Introduction to Machine Learning

Ning Xiong

Mälardalen University

Agenda

- What is machine learning?
- Three classes of learning problems (supervised, unsupervised, reinforcement)
- Overview of some learning techniques
  - k-means clustering
  - linear regression
  - logistic regression
  - artificial neural network
  - Q-learning

Part 1: What is Machine learning

Inspiration from Human

Learn from experience

Learn from data

Follow instruction

What is Machine Learning

Machine Learning is to build computer programs that can improve with experience at some task

- improve over some Task T
- with respect to a performance measure P
- based on experience E

Applications of ML

- Learning to recognize spoken words
- Learning to drive autonomous vehicles
  (Google self-driving car)
- Learning to diagnose in medical environments
- Learning for diagnosis and progonosis for machine maintenance
- Learning for adaptive control of robots or industrial processes

Three classes of learning problems

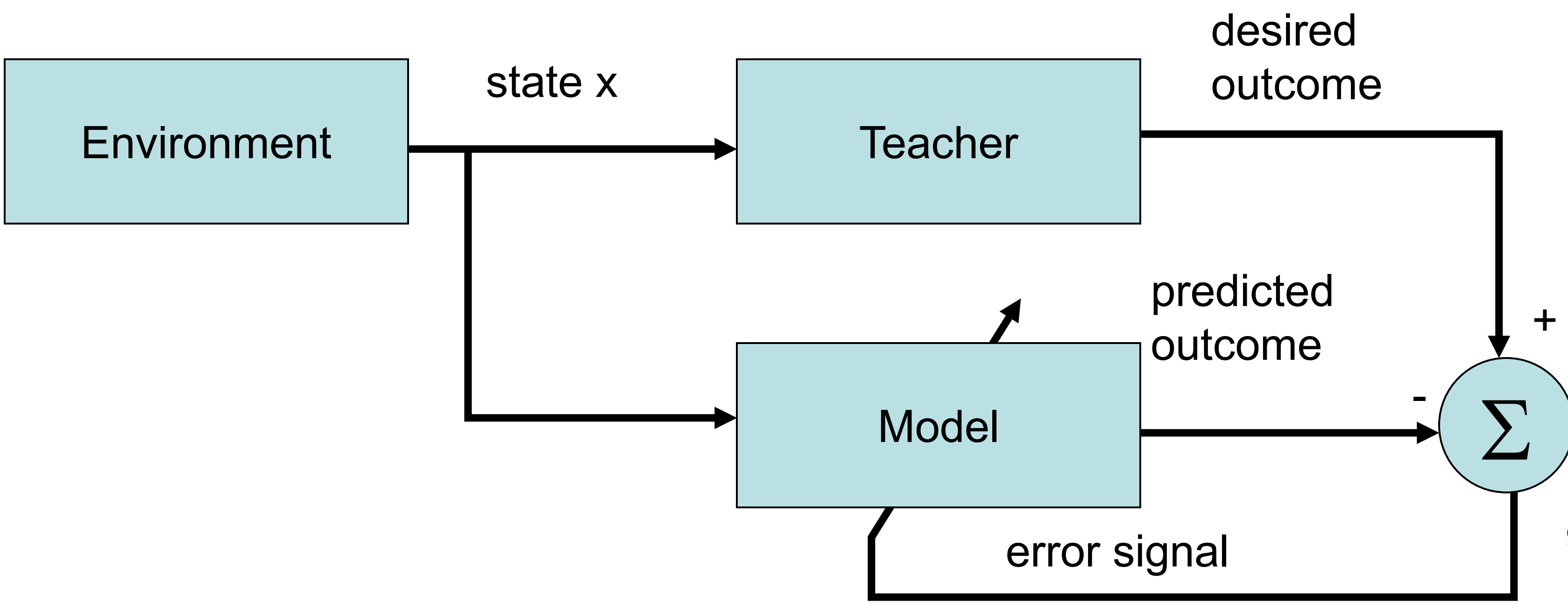
Learning Problems

The nature of experience causes different kinds of learning problems

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised Learning

- A teacher available to give desired outcome
- Direct experience represented by a set of input-output examples ($\mathbf{x}_i$,$\mathbf{y}_i$), as training examples
- minimize the error between the predicted outcome of the learner and the desired outcome

