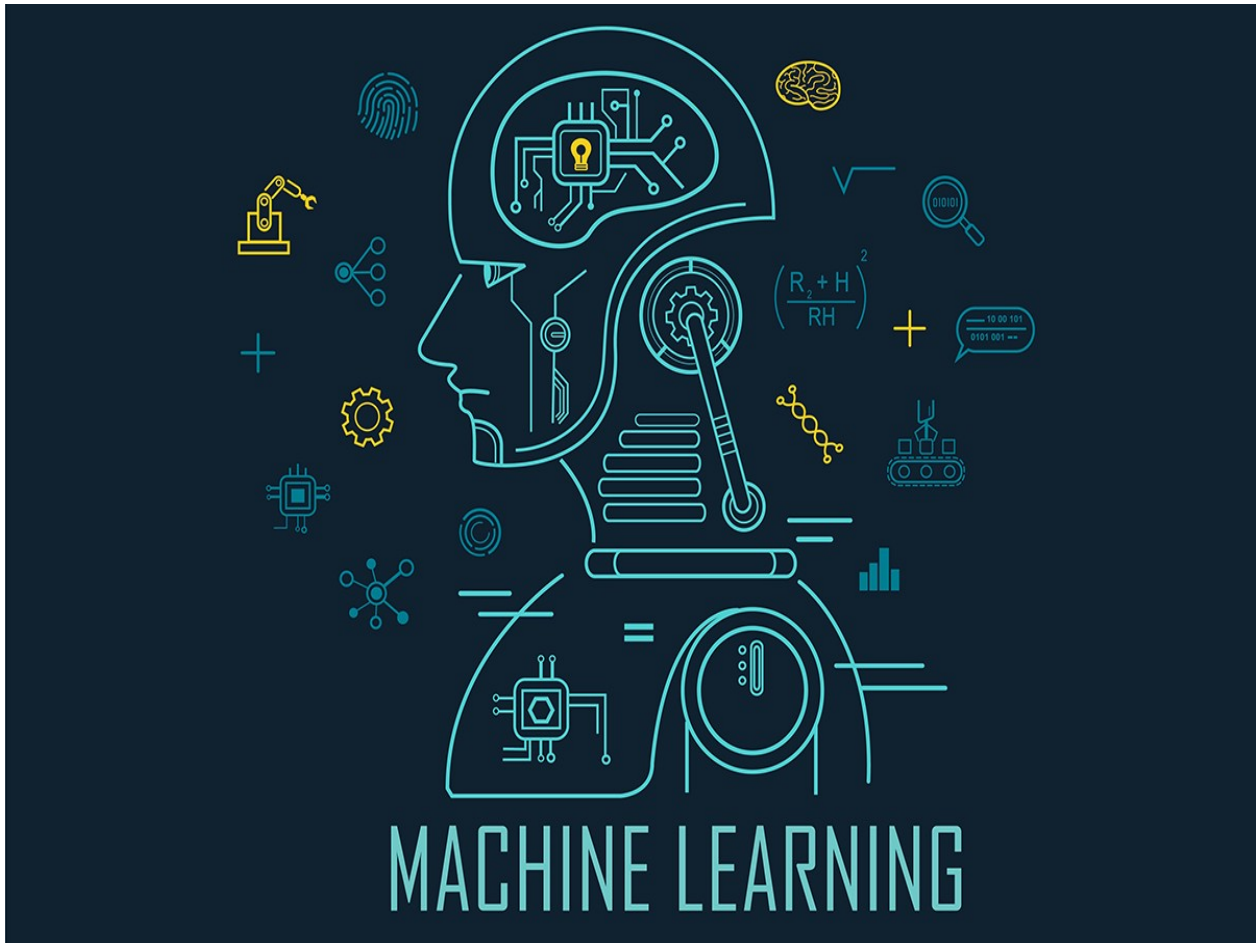
Machine Learning 1
Introduction



A definition:

"A program learns from experience E of performing task T with a performance measure P"

- Tom Mitchell 1998

A definition:

"A program learns from experience E of performing task T with a performance measure P"

- Tom Mitchell 1998

• The task is generally quite narrow in scope, largely because of issues in well defining a general performance measure.

Type I

Supervised learning

- 'Right' answers are known for some set of examples which we will call the **training set.** This gives a well defined performance measure.

Type I

Supervised learning

- 'Right' answers are known for some set of examples which we will call the **training set.** This gives a well defined performance measure.
- Task T can be a **classification** problem or a **continuous prediction**.

