COMP3420 — AI for Text and Vision

Week 02 Lecture 1: Machine Learning for Image Classification

Diego Mollá

Department of Computer Science
Macquarie University

COMP3420 2023H1

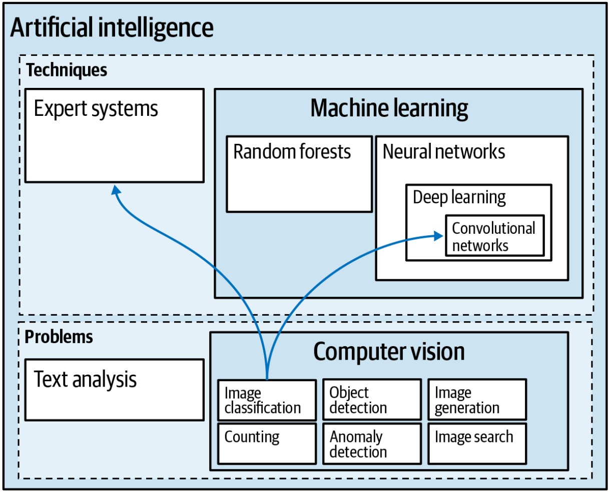
Programme

Reading

- Deep Learning book, Chapter 2
- Computer Vision book, Chapters 1 & 2

Programme

Computer Vision as a Subfield of AI



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

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

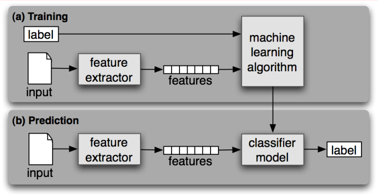
Given

Training data annotated with class information.

Goal

Build a model which will allow classification of new data.

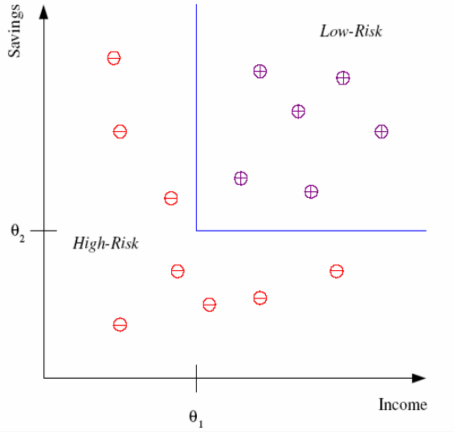
Method



(figure from NLTK book)

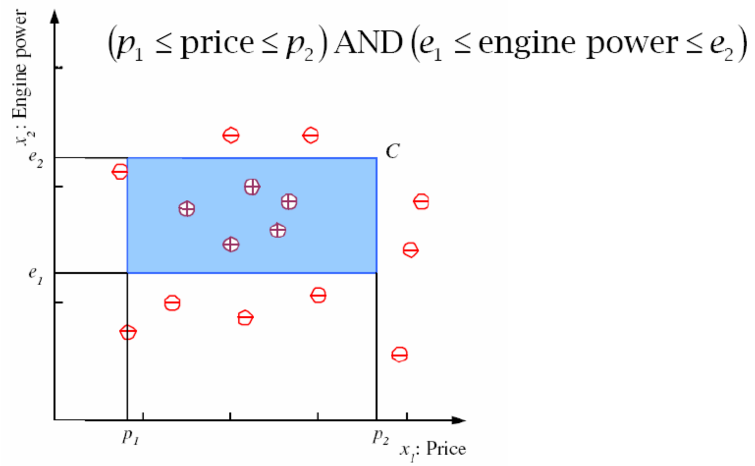
- Feature extraction: Convert samples into vectors.
- Training: Automatically learn a model.
- Classification: Apply the model on new data.

Supervised Learning Example: Bank Customers



(from Alpaydin (2004))

Supervised Learning Example: Family Cars



(from Alpaydin (2004))

The Development Set I

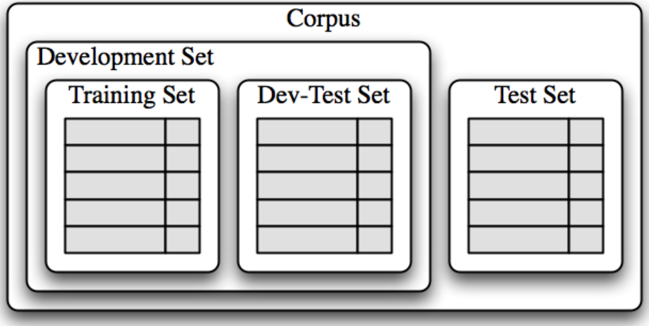
Important

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

Development and Test Sets

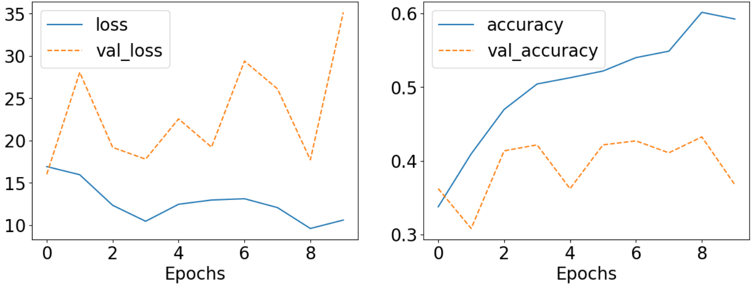
- Put aside a test set and don't even look at its contents.
- Use the remaining data as a development set.
  - Separate the development set into training and dev-test sets.
  - Use the training set to train the statistical classifiers.
  - Use the dev-test set (also called validation set) to fine-tune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

The Development Set II



(image from NLTK book)

Identifying Over-fitting



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

Programme

1. Machine Learning for Image Classification
2. Deep Learning
    - A Neural Network
    - Deep Learning
3. Classification in Keras

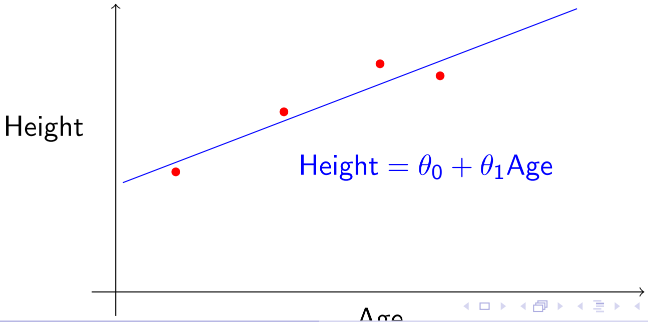
What is Deep Learning?

- 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.

Programme

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:

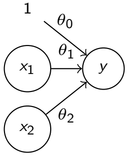


Linear Regression with Multiple Variables

- 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.



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.

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

Where

  - $X = \{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}$ is the training data, and
  - $Y = \{y^{(1)}, y^{(2)}, \cdots, y^{(n)}\}$ are the labels of the training data.

- In linear regression:
  - $f(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_p x_p^{(i)}$
  - $L(X, Y) = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2$
    This loss is the mean squared error.

Optimisation Problems in Other Approaches

Logistic Regression

Logistic regression is commonly used for classification

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

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

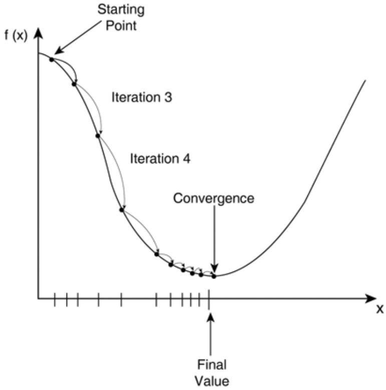
$$\frac{\partial}{\partial\theta_0} 1/n \sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \cdots - \theta_p x_p^{(i)})^2 = 0$$
$$\frac{\partial}{\partial\theta_1} 1/n \sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \cdots - \theta_p x_p^{(i)})^2 = 0$$
$$\ldots$$
$$\frac{\partial}{\partial\theta_p}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

Gradient Descent

- Solving the system of equations $\frac{\partial L}{\partial \theta_0} L(X, Y) = 0, \frac{\partial}{\partial \theta_1} L(X, Y) = 0, \ldots$ 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 $\theta_0, \ldots, \theta_p$
2. Repeat until convergence:
   For $j = 1, 2, \cdots p$:
   $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X, Y)$

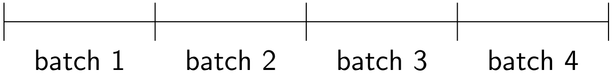


Batch Gradient Descent

- 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.

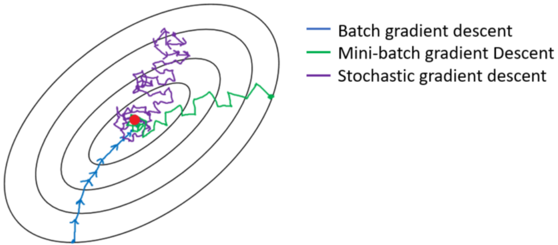
Mini-batch Gradient Descent

- 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.



Mini-Batch Gradient Descent Algorithm

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



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.

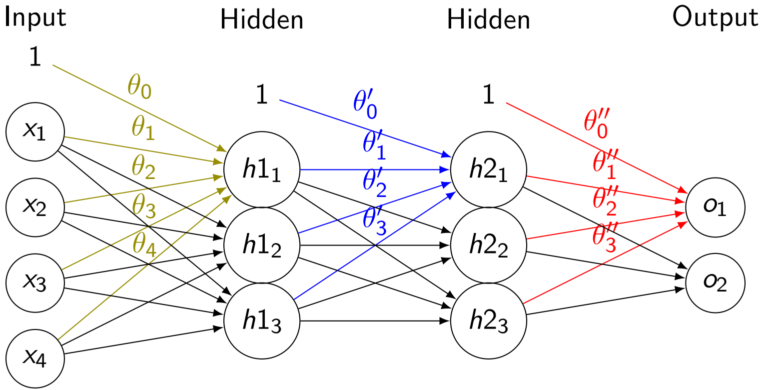
Programme

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 $f(x_1, x_2, \cdots, x_p)$. 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

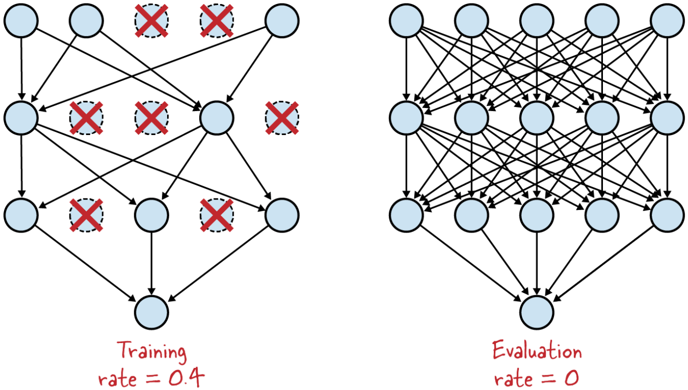
a.k.a. multilayer perceptron (MLP)



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

Dropout

This is a simple and effective technique to combat overfitting.



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

Programme

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.