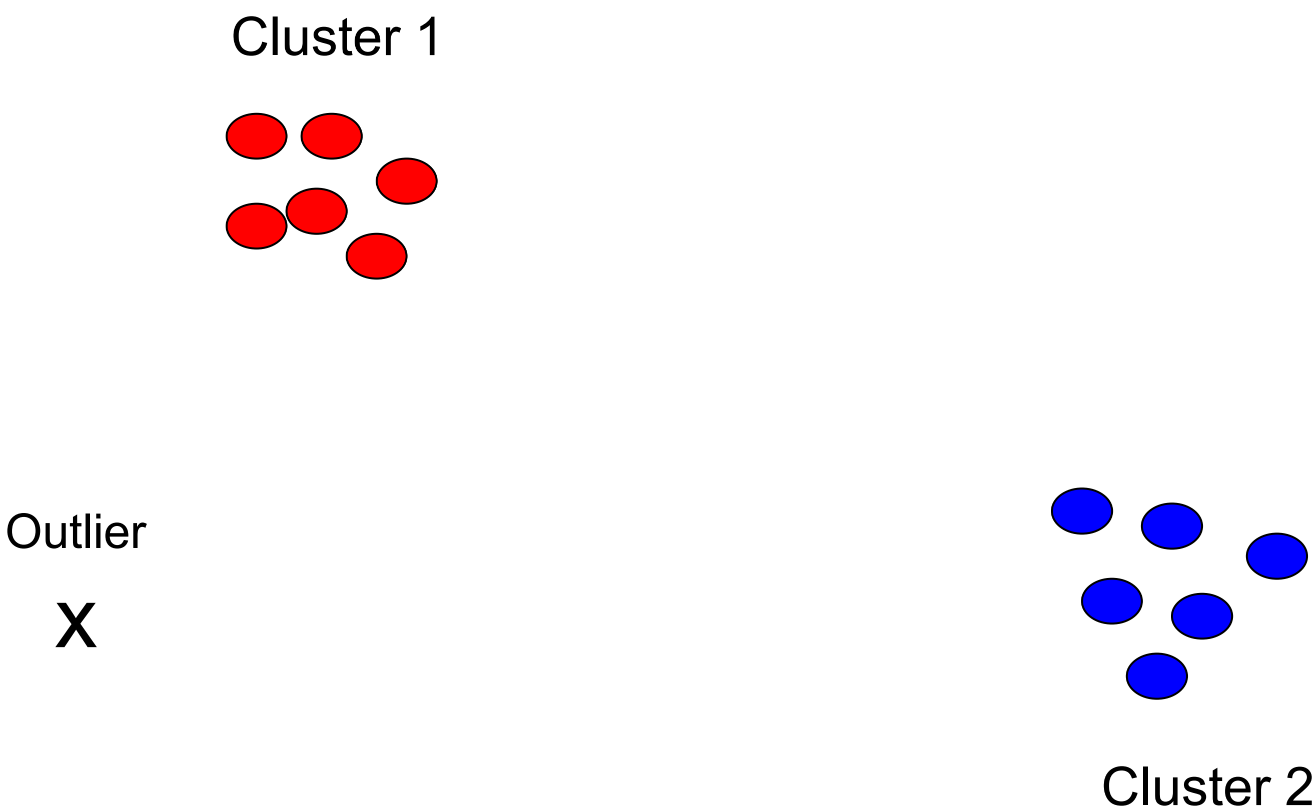


Unsupervised Learning

- No teacher or supervisor available
- No desired outputs in the experiences
- Clustering: Grouping similar instances
- Data dimension reduction: creating fewer new features

