Welcome to the Birthplace of Deep Learning!

Computation Intelligence and it’s Application in Mechatronics

Winter 2025

Amirkabir University of Technology
(Tehran Polytechnic)

Course Overview

What you are going to learn:

Title	Content	Weeks
Perceptron	An introduction to our course and ANN	1
MLP	MLP, keras API, training and optimizing a deep NN	2
RBF and Hopfield	A quick introduction to RBF and Hopfield implementation	1
CNN	CNN layers, kernels, dilated CNN, well-known CNN architectures, …	2
RNN	RNN, LSTM, GRU, CNN and LSTM integration, …	2
Transfer Learning, Transformers	Using pre-trained models, attention, and transformers	1
Identification and Control	Online training of NN with application in identification and control	1
Reinforcemet Learning	RL introduction, NN policy, policy gradient, DQL, A2C, ….	3
Few-Shot Learning	Meta learning based few-shot learning	1
Fuzzy Logic	Fuzzy logic inference, fuzzy clustering, ANFIS, ….	1

Course Overview

What you won’t learn in this course:

- Natural language processing (NLP)
- Generative Adversarial Networks (GAN)
- Autoencoders (AE)
- Machine-learning Algorithms including: Bayesian, SVM, GPR, Decision Tree, …

Prerequisties

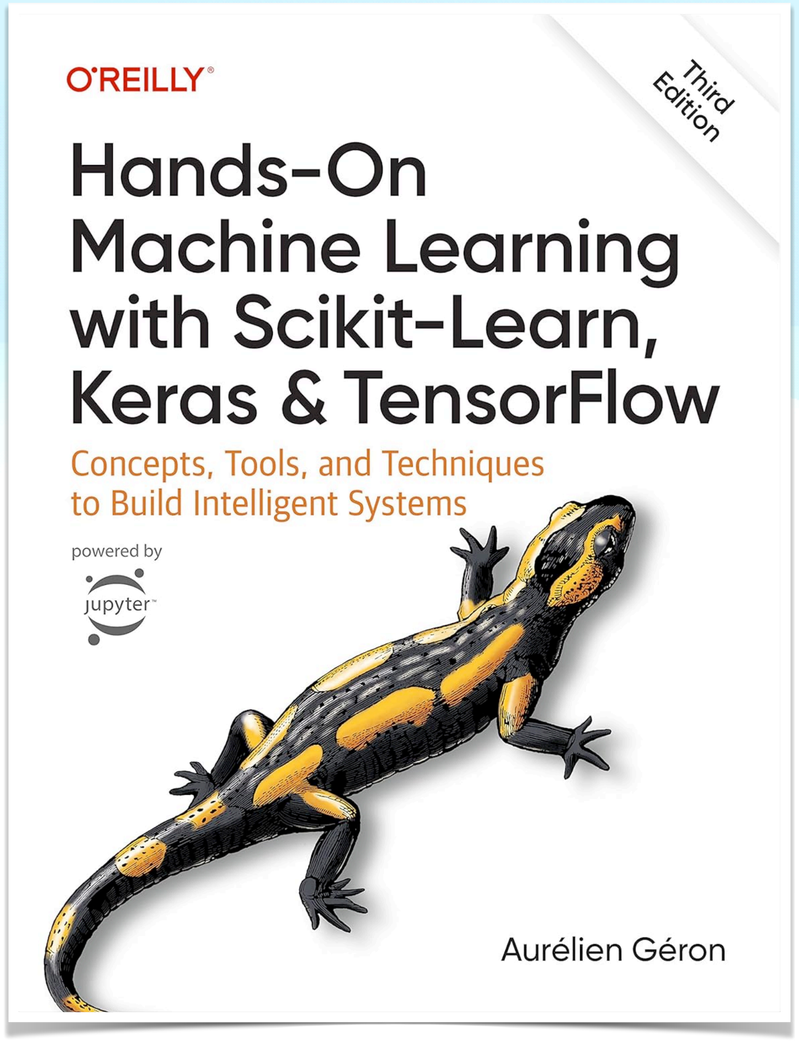
What you should know before starting:

- Good Python programming knowledge (NumPy, Pandas, matplotlib, working with datasets, …)
- Computational Intelligence Fundamentals
- Linear algebra, probability, and calculus (basic level)

