

# COMP3420 — AI for Text and Vision

Week 02 Lecture 1: Machine Learning for Image Classification

Diego Mollá

COMP3420 2023H1

## Abstract

This lecture will focus on the task of image classification by using statistical classifiers. We will focus on key concepts of deep learning, and illustrate the simplest type of deep learning: the feedforward networks.

Update February 24, 2023

## Contents

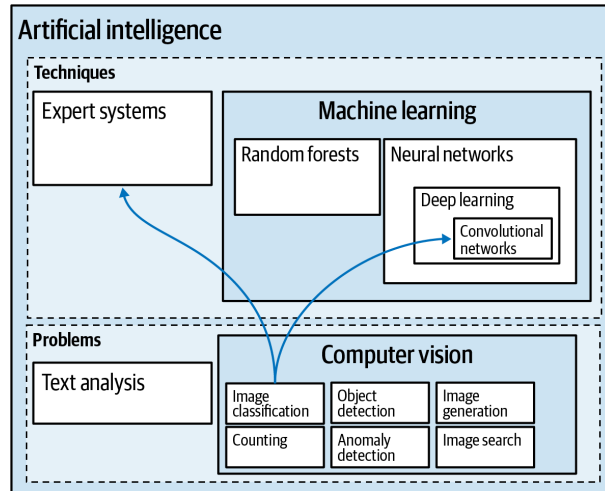
<b>1</b>	<b>Machine Learning for Image Classification</b>	<b>1</b>
<b>2</b>	<b>Deep Learning</b>	<b>5</b>
2.1	A Neural Network . . . . .	5
2.2	Deep Learning . . . . .	10
<b>3</b>	<b>Classification in Keras</b>	<b>11</b>

## Reading

- Deep Learning with Python, 2nd Edition, Chapter 2
- Practical Machine Learning for Computer Vision, Chapters 1& 2

## 1 Machine Learning for Image Classification

Computer Vision as a Subfield of AI



(Figure 1-3 from Lakshmanan et al. (2021))

Caption of Figure 1-3 from book “Practical Deep Learning for Computer Vision”:

Figure 1-3. Computer vision is a subfield of AI that tries to mimic the human visual system; while it used to rely on an expert systems approach, today it’s done with machine learning.

## Image Classification

### What is Image Classification?

Classify images into one of a *fixed predetermined* set of categories.

- The number of categories is predetermined.
- The actual categories are predetermined.
- This task is *not* about detecting objects in the image.

*Example: Classify images of flowers*



## Supervised Machine Learning

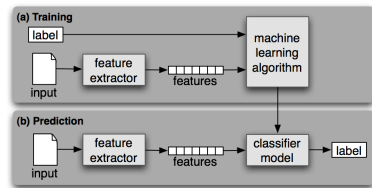
### Given

Training data annotated with class information.

### Goal

Build a *model* which will allow classification of new data.

### Method



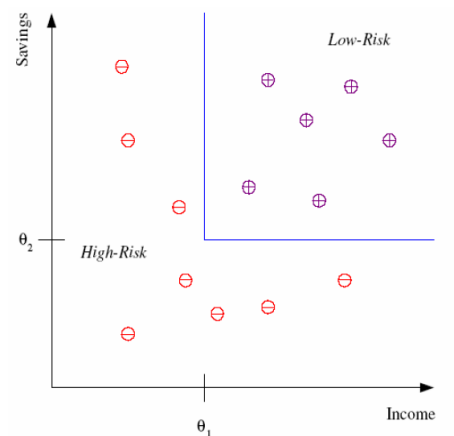
(figure from NLTK book)

- Feature extraction: Convert samples into vectors.
- Training: Automatically learn a model.
- Classification: Apply the model on new data.

(Caption for figure from NLTK book)

Figure 1.1: Supervised Classification. (a) During training, a feature extractor is used to convert each input value to a feature set. These feature sets, which capture the basic information about each input that should be used to classify it, are discussed in the next section. Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model. (b) During prediction, the same feature extractor is used to convert unseen inputs to feature sets. These feature sets are then fed into the model, which generates predicted labels.

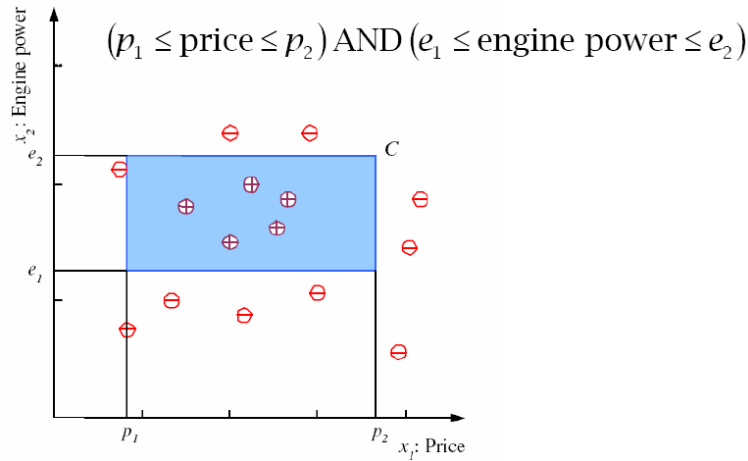
### Supervised Learning Example: Bank Customers



(from Alpaydin (2004))

This is an example of a training dataset where each circle corresponds to one data instance with input values in the corresponding axes and its sign indicates the class. For simplicity, only two customer attributes, income and savings, are taken as input and the two classes are low-risk ('+') and high-risk ('-'). An example model that separates the two types of examples is also shown. The model has two parameters,  $\theta_1$  and  $\theta_2$ , which need to be set. These parameters are either set manually by observing the training data, or automatically by training the model using supervised machine learning. Image from: E. Alpaydin. 2004. Introduction to Machine Learning. ©The MIT Press.

### Supervised Learning Example: Family Cars



(from Alpaydin (2004))

Another example where now the model has four parameters. The class of family car is a rectangle in the price-engine power space. From: E. Alpaydin. 2004. Introduction to Machine Learning. ©The MIT Press.

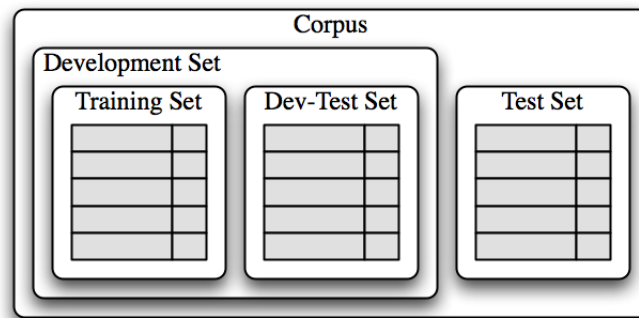
## The Development Set

### Important

Always test your system with data that has not been used for development (Why ...?)

## Development and Test Sets

- Put aside a test set and don't even look at its contents.
- Use the remaining data as a development set.
  - Separate the development set into training and dev-test sets.
  - Use the training set to train the statistical classifiers.
  - Use the dev-test set (also called validation set) to fine-tune the classifiers and conduct error analysis.
  - Use the test set for the final system evaluation once all decisions and fine-tuning have been completed.

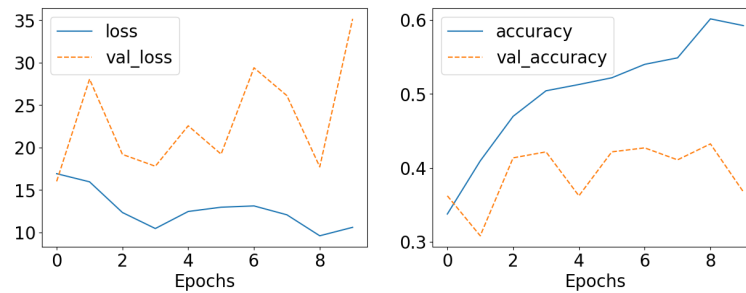


(image from NLTK book)

(Caption of figure from NLTK book)

Figure 1.3: Organization of corpus data for training supervised classifiers. The corpus data is divided into two sets: the development set, and the test set. The development set is often further subdivided into a training set and a dev-test set.

## Identifying Over-fitting



(we will see plots like this in this week's lecture notebooks)

We can observe over-fitting when the evaluation results on the training set is much better than those on the test set. Over-fitting is generally lesser as we increase the size of the training set.

The figure shows the evolution of the loss and accuracy as we increase the number of epochs during training (we will learn about epochs later in this lecture). In this example, the system overfits from about 2 epochs. Overfitting usually increases as we increase the number of training epochs, as we observe in the figures.

We can also see that the accuracy of the training data keeps increasing as we increase the number of training epochs. This is usually the case, but accuracy stalls or may decrease after a number of epochs, in this case after just 2 epochs. This means that the optimum number of epochs needed to train this system is 2.

## 2 Deep Learning

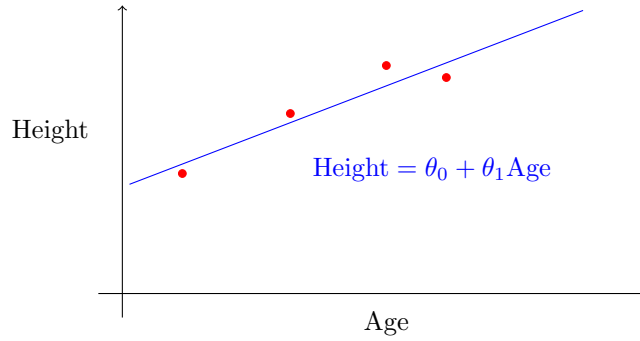
### What is Deep Learning?

- Deep learning is an extension to the neural networks first developed during the late 20th century.
- The main differences between deep learning and the early neural networks are:
  1. A principled manner to combine simple neural network architectures to build complex architectures.
  2. Better algorithms to train the architectures.
- Besides improvements in the theory, three main drivers of the success of deep learning are:
  1. The availability of large training data.
  2. The availability of much faster computers.
  3. Massive parallel methods that use specialised hardware.
    - Graphic Processing Units.

### 2.1 A Neural Network

#### Linear Regression: The Simplest Neural Network

- Linear regression is one of the simplest machine learning methods to predict a numerical outcome.
- For example, we want to predict the height of a person based on its age.
- Based on the training data, linear regression will try to find the line that best fits the training data:



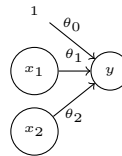
In this example, the neural network needs to learn the values of  $\theta_0$  and  $\theta_1$  that generates the line that best fits the training data.

### Linear Regression with Multiple Variables

- For example, we want to predict the value of a house based on two features:
  - $x_1$  Area in squared metres.
  - $x_2$  Number of bedrooms.
- We can predict the value based on a linear combination of the two features:

$$f(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- Where  $\theta_0, \theta_1, \theta_2$  are learnt during the training stage.



The graphical representation shows how we can represent linear regression as a very simple neural network:

- The circles represent neurons (cells).
- The arrows represent connections between the cells.
- There is a “fixed” neuron that always has the value 1. This neuron is called the “bias term”, and is used so that the formula to compute  $y$  can use  $\theta_0$ :

$$y = \theta_0 1 + \theta_1 x_1 + \theta_2 x_2$$

### Supervised Machine Learning as an Optimisation Problem

- The machine learning approach will attempt to learn the parameters of the learning function that minimise the loss (prediction error) in the training data.

$$\Theta = \operatorname{argmin}_{\Theta} L(X, Y)$$

Where

- $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$  is the training data, and

–  $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(n)}\}$  are the labels of the training data.

- In linear regression:

–  $f(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_p x_p^{(i)}$

–  $L(X, Y) = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - f(x^{(i)}))^2$

This loss is the mean squared error.

The notation of the above samples is explained here:

- $\Theta$  stands for all the parameters  $\theta_0, \theta_1, \theta_2, \dots, \theta_p$ .
- $x^{(1)}$  represents the first sample in the training data,  $x^{(2)}$  represents the second sample, and so on. Each sample is a vector of features, so the sample at position  $i$  is represented as  $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_p^{(i)})$ .
- $y^{(i)}$  represents the label of the sample at position  $i$ , that is, the label of  $x^{(i)}$ .

## Optimisation Problems in Other Approaches



### Logistic Regression


Logistic regression is commonly used for classification

- $f(x^{(i)}) = \frac{1}{1 + e^{-\theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)}}}$
- $L(X, Y) = -\frac{1}{n} \sum_{i=1}^n (y^{(i)} \times \log f(x^{(i)}) + (1 - y^{(i)}) \times \log (1 - f(x^{(i)})))$   
This loss is called cross-entropy.

### Support Vector Machines

Initially, SVM was formulated differently but it can also be seen as:

- $f(x^{(i)}) = \text{sign}(p(x^{(i)}))$   
 $p(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_p x_p^{(i)}$
- $L(X, Y) = \frac{1}{n} \max\{0, 1 - y^{(i)} \times p(x^{(i)})\}$   
This is called the hinge loss.

The symbol  means that the contents of this slide is supplementary and will not be asked in the exam.

The only thing that you need to learn from this slide is that there are multiple types of loss functions, and a particular machine learning problem will need to use the appropriate loss function. We will talk about this further but, in general:

- The mean squared error is used for regression problems.
- Cross-entropy is used in classification problems.
- The hinge loss is used when we want to implement Support Vector Machines.