Type I

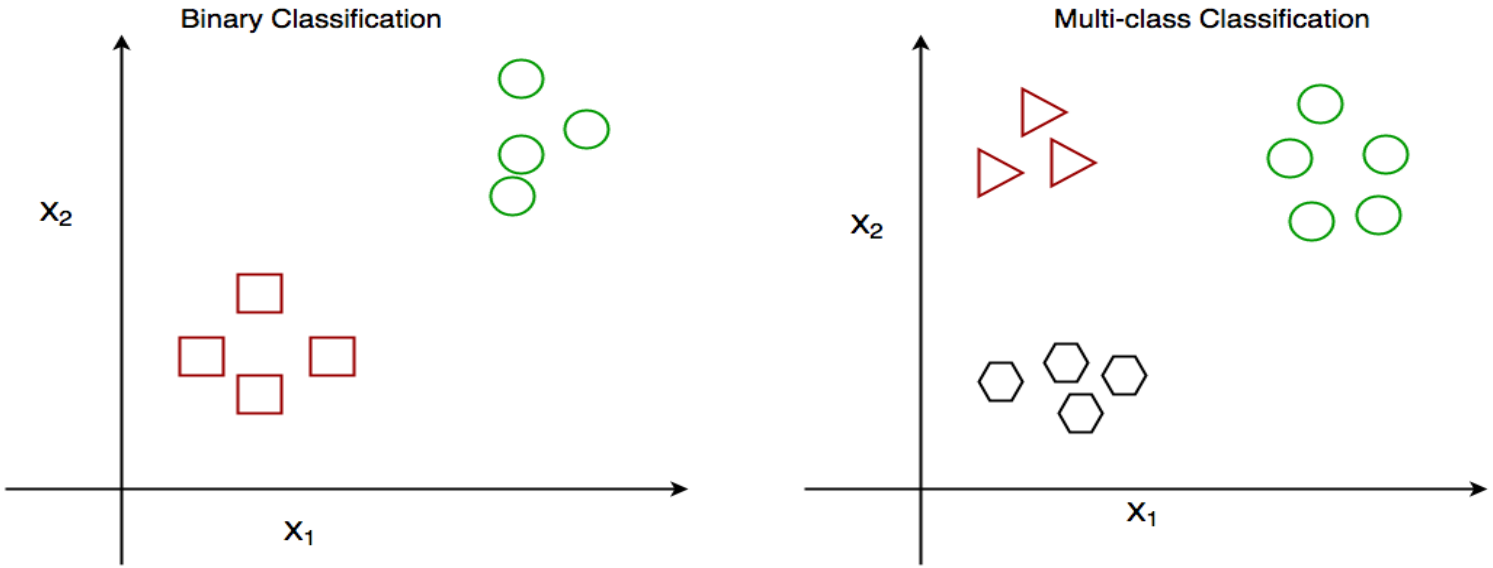
Supervised learning

- 'Right' answers are known for some set of examples which we will call the **training set.** This gives a well defined performance measure.
- Task T can be a **classification** problem or a **continuous prediction**.
- Some examples: face recognition, patient diagnosis, spam detection, galaxy classification, power spectrum classification ...

Type I

Supervised learning

- 'Right' answers are known for some set of examples which we will call the **training set.** This gives a well defined performance measure.
- Task T can be a **classification** problem or a **continuous prediction**.
- Some examples: face recognition, patient diagnosis, spam detection, galaxy classification, power spectrum classification ...



Type II

Un-supervised learning

- 'Right' answers are **NOT** known. *What is the performance measure??*

Type II

Un-supervised learning

- 'Right' answers are **NOT** known. *What is the performance measure??*
- Task T is generally to identify trends, correlations and/or probability distributions without prior knowledge.

Type II

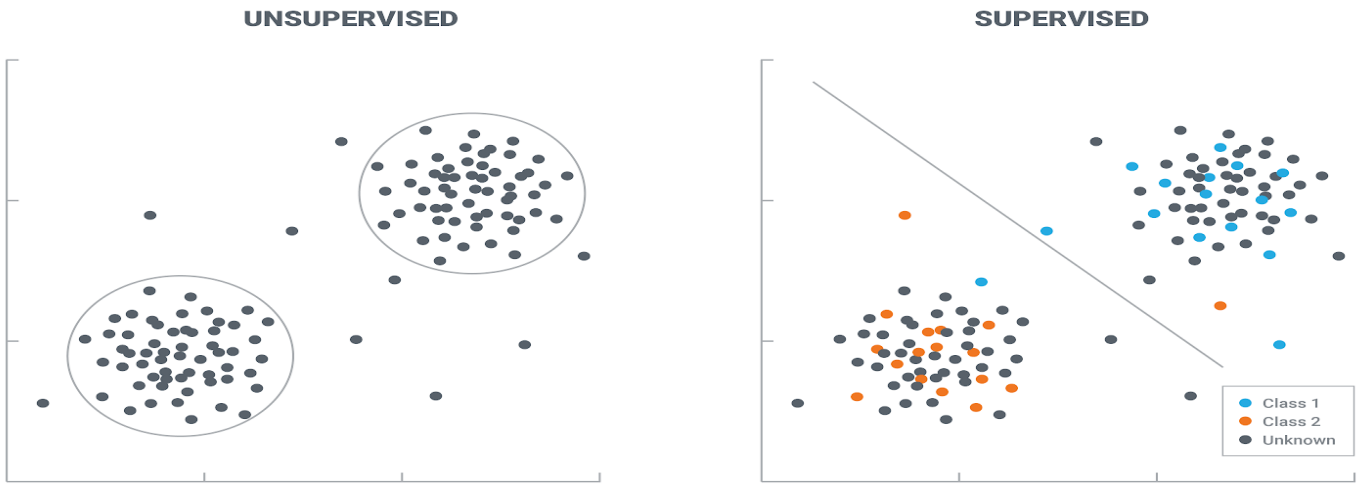
Un-supervised learning

- 'Right' answers are **NOT** known. *What is the performance measure??*
- Task T is generally to identify trends, correlations and/or probability distributions without prior knowledge.
- Some examples for cosmology could include identification of features in the correlation function from galaxy cluster data, void finding, anomaly detection

Type II

Un-supervised learning

- 'Right' answers are **NOT** known. *What is the performance measure??*
- Task T is generally to identify trends, correlations and/or probability distributions without prior knowledge.
- Some examples for cosmology could include identification of features in the correlation function from galaxy cluster data, void finding, anomaly detection



We will focus on supervised learning

We will start with some definitions and notation ...

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.
- **Cost** will refer to the performance measure **P**. In supervised learning, this measures how different a prediction is from the 'right' result. This is usually denoted as J.

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.
- **Cost** will refer to the performance measure **P**. In supervised learning, this measures how different a prediction is from the 'right' result. This is usually denoted as J.

In general I'll also use

m: the number of examples we have / how large the training set is.

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.
- **Cost** will refer to the performance measure **P**. In supervised learning, this measures how different a prediction is from the 'right' result. This is usually denoted as J.

In general I'll also use

m: the number of examples we have / how large the training set is.

n: the number of features describing each data example.

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.
- **Cost** will refer to the performance measure **P**. In supervised learning, this measures how different a prediction is from the 'right' result. This is usually denoted as J.

In general I'll also use

m: the number of examples we have / how large the training set is.

n: the number of features describing each data example.

$\vec{x}^{(i)}$ **:** will be our feature vector for the i^{th} example.

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.
- **Cost** will refer to the performance measure **P**. In supervised learning, this measures how different a prediction is from the 'right' result. This is usually denoted as J.

In general I'll also use

m: the number of examples we have / how large the training set is.

n: the number of features describing each data example.

$\vec{x}^{(i)}$ **:** will be our feature vector for the i^{th} example.

$\vec{y}^{(i)}$ **:** will be the correct output (class or value) for the i^{th} example.

We will focus on supervised learning

We will start with some definitions and notation ...

- **Features** will refer to parameters or functions of parameters that we choose to describe the data.
- **Cost** will refer to the performance measure **P**. In supervised learning, this measures how different a prediction is from the 'right' result. This is usually denoted as J.

In general I'll also use

m: the number of examples we have / how large the training set is.

n: the number of features describing each data example.

$\vec{x}^{(i)}$ **:** will be our feature vector for the i^{th} example.

