Examples of Predicting A Number

* Predict price of a house given its location, floor area, #rooms etc
* Predict income of a person given their age, gender, handset model, web pages browsed
* Predict temperature outside tomorrow given weather forecast and past measurements at your location
* Predict whether distance between mobile handsets given Bluetooth received signal strength, handset models, type of location (house, bus, office, supermarket etc), whether handsets in a pocket/bag or not
Data http://www-bcf.usc.edu/~gareth/ISL/data.html

Data consists of the advertising budgets for three media (TV, radio and newspapers) and the overall sales in 200 different markets.

<table>
<thead>
<tr>
<th>TV</th>
<th>Radio</th>
<th>Newspaper</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>230.1</td>
<td>37.8</td>
<td>69.2</td>
<td>22.1</td>
</tr>
<tr>
<td>44.5</td>
<td>39.3</td>
<td>45.1</td>
<td>10.4</td>
</tr>
<tr>
<td>17.2</td>
<td>45.9</td>
<td>69.3</td>
<td>9.3</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Example: Advertising Data

Suppose we want to predict sales in a new area?

Predict sales when the TV advertising budget is increased to 350?

... Draw a line that fits through the data points
**Some Notation**

Training data:

<table>
<thead>
<tr>
<th>TV ($x$)</th>
<th>Sales ($y$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>230.1</td>
<td>22.1</td>
</tr>
<tr>
<td>44.5</td>
<td>10.4</td>
</tr>
<tr>
<td>17.2</td>
<td>9.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- $m$=number of training examples
- $x$=“input” variable/features
- $y$=“output” variable/“target” variable
- $(x^{(i)}, y^{(i)})$ the $i$th training example
- $x^{(1)} = 230.1$, $y^{(1)} = 22.1$
- $x^{(2)} = 44.5$, $y^{(2)} = 10.4$
**Model**

- Prediction: \( \hat{y} = h_\theta(x) = \theta_0 + \theta_1 x \)
- \( \theta_0, \theta_1 \) are (unknown) parameters
- Often abbreviate \( h_\theta(x) \) to \( h(x) \)

\[
\begin{align*}
\theta_0 &= 15, \quad \theta_1 = 0 \\
\theta_0 &= 0, \quad \theta_1 = 0.1 \\
\theta_0 &= 15, \quad \theta_1 = 0.1
\end{align*}
\]
Cost Function: How to choose model parameters $\theta$?

- Prediction: $\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$

- Idea: Choose $\theta_0$ and $\theta_1$ so that $h_\theta(x^{(i)})$ is close to $y^{(i)}$ for each of our training examples $(x^{(i)}, y^{(i)}), i = 1, \ldots, m$.

- Least squares case: select the values for $\theta_0$ and $\theta_1$ that minimise cost function:

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Note: cost function is a sum over prediction error at each training point so can also write as

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} l_i(\theta_0, \theta_1)$$

where $l_i(\theta_0, \theta_1) = (h_\theta(x^{(i)}) - y^{(i)})^2$. 
Suppose our training data consists of just two observations: 
(3, 1), (2, 1), and to keep things simple we know that $\theta_0 = 0$.

The cost function is

$$\frac{1}{2} \sum_{j=1}^{2} (y^{(j)} + \theta_1 x^{(j)})^2 = \frac{1}{2} (1 - 3\theta_1)^2 + (1 - 2\theta_1)^2$$

What value of $\theta_1$ minimises $(1 - 3\theta_1)^2 + (1 - 2\theta_1)^2$?
Example: Advertising Data

The diagram depicts a 3D plot of the function $J(\theta_0, \theta_1)$, where $J$ represents the cost function in a linear regression model. The axes represent $\theta_0$ and $\theta_1$, with the z-axis showing the value of $J$ multiplied by $10^5$. The plot shows how the cost function changes with different values of $\theta_0$ and $\theta_1$. The peaks and valleys indicate the optimization landscape for finding the minimum cost, which corresponds to the best fit of the model to the data.
**Example: Advertising Data**

- Least square linear fit
- Residuals are the difference between the value predicted by the fit and the measured value.
  - Do the residuals look “random” or do they have some “structure” \(^*\)? Is our model satisfactory?  
- We can use the residuals to estimate a confidence interval for the prediction made by our linear fit.
Summary: Linear Regression With One Feature

* Feature: $x$
* Linear Model: $h_\theta(x) = \theta_0 + \theta_1 x$
* Parameters: $\theta_0$, $\theta_1$
* Cost Function: $J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
* Optimisation: Select $\theta_0$ and $\theta_1$ that minimise $J(\theta_0, \theta_1)$ "least squares" – why?
Gradient Descent

Need to select $\theta_0$ and $\theta_1$ that minimise $J(\theta_0, \theta_1)$. Brute force search over pairs of values of $\theta_0$ and $\theta_1$ is inefficient, can we be smarter?

