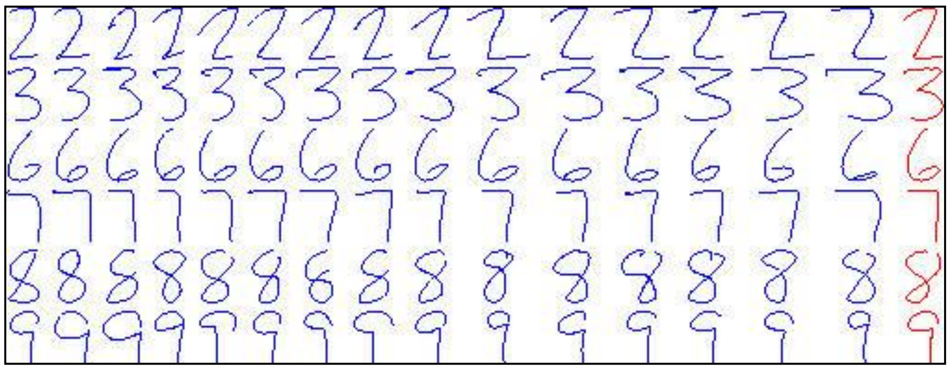
Machine Learning

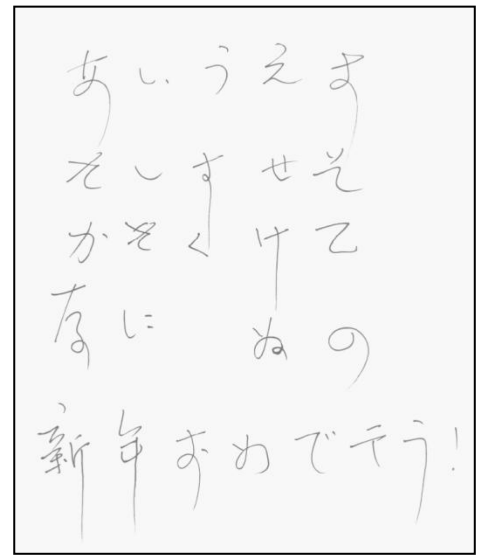
Lecture 1

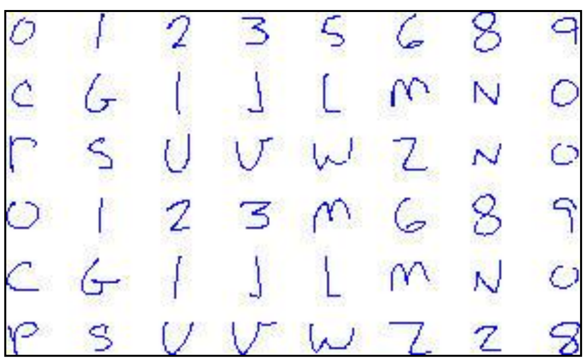
Introduction

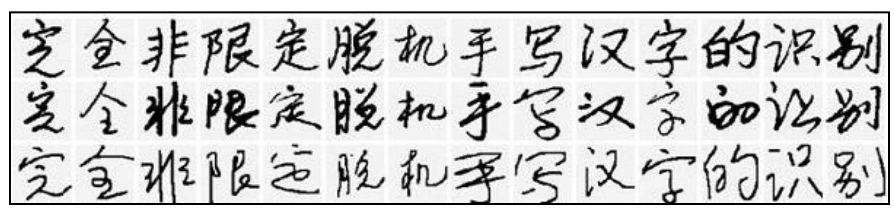
Motivating Problems

- Handwritten Character Recognition



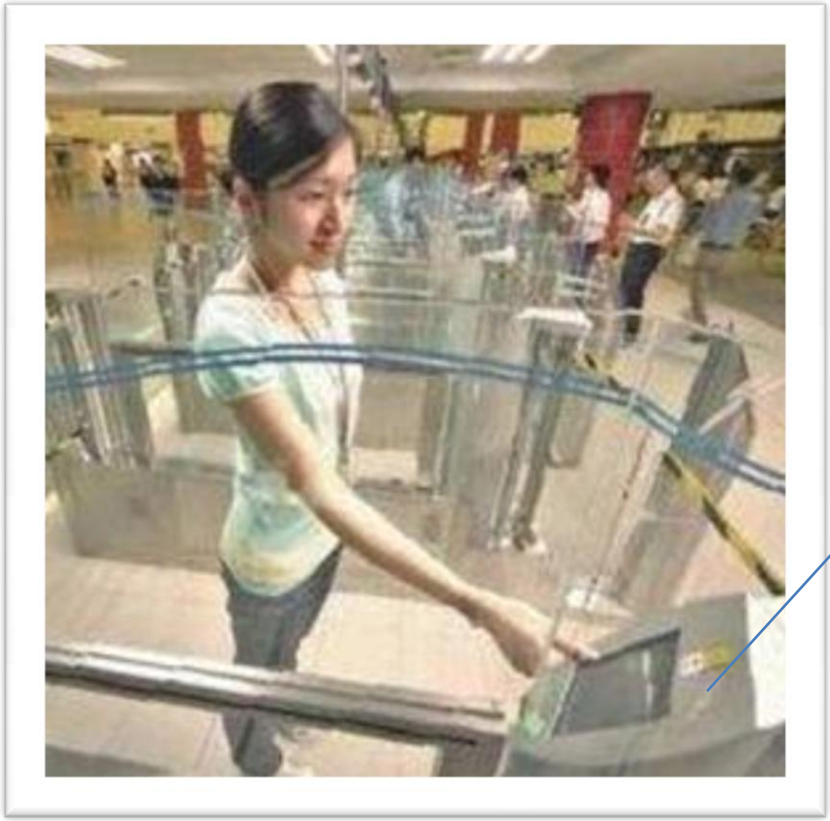






Motivating Problems

- Fingerprint Recognition (e.g., border control)



Motivating Problems

- Face Recognition (security access to buildings etc)

Can Machines Learn to Solve These Problems?

Or, to be more precise

- Can we program machines to learn to do these tasks?

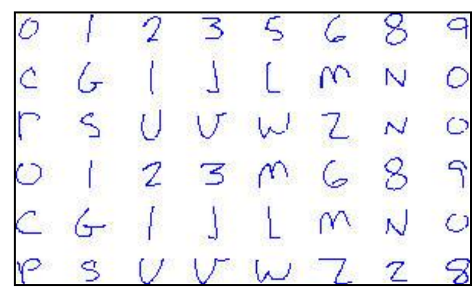
Definition of Learning

- A computer program is said to learn from experience ***E*** with respect to some class of tasks ***T*** and performance measure ***P***, if its performance at tasks in ***T***, as measured by ***P***, improves with experience ***E***

(Mitchell, Machine Learning, McGraw-Hill, 1997)

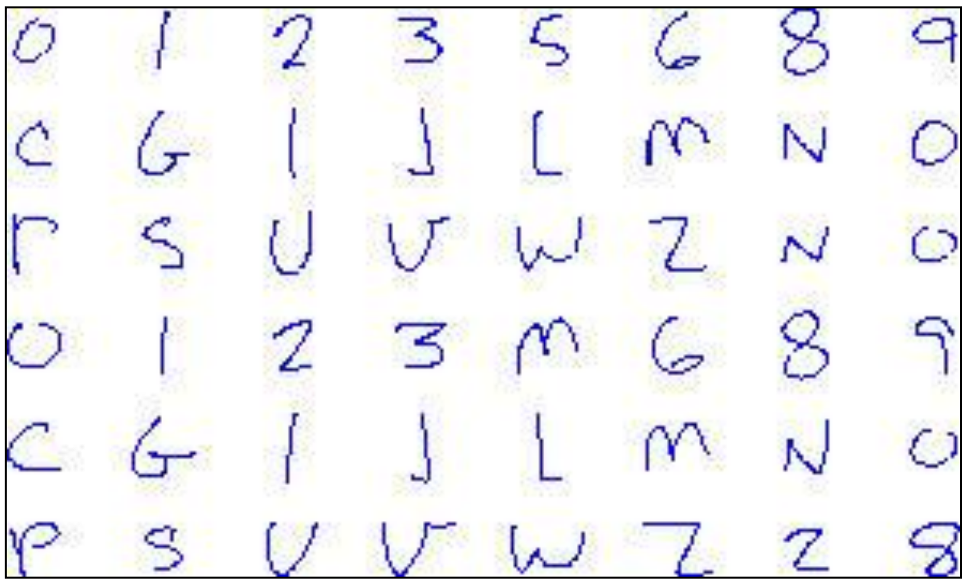
Definition of Learning

- What does this mean exactly?
  - Handwriting recognition problem
    - Task ***T***: Recognizing hand written characters
    - Performance measure ***P***: percent of characters correctly classified
    - Training experience ***E***: a database of handwritten characters with given classifications



Design a Learning System

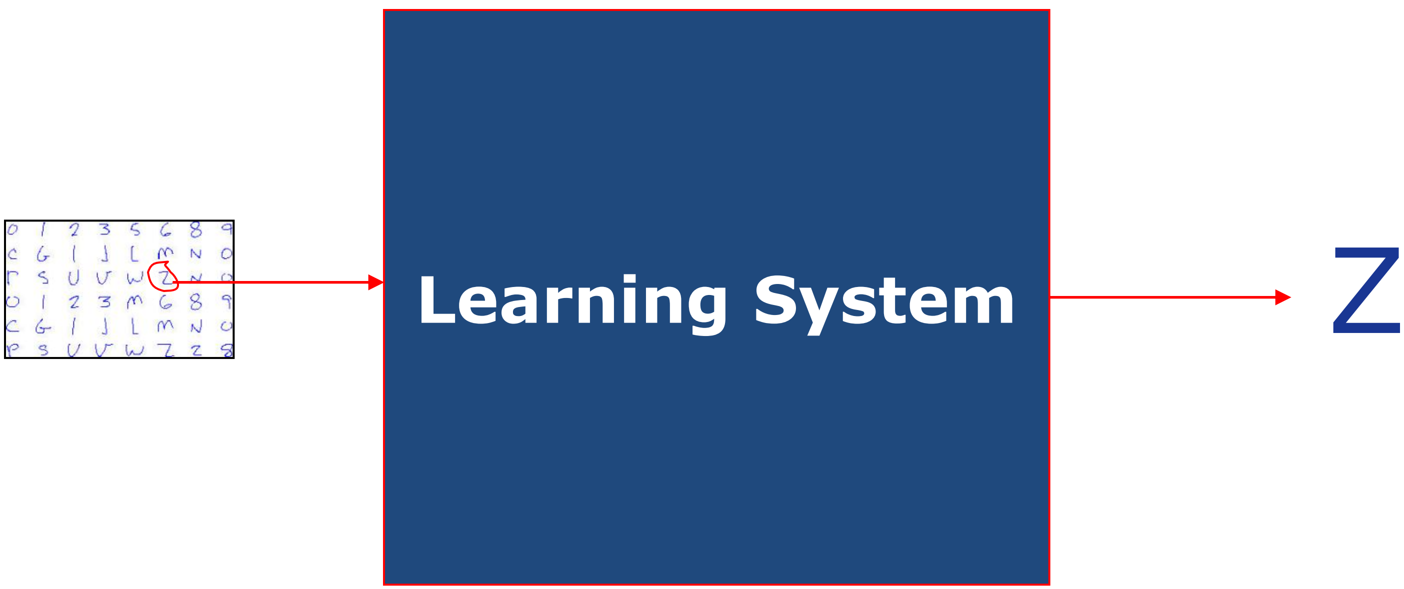
- We shall use handwritten Character recognition as an example to illustrate the design issues and approaches



Design a Learning System

Step 0:

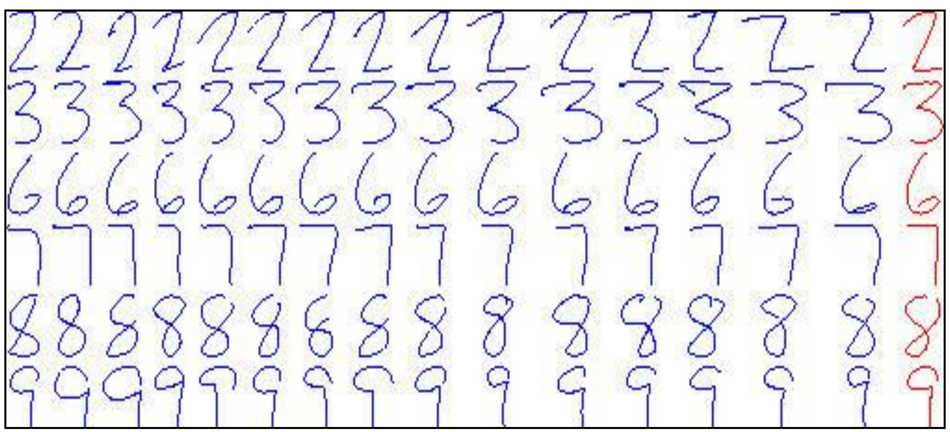
– Lets treat the learning system as a black box



Design a Learning System

Step 1: Collect Training Examples (Experience).

- Without examples, our system will not learn (so-called learning from examples)

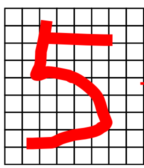


2
3
6
7
8
9

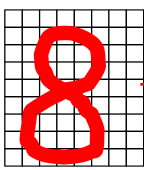
Design a Learning System

Step 2: Representing Experience

– Choose a representation scheme for the experience/examples



(1,1,0,1,1,1,1,1,1,1,0,0,0,0,1,1,1, 1,1,0, …., 1) 64-d Vector



(1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1, 1,1,0, …., 1) 64-d Vector

- The sensor input represented by an n-d vector, called the **feature vector**, **X** = (x_1, x_2, x_3, …, x_n)

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience/examples
  - The sensor input represented by an n-d vector, called the feature vector, **X** = (x1, x2, x3, …, xn)
  - To represent the experience, we need to know what **X** is.
  - So we need a corresponding vector **D**, which will record our knowledge (experience) about **X**
  - The experience **E** is a pair of vectors **E** = (**X**, **D**)

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience/examples

  - The experience **E** is a pair of vectors **E** = (**X**, **D**)

- So, what would **D** be like? There are many possibilities.

Design a Learning System

Step 2: Representing Experience

- So, what would **D** be like? There are many possibilities.

- Assuming our system is to recognise 10 digits only, then D can be a 10-d binary vector; each correspond to one of the digits

D = (d0, d1, d2, d3, d4, d5, d6, d7, d8, d9)

e.g,

if X is digit 5, then d5=1; all others =0

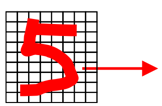
If X is digit 9, then d9=1; all others =0

Design a Learning System

Step 2: Representing Experience

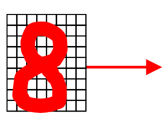
- So, what would **D** be like? There are many possibilities.
- Assuming our system is to recognise 10 digits only, then D can be a 10-d binary vector; each correspond to one of the digits

D = (d0, d1, d2, d3, d4, d5, d6, d7, d8, d9)



X = (1,1,0,1,1,1,1,1,1,1,0,0,0,0,1,1,1, 1,1,0, …., 1); 64-d Vector

D= (0,0,0,0,0,1,0,0,0,0)



X= (1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1, 1,1,0, …., 1); 64-d Vector

D= (0,0,0,0,0,0,0,0,1,0)

Design a Learning System

Step 3: Choose a Representation for the Black Box

- We need to choose a function F to approximate the block box. For a given X, the value of F will give the classification of X. There are considerable flexibilities in choosing F

Design a Learning System

Step 3: Choose a Representation for the Black Box

– F will be a function of some adjustable parameters, or weights, $W = (w1, w2, w3, \ldots w_N)$, which the learning algorithm can modify or learn

Design a Learning System

Step 4: Learning/Adjusting the Weights

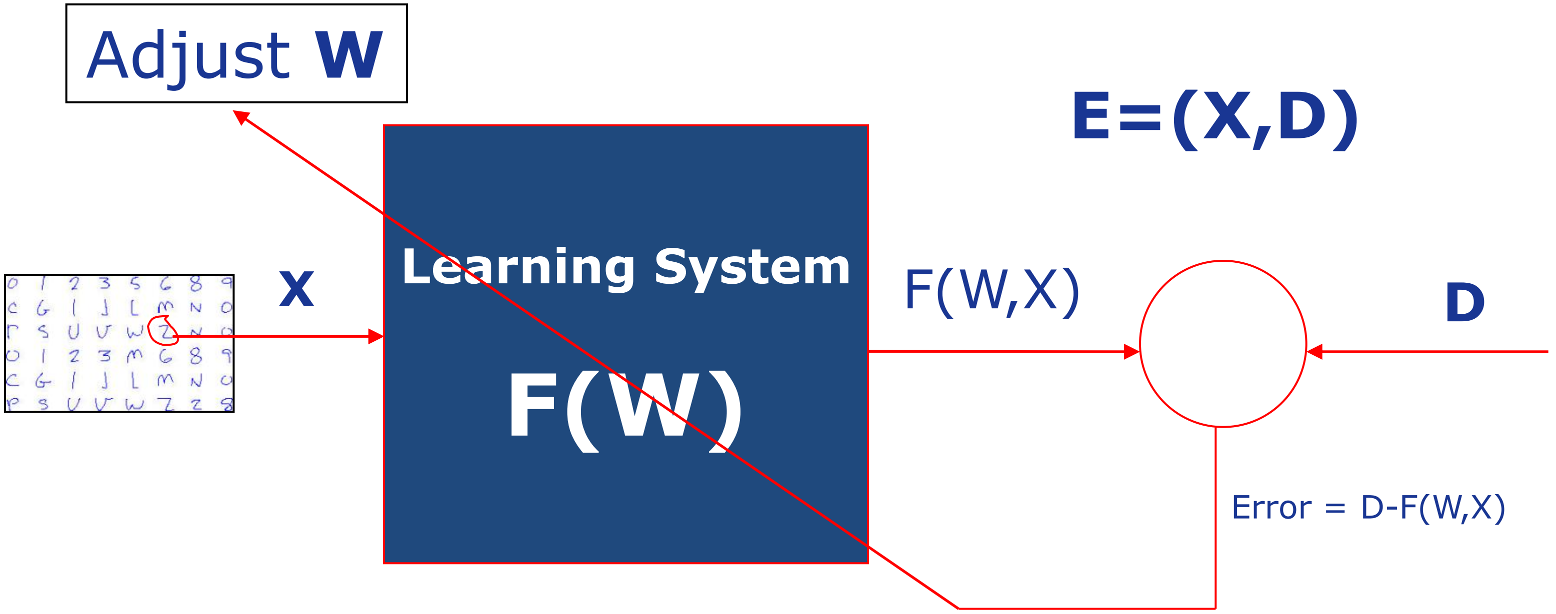
- We need a learning algorithm to adjust the weights such that the experience/prior knowledge from the training data can be learned into the system:

$$E=(X,D)$$

$$F(W,X) = D$$

Design a Learning System

Step 4: Learning/Adjusting the Weights



Design a Learning System

Step 5: Use/Test the System

- Once learning is completed, all parameters are fixed. An unknown input **X** is presented to the system, the system computes its answer according to F(**W**,**X**)

Revision of Some Basic Maths

- Vector and Matrix
  - Row vector/column vector/vector transposition
  - Vector length/norm
  - Inner/dot product
  - Matrix (vector) multiplication
  - Linear algebra
  - Euclidean space
- Basic Calculus
  - Partial derivatives
  - Gradient
  - Chain rule

Revision of Some Basic Maths

- Inner/dot product

$\boldsymbol{x}$ = $[x_1, x_1, \ldots, x_\mathrm{n}]^\mathrm{T}$ **,** $\boldsymbol{y}$ = $[y_1, y_1, \ldots, y_\mathrm{n}]^\mathrm{T}$

Inner/dot product of *x* and *y*, $\mathrm{x}^\mathrm{T}\mathrm{y}$

$$x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = \sum_{i=1}^{n} x_i y_i$$

- Matrix/Vector multiplication

Revision of Some Basic Maths

- Vector space/Euclidean space

  - A vector space V is a set that is closed under finite vector addition and scalar multiplication.

  - The basic example is n-dimensional Euclidean space, where every element is represented by a list of n real numbers

  - An n-dimensional real vector corresponds to **a point** in the Euclidean space.

    [1, 3] is a point in 2-dimensional space

    [2, 4, 6] is point in 3-dimensional space

Revision of Some Basic Maths

- Vector space/Euclidean space
  - Euclidean space (Euclidean distance)

$$\|X - Y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 \ldots + (x_n - y_n)^2}$$

  - Dot/inner product and Euclidean distance
    - Let *x* and *y* are two normalized n vectors, ||*x*||= 1, ||*y*||=1, we can write

$$\|X - Y\|^2 = (X - Y)^T (X - Y) = 2 - 2X^T Y$$

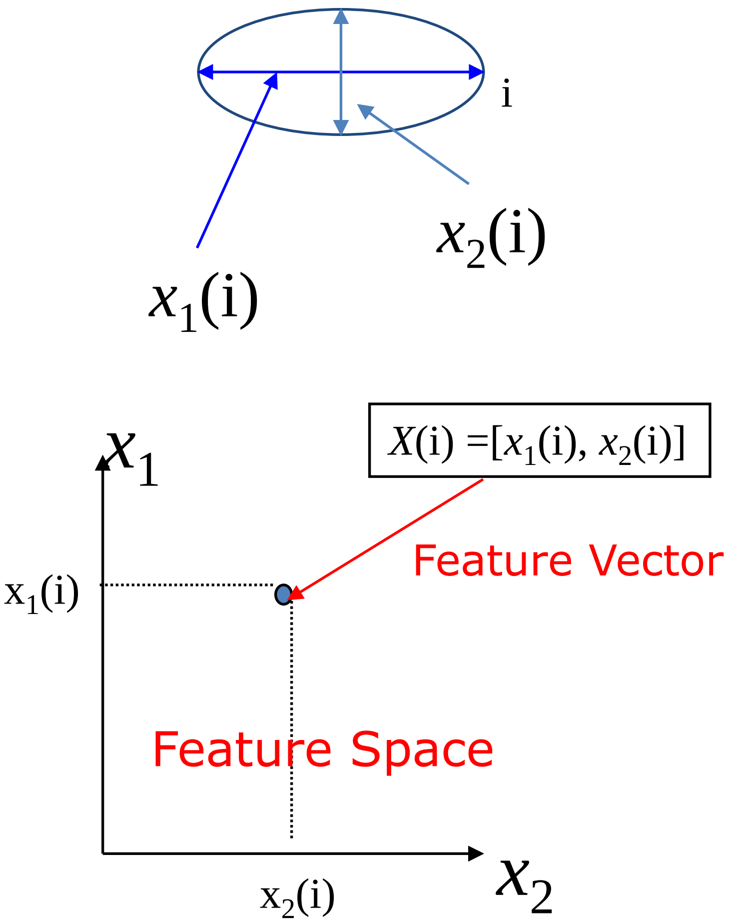
    - Minimization of Euclidean distance between two vectors corresponds to maximization of their inner product.
  - Euclidean distance/inner product as similarity measure

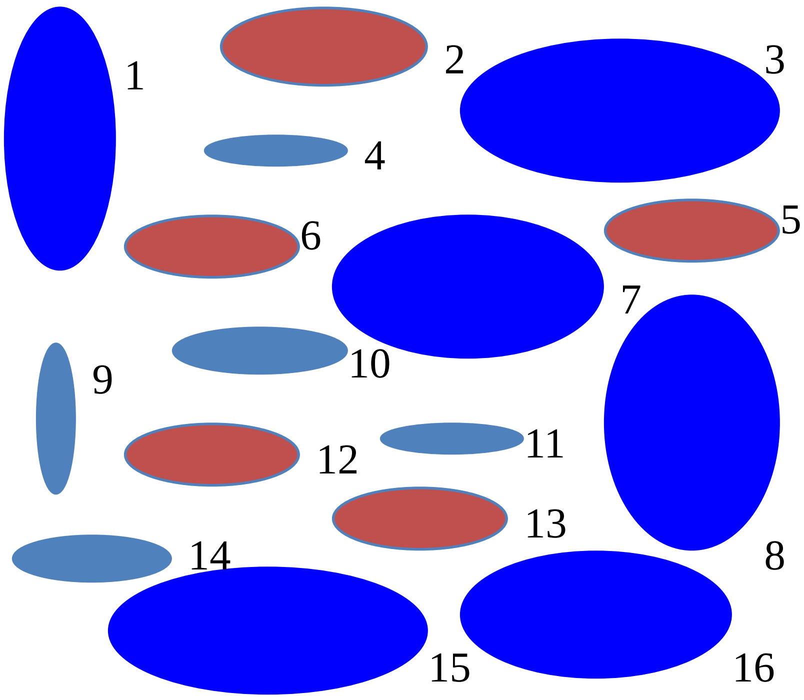
Revision of Some Basic Maths

- Basic Calculus
  - Multivariable function: $$y(x) = f(x_1, x_2, \dots, x_n)$$
  - Partial derivative: gives the direction and speed of change of *y*, with respect to *x*i
  - Gradient $$\nabla f = \left[\frac{\partial f}{\partial x_1}, \dots\dots \frac{\partial f}{\partial x_n}\right]$$
  - **Chain rule**: Let *y* = *f* (*g*(*x*)), *u* = *g*(*x*), then $$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$
  - Let *z* = *f*(*x*, *y*), *x* = *g*(*t*), *y* = *h*(*t*), then $$\frac{dz}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

Feature Space

- Representing real world objects using **feature vectors**

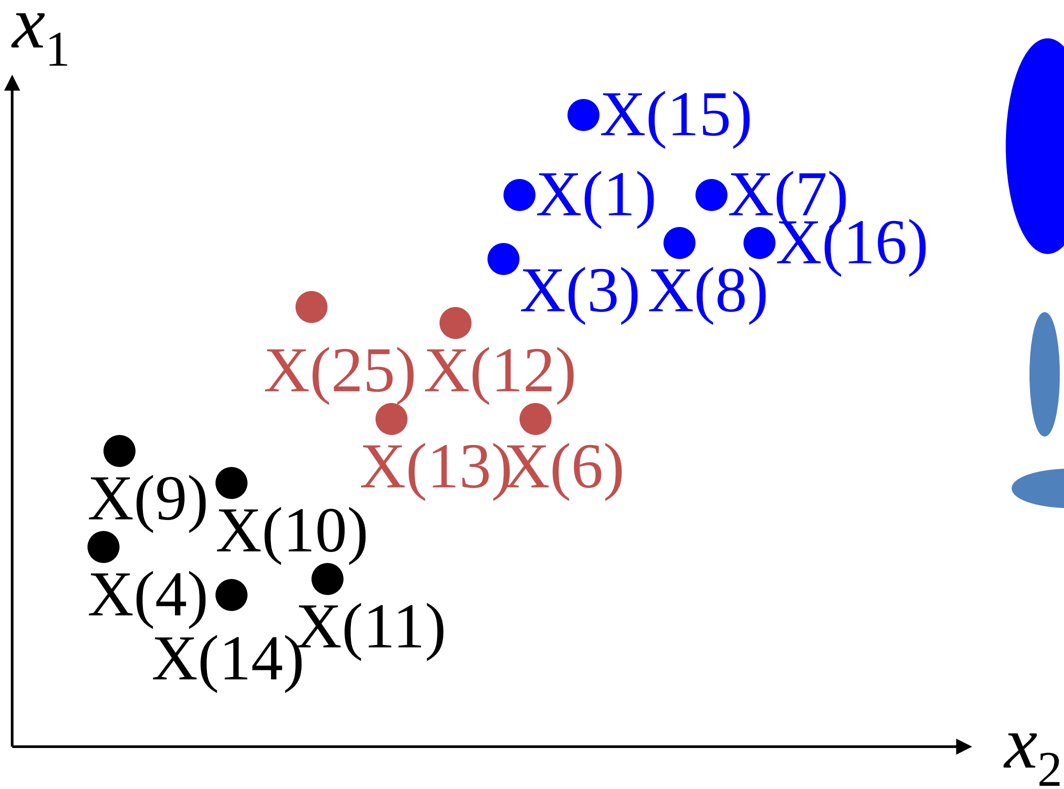


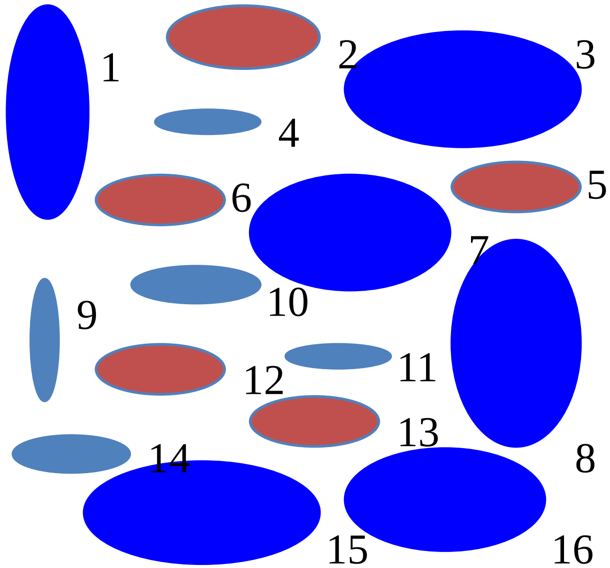


Elliptical blobs (objects)

Feature Space

- From Objects to Feature Vectors to Points in the Feature Spaces





Elliptical blobs (objects)

Representing General Objects

- ✪ Feature vectors of
  - Faces
  - Cars
  - Fingerprints
  - Gestures
  - Emotions (a smiling face, a sad expression etc)
  - …

Further Reading

- T. M. Mitchell, Machine Learning, McGraw-Hill International Edition, 1997

  Chapter 1

Tutorial/Exercise Questions

1. Describe informally in one paragraph of English, the task of learning to recognize handwriting numerical digits.

2. Describe the various steps involved in designing a learning system to perform the task of question 1, give as much detail as possible the tasks that have to be performed in each step.

3. For the tasks of learning to recognize human faces and fingerprints respectively, redo questions 1 and 2.

4. In the lecture, we used a very long binary vector to represent the handwriting digits, can you think of other representation methods?