CS4618: Artificial Intelligence I

Prediction

Derek Bridge
School of Computer Science and Information Technology
University College Cork

Initialization

In [1]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Prediction

- We want to create programs that make **predictions**
  - (In everyday use, prediction is about the future; we use the word more generally in AI)
- Generically, we refer to such programs as **estimators**
- There are two main types of prediction, and hence two types of estimator:

In both, we are given a vector $\boldsymbol{x}$ of feature values that describes some object:

- **Regression** means predicting a **target value**, which is numeric (real-valued)
  - e.g. given a vector of feature values that describe a house, predict the selling price of the house
- **Classification** means predicting the object's **class** from a finite (and usually small) set of classes
  - e.g. given a vector of feature values that describe an email, predict whether the email is spam or ham

Notation

- We continue to use $\boldsymbol{x}$ for an object
- We will use y for the target value (in regression) or class label (in classification)
- Actually, we will be even more precise:
  - We will use y for the *actual* target value/class label
  - We will use $\hat{y}$ for a *predicted* target value/class label

There's more to say about classification

- Classification means predicting an object's class from a finite (and usually small) set of classes
  - We assume we have a finite set of **labels**, C, one per class
  - Given an object $\boldsymbol{x}$, our task is to assign one of the labels $\hat{y} \in C$ to the object.
- We will often use integers for the labels
  - E.g. given an email, a spam filter predicts $\hat{y} \in \{0, 1\}$, where 0 means ham and 1 means spam
  - But a classifier should not treat these as continuous, e.g. it should never output 0.5
  - Furthermore, where there are more than two labels, we should not assume a relationship between the labels
    - Suppose there are three classes $\{1, 2, 3\}$
    - Suppose we are classifying object $\boldsymbol{x}$ and we happen to know that its *actual* class label is $y = 3$
    - One classifier predicts $\hat{y} = 1$
    - Another classifier predicts $\hat{y} = 2$
    - Which classifier has done better?

A variation of classification

- Given an object $\boldsymbol{x}$, a classifier outputs a label, $\hat{y} \in C$
- Instead, a classifer could output a probability distribution over the labels C
  - E.g. given an email $\boldsymbol{x}$, a spam filter might output $\langle 0.2, 0.8 \rangle$ meaning $P(y = ham \mid \boldsymbol{x}) = 0.2$ and $P(y = spam \mid \boldsymbol{x}) = 0.8$
  - The probabilities must sum to 1.
- We can convert such a classifier into a more traditional one by taking the probability distribution and selecting the class with the highest probability:

$$\arg\max_{\hat{y} \in C} P(\hat{y} \mid \boldsymbol{x})$$

Types of Classification

- We distinguish two types of classification:
  - **Binary classification**, in which there are just two classes, i.e. $|C| = 2$, e.g. fail/pass, ham/spam, benign/malignant
  - **Multiclass classification**, where there are more than two classes, i.e. $|C| > 2$, e.g. let's say that a post to a forum or discussion board can be a question, an answer, a clarification or an irrelevance
- In fact, there are even more types of classification, but we will not be studying them further:
  - In **multilabel classification**, the classifier can assign $\boldsymbol{x}$ to more than one class
    - I.e. it outputs a *set* of labels, $\hat{y} \subseteq C$.
    - E.g. consider a movie classifier where the classes are genres, e.g. $C = \{comedy, action, horror, musical, romance\}$
    - The classifier's output for *The Blues Brothers* should be $\{comedy, action, musical\}$.

    Do **not** confuse this with *multiclass* classification
  - In **ordered classification**, there is an *ordering* defined on the classes
    - The ordering matters in measuring the performance of the classifier
    - E.g. consider a classifier that predicts a student's degree class, i.e. $C = \{Ordinary, 3rd, 2ii, 2i, 1st\}$
    - Suppose for student $\boldsymbol{x}$, the actual class $y = 1st$
    - One classifier predicts $\hat{y} = 2ii$
    - Another classifier predicts $\hat{y} = 2i$
    - Which classifier has done better?