Materials & References

Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems

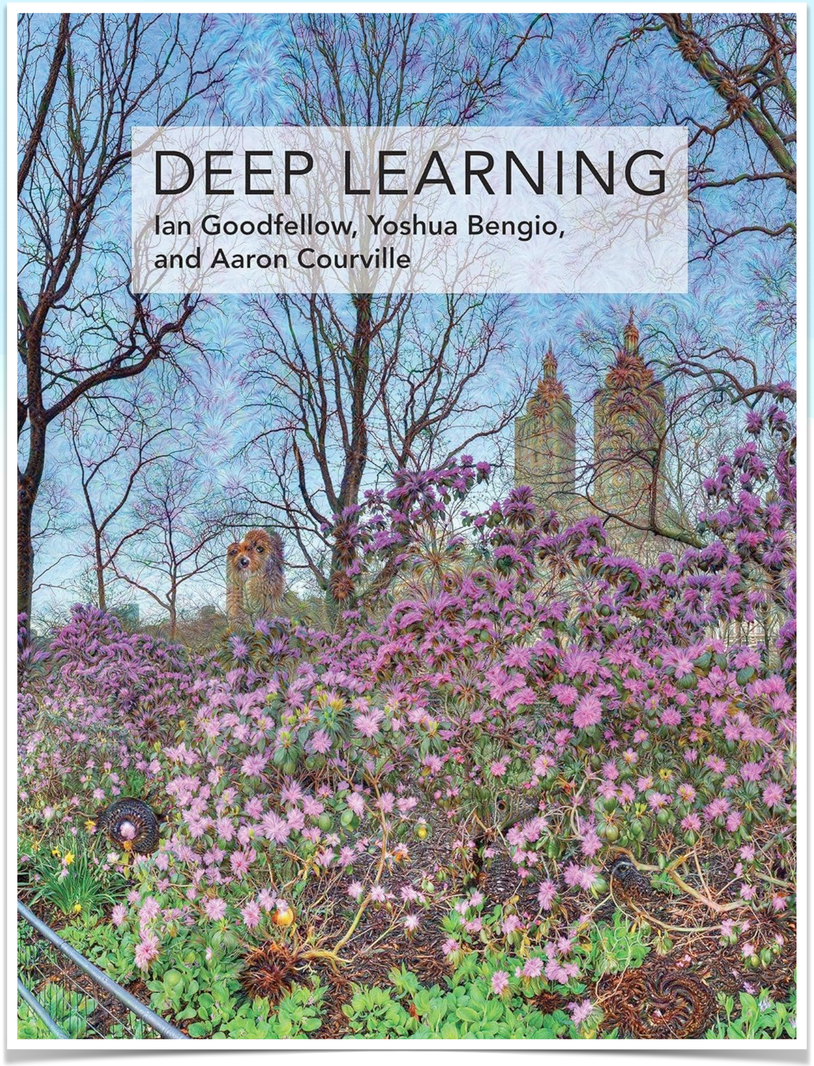
Github: https://github.com/ageron/handson-ml2



Materials & References

Deep Learning

“Written by three experts in the field, Deep Learning is the only comprehensive book on the subject.”
—Elon Musk, cochair of OpenAI; cofounder and CEO of Tesla and SpaceX



Materials & References

Online Materials

- Tensorflow Playground: https://playground.tensorflow.org
- GeeksforGeeks: https://www.geeksforgeeks.org/
- Github: https://github.com/
- Kaggle: https://www.kaggle.com/
- Papers with code: https://paperswithcode.com/

Course Github

https://github.com/Ella-Mousavi/CI-Winter-2025/

My Emails

e.a.mousavi@aut.ac.ir
mousavi.ella@gmail.com


From a Single Neuron to Deep Learning
Your Journey Begins Now!