* Start with some $\theta_0$ and $\theta_1$

* Repeat:
  
  Update $\theta_0$ and $\theta_1$ to new value which makes $J(\theta_0, \theta_1)$ smaller

* When curve is “bowl shaped” or convex then this must eventually find the minimum.
Gradient Descent

* Start with some $\theta_0$ and $\theta_1$
* Repeat:
  Update $\theta_0$ and $\theta_1$ to new value which makes $J(\theta_0, \theta_1)$ smaller
* When curve has several minima then we can’t be sure which we will converge to.
* Might converge to a local minimum, not the global minimum
Gradient Descent

Repeat:

\[ \delta_0 := -\alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1), \quad \delta_1 := -\alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \]

\[ \theta_0 := \theta_0 + \delta_0, \quad \theta_1 := \theta_1 + \delta_1 \]

\(\alpha\) is called the *step size* or *learning rate*, its value needs to be selected appropriately (not too large, not too small).

Why does this work?

* \[ \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \approx \frac{J(\theta_0 + \delta_0, \theta_1) - J(\theta_0, \theta_1)}{\delta_0} \] for \(\delta_0\) sufficiently small.

* \[ J(\theta_0 + \delta_0, \theta_1) \approx J(\theta_0, \theta_1) + \delta_0 \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \]

* When \(\delta_0 = -\alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)\) then

\[ J(\theta_0 + \delta_0, \theta_1) \approx J(\theta_0, \theta_1) - \alpha \left( \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \right)^2 \]
For $J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})^2$ with $h_{\theta}(x) = \theta_0 + \theta_1 x$:

* $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{2}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})$

* $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{2}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$

So gradient descent algorithm is:

* repeat:

  $\delta_0 := -\frac{2}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})$

  $\delta_1 := -\frac{2}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$

  $\theta_0 := \theta_0 + \delta_0$, $\theta_1 := \theta_1 + \delta_1$
Practicalities: Normalising Data

* When using gradient descent (and also more generally) its a good idea to **normalise** your data i.e. scale and shift the inputs and outputs so that they lie roughly between 0 → 1 or -1 → 1.
* It's ok if data range spans 1 → 100, problem is when range is very large e.g. 1 → 10^6 → large ranges (i) mess up numerics, (ii) larger valued data tends to dominate cost function and training focuses on that data.
Commonly replace $x_j$ with $\frac{x_j - \mu_j}{\sigma_j}$ where:

* Shift $\mu_j = \frac{1}{m} \sum_{i=1}^{n} x_j^{(i)}$ tries to make features have approximately zero mean (do not apply to $x_0 = 1$ though)

* Scaling factor $\sigma_j = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu)^2}$ tries to make features mostly lie between -1 and 1 or can use $\sigma_j = max(x_j) - min(x_j)$

* E.g. in advertising data TV budget values lie in range 0.7 to 296.4 with mean 147, so rescaling as $\frac{TV - 147}{296}$ gives a feature with values in interval $-0.5 \leq x_1 \leq 0.5$
Practicalities: When to stop?

- “Debugging”: How to make sure gradient descent is working correctly → \( J(\theta) \) should decrease after every iteration
- Stopping criteria: stop when decreases by less than e.g. \( 10^{-2} \) or after a fixed number of iterations e.g. 200, whichever comes first.
Selecting step size $\alpha$ too small will mean it takes a long time to converge to minimum.

But selecting $\alpha$ too large can lead to us overshooting the minimum.

We need to adjust $\alpha$ so that algorithm converges in a reasonable time.

There are also many automated approaches for adjusting $\alpha$ at each iteration. E.g. using line search (at each gradient descent iteration try several values of $\alpha$ until find one that causes descent).
We’ll use python and usually sklearn
https://scikit-learn.org/stable/index.html in examples and assignments

Sometimes you’ll be asked to implement things from scratch rather than calling the sklearn function → to help you understand what’s happening “under the hood”

Linear regression:

```python
typing numpy as np
Xtrain = np.arrangel(0,1,0.01).reshape(-1, 1)
ytrain = 10*Xtrain + np.random.normal(0.0,1.0,100).reshape(-1, 1)
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(Xtrain.reshape(-1, 1), ytrain.reshape(-1, 1))
print(model.intercept_, model.coef_)
```

Typical output:
0.0175381 9.77234168
Plotting model predictions:

```python
import matplotlib.pyplot as plt
plt.rc('font', size=18)
plt.rcParams['figure.constrained_layout.use'] = True
plt.scatter(Xtrain, ytrain, color='black')
plt.plot(Xtrain, ypred, color='blue', linewidth=3)
plt.xlabel("input x"); plt.ylabel("output y")
plt.legend(["predictions","training data"]) plt.show()
```
In your assignments and individual project reports:

⋆ Always label axes in plots
⋆ Make sure text and numbers are legible and plot is easy to read (use colours, adjust line width/marker size etc). **If really illegible, you should expect to lose marks.**
⋆ Always clearly explain what data a plot shows - giving code is **not** enough, you must explain in english. **If you don’t do this you should expect to lose marks.**
Linear Regression with Multiple Variables

Advertising example:

<table>
<thead>
<tr>
<th></th>
<th>TV $x_1$</th>
<th>Radio $x_2$</th>
<th>Newspaper $x_3$</th>
<th>Sales $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>230.1</td>
<td>37.8</td>
<td>69.2</td>
<td>22.1</td>
</tr>
<tr>
<td>2</td>
<td>44.5</td>
<td>39.3</td>
<td>45.1</td>
<td>10.4</td>
</tr>
<tr>
<td>3</td>
<td>17.2</td>
<td>45.9</td>
<td>69.3</td>
<td>9.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- $n$ = number of features (3 in this example)
- $(x^{(i)}, y^{(i)})$ the $i$th training example e.g.

$$x^{(1)} = [230.1, 37.8, 69.2]^T = \begin{bmatrix} 230.1 \\ 37.8 \\ 69.2 \end{bmatrix}$$

- $x_{j}^{(i)}$ is feature $j$ in the $i$th training example, e.g. $x_{2}^{(1)} = 37.8$
Model: \( h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \)
e.g. \( h_\theta(x) = 15 + 0.1 \cdot x_1 - 5 \cdot x_2 + 10 \cdot x_3 \)

\begin{align*}
\text{Sales} & = \theta_0 + 0.1 \cdot x_1 - 5 \cdot x_2 + 10 \cdot x_3 \\
\text{TV} & \\
\text{Radio} & \\
\text{Newspaper} & 
\end{align*}

* For convenience, define \( x_0 = 1 \)

* Feature vector \( x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \)

* Parameter vector \( \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \)

\( h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x \)
Linear Regression with Multiple Variables

* Model: $h_\theta(x) = \theta^T x$
  (with $\theta$, $x$ now $n+1$-dimensional vectors)

* Cost Function: $J(\theta_0, \theta_1, \ldots, \theta_n) = J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

* Optimisation: Select $\theta$ that minimises $J(\theta)$

* As before, can find $\theta$ using e.g using gradient descent:
  * Start with some $\theta$
  * Repeat:
    * for $j=0$ to $n$ \{ $\delta_j := -\alpha \frac{\partial}{\partial \theta_j} J(\theta)$ \}
    * for $j=0$ to $n$ \{ $\theta_j := \theta_j + \delta_j$ \}
For $J(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ with $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$:

1. $\frac{\partial}{\partial \theta_0} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$
2. $\frac{\partial}{\partial \theta_1} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$
3. $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

So gradient descent algorithm is:

* Start with some $\theta$
* Repeat:
  * for $j=0$ to $n$ \{ $\delta_j := -\frac{2\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ \}
  * for $j=0$ to $n$ \{ $\theta_j := \theta_j + \delta_j$ \}
Example: Advertising Data

* How is the impact of the advertising spend on TV and radio related, if at all?
* Perhaps a quadratic fit would be better? If so, what does that imply for how we allocate our advertising budget?