We need to say more about binary classification

- In binary classification, there are two classes
- It is common to refer to one class (the one labelled 0) as the **negative class** and the other (the one labelled 1) as the **positive class**
- It doesn't really matter which is which
  - But, usually, we treat the class we're trying to identify, or the class that requires special action, as the positive class
  - E.g. in spam filtering, ham is the negative class; spam is the positive class
  - What about tumour classification?
- (This terminology is extended to other things too, e.g. we can refer to **negative examples** and **positive examples**)

Class exercises

- Consider:
  - Predicting tomorrow's rainfall
  - Predicting whether we will have a white Christmas
  - Predicting the sentiment of a tweet (negative, neutral or positive)
  - Predicting a person's sexual orientation
  - Predicting a person's opinion of a movie on a rating scale of 1 star (rotten) to 5 stars (fab)
- Answer the following:
  - Which are regression and which classification?
  - If classification, which are binary and which are multiclass?
  - If binary, which is the positive class and which the negative?
- Clustering and classification are easily confused: in both, we 'assign objects to groups'
  What is the *key* difference between them?

Building a model: ask an expert

- So how, for example, do we build a regression system that can predict the selling price of your house?
- We ask Ann
  - She's an auctioneer — an expert at predicting Cork city house prices
  - But we don't ask her to predict your house price
  - We ask her for a *general* method for predicting Cork house prices
- She tells us that her rule-of-thumb is that prices start at 25k€ and increase by 1.5k€ for every extra square metre of floor area:

$$y = 25 + 1.5 flarea$$

  - So, she predicts your house (floor area of 114 square metres) will sell for
    $\hat{y} = 25 + 1.5 \times 114 = 196$k€

Models

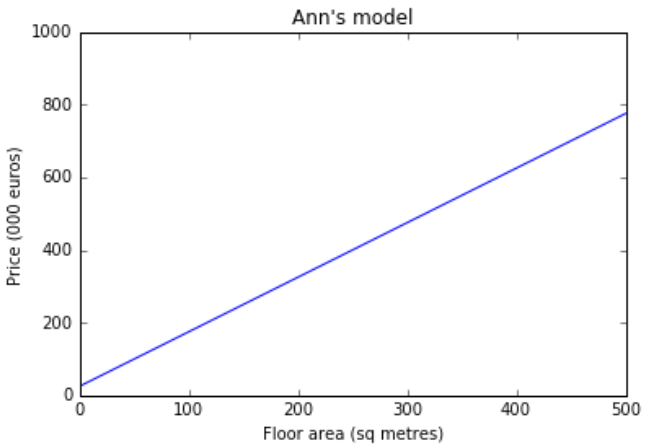
- Ann has given us a **model**
- In very abstract terms, a model is an approximation of some part of reality that enables us to make predictions about that reality
- In very concrete terms for this module, a model is a formula (or function or procedure or set of rules…) that expresses the relationship between the the thing being predicted (target value or class) and the features
- (It so happens that Ann's is a **linear model** — see future lecture)

In [3]:
```
# Ann's model
def f_ann(flarea):
    return 25 + 1.5 * flarea
```

In [4]:
```
# Predicting the selling price of your house
f_ann(114)
```

Out[4]: 196.0

In [5]:
```
# Plotting the predictions made by Ann's model
fig = plt.figure()
plt.title("Ann's model")
xvals = np.linspace(0, 500, 2)
plt.plot(xvals, f_ann(xvals))
plt.xlabel("Floor area (sq metres)")
plt.xlim(0, 500)
plt.ylabel("Price (000 euros)")
plt.ylim(0, 1000)
plt.show()
```



Ben's Model

- We might also ask Ben, another Cork auctioneer, and he might give us a different model, e.g.