The Perceptron

Threshold Logic Unit (TLU)

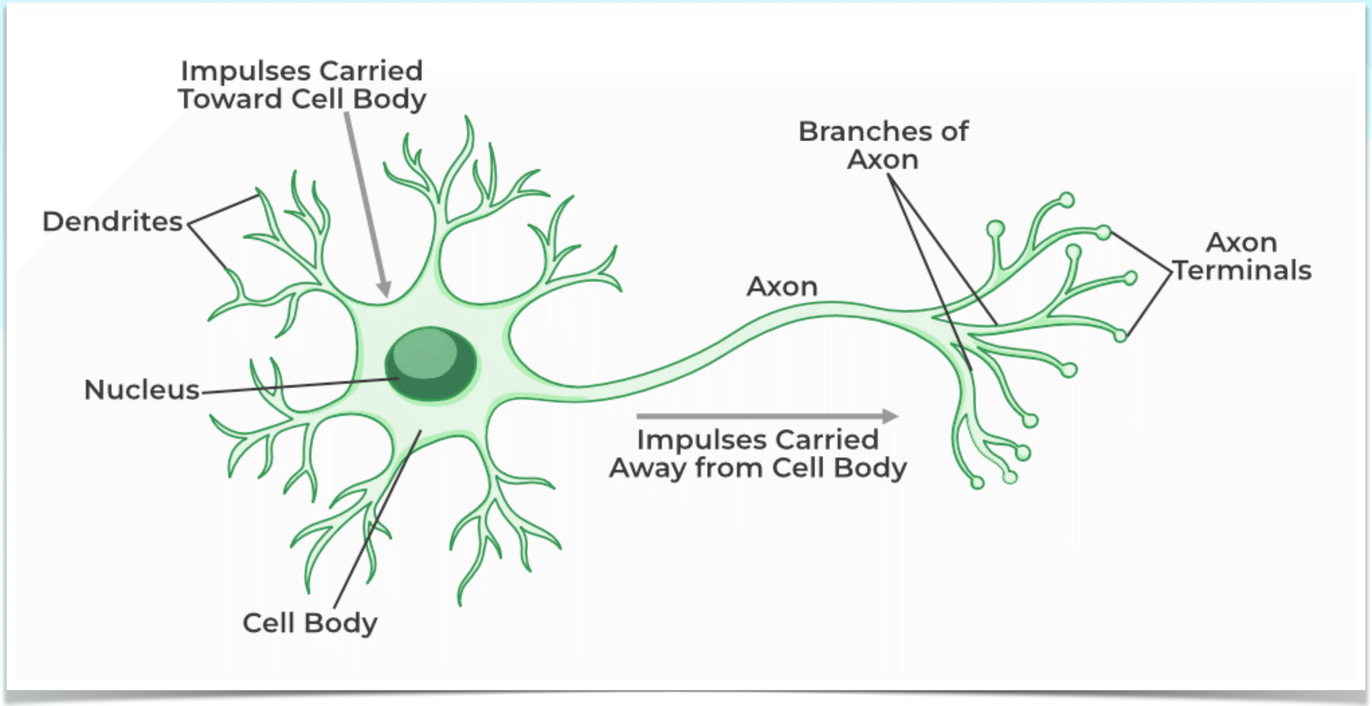


Image from: https://www.geeksforgeeks.org/single-layer-perceptron-in-tensorflow/

The Perceptron

Threshold Logic Unit (TLU)

