

WPI

Artificial Intelligence

CS 534

Week 5



Let's review the basic
clustering algorithms

Major categories of clustering algorithms

- Sequential: A single clustering is produced. One or few sequential passes on the data
- Hierarchical: A sequence of (nested) clusterings is produced
 - Agglomerative
 - Matrix theory
 - Graph theory
 - Divisive
 - Combinations of the above (e.g., the Chameleon algorithm.)

Basic sequential algorithmic scheme (BSAS)

- $m = 1 \setminus \{\text{number of clusters}\} \setminus$
- $C_m = \{\mathbf{x}_1\}$
- For $i=2$ to N
 - Find C_k : $d(\mathbf{x}_i, C_k) = \min_{1 \leq j \leq m} d(\mathbf{x}_i, C_j)$
 - If $(d(\mathbf{x}_i, C_k) > \Theta)$ AND $(m < q)$ then
 - $m = m + 1$
 - $C_m = \{\mathbf{x}_i\}$
 - Else
 - $C_k = C_k \cup \{\mathbf{x}_i\}$
 - Where necessary, update representatives (*)
 - End {if}
- End {for}

Basic sequential algorithmic scheme (BSAS)

- $m = 1 \setminus \{\text{number of clusters}\} \setminus$
- $C_m = \{\mathbf{x}_1\}$
- For $i=2$ to N
 - Find C_k : $d(\mathbf{x}_i, C_k) = \min_{1 \leq j \leq m} d(\mathbf{x}_i, C_j)$
 - If $(d(\mathbf{x}_i, C_k) > \Theta)$ AND $(m < q)$ then
 - $m = m + 1$
 - $C_m = \{\mathbf{x}_i\}$
 - Else
 - $C_k = C_k \cup \{\mathbf{x}_i\}$
 - Where necessary, update representatives (*)
 - End {if}
- End {for}

(*) When the mean vector \mathbf{m}_C is used as representative of the cluster C with n_C elements, the updating in the light of a new vector \mathbf{x} becomes:

$$\mathbf{m}_C^{\text{new}} = (n_C \mathbf{m}_C + \mathbf{x}) / (n_C + 1)$$

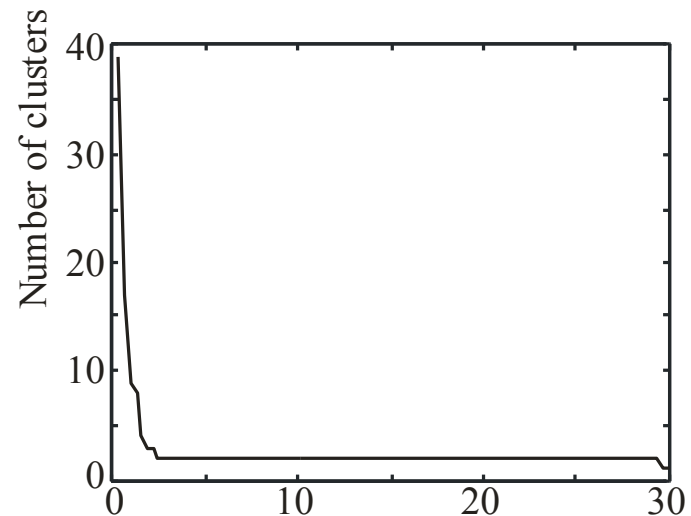
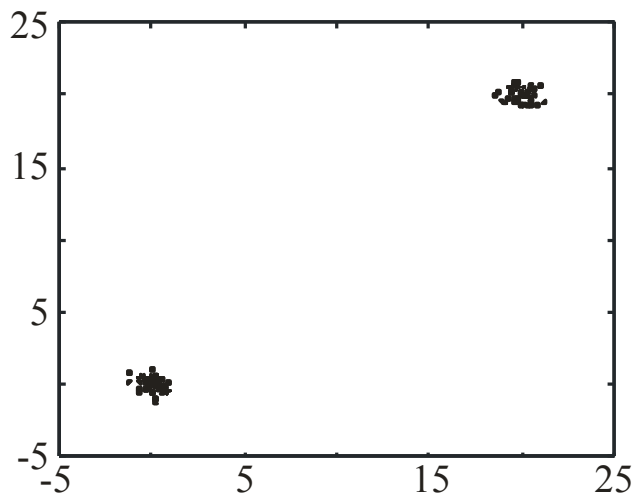
Remarks

- The order of presentation of the data in the algorithm plays important role in the clustering results. Different order of presentation may lead to totally different clustering results, in terms of the number of clusters as well as the clusters themselves
- In BSAS the decision for a vector x is reached prior to the final cluster formation
- BSAS perform a single pass on the data. Its complexity is $O(N)$
- If clusters are represented by point representatives, compact clusters are favored

Estimating number of clusters in the data set

Let $BSAS(\Theta)$ denote the $BSAS$ algorithm when the dissimilarity threshold is Θ

- For $\Theta = a$ to b step c
 - Run s times $BSAS(\Theta)$, each time presenting the data in a different order
 - Estimate the number of clusters m_Θ , as the most frequent number resulting from the s runs of $BSAS(\Theta)$
- Next Θ
- Plot m_Θ versus Θ and identify the number of clusters m as the one corresponding to the widest flat region in the above graph.



A two-threshold sequential scheme (TTSAS)

- The formation of the clusters, as well as the assignment of vectors to clusters, is carried out concurrently (like **BSAS** and **unlike MBSAS**)
- Two thresholds Θ_1 and Θ_2 ($\Theta_1 < \Theta_2$) are employed
- The **general idea** is the following:
 - If the distance $d(\mathbf{x}, C)$ of \mathbf{x} from its closest cluster, C , is greater than Θ_2 then:
 - *A new cluster represented by \mathbf{x} is formed.*
 - Else if $d(\mathbf{x}, C) < \Theta_1$ then
 - *\mathbf{x} is assigned to C .*
 - Else
 - *The decision is postponed to a later stage.*
 - End {if}

The unassigned vectors are presented iteratively to the algorithm until all of them are classified

Remarks

- In practice, a few passes (≥ 2) of the data set are required
- TTSAS is less sensitive to the order of data presentation, compared to BSAS

Refinement stages

The problem of **closeness of clusters**: *"In all the above algorithms it may happen that two formed clusters lie very close to each other"*.

A simple merging procedure:

- (A) Find C_i, C_j ($i < j$) such that
$$d(C_i, C_j) = \min_{k, r=1, \dots, m, k \neq r} d(C_k, C_r)$$
- If $d(C_i, C_j) \leq M_1$ then
 - $\backslash \{M_1 \text{ is a user-defined threshold} \backslash \}$
 - Merge C_i, C_j to C_i and eliminate C_j
 - If necessary, update the cluster representative of C_i
 - Rename the clusters C_{j+1}, \dots, C_m to C_j, \dots, C_{m-1} , respectively
 - $m = m - 1$
 - Go to (A)
- Else
 - Stop
- End {if}

The problem of sensitivity to the order of data presentation

- The problem of **sensitivity to the order of data presentation**:
"A vector x may have been assigned to a cluster C_i at the current stage but another cluster C_j may be formed at a later stage that lies closer to x "

A simple reassignment procedure

- For $i=1$ to N
 - Find C_j such that $d(\mathbf{x}_i, C_j) = \min_{k=1, \dots, m} d(\mathbf{x}_i, C_k)$
 - Set $b(i)=j$ \{ $b(i)$ is the index of cluster that lies closest to \mathbf{x}_i \}
- End {for}
- For $j=1$ to m
 - Set $C_j = \{\mathbf{x}_i \in X : b(i)=j\}$
 - If necessary, update representatives
- End {for}

Hierarchical clustering algorithms

- Produce a **hierarchy** of (**hard**) clusterings instead of a **single** clustering
- Applications in:
 - Social sciences
 - Biological taxonomy
 - Modern biology
 - Medicine
 - Archaeology
 - Computer science and engineering

Definition

Let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\mathbf{x}_i = [x_{i1}, \dots, x_{il}]^T$. Recall that:

- In hard clustering each vector belongs **exclusively** to a single cluster.
- An m -(hard) clustering of X , K , is a partition of X into m sets (clusters) C_1, \dots, C_m , so that:

- $C_i \neq \emptyset, i = 1, 2, \dots, m$
- $\bigcup_{i=1}^m C_i = X$
- $C_i \cap C_j = \emptyset, i \neq j, i, j = 1, 2, \dots, m$

By the definition: $K = \{C_j, j=1, \dots, m\}$

- Definition: A clustering K_1 containing k clusters is said to be **nested** in the clustering K_2 containing r ($< k$) clusters, if **each** cluster in K_1 is a subset of a cluster in K_2 .

We write $K_1 \subset K_2$

➤ **Example:** Let $K_1 = \{\{x_1, x_3\}, \{x_4\}, \{x_2, x_5\}\}$, $K_2 = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}\}$,

$$K_3 = \{\{x_1, x_4\}, \{x_3\}, \{x_2, x_5\}\}, K_4 = \{\{x_1, x_2, x_4\}, \{x_3, x_5\}\}.$$

It is $K_1 \subset K_2$, **but not** $K_1 \subset K_3$, $K_1 \subset K_4$, $K_1 \subset K_1$.

➤ **Remarks:**

- Hierarchical clustering algorithms produce a **hierarchy of nested clusterings**
- They involve N steps at the most.
- At each step t , the clustering K_t is produced by K_{t-1} .

➤ **Main categories:**

- **Agglomerative** clustering algorithms: Here $K_0 = \{\{x_1\}, \dots, \{x_N\}\}$, $K_{N-1} = \{\{x_1, \dots, x_N\}\}$ and $K_0 \subset \dots \subset K_{N-1}$.
- **Divisive** clustering algorithms: Here $K_0 = \{\{x_1, \dots, x_N\}\}$, $K_{N-1} = \{\{x_1\}, \dots, \{x_N\}\}$ and $K_{N-1} \subset \dots \subset K_0$.

Agglomerative Algorithms

- Let $g(C_i, C_j)$ a proximity function between two clusters of X .

Generalized Agglomerative Scheme (GAS)

– Initialization

- Choose $K_0 = \{\{\mathbf{x}_