$\vec{y}^{(i)}$ **:** will be the correct output (class or value) for the i^{th} example.

$\vec{\theta}$ **:** will denote a vector of (fitted) coefficients that transform the feature vector into a prediction.

Let's take a basic example...

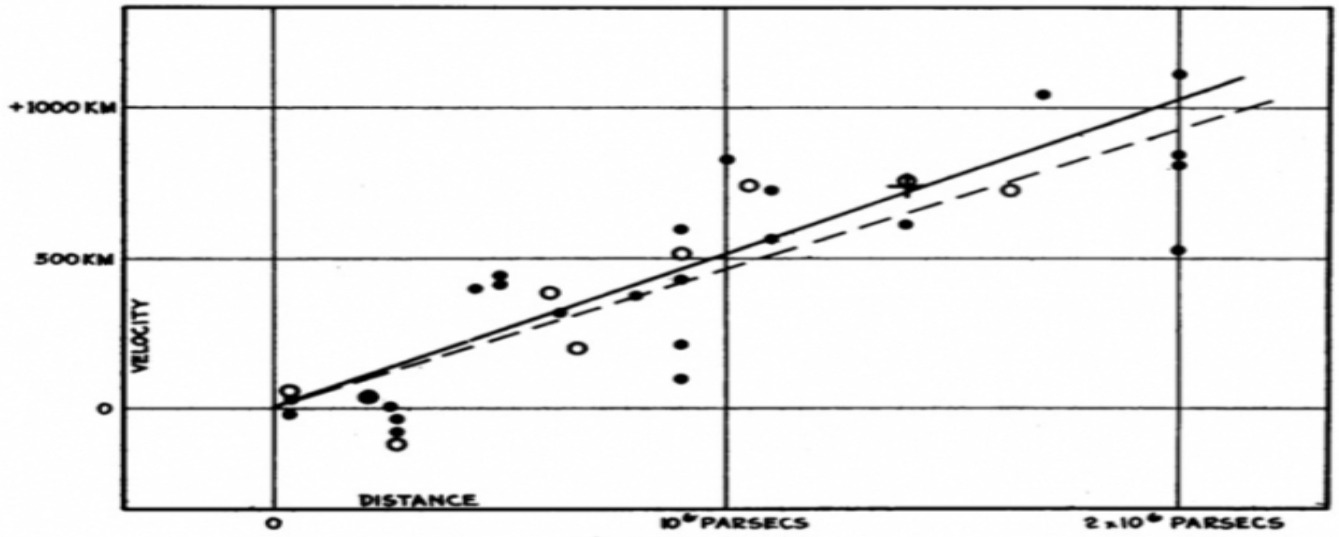


FIGURE 1

Velocity-Distance Relation among Extra-Galactic Nebulae.

Let's take a basic example...

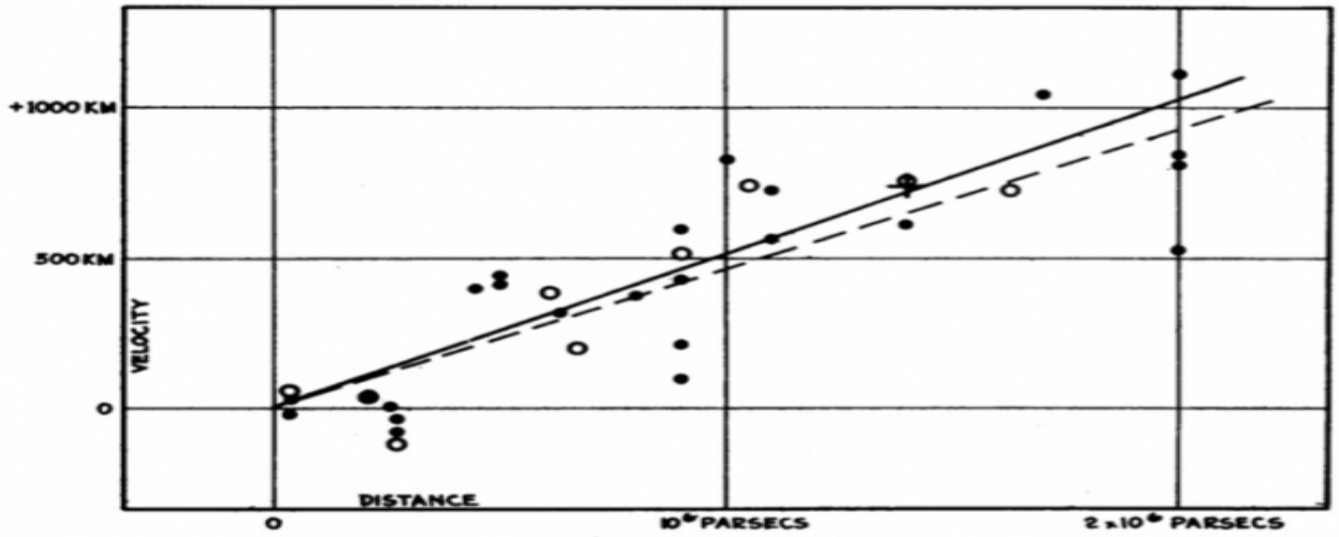


FIGURE 1
Velocity-Distance Relation among Extra-Galactic Nebulae.

The features are $\{\mathrm{bias} = 1, r\}$. Say our training set is given by the points on the plot. We can use fit a straight line to the data to get new predictions, i.e. **linear regression** . The performance measure or 'cost' could be a χ^2

Let's take a basic example...

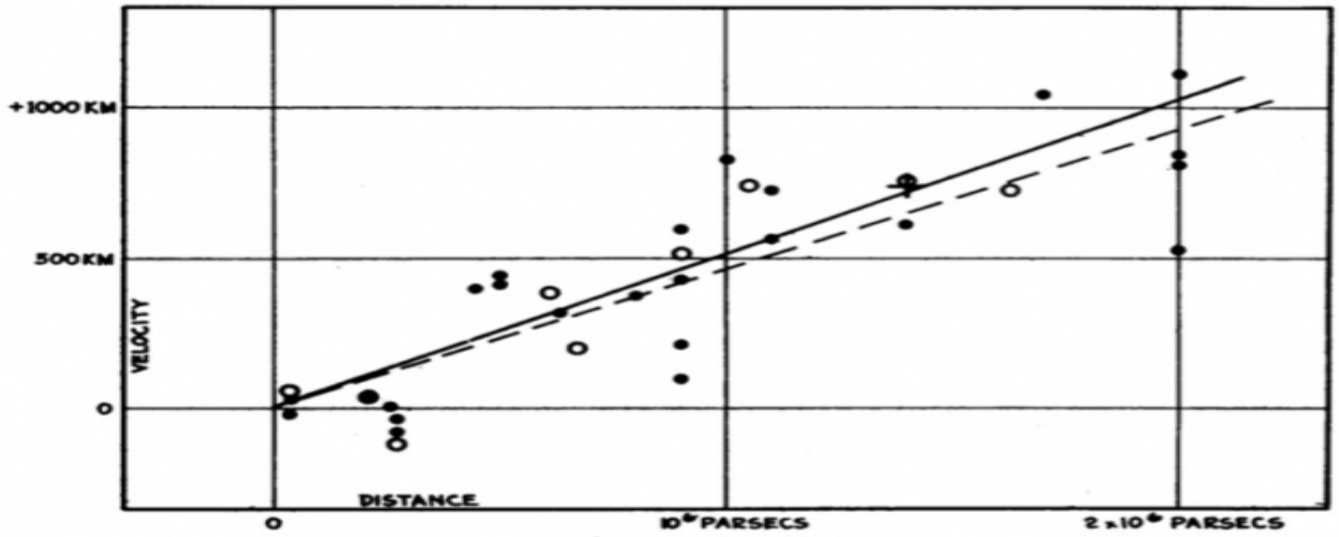


FIGURE 1
Velocity-Distance Relation among Extra-Galactic Nebulae.

The features are $\{\text{bias} = 1, r\}$. Say our training set is given by the points on the plot. We can use fit a straight line to the data to get new predictions, i.e. **linear regression** . The performance measure or 'cost' could be a χ^2

$$\min_\theta J(\theta) = \min_\theta \frac{1}{2m} \sum_{i=1}^{m} \frac{(\theta^T \vec{x}^{(i)} - \vec{y}^{(i)})^2}{(\sigma^i)^2} \tag{1}$$

Let's take a basic example...

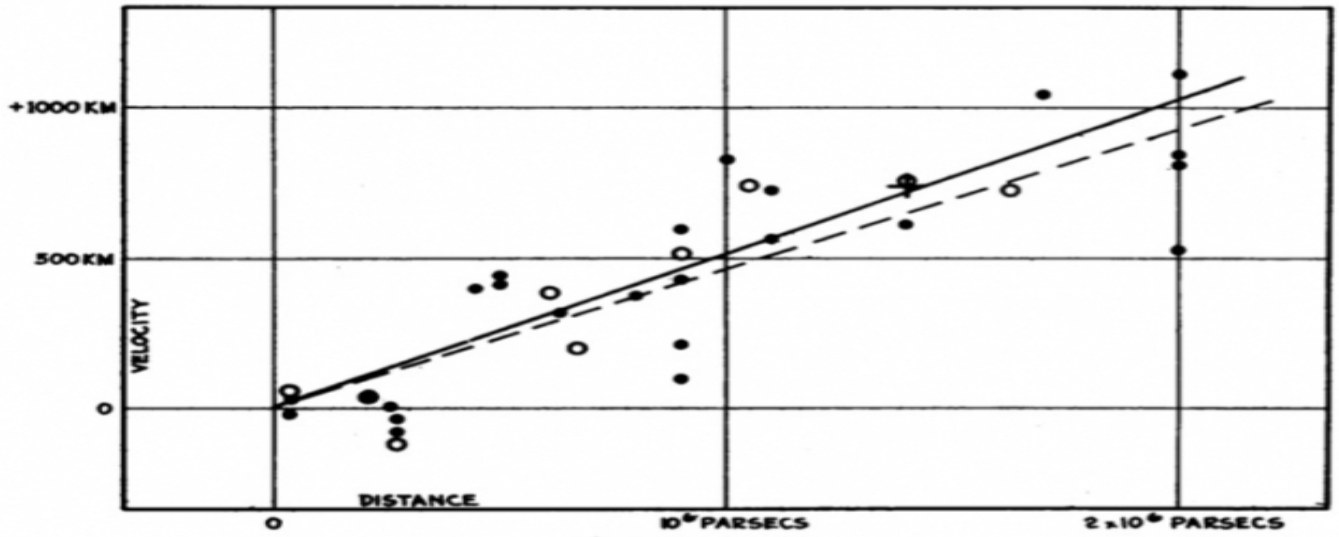


FIGURE 1
Velocity-Distance Relation among Extra-Galactic Nebulae.

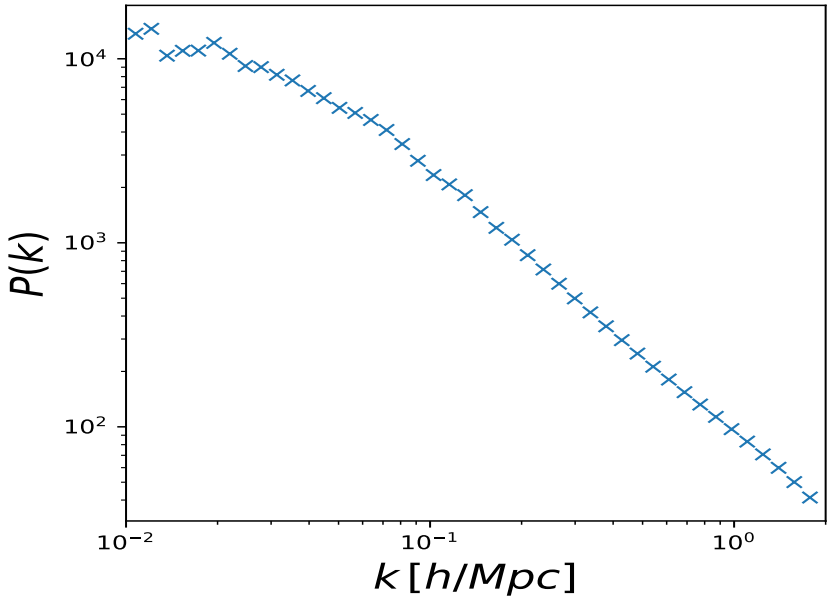
The features are $\{\text{bias} = 1, r\}$. Say our training set is given by the points on the plot. We can use fit a straight line to the data to get new predictions, i.e. **linear regression** . The performance measure or 'cost' could be a χ^2

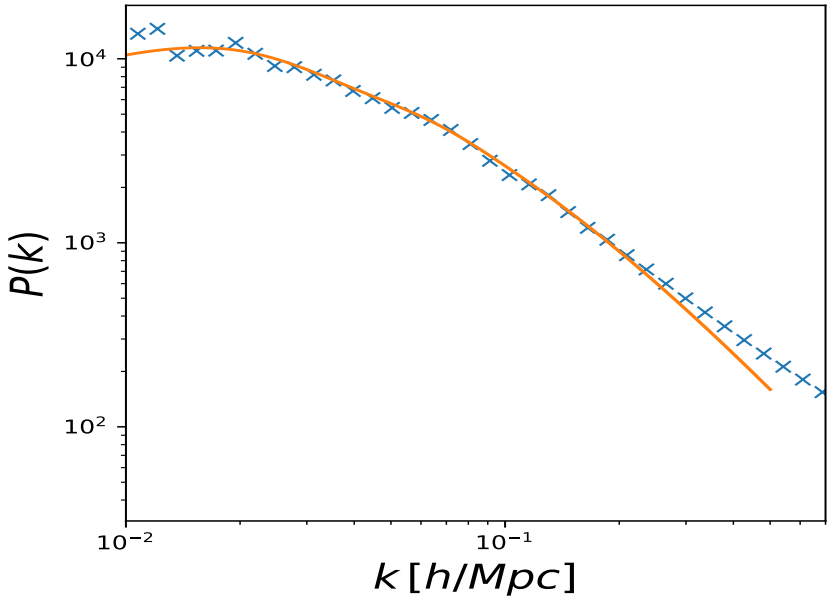
$$\min_\theta J(\theta) = \min_\theta \frac{1}{2m} \sum_{i=1}^{m} \frac{(\theta^T \vec{x}^{(i)} - \vec{y}^{(i)})^2}{(\sigma^i)^2} \tag{1}$$

where in the end Edwin Hubble found $\theta = \{0, H_0\}$.

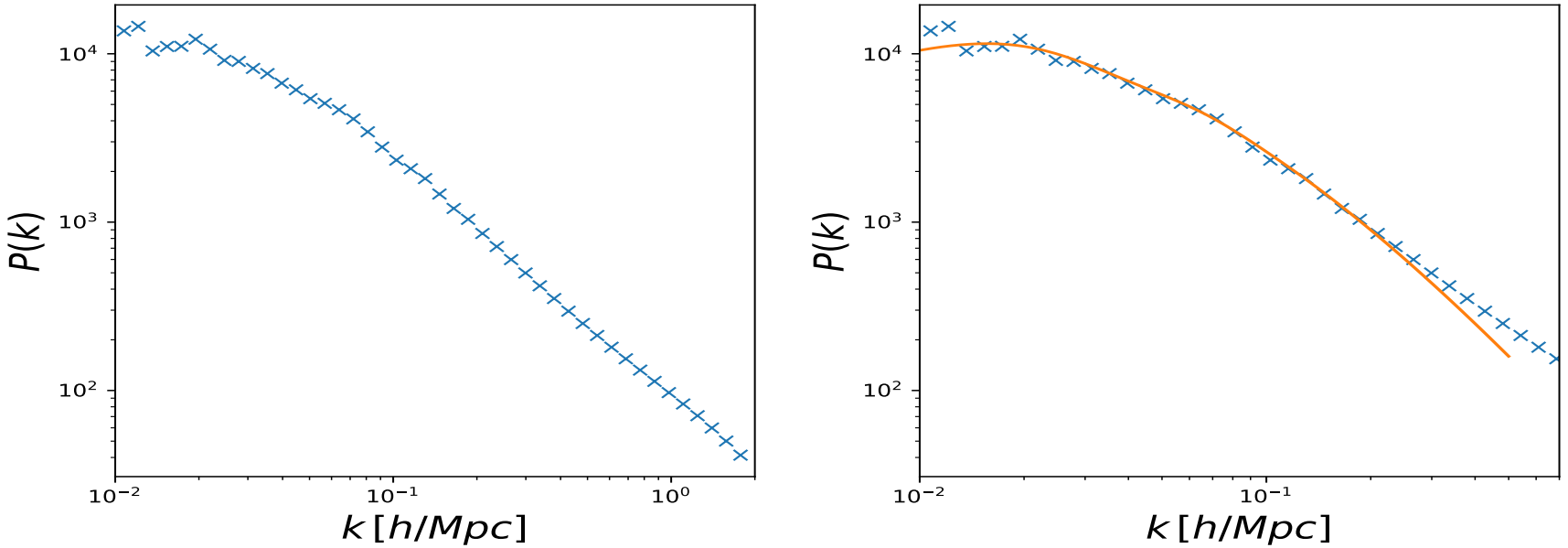
We can extend to **polynomial /multi-variate linear** regression with a familiar example in cosmology, the power spectrum $P(k)$ of dark matter:

We can extend to **polynomial /multi-variate linear** regression with a familiar example in cosmology, the power spectrum $P(k)$ of dark matter:





We can extend to **polynomial /multi-variate linear** regression with a familiar example in cosmology, the power spectrum $P(k)$ of dark matter:



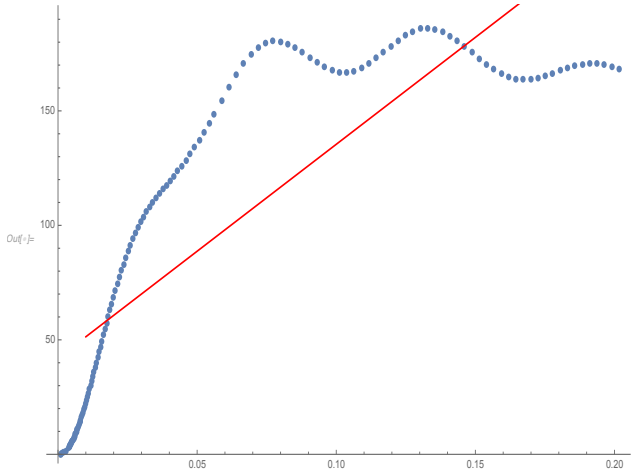
In this case, **m** is the number of k-bins, **n** is the number of features.
Let us then take features $\{1, k, k^2, k^3 ... k^n\}$

$$J(\theta) = \chi^2 = \sum_{k_{\text{bins}}} \frac{[\theta^T \vec{x}^{(i)} - P^{\text{measured}}(k_i)]^2}{\sigma_i^2}. \tag{2}$$

`See python example 1`

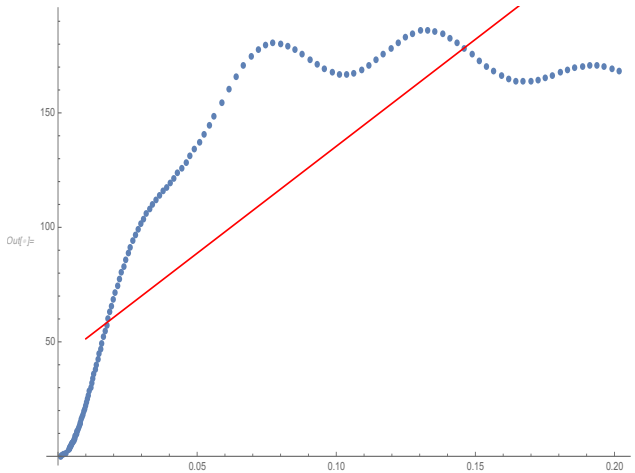
A note on variance and bias

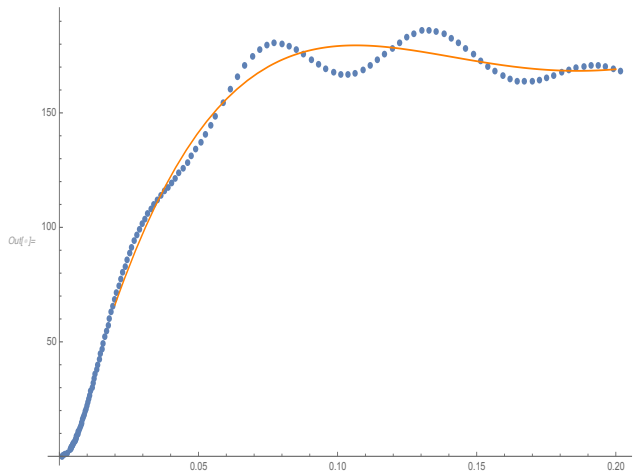
High bias:



A note on variance and bias

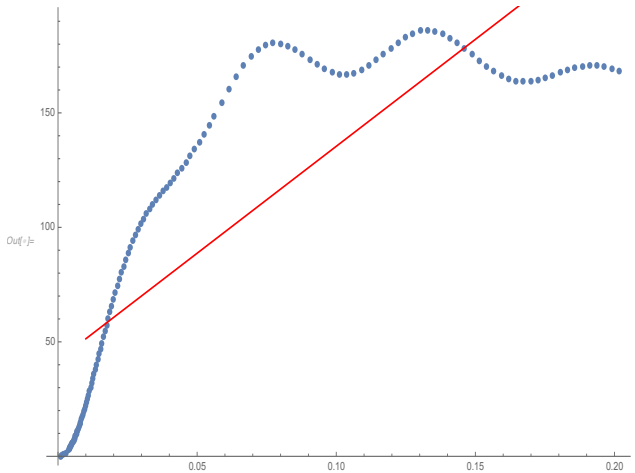
High bias:

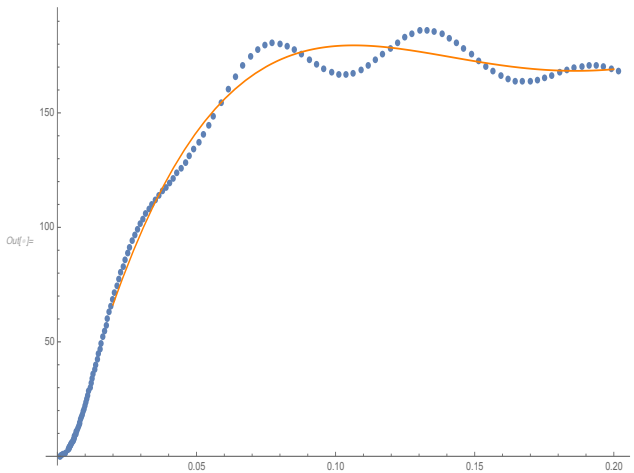




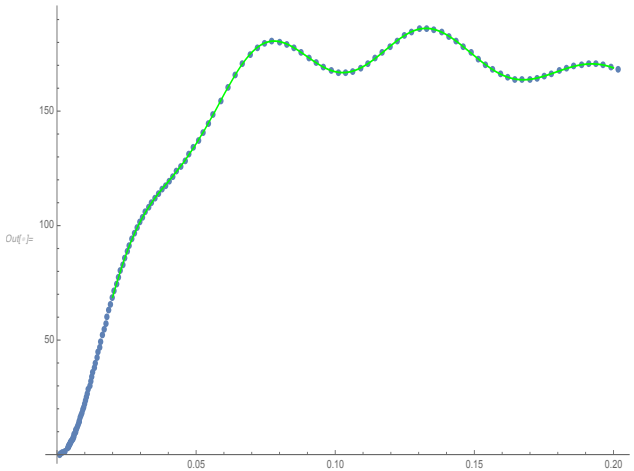
A note on variance and bias

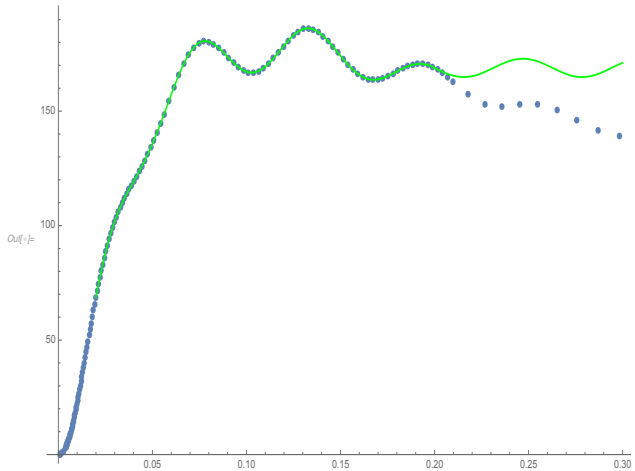
High bias:





High variance:





Some notes before proceeding to classification problems ....

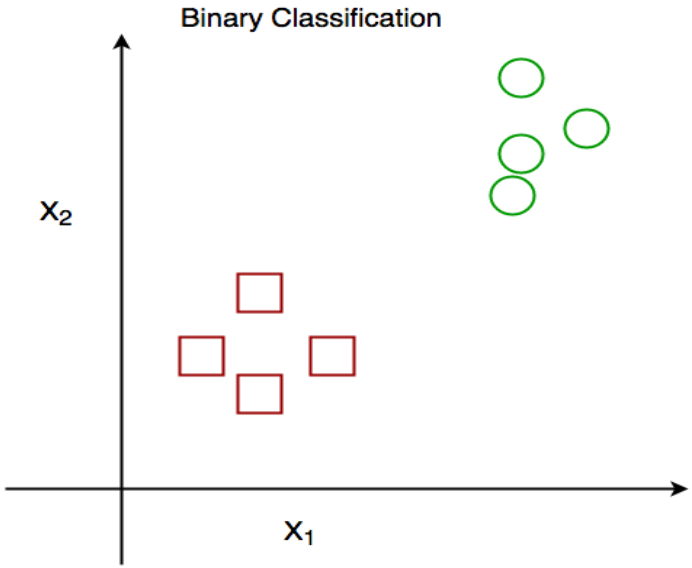
Some notes before proceeding to classification problems ....

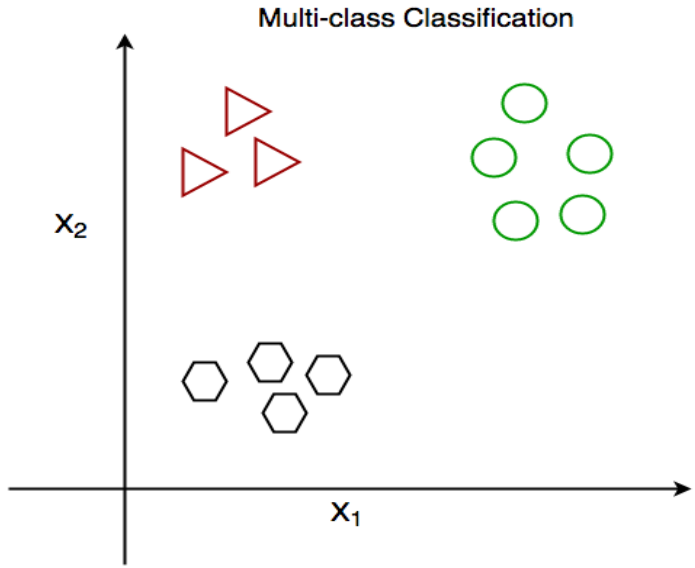
- We can minimize the cost using **gradient descent** or **normal equation** in practice. Many optimised packages for this in many languages.

Some notes before proceeding to classification problems ....

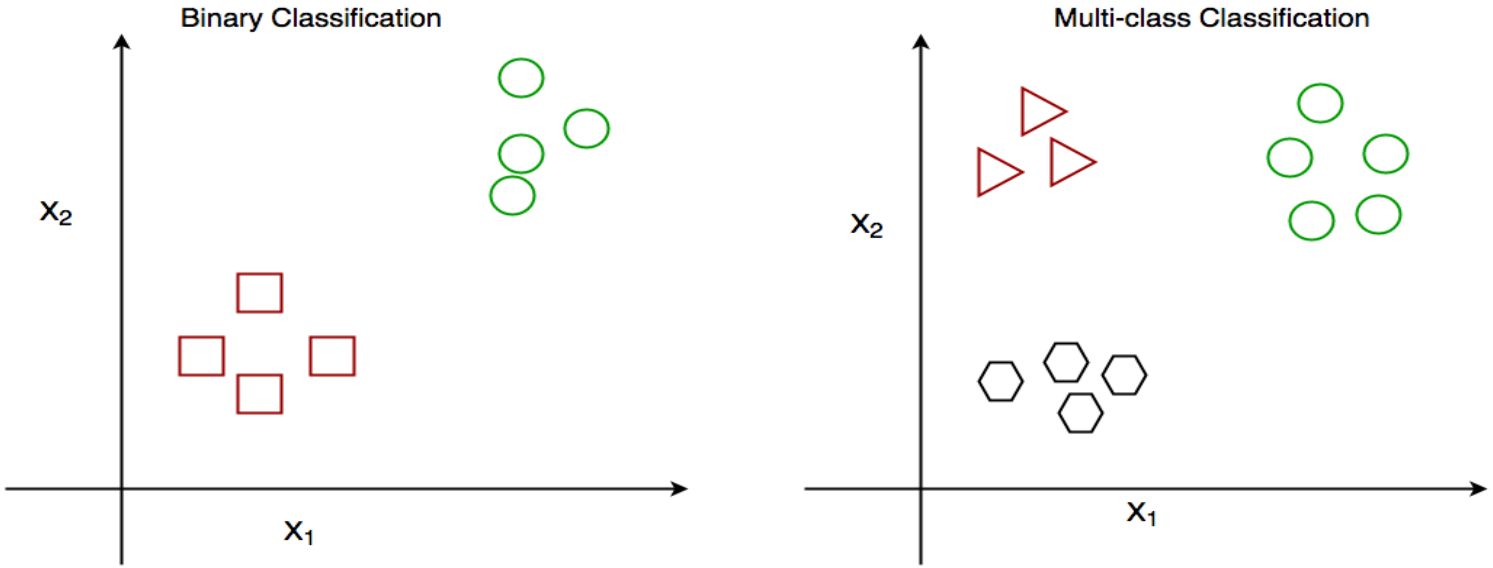
- We can minimize the cost using **gradient descent** or **normal equation** in practice. Many optimised packages for this in many languages.
- For polynomial regression, **feature scaling** is very important - renormalise features to be of the same order + mean normalisation ($\mu_x = 0$).

What about classification?

