

ICDSS Lecture 3: Linear Regression & Logistic Regression

Instructor: *Kenton Kwok,
4th Year Physics*

Machine Learning is methods that leverage data to improve performance on some set of tasks. Training data is used to make predictions without being explicitly programmed to do so.

Supervised Learning

Predict values:
e.g. price of a house

Regression

Distinguish
between **classes**

Classification

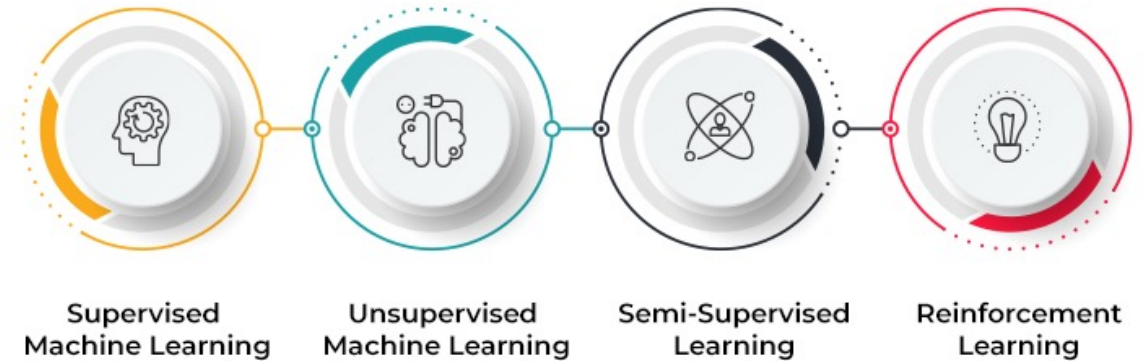


There are many supervised learning algorithms!

- Linear Regression
- Polynomial Regression
- Logistic Regression
- Support Vector Machines
- Decision Trees
- K-Nearest Neighbours
- Neural Networks and their Variants