## Solving the Optimisation Problem



- A common approach to find the minimum of the loss function is to find the value where the gradient of the loss function is zero.
- This results in a system of equations that can be solved.

### System of equations in linear regression

$$\frac{\partial}{\partial \theta_0} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)})^2 = 0$$

$$\frac{\partial}{\partial \theta_1} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)})^2 = 0$$

...

$$\frac{\partial}{\partial \theta_p} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)})^2 = 0$$

## Gradient Descent



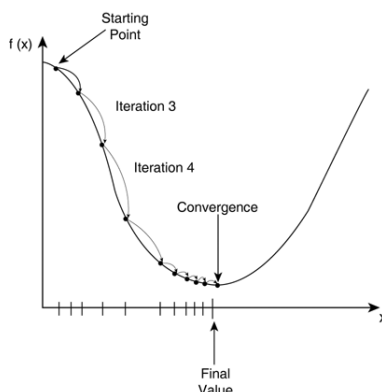
- Solving the system of equations  $\frac{\partial L}{\partial \theta_0} L(X, Y) = 0, \frac{\partial}{\partial \theta_1} L(X, Y) = 0, \dots$  can be too time-consuming.
- e.g. in linear regression, the complexity of computing the formula that solves the system of equations is  $O(n^3)$ .
- Some loss functions are very complex (e.g. in deep learning approaches) and it is not practical to attempt to solve the equations at all.
- Gradient descent is an iterative approach that finds the minimum of the loss function.

## Gradient Descent Algorithm



1. Assign initial random values to  $\theta_0, \dots, \theta_p$
2. Repeat until convergence:

For  $j = 1, 2, \dots, p$ :  
$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X, Y)$$



## Batch Gradient Descent



- There are automated methods to compute the derivatives of many complex loss functions.
  - This made it possible to develop the current deep learning approaches.
- Note, however, that every step of the gradient descent algorithm requires to process the entire training data.
- This is what is called batch gradient descent.

## Mini-batch Gradient Descent



- In mini-batch gradient descent, only part of the training data is used to compute the gradient of the loss function.
- The entire data set is partitioned into small batches, and at each step of the gradient descent iterations, only one batch is processed.
  - If the batch size is 1, this is usually called stochastic gradient descent.



- When all batches are processed, we say that we have completed an epoch and start processing the first batch again.

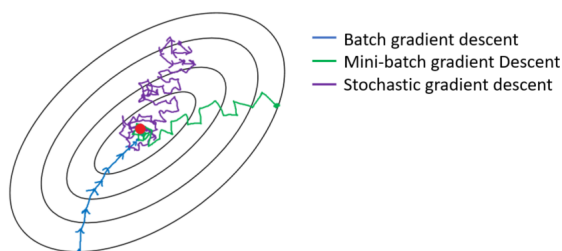


The above figure illustrates a case where the entire training data is partitioned into 4 batches. In this example, an epoch will be completed every 4 iterations of the mini-batch gradient descent algorithm (where an “iteration” means the execution of an iteration of the loop in *emphasis* in the algorithm below).

### Mini-Batch Gradient Descent Algorithm



1.  $\theta_0 = 0, \dots, \theta_p = 0$
2. Repeat until (near) convergence:
  - (a) Shuffle  $(X, Y)$  and split it into  $n$  mini-batches  $(X_0, Y_0), \dots, (X_n, Y_n)$ .
  - (b) For every mini-batch  $(X_i, Y_i)$ :
    - i. For  $j = 1, 2, \dots, p$ :
 
$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X_i, Y_i)$$



<https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3>

The figure shows the trajectories of the parameters  $\Theta$  towards the minimum. In the figure, only two dimensions are shown, and the ovals mean regions of the loss function with equal loss value. In batch gradient descent, the values of  $\Theta$  move directly towards a minimum until the minimum loss is reached, at which point the values are stationary. In mini-batch and stochastic gradient descent, the path towards reaching the minimum is more erratic and the values will not become stationary.

### Batch vs. Mini-Batch Gradient Descent



#### Batch Gradient Descent

- At each iteration step, we take the most direct path towards reaching a minimum.
- The algorithm converges in a relatively small number of steps.
- Each step may take long to compute (if the training data is large).

#### Mini-batch Gradient Descent

- At each iteration step, some random noise is introduced and we take a path roughly in the direction towards the minimum.
- The algorithm reaches near convergence in a larger number of steps.
- Each step is very quick to compute.