$$y = 20 + 50bdrms + 10bdrms^2$$

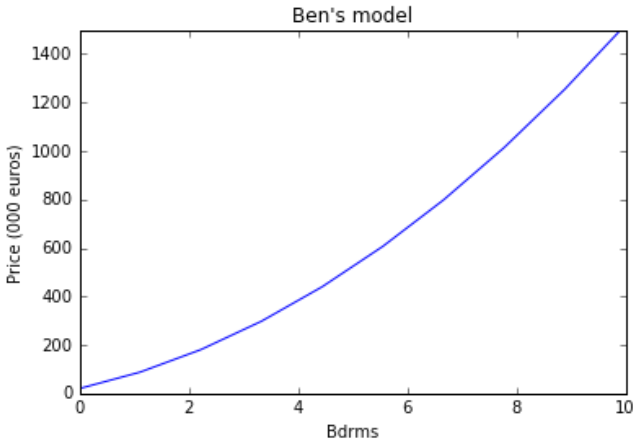
- (Ben' model is not a linear model — see future lecture)

In [6]:
```
# Ben's model
def f_ben(bdrms):
    return 20 + 50 * bdrms + 10 * bdrms ** 2
```

In [7]:
```
# Predicting the selling price of your house
f_ben(3)
```

Out[7]:
```
260
```

In [8]:
```
# Plotting the predictions made by Ben's model
fig = plt.figure()
plt.title("Ben's model")
xvals = np.linspace(0, 10, 10)
plt.plot(xvals, f_ben(xvals))
plt.xlabel("Bdrms")
plt.xlim(0, 10)
plt.ylabel("Price (000 euros)")
plt.ylim(0, 1500)
plt.show()
```



Which Model is Better?

- Ann's and Ben's models make different predictions
  - Ann predicts your house will sell for $y = 25 + 1.5 \times 114 = 196$k€
  - Ben predicts your house will sell for $y = 20 + 50 \times 3 + 10 \times 3^2 = 260$k€
- So we might ask: which is better?
  - A complicated question — to be explored in this module and the next module
  - For now, suppose your house sells for 210000€. Ann's prediction (196000€) is closer than Ben's (260000€), so we have some evidence that Ann's model is better

Building a model: learn from data

- Rather than ask an expert, we want to **learn** a model from data
- Suppose we collect a dataset of **labeled examples**
  - Each example describes a house by giving values for the various features
  - But now, also, each example gives the *actual* selling price of the house
- We take some or all of these examples, call them the **training set**, and give them to the learning algorithm
- As best it can, the learning algorithm finds a model based on the labeled examples in the training set

Datasets of labeled examples

- As before: m examples, n features
- But a **labeled example** is a *pair*, comprising a vector of feature values and the value of the target (regression) or the class label (classification)

$$\langle \boldsymbol{x}, y \rangle$$

- So a **labeled dataset** looks like this:

$$\{\langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle, \langle \boldsymbol{x}^{(2)}, y^{(2)} \rangle, \ldots, \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle\}$$

  - E.g. regression: features are floor area, bedrooms and bathrooms; target is selling price (thousands of €)

$$\{\langle \begin{bmatrix} 92.9 \\ 3 \\ 2 \end{bmatrix}, 175 \rangle, \langle \begin{bmatrix} 171.9 \\ 4 \\ 3 \end{bmatrix}, 435 \rangle, \langle \begin{bmatrix} 79 \\ 3 \\ 1 \end{bmatrix}, 85 \rangle \cdots\}$$

  - E.g. classification: features are lecture and lab attendance (%) and CAO points; class labels are 0 = pass, 1 = fail

$$\{\langle \begin{bmatrix} 60 \\ 45 \\ 500 \end{bmatrix}, 1 \rangle, \langle \begin{bmatrix} 20 \\ 80 \\ 350 \end{bmatrix}, 0 \rangle, \langle \begin{bmatrix} 90 \\ 70 \\ 400 \end{bmatrix}, 0 \rangle \cdots\}$$

- From a labeled datset, we can construct a matrix $\boldsymbol{X}$ *and* a vector $\boldsymbol{y}$ as follows:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^{(1)} & \boldsymbol{x}_2^{(1)} & \ldots & \boldsymbol{x}_n^{(1)} \\ \boldsymbol{x}_1^{(2)} & \boldsymbol{x}_2^{(2)} & \ldots & \boldsymbol{x}_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{x}_1^{(m)} & \boldsymbol{x}_2^{(m)} & \ldots & \boldsymbol{x}_n^{(m)} \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