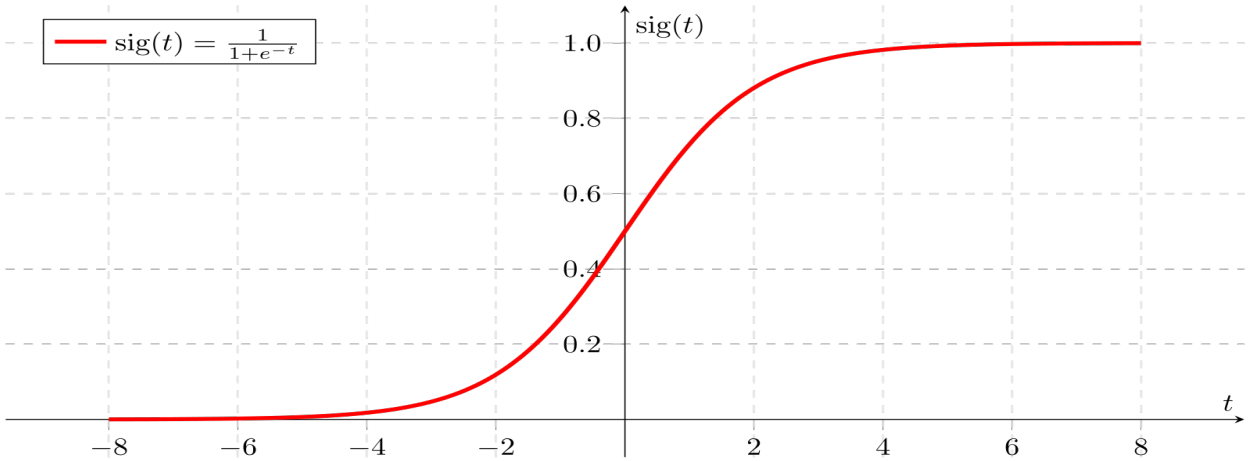




What about classification?



We can use **logistic regression** and the **sigmoid function** to handle this.



Consider binary classification (2 classes)

1. Let $y = 1$ (class 1) and $y = 0$ (class 2) denote the the two classes.
2. Now set our prediction to be given by

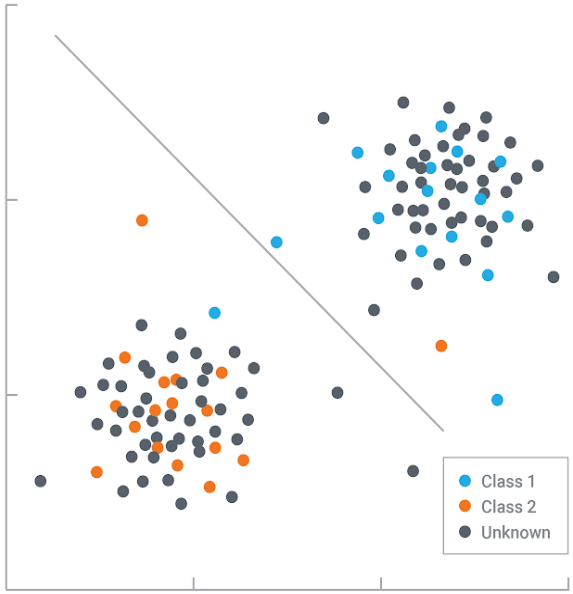
$$h(\theta^T x) = \begin{cases} 1 & \text{if } sig(\theta^T x) \geq 0.5 \\ 0 & \text{if } sig(\theta^T x) < 0.5 \end{cases}, y = \begin{cases} 1 & \text{if } \theta^T x \geq 0.5 \\ 0 & \text{if } \theta^T x < 0.5 \end{cases}$$

Consider binary classification (2 classes)

1. Let $y = 1$ (class 1) and $y = 0$ (class 2) denote the the two classes.
2. Now set our prediction to be given by

$$h(\theta^T x) = \begin{cases} 1 & \text{if } sig(\theta^T x) \geq 0.5 \\ 0 & \text{if } sig(\theta^T x) < 0.5 \end{cases}, y = \begin{cases} 1 & \text{if } \theta^T x \geq 0.5 \\ 0 & \text{if } \theta^T x < 0.5 \end{cases}$$

$\theta^T x$ then draws out what is called the **decision boundary**.



Some notes to bring this all together

- Can use **polynomial features** to get complicated decision boundaries.

Some notes to bring this all together

- Can use **polynomial features** to get complicated decision boundaries.
- Should not use least square error for the cost because sigmoid is non-linear and makes the cost **non-convex** — hard to find global min.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[y^i \log[h(\theta^T x^i)] + (1 - y^i) \log[1 - h(\theta^T x^i)] \right].$$

Some notes to bring this all together

- Can use **polynomial features** to get complicated decision boundaries.
- Should not use least square error for the cost because sigmoid is non-linear and makes the cost **non-convex** — hard to find global min.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[y^i \log[h(\theta^T x^i)] + (1 - y^i) \log[1 - h(\theta^T x^i)] \right]. \quad (3)$$

- For multi-class classification, we can use the **one vs all** method:
  1. Delimit classes numerically as $\{0, 1, 2, 3, ...k\}$.
  2. Train for k-sets of optimised θ, one for each class.
  3. For a new example $\bar{x}$, choose the class j that gives the maximum value of $sig(\theta_j^T \bar{x})$.

`See python example 1`

Example 1: Spam classifier

Example 1: Spam classifier

Features could be word frequency or appearance: $x_j = 1, 0$ if some word appears or not.

Then $\theta^T x^i = \theta_0 + \theta_1 x_1^i + \theta_4 x_4^i$.... etc.

Example 1: Spam classifier

Features could be word frequency or appearance: $x_j = 1, 0$ if some word appears or not.

Then $\theta^T x^i = \theta_0 + \theta_1 x_1^i + \theta_4 x_4^i ....$ etc.

If we want to include frequency, then we might set $x_j = n_j$ where n_j is the number of times the j^{th} word appears.

Summary so far:

- Machine learning can be classified as supervised and unsupervised learning.

Summary so far:

- Machine learning can be classified as supervised and unsupervised learning.
- In supervised learning we have the answers in the form of a training set. The correct answer could be a predicted value or a classification.

Summary so far:

- Machine learning can be classified as supervised and unsupervised learning.
- In supervised learning we have the answers in the form of a training set. The correct answer could be a predicted value or a classification.
- We can form a hypothesis for the prediction or classification using linear/polynomial regression OR logistic regression + one vs all.

Summary so far:

- Machine learning can be classified as supervised and unsupervised learning.
- In supervised learning we have the answers in the form of a training set. The correct answer could be a predicted value or a classification.
- We can form a hypothesis for the prediction or classification using linear/polynomial regression OR logistic regression + one vs all.
- Feature selection is very important!

Example 2: Puzzle classifier

Example 2: Puzzle classifier

Example 2: Puzzle classifier

Features are shades of blue:

- $x_j = 1, 0$ if average of pixels on a piece are in some blue category j.
- $x_j = \bar{B}$ in RGB context. We should rescale/normalise this feature.