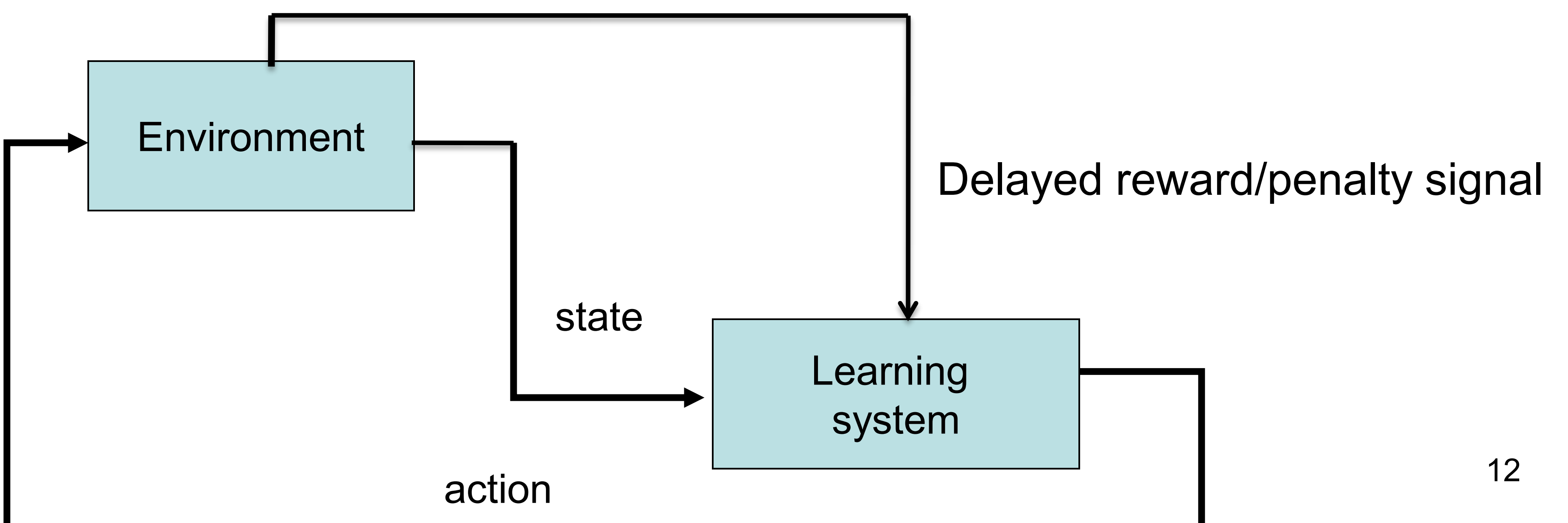
Unsupervised Learning: Clustering

Objects similar in their attributes are clustered in the same group



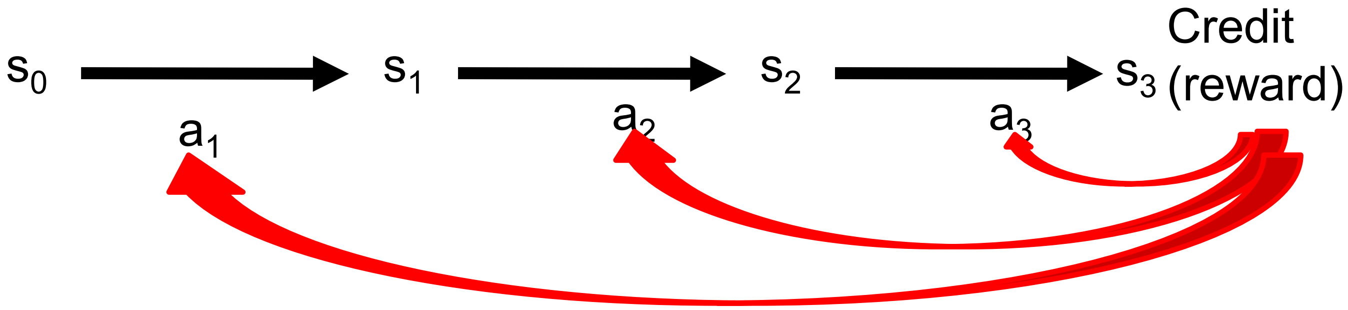
Reinforcement Learning

- Aim to obtain optimal action strategy without teacher
- learning through interaction with the environment
- exploration of states and actions
- goal: maximize accumulated rewards for the long term
- Indirect experience in form of delayed reward signal (temporal credit assignment problem)



Credit Assignment Problem

Assigning credit or blame for the overall outcomes (delayed reward) to each of the internal decisions made by the learning machine which contributed to the overall outcomes



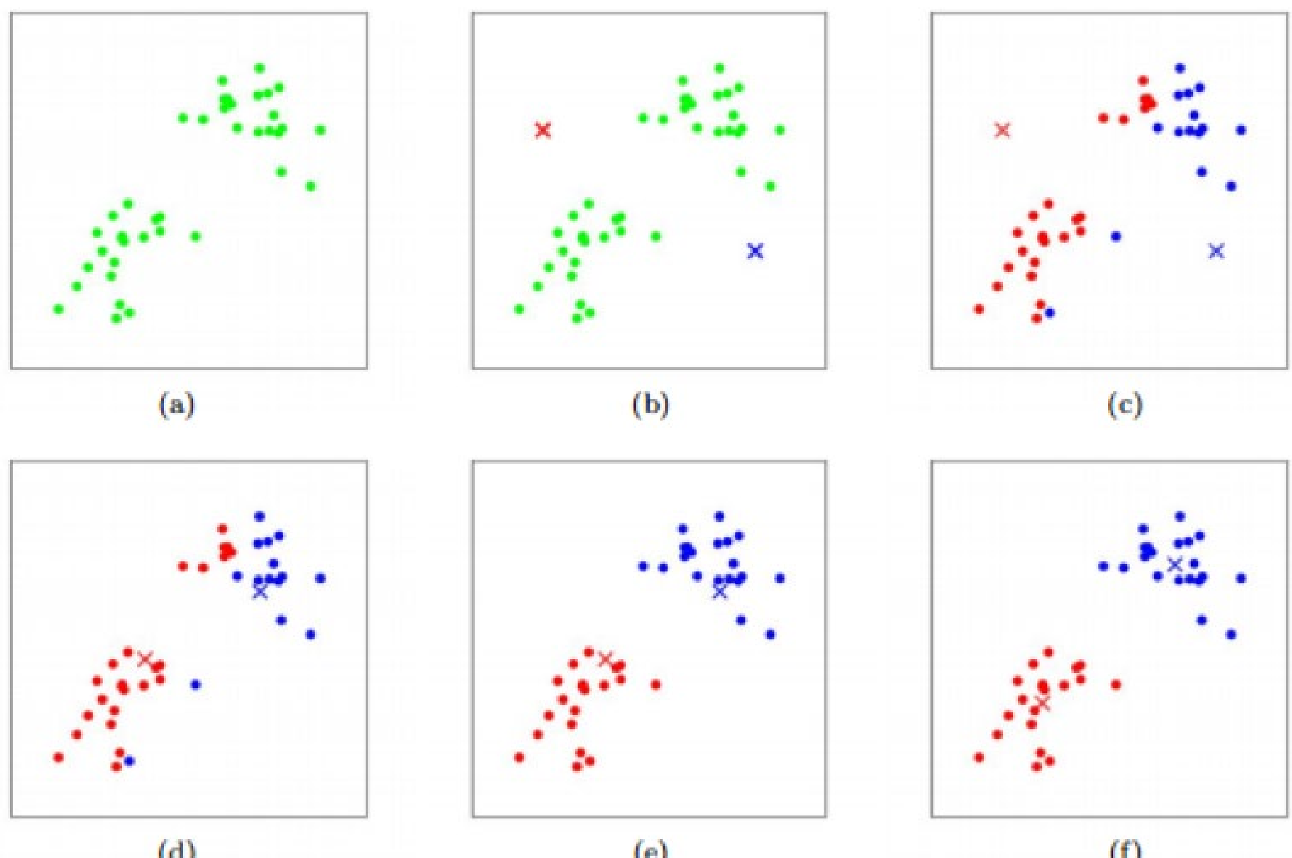
Overview of learning techniques

K-Means Clustering (unsupervised)

Task: divide the examples in to K clusters based on similarity

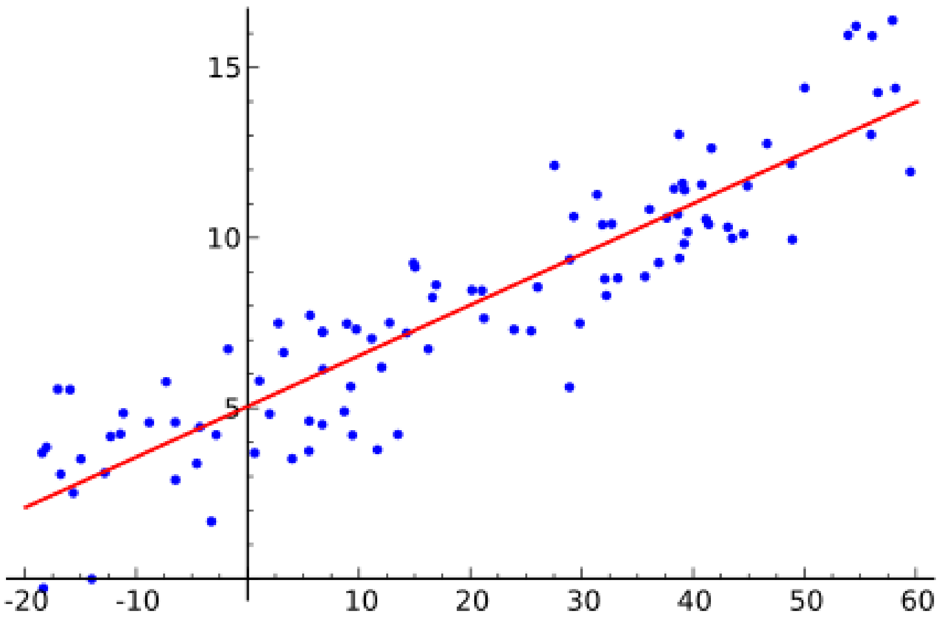
Initialize: number of clusters K; the initial centroids of the clusters

Iteration: assign data to clusters → recalculate the centroids



Linear Regression (Supervised)

Build a linear function to model the relation of output variable y with the input variables $x_1, \ldots., x_n$



$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Finding the parameters to best fit the training samples

$$\min \quad \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

Linear Regression

Suppose (x_{i1}, x_{i2}, …., x_{in}, y_i) (i=1 … m) are the training examples.

Let

$$\boldsymbol{X}=\begin{bmatrix} 1 & x_{11} & & x_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{m1} & & x_{mn} \end{bmatrix} \qquad \boldsymbol{Y}=\begin{bmatrix} y_1 \\ \cdots \\ y_m \end{bmatrix} \qquad \boldsymbol{W}=\begin{bmatrix} w_0 \\ \cdots \\ w_n \end{bmatrix}$$

$$E(\mathbf{W})=\frac{1}{m}\|\boldsymbol{XW}-\boldsymbol{Y}\|^2 = \frac{1}{m}(\boldsymbol{XW}-\boldsymbol{Y})^T(\boldsymbol{XW}-\boldsymbol{Y})$$

$$=\frac{1}{m}(\boldsymbol{W}^T\boldsymbol{X}^T\boldsymbol{XW} - 2\boldsymbol{W}^T\boldsymbol{X}^T\boldsymbol{Y} + \boldsymbol{Y}^T\boldsymbol{Y})$$

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}} = (\mathbf{2}\boldsymbol{X}^T\mathbf{XW} - 2\boldsymbol{X}^T\boldsymbol{Y})/m = 0$$

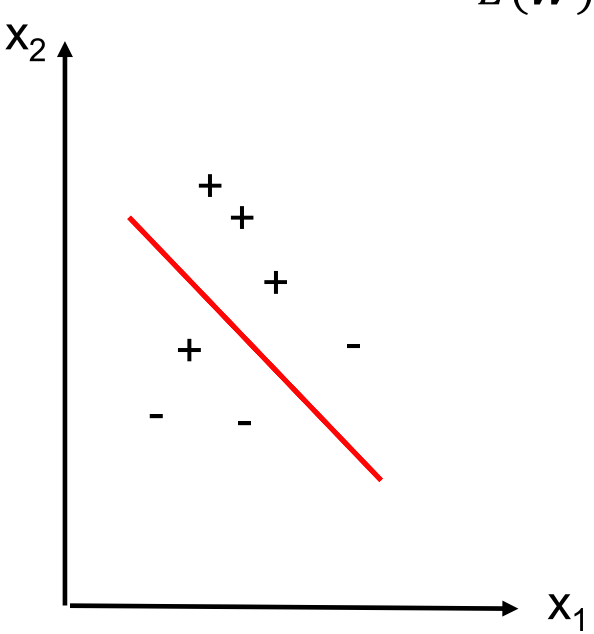
$$\boldsymbol{W} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

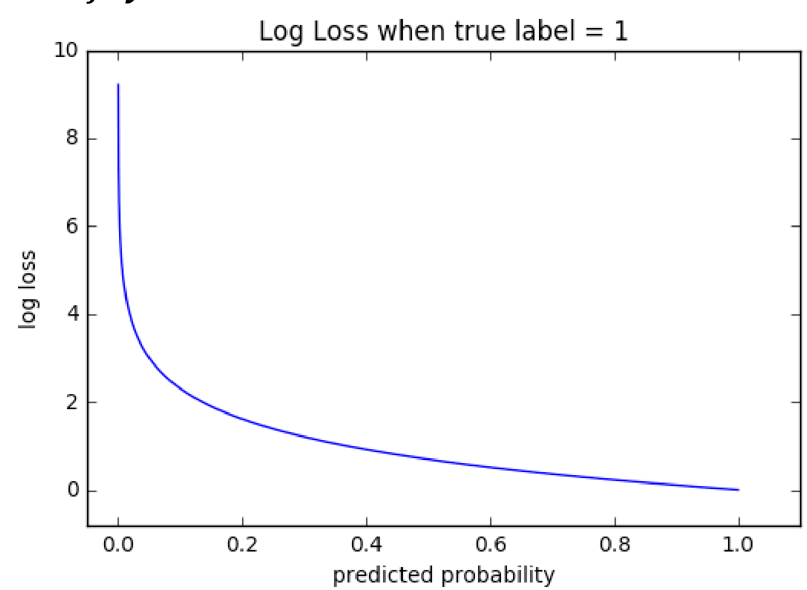
Least square estimation

Logistic Regression (Supervised)

- Build a linear surface to separate training examples into two classes
- Using linear surface to convert to class probability: $z = \dfrac{e^{w_0+w_1x_1+w_2x_2}}{1+e^{w_0+w_1x_1+w_2x_2}}$
- Finding parameters to minimize log loss function

$$E(\boldsymbol{W}) = \begin{cases} -\log(Z) & if\ y = 1 \\ -\log(1-z) & if\ y = 0 \end{cases}$$

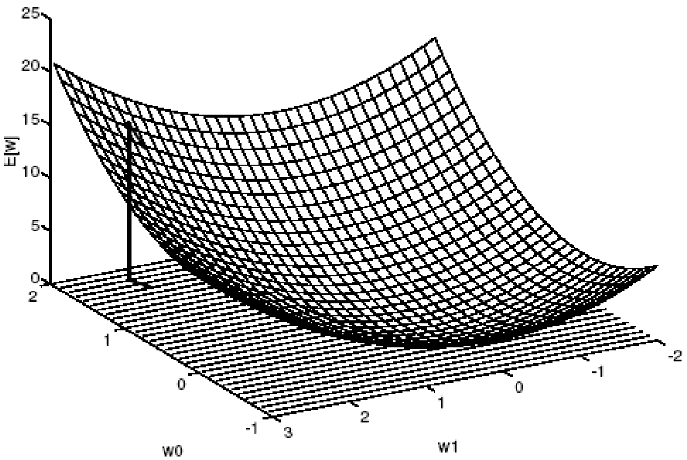




Gradient descent for learning

$$Gradient = [\frac{\partial E}{\partial W_j}]$$

$$w_j = w_j - \eta \frac{\partial E}{\partial w_j}$$



η is the learning rate specifying the step size in the gradient search

Getting the gradient

Rewrite the loss function for a training example as

$$E(W) = -ylog(z) - (1-y)\log(1-z)$$

z: the predicted probability for the example

$$z = \frac{e^{w_0+\sum_{j=1}^{n} w_j x_j}}{1+e^{w_0+\sum_{j=1}^{n} w_j x_j}} = \frac{1}{1+e^{-(w_0+\sum_{j=1}^{n} w_j x_j)}}$$

y: the desired probability (1 or 0) for the example

$$\frac{\partial E}{\partial w_j} = (z-y)x_j \qquad \frac{\partial E}{\partial w_0} = (z-y)$$

Learning for logistic regression

Initialize parameters w_i with random values

Repeat

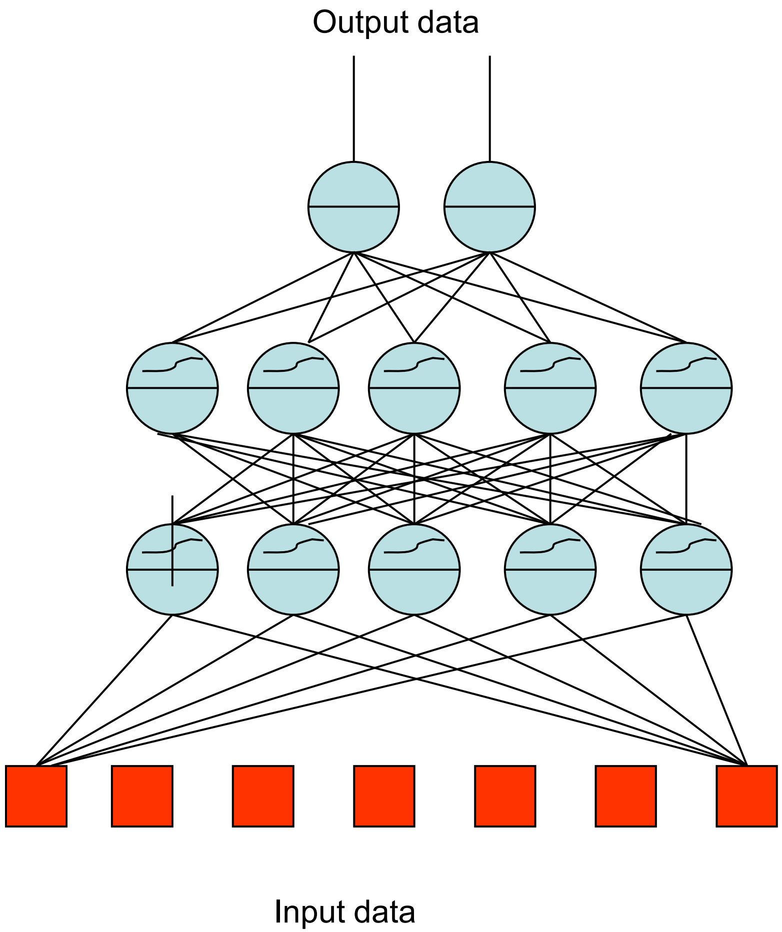
For each training example $\{(x_1,\ldots x_n)^p, y^p\}$ do

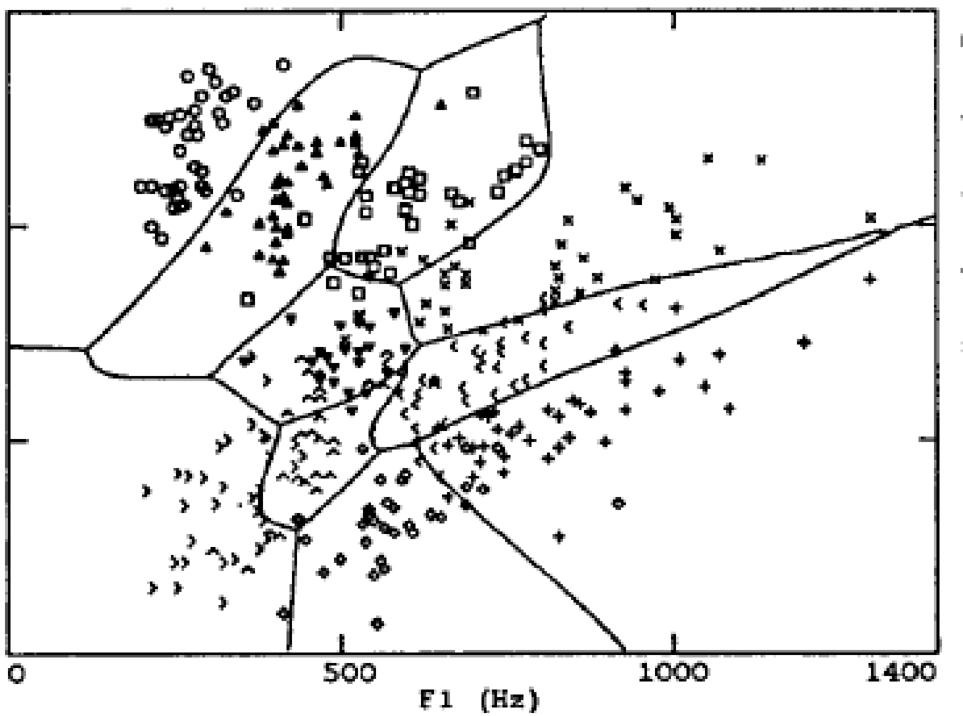
1. Use the inputs $(x_1,\ldots,x_n)^p$ to calculate the predicted probability
2. Compute the gradients for all parameters
3. Update all the parameters based on gradients

until a certain criterion is satisfied

Artificial Neural Network (ANN)

To directly represent complex nonlinear relation, we can build an artificial neural network consisting of many units.



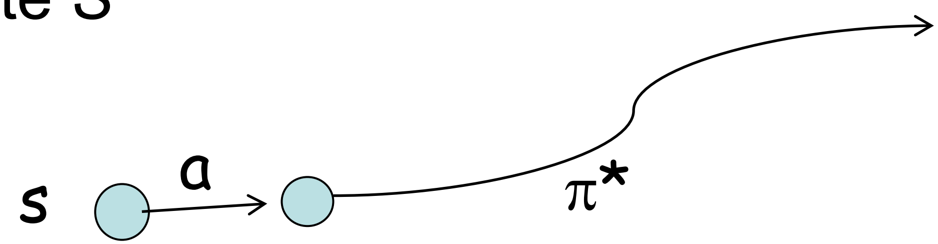


- More details of ANN and its learning will be presented in Lecture 8

Q learning (reinforcement)

- Goal: Learn optimal action policy π^*: s → a, to maximize the total sum of rewards

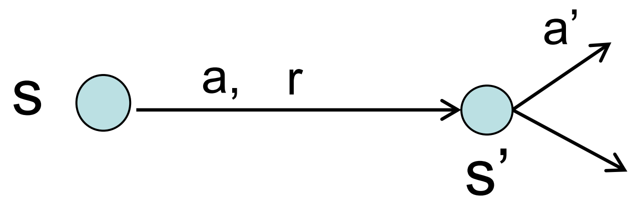
- Target function: $Q^*(s,a)$ referred as optimal value of action in state *S*



- $Q^*(s, a)$ is defined as the accumulated reward that will be obtained from state *s* by first doing action *a* then following optimal policy

- Optimal decision making: $\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a)$

Q learning rule



$$Q^*(s,a) = r + \gamma \max_{\forall a'} Q^*(s',a')$$

$$\Downarrow$$

Q learning rule: $Q(s,a) = r + \gamma \max_{\forall a'} Q\,(s',a')$

Through many interactions: $Q(s,a) \rightarrow Q^*(s,a)$

A table for estimtes of Q* values

	a_1	a_2	…..	a_m
s_1	*	*	*	*
s_2	*	*	*	*
…	*	*	*	*
s_n	*	*	*	*

- Each time update one entity in the table following the Q-learning rule