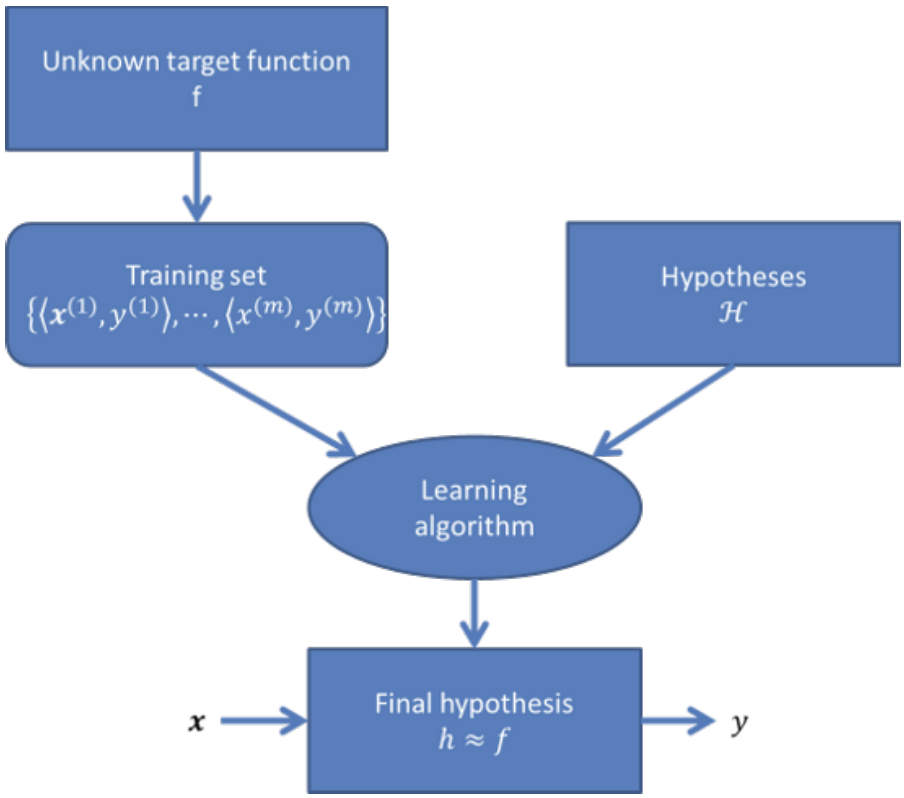
  - In the matrix rows are examples, columns are features
  - The vector gives corresponding target values/class labels
  - E.g

$$\boldsymbol{X} = \begin{bmatrix} 92.9 & 3 & 2 \\ 171.9 & 4 & 3 \\ 79 & 3 & 1 \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} 175 \\ 435 \\ 85 \end{bmatrix}$$

  - E.g.

$$\boldsymbol{X} = \begin{bmatrix} 60 & 45 & 500 \\ 20 & 80 & 350 \\ 90 & 70 & 400 \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Learning from data



Terminology

- We will say that the algorithm **learns** a model
- We could also say that we are **training** the algorithm on the data
- We could also say that the algorithm **fits** a model to the training set
- We could also call it **function approximation**

Types of Learning in AI

- **Reinforcement learning**:
  - The agent receives rewards (or punishments) after executing actions
  - The rewards (or punishments) act as positive (or negative) reinforcement
  - The agent learns a policy that defines which actions to perform in which situations to maximize reward over time
- **Unsupervised learning**:
  - The agent learns from an unlabeled dataset
  - The goal is to find structure within the dataset
  - Clustering and most forms of dimensionality reduction are examples of unsupervised learning but there are other examples of unsupervised learning (not covered) such as anomaly detection and association rule mining
- **Supervised learning**:
  - The agent learns from a labeled dataset
  - The goal is to generalise from the labeled dataset to learn how to predict target values/class labels when given feature values
  - Learning models for regression and classification are examples of supervised learning
- **Semisupervised learning**:
  - The agent learns from a dataset, only a (small) subset of which is labeled
  - The goal is usually the same as in supervised learning but making use of the unlabeled data to compensate for the low volume of labeled data

In [ ]: