore, the **input** is the set of an applicant's features and the **output**, that we want to predict, is the applicant's credit grade.
- The mapping from the input to the output is modelled using a NN:



x_{i1}: 1st feature of applicant i
y_1: probability that i has credit score 1

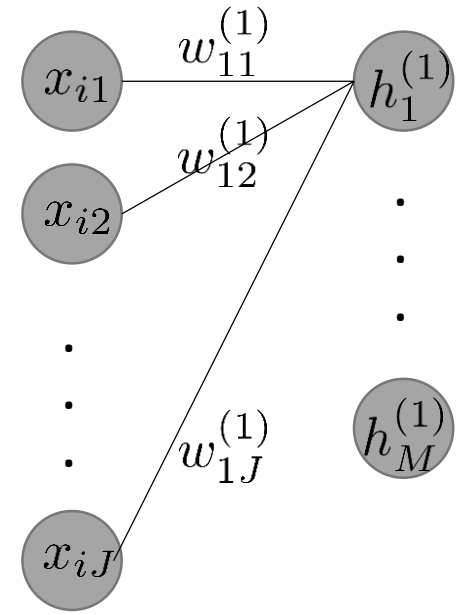
Feed-forward neural network

- Assume for a set of observations $i = 1, \ldots, n$ a **feature input** of length J, such that the feature of the i-th observation is denoted as $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})'$



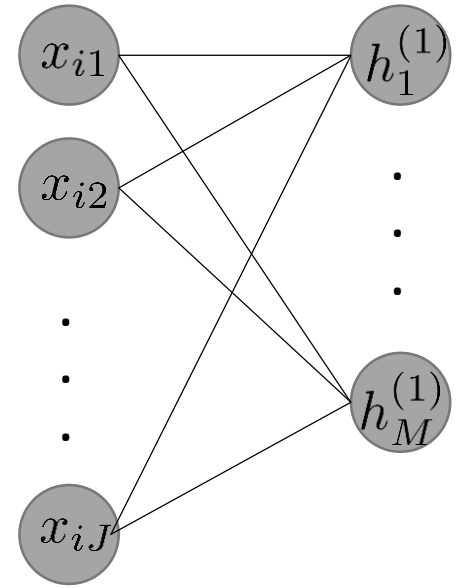
Feed-forward neural network

- The feature input $\mathbf{x}_i$ is connected to **hidden layer (1)** with M nodes as follows $h_m^{(1)}(\mathbf{x}_i, \mathbf{w}^{(1)}) = \sum_j w_{mj}^{(1)} x_{ij} + b_m^{(1)}$, where $\mathbf{w}^{(1)}$ is a weight matrix of size $M \times J$.



Feed-forward neural network

- The feature input $\mathbf{x}_i$ is connected to **hidden layer (1)** with M nodes as follows $h_m^{(1)}(\mathbf{x}_i, \mathbf{w}^{(1)}) = a(\sum_j w_{mj}^{(1)} x_{ij} + b_m^{(1)})$ where $\mathbf{w}^{(1)}$ is a weight matrix of size $M \times J$ and $a(\cdot)$ is an activation function.



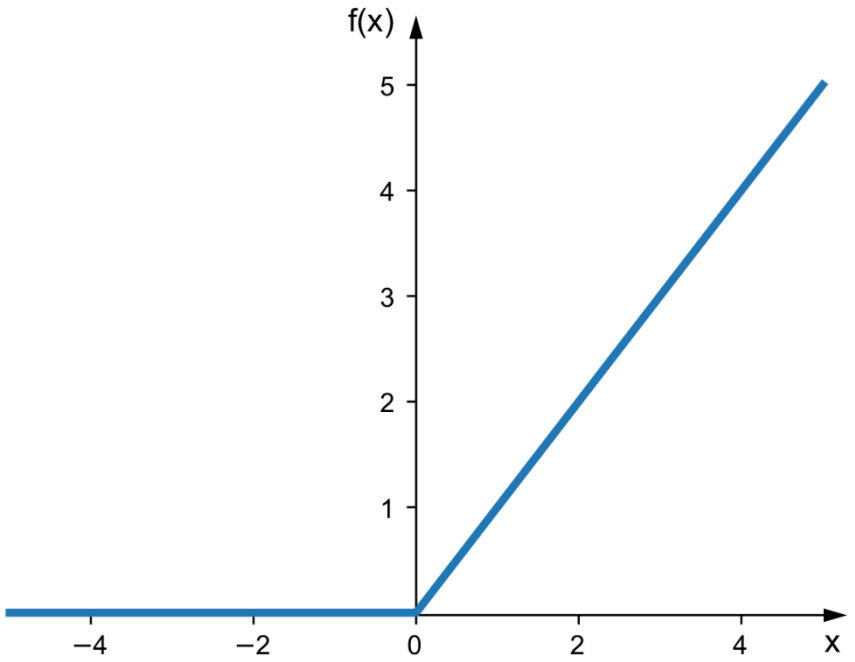
Feed-forward neural network

- **Activation function**: is a differentiable, non-linear function that connects the input layer (hidden layer) with another succeeding hidden layer to determine whether a **node is active or not**.

- Example: rectified linear unit (**ReLU**)

$$f(x) = \max(0, x)$$

  - Yields a non-linear output (from a linear input)
  - Close to linear due to its piece-wise definition
  - Easy to differentiate for gradient computation

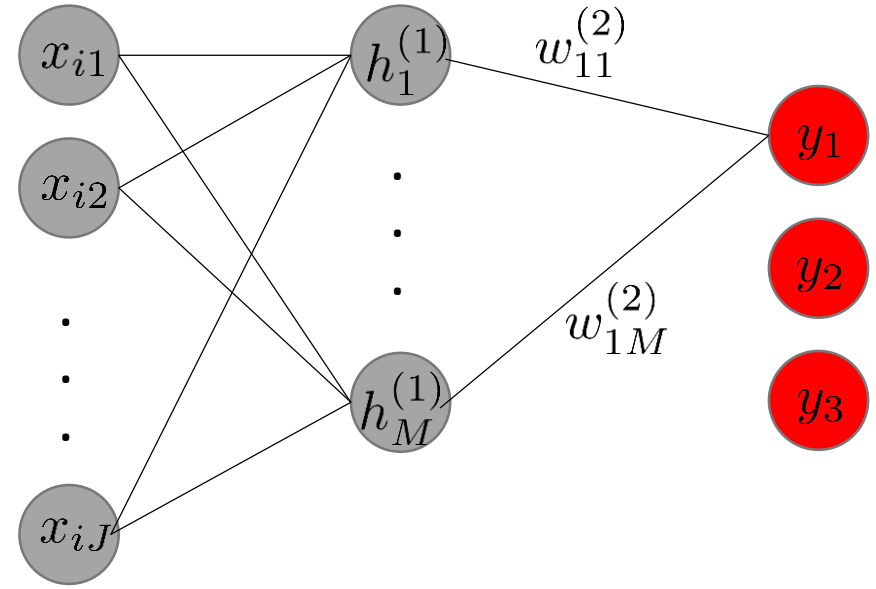


Source: https://sebastianraschka.com/faq/docs/relu-derivative.html

Feed-forward neural network

Output layer.

$$y_k(h_1^{(1)}, \ldots, h_M^{(1)}, \mathbf{w}^{(2)}) = a(\sum_m w_{km}^{(2)} h_m^{(1)} + b_k^{(2)})$$

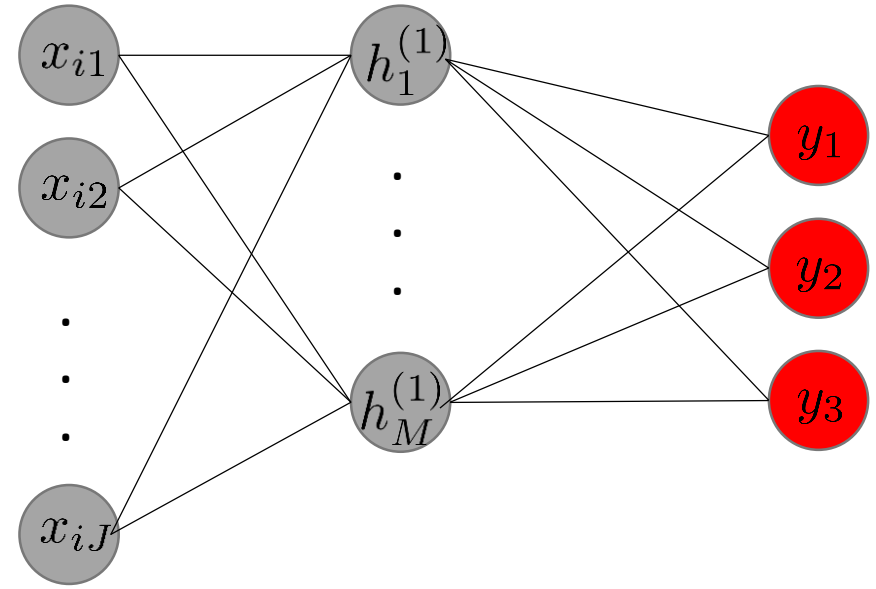


inputs **hidden layer** **output layer**

Feed-forward neural network

Output layer.

$$\hat{y}_k(h_1^{(1)}, \ldots, h_M^{(1)}, \mathbf{w}^{(2)}) = a(\sum_m w_{km}^{(2)} h_m^{(1)} + b_k^{(2)})$$



Activation function for output

- The functional form depends on whether the NN describes a **regression or classification** problem.
- For multi-class classification (what we will do) – **softmax** function:

$$a(z_k) = \frac{\exp(z_k)}{\sum_{k'} \exp(z'_k)}$$

- Where $z_k = \sum_m w_{km}^{(2)} h_m^{(1)} + b_k^{(2)}$ and the predicted class for a given observation is the one with maximum support, i.e.,

$$y_k^* = \max_k \hat{y}_k$$

- For regression – a linear function

Loss function

- Is the loss when a wrong decision is made by the NN
- Depends on optimization problem, we will for now focus on classification problems
- Idea: find a set of weights in the NN such that the loss function is minimized, i.e., we find a set of parameters that optimally fit the observed data (potentially subject to constraints → future course)
- We have a multi-class problem vs. a binary classification problem (or even a multi-label problem…)
- We will use the categorical cross-entropy loss (for binary problems):

$$\sum_i -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Hyperparamters (not complete) – actually the hardest part in applied Deep Learning

- Number of nodes
- Number of layers
- Activation functions
- Optimizer (optimal weights to minimize loss function)
- Batch size (later covered when talking about Keras)

- Techniques to prevent overfitting (not covered today):
  - Regularization
  - Dropout layers
  - …

Example in Python using Keras

- There are other Deep Learning libraries in Python as well, but Keras is the easiest framework to learn.
- Others are, e.g., Tensorflow or PyTorch
- Our **classification problem** using the Lendingclub data:
  - We want to classify the grade of a debtor conditional on some of his/her **characteristics**:
    - Employment length (numerical)
    - Annual income (numerical)
    - Debt-to-income ratio (numerical)
    - Home ownership (categorical)

Some basics before we dive into keras

- Batch size: faster and less memory consuming than using the full data set, might be at the cost of less precise gradient computations
- Number of epochs: number of times the dataset is used (during the training process) for estimating the model parameters
- What to do with categorical features? → one-hot encoding, i.e., introducing a dummy for each category for a given feature
  - Feature “home_ownership”: (“RENT”, “OWN”, “MORTGAGE”, “RENT”) gives

RENT	OWN	MORTGAGE
1	0	0
0	1	0
0	0	1
1	0	0

Tasks (together)

- Create a feed forward neural network with
  - 1 input layer with #nodes is the number of features (let's see!)
  - 1 hidden layer with 20 nodes
  - 1 output layer that has seven outputs, i.e., debtor's grade
- Train network with
  - Batch size 50
  - Five epochs

Tasks

1. Hyperparameter optimization
   - Create a loop, run the model for different number of hidden layers (1,2,…,10) and report the respective accuracies for a test set.
   - Create a plot for the accuracies w.r.t. number of hidden layers.

2. Evaluation
   - A higher number of hidden layers means that the NN can mimic more complex functions, but might also be prone to overfit.
   - Does the model fit using the test data improve with an increasing number of layers? If yes, is the effect large?

References

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press, available at https://www.deeplearningbook.org/
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Keras: The Python Deep Learning library, https://keras.io/