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$h_w(x) = f(z)$$

$$f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

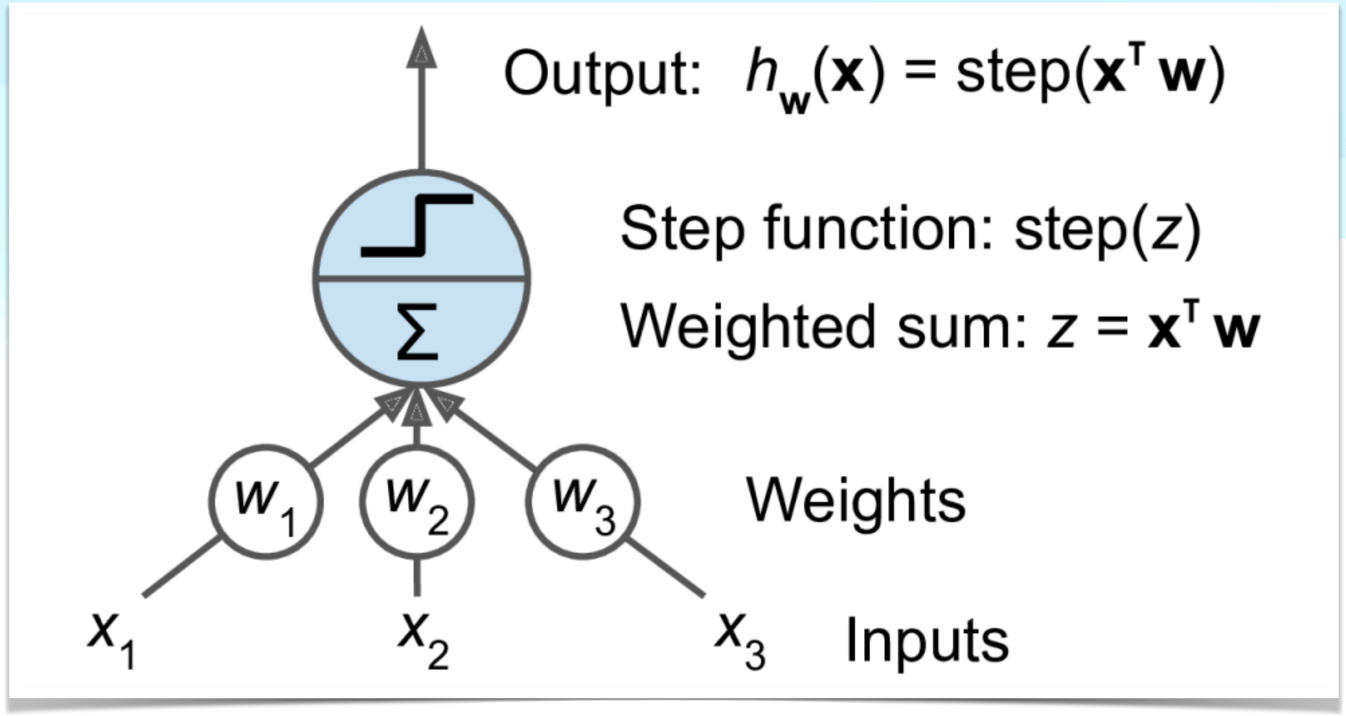


Image from: Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow

Activation Functions in the Perceptron

Standard Step Function

$$f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Sign Function

$$\text{sgn}(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z = 0 \\ -1, & \text{if } z < 0 \end{cases}$$

Image from: Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow

Architecture of a Perceptron

$$y_j = f(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b)$$

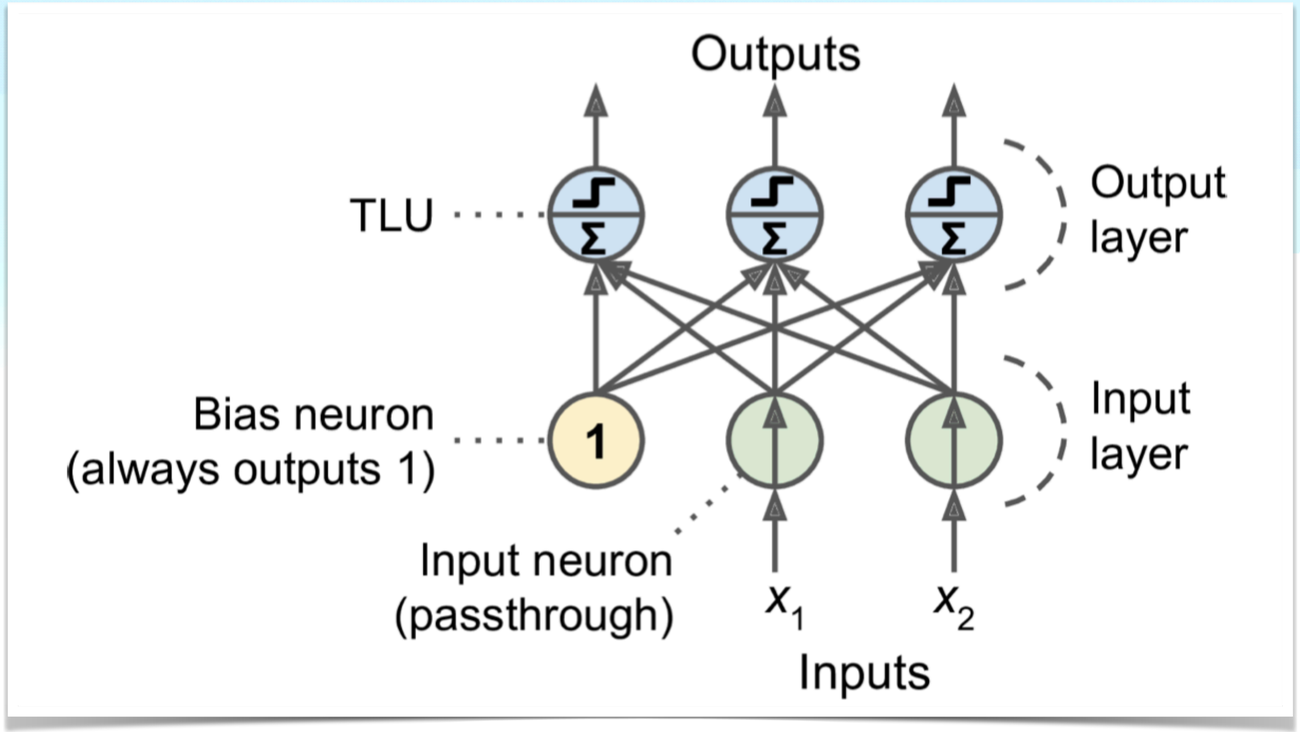


Image from: Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow

Perceptron Learning Rule

How Does a Perceptron Learn?

- Learning is based on adjusting weights using **the perceptron learning rule**:

$$w_{i,j}^{nextstep} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

$w_{i,j}$ → Connection weight between the **ith input** and the **jth output neuron**.

x_i → **ith input value** of the current training instance.

$\hat{y}_j$ → **Predicted output** of the jth output neuron for the current training instance.

y_j → **Target output** of the jth output neuron.

η → **Learning rate**, controls the step size for updates.

Perceptron Convergence Theorem

- The Perceptron Learning Algorithm will converge if the training data is linearly separable.
- Rosenblatt's Perceptron Convergence Theorem states that:
    - If a solution exists, the perceptron **will find it in a finite number of steps**.
    - If the data is **not linearly separable**, the algorithm will **never converge** (it keeps oscillating).

Perceptron Learning Algorithm

```
Initialize weights w randomly
Set learning rate η

Repeat until convergence OR max_iterations reached:

    For each training sample (x, y):
        Compute output: y_hat = step_function(w · x + b)

        If y_hat ≠ y:  # Only update if misclassified
            Update weights: w = w + η (y - y_hat) * x

        else:  # No weight updates in this iteration → Converged
            Stop training
            Break
```

Implementing Perceptron in Python

sklearn.linear_model.Perceptron

class `sklearn.linear_model.`**`Perceptron`**(**, penalty=None, alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, eta0=1.0, n_jobs=None, random_state=0, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False*)

`decision_function`(X)	Predict confidence scores for samples.
`densify`()	Convert coefficient matrix to dense array format.
`fit`(X, y[, coef_init, intercept_init, ...])	Fit linear model with Stochastic Gradient Descent.
`get_metadata_routing`()	Get metadata routing of this object.
`get_params`([deep])	Get parameters for this estimator.
`partial_fit`(X, y[, classes, sample_weight])	Perform one epoch of stochastic gradient descent on given samples.
`predict`(X)	Predict class labels for samples in X.
`score`(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
`set_fit_request`(*[, coef_init, ...])	Request metadata passed to the `fit` method.
`set_params`(**params)	Set the parameters of this estimator.
`set_partial_fit_request`(*[, classes, ...])	Request metadata passed to the `partial_fit` method.
`set_score_request`(*[, sample_weight])	Request metadata passed to the `score` method.
`sparsify`()	Convert coefficient matrix to sparse format.

Unlocking the Full Potential!

“Perceptrons can recognize basic patterns, but they struggle with more intricate structures. By introducing hidden layers and non-linear activation functions, we unlock the true power of neural networks—welcome to Multi-Layer Perceptrons!”

Multi-Layer Perceptron (MLP)

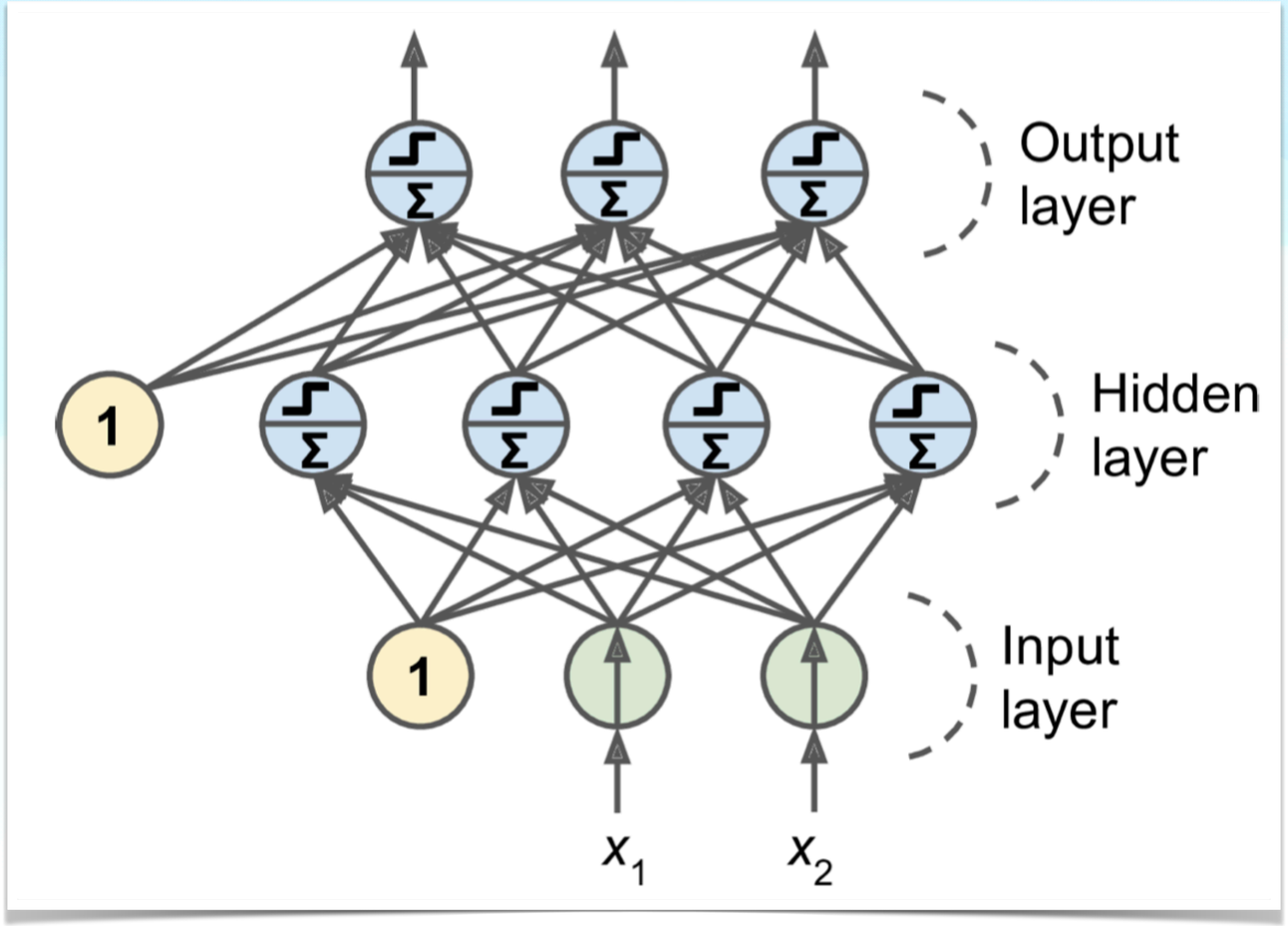


Image from: Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow

Multi-Layer Perceptron (MLP)

The one big issue

How to train it?

Multi-Layer Perceptron (MLP)

What’s coming next

Training MLP via backpropagation algorithm

Introducing advanced activation functions

Introducing Keras API

Building, compiling, training, and evaluating your first deep network

For Next Week

Your turn

- Think of the solutions for “The One Big Issue”
- Install Visual Studio or Conda
- Install Tensorflow and sklearn
- Improve your Python knowledge

And do your first homework :)

GOOD LUCK!