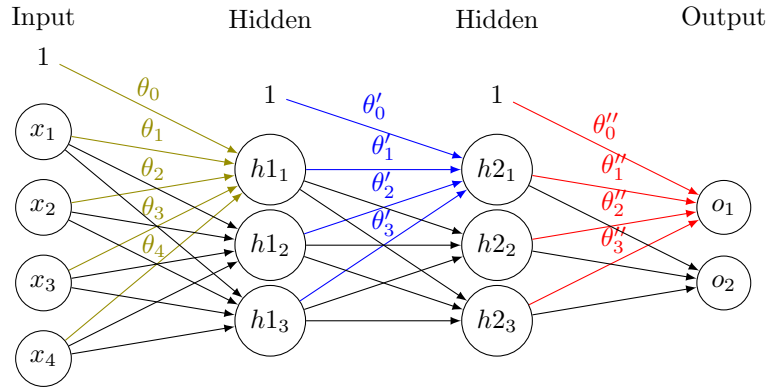
## 2.2 Deep Learning

### A Deep Learning Architecture

- A deep learning architecture is a large neural network.
- The principle is the same as with a simple neural network:
  1. Define a complex network that generates a complex prediction  $f(x_1, x_2, \dots, x_p)$ . This is normally based on simpler building blocks.
  2. Define a loss function  $L(X, Y)$ . There are some popular loss functions for classification, regression, etc.
  3. Determine the gradient of the loss function. This is done automatically.

#### A feedforward neural network

*a.k.a. multilayer perceptron (MLP)*



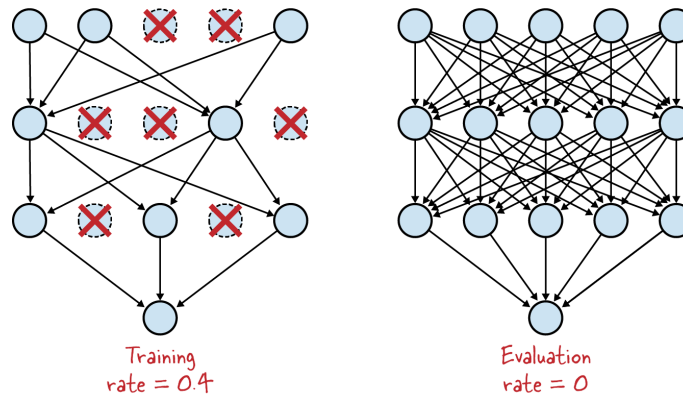
- $h1_1 = f_{h11}(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4)$
- $h2_1 = f_{h21}(\theta'_0 + \theta'_1 h1_1 + \theta'_2 h1_2 + \theta'_3 h1_3)$
- $o_1 = f_{o1}(\theta''_0 + \theta''_1 h2_1 + \theta''_2 h2_2 + \theta''_3 h2_3)$

A feedforward neural network is one of the earliest deep learning architectures:

- The network is composed of a sequence of densely connected layers.
- The first layer is the input layer and it receives the vector that represents the input data.
- The last layer is the output layer and it has one node per classification label.
- There may be one or more intermediate layers called hidden layers.
- The layers are densely connected: each node in a layer is connected to each node in the next layer. The output of a node is a function of the weighted sum of all nodes from the previous layer, where the weights are the parameters that will be learned at the training stage.
- The function that applies to a node is called the *activation function*. There are several popular activation functions. We will see some of them later.

## Dropout

This is a simple and effective technique to combat overfitting.



(Figure 2-22 from Lakshmanan et al. (2021))

Dropout is one of the oldest regularisation techniques and it is useful when the model has many parameters and/or the data set is small, and there is risk of overfitting. The approach is very simple: during training, “drop” neurons randomly according to a pre-set dropout rate. In practice, the dropped neuron’s outputs are set to zero. Different neurons will be dropped at each training iteration. During testing, all neurons are used. This forces the network to include some redundancy in the remaining neurons, and this in turn makes the network be more resilient against overfitting.

In the figure, there are three feedforward layers that have dropout with a dropout rate of 0.4. Dropout can be applied to only some of the layers, and there may be different dropout rates for each layer.

## 3 Classification in Keras

### Classification in Keras

- This section is based on jupyter notebooks provided by the unit textbooks.
- Study these notebooks carefully since they contain important information about how neural networks are constructed and how they operate.
- The notebooks also introduce important terminology that you need to understand.

### Take-home Messages

1. Explain and demonstrate the need for separate training and test set.
2. Using Keras, implement image classifiers.
3. Detect over-fitting.
4. Perform hyperparameter fine-tuning.

### What’s Next

#### Week 3

- Convolutional networks for image classification.
- Deadline assignment 1.

## Reading

- Computer Vision book, chapter 3.
- Deep Learning book, chapter 8.