TYPES OF MACHINE LEARNING

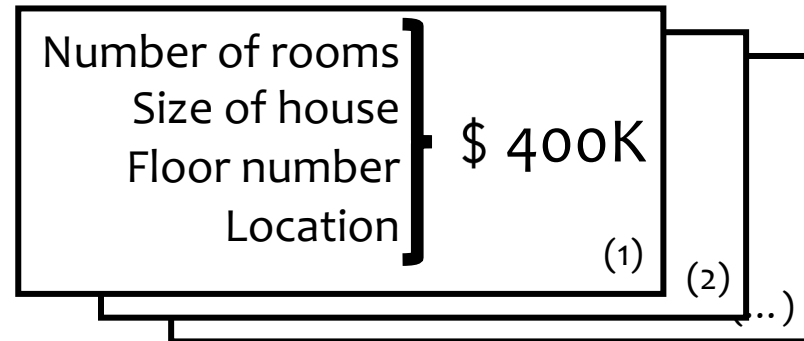


Linear Regression

Linear Regression: supervised learning

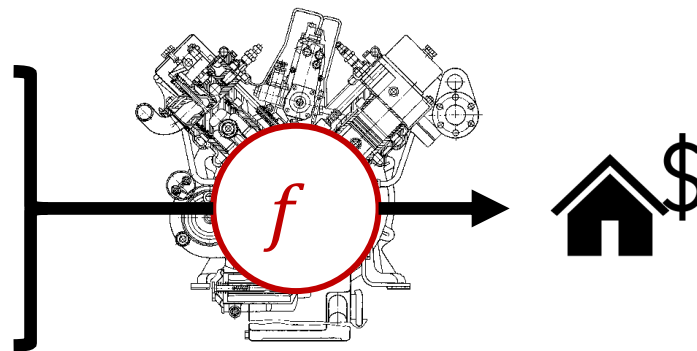
Problem: **predict** house price 

Dataset: house prices



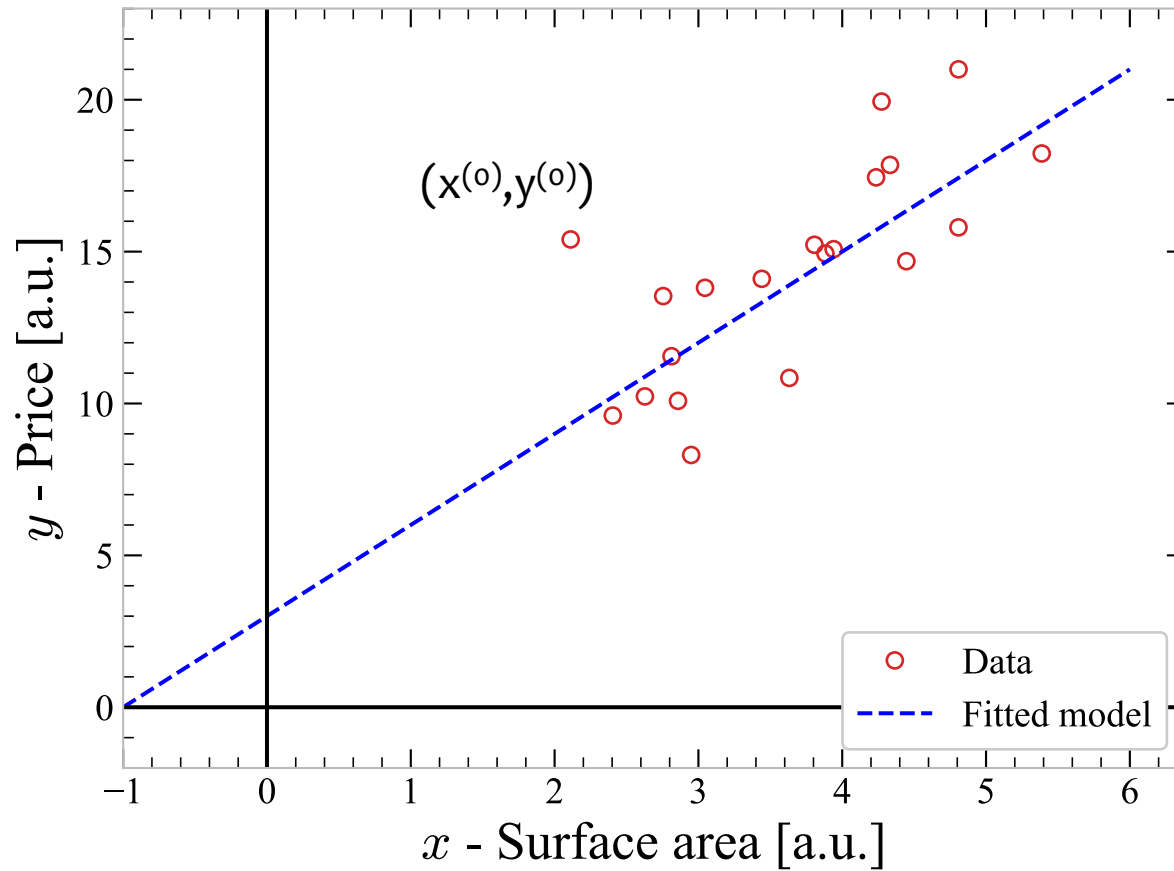
Model

Number of rooms
Size of house
Floor number
Location



Fit the model to the data, so that the function gives **good estimations** of the house price.

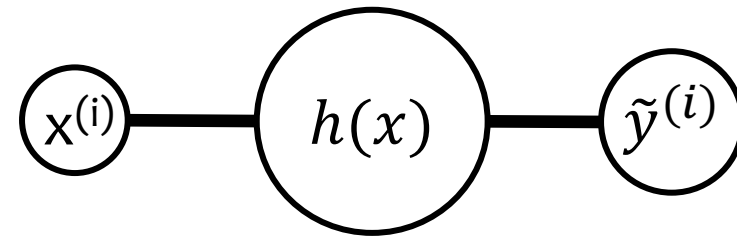
Supervised Learning: predicting house prices



X : Surface area, known **feature**

Y : Price of house, quantity that we want to predict

\tilde{y} : Model prediction



We need to find the values of a_0 and a_1 that **better fit** the data.

$$\mathbf{a} = (a_0, a_1)$$

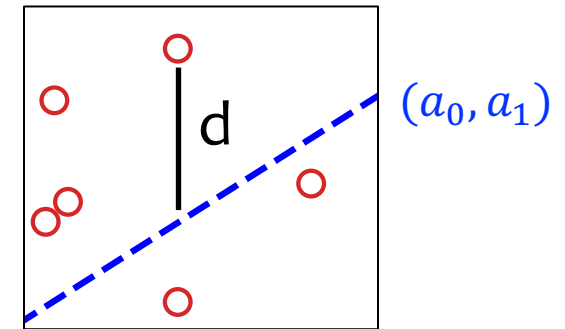
$$\tilde{y}^{(i)} = h(x^{(i)}; \mathbf{a}) = a_0 + a_1 x^{(i)}$$

Is our model good? The **cost function**

$$\tilde{y}^{(i)} = h(x^{(i)}; \mathbf{a}) = a_0 + a_1 x^{(i)}$$

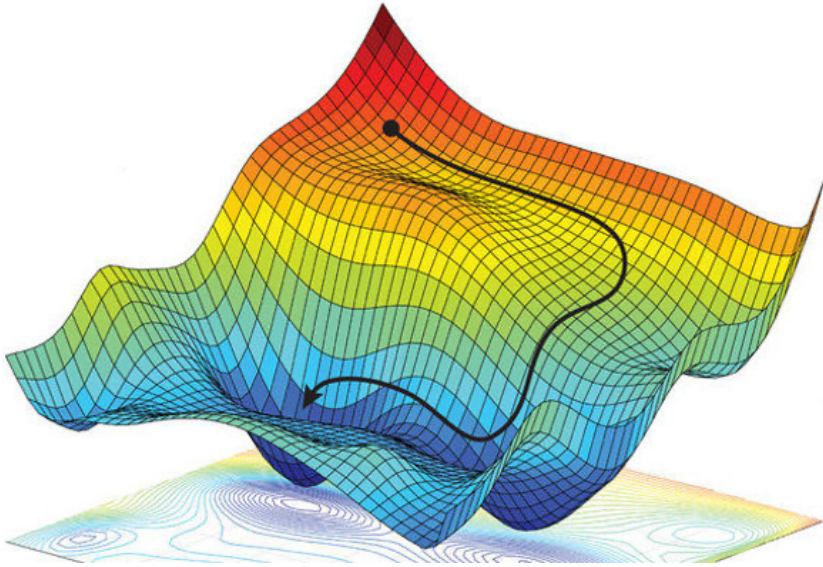
The **cost function** returns a number that tells us how well our model is fitting the data

$$\text{MSE: } C(\mathbf{a}) = \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - h(x^{(i)}; \mathbf{a}) \right)^2$$



~~We need to find the values of a_0 and a_1 that better fit the data.~~

We need to find the value of $\mathbf{a} = (a_0, a_1)$ that minimizes the cost function. This process is called **training the model**.



[A. Amini, Spatial Uncertainty Sampling for End-to-End Control]

$$C(\mathbf{a}) = \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - h(x^{(i)}; \mathbf{a}) \right)^2$$

$$\mathbf{a} = (a_0, a_1)$$

It is possible to get stuck in the **local minimum**, however we wish to find the **global minimum**.

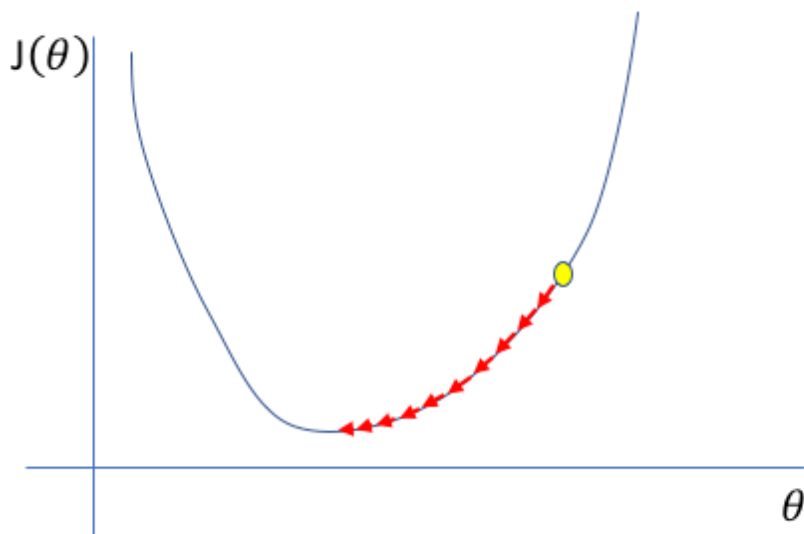
Update an initial guess to move towards the minimum of C:

$$(a_0^{(0)}, a_1^{(0)}) \rightarrow (a_0^{(1)}, a_1^{(1)}) \rightarrow (a_0^{(2)}, a_1^{(2)}) \rightarrow (a_0^{(3)}, a_1^{(3)}) \rightarrow \dots$$

Use **available algorithms** to perform this **updates**: in most cases you don't have to code these yourselves! These are called **OPTIMIZERS**.

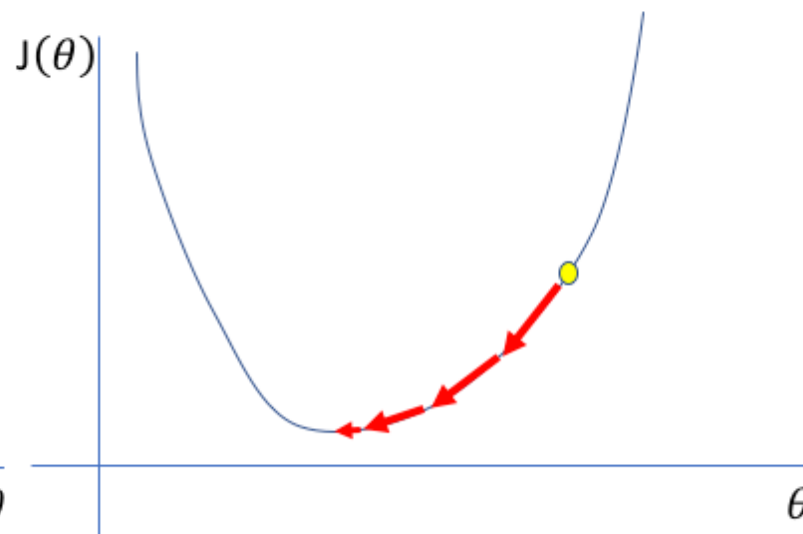
Optimizing the cost function: learning rate

Too low



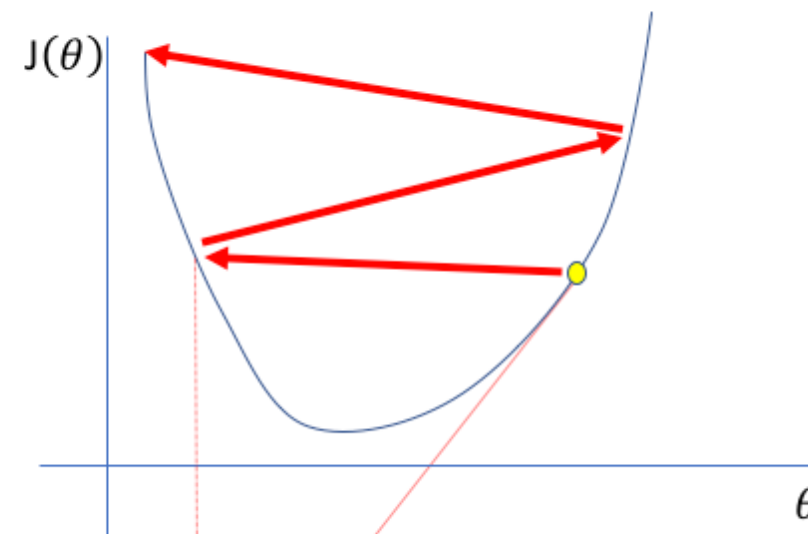
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high

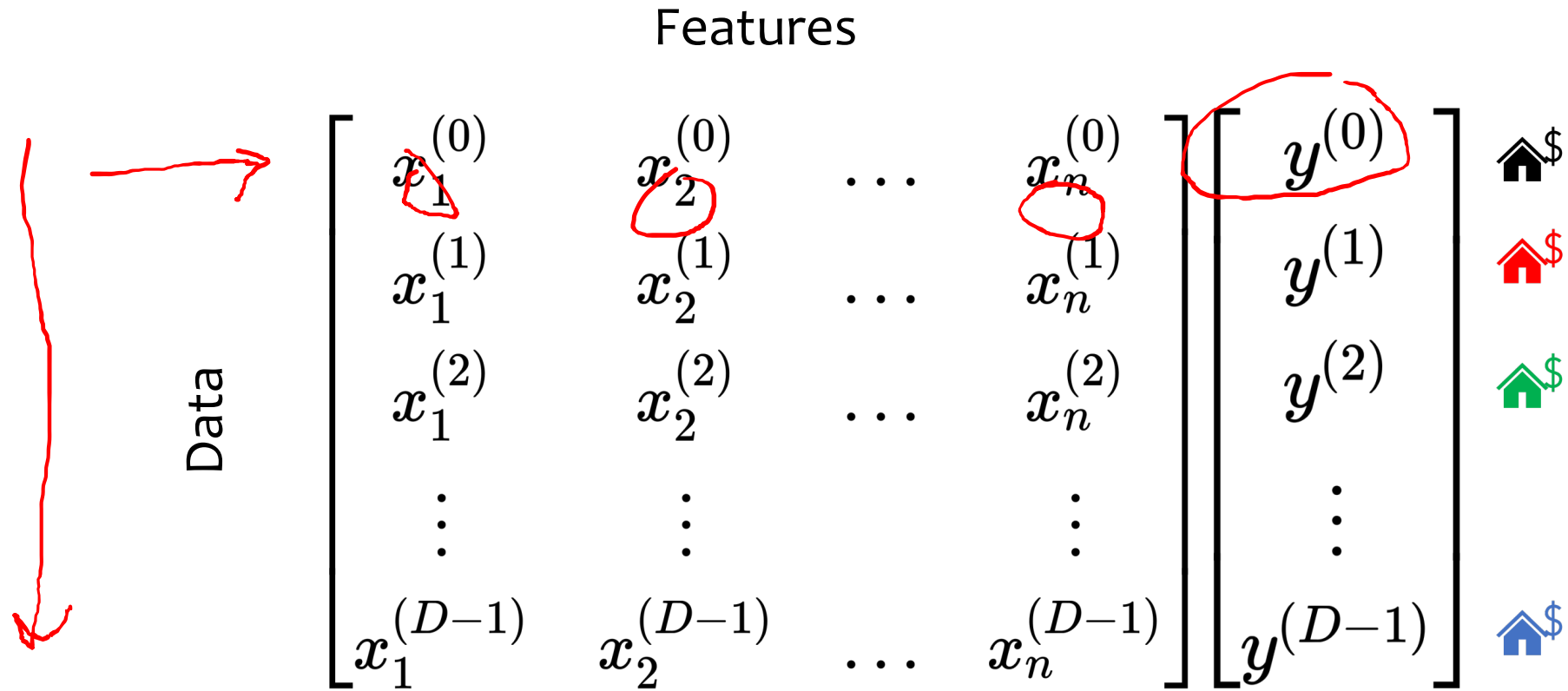


Too large of a learning rate causes drastic updates which lead to divergent behaviors

For all i parameters

$$\text{Update } a_i \mapsto a_i - \lambda \frac{\partial C(\mathbf{a})}{\partial a_i}$$

Linear regression on more dimensions



More generally, we may have more than 1 feature.

$$\tilde{y} = h(\mathbf{x}^{(i)}; \mathbf{a}) = a_0 + a_1 \underline{x_1} + a_2 \underline{x_2} + \dots + a_n x_n$$

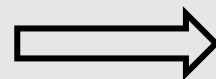


Generalised line in n dimensions

Few details on notation...

$$\mathbf{x} = [1, x_1, x_2, \dots, x_n]$$

$$\mathbf{a} = [a_0, a_1, a_2, \dots, a_n]$$



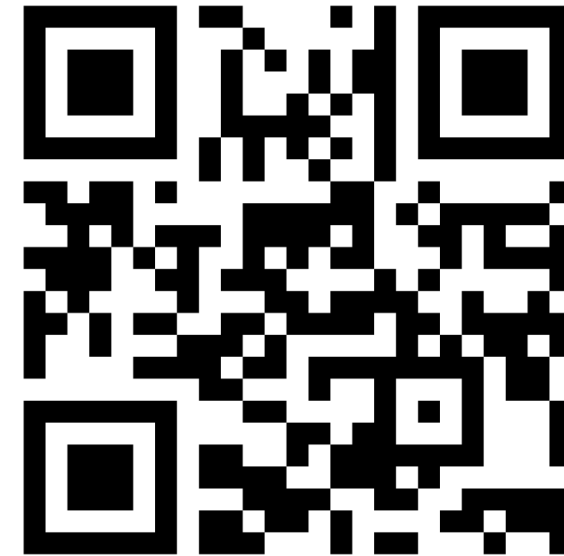
$$\tilde{y} = h(\mathbf{x}^{(i)}; \mathbf{a}) = \mathbf{a}^T \mathbf{x}$$

What is the Cost function going to be?

$$1) \quad C(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{D-1} y^{(i)^2} - (\mathbf{a}^T \mathbf{x}^{(i)})^2$$

$$2) \quad C(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{D-1} (y^{(i)} - \mathbf{a}^T \mathbf{x}^{(i)})^2$$

$$3) \quad C(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{D-1} |y^{(i)} - \mathbf{a}^T \mathbf{x}^{(i)}|$$



Multi variable linear regression:

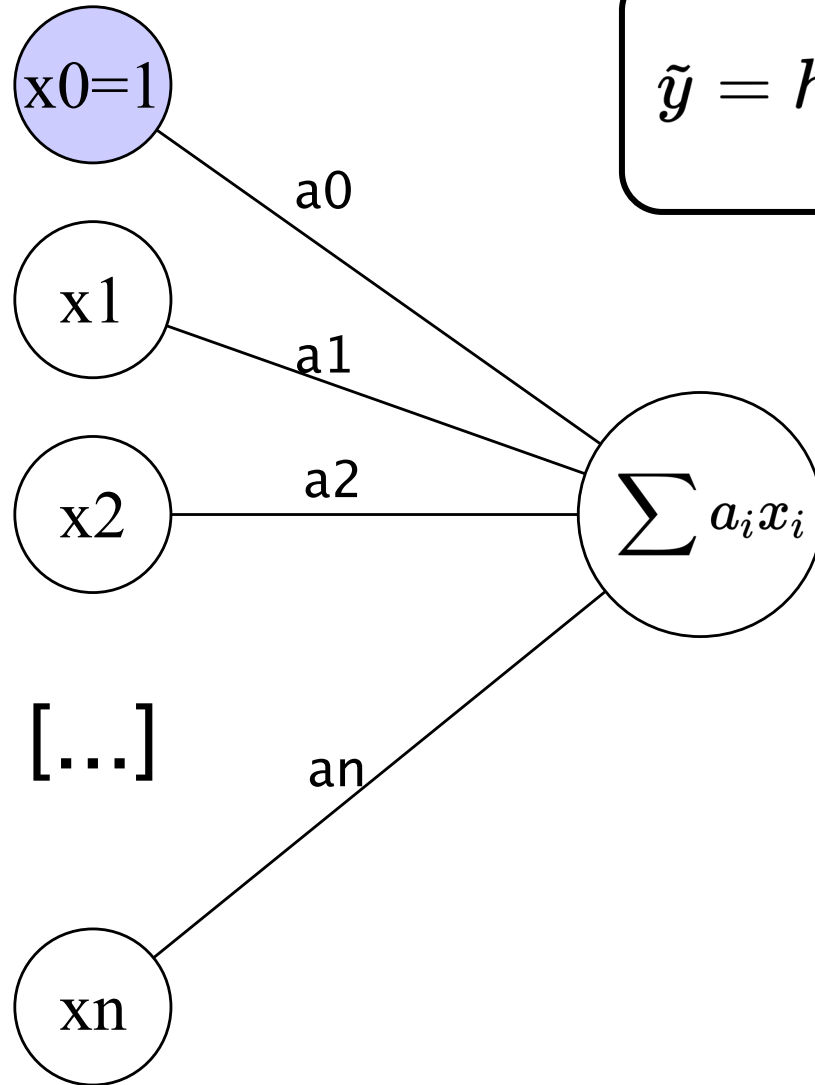
$$\tilde{y} = h(\mathbf{x}^{(i)}; \mathbf{a}) = \mathbf{a}^T \mathbf{x}$$

$$\mathbf{x} = [1, x_1, x_2, \dots, x_n]$$
$$\mathbf{a} = [a_0, a_1, a_2, \dots, a_n]$$

$$C(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{D-1} (y^{(i)} - \mathbf{a}^T \mathbf{x}^{(i)})^2$$

Use an optimizer to minimize the cost function (come to next workshop!).

A provocative way of picturing linear regression



$$\tilde{y} = h(\mathbf{x}^{(i)}; \mathbf{a}) = a_0 + a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

Handwritten red correction: $\cancel{x_0=1}$

$$\tilde{y} = h(\mathbf{x}^{(1)}; \mathbf{a}) = \mathbf{a}^T \mathbf{x}^{(1)} = [a_0 \quad a_1 \quad a_2 \quad a_3] \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = a_0 + a_1 x + a_2 x_2 + a_3 x_3$$

\mathbf{a} is the model's parameter vector, containing the bias term a_0 and feature weights $a_1, a_2 \dots$

\mathbf{x} is the feature vector, containing $x_0 = 1$ to x_3

h is the hypothesis function, using the model parameters \mathbf{a}

$$C(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{D-1} (y^{(i)} - \mathbf{a}^T \mathbf{x}^{(i)})^2$$

1. What does i in $\mathbf{x}^{(i)}$ mean?
2. What is D ?
3. What is N ?

- Dataset:

$$\begin{bmatrix} x_1^{(0)} & x_1^{(0)} & \dots & x_n^{(0)} \\ x_1^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(D)} & x_1^{(D)} & \dots & x_n^{(D)} \end{bmatrix} \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(D)} \end{bmatrix} \begin{matrix} \text{house}^{\$} \\ \text{red house}^{\$} \\ \text{green house}^{\$} \\ \vdots \\ \text{blue house}^{\$} \end{matrix}$$

- Normalise data.

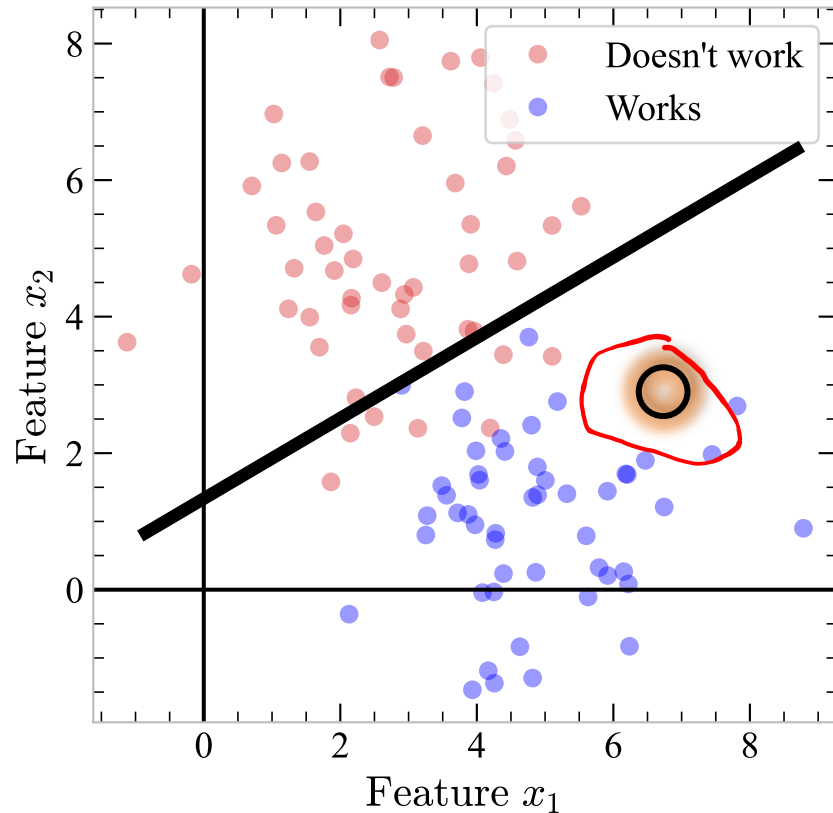
- Model: $\tilde{y} = h(\mathbf{x}^{(i)}; \mathbf{a}) = \underline{\mathbf{a}}^T \mathbf{x}$

- Cost function: $C(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{D-1} (y^{(i)} - \mathbf{a}^T \mathbf{x}^{(i)})^2$

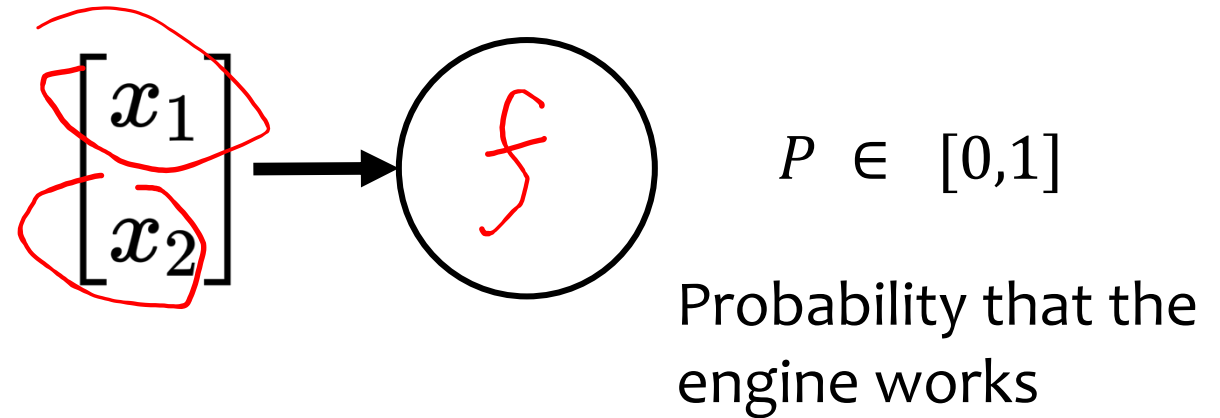
- Choose any available optimizer and train the model to minimise the **cost function**.

- Evaluate model. Is it performing good?

Logistic Regression



We want to predict the probability that the engine will work.



 Will this engine work?

Do this using **Logistic Regression**

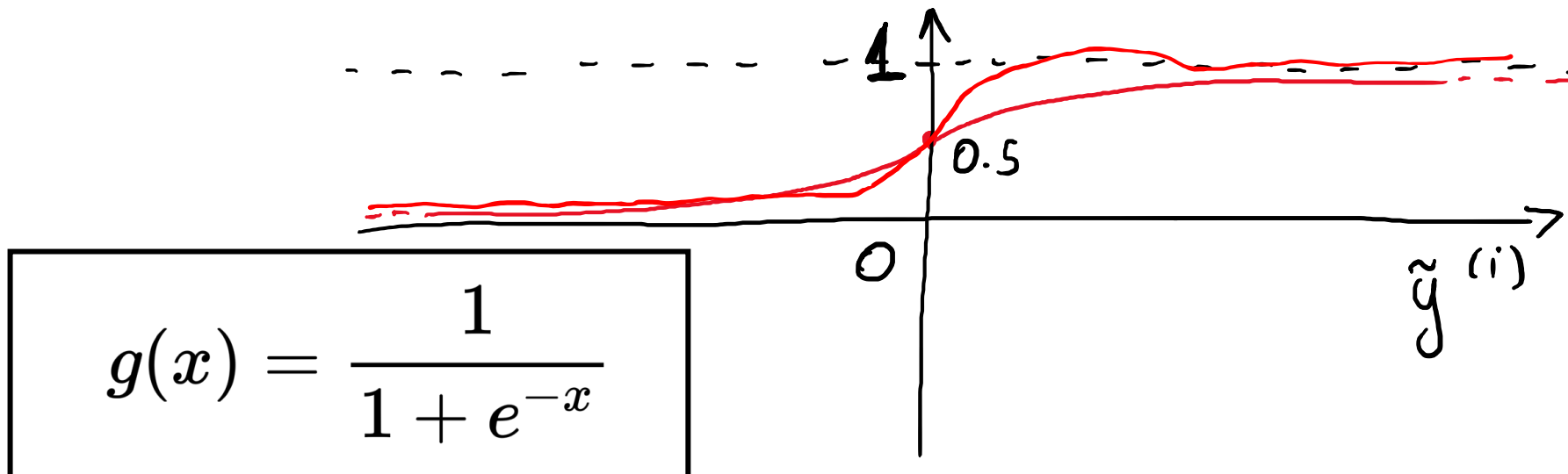
Linear regression or Logistic Regression?

- We want to predict a probability that is either 0 = 0% or 1 = 100%.
- Linear regression gave us a value that **is not between 0 and 1**

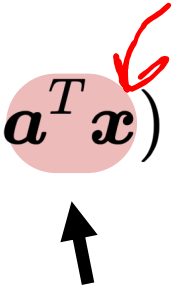
$$\tilde{y}^{(i)} = h(x^{(i)}; \mathbf{a}) = a_0 + a_1 x^{(i)}$$

$$g(\tilde{y}^{(i)})$$

- Wrap the result of the linear regression with another function.



Logistic regression model:

$$h_a = g(\mathbf{a}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{a}^T \mathbf{x}}}$$


Output from linear regression

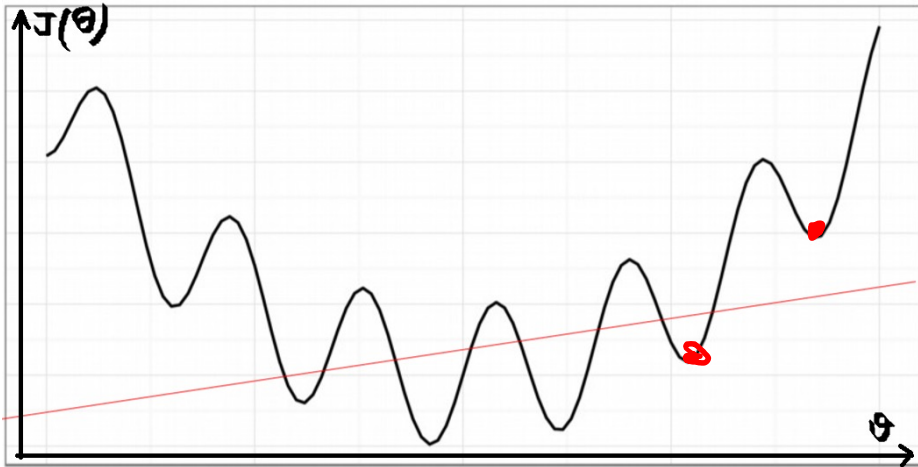
Sigmoid function:

$$g(x) = \frac{1}{1 + e^{-x}}$$

Given some parameters x , obtain the probability $P(x)$ of the engine working or not.

1. **Task** ✓
2. **Model** ✓
3. **Cost function:** is the model performing good? ✓
4. **Optimization algorithm:** train the model such that the cost function is minimised.

Cost function: a problem of convexity



Why not our usual?

$$C(\mathbf{a}) = \frac{1}{2} \sum_{i=0}^{D-1} \left(\underbrace{h_{\mathbf{a}}(x^{(i)})}_{\text{red underline}} - \underbrace{y^{(i)}}_{\text{red circle}} \right)^2$$

This function has many local minima and hence it is not **convex**.
Hard to minimise!

Another approach comes from **maximising the likelihood!**

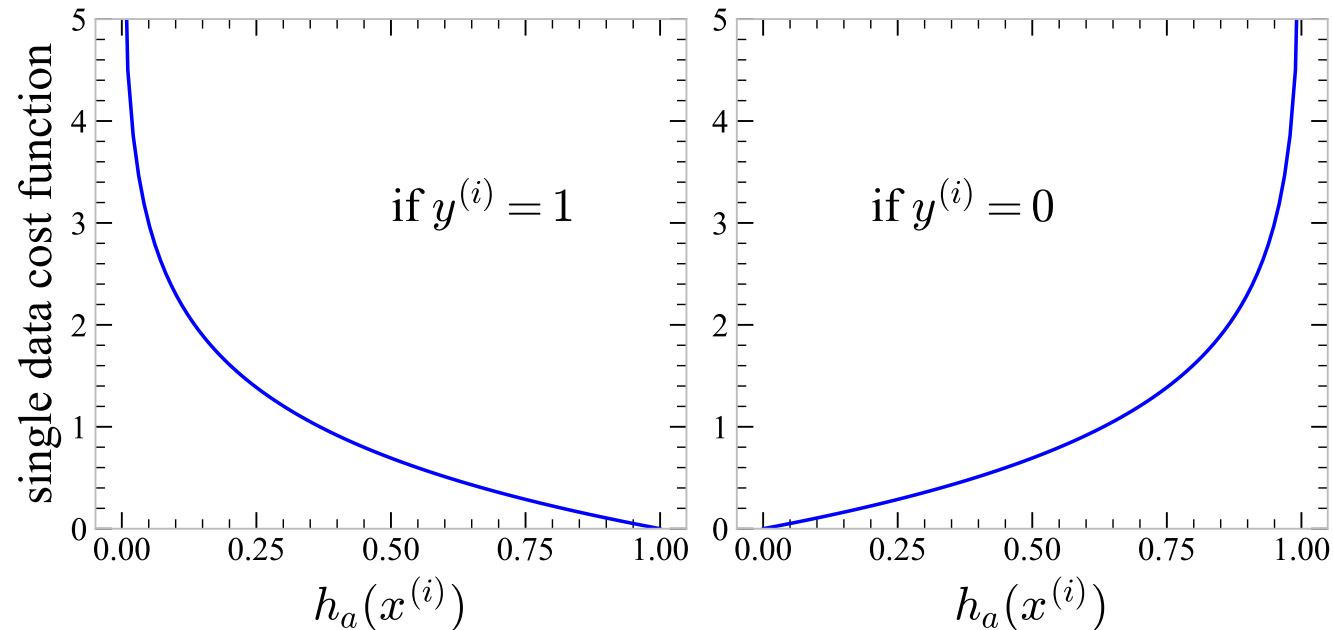
<https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/pdfs/40%20LogisticRegression.pdf>

Cost function (Log-loss):

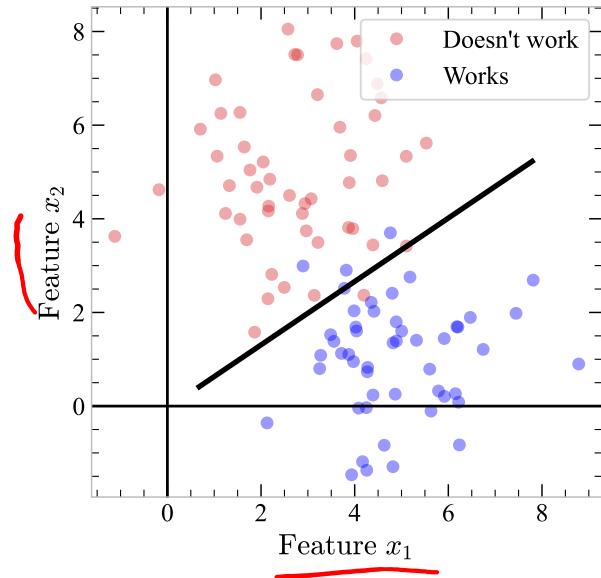
$$C(\mathbf{a}) = \sum_{i=0}^{D-1} y^{(i)} \log \left[g \left(\mathbf{a}^T \mathbf{x}^{(i)} \right) \right] + \left(1 - y^{(i)} \right) \log \left[1 - g \left(\mathbf{a}^T \mathbf{x}^{(i)} \right) \right]$$

Look at one term:

$$y^{(i)} \log \left[g \left(\mathbf{a}^T \mathbf{x}^{(i)} \right) \right] + \left(1 - y^{(i)} \right) \log \left[1 - g \left(\mathbf{a}^T \mathbf{x}^{(i)} \right) \right]$$



Decision boundary



Logistic regression:
$$h_a(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{a}^T \mathbf{x}}}$$

We call the **decision boundary** the points where

$$h_a(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{a}^T \mathbf{x}}} = 0.5$$

Hence,
$$e^{-\mathbf{a}^T \mathbf{x}} = 1 \quad \therefore \quad -\mathbf{a}^T \mathbf{x} = \log 1 = 0$$

$$\therefore \mathbf{a}^T \mathbf{x} = a_0 + a_1 x_1 + a_2 x_2 = 0$$

$$x_2 = -\frac{a_1}{a_0} x_1 - \frac{a_2}{a_0}$$

This is the equation of a **line**! This is a **linear decision boundary**!

Instead of doing a binary classifier (yes/no), what if we want to classify multiple things?

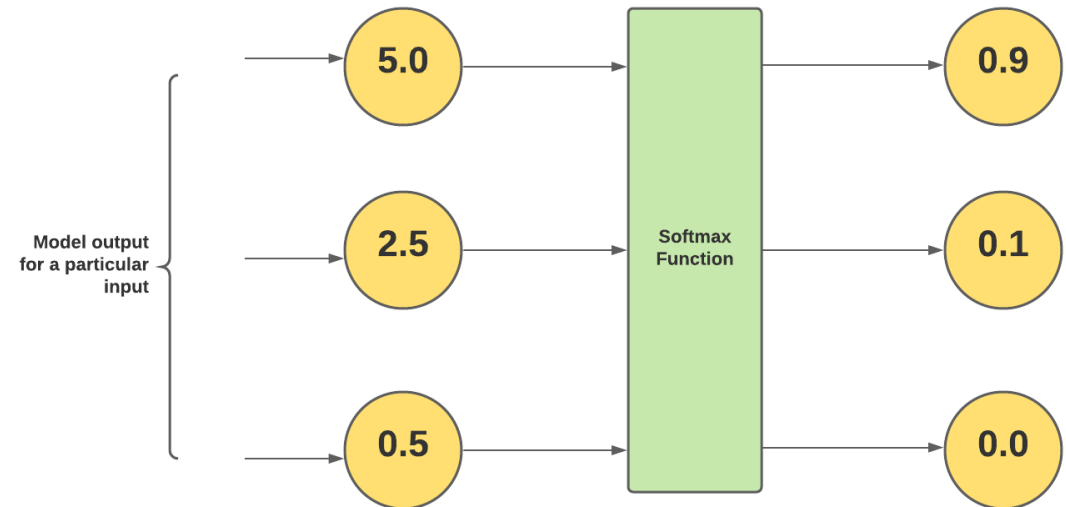
Have separate scores/ parameter values for each class k

$$s_k(\mathbf{x}) = \mathbf{a}^{(k)T} \mathbf{x}$$

Estimate probability via Softmax function instead of the logistic function

$$p_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp(s_k(\mathbf{x}))}{\sum_{j=1}^K \exp(s_j(\mathbf{x}))}$$

This is basically a weighted sum of s_k passed through an exponential function



Loss function: Cross Entropy

$$C(a) = -\frac{1}{m} \sum_{i=1}^D \sum_{k=1}^K y_k^{(i)} \log(p_k^{(i)})$$

First sum is over the D data instances

Second sum is over the K classes

When there are two classes, this is the same as the log-loss!

Model validation

- Overfitting vs underfitting (bias vs variance)

How do we measure how well our model is performing?

Can use metrics

- Precision vs Recall
- Accuracy

Mini-exercises

- Implement MSE and log-loss and cross entropy loss functions
- Implement Logistic Regression in Scikit-learn

Questions?

For an alternative explanation to the concepts today, go to this interactive website that Amazon have made: <https://mlu-explain.github.io/>

A good resource to practical implementations of this is Hands on Machine Learning by Aurélien Geron (Chapter 2-3)

A full mathematical resource can be found here (Andrew Ng Stanford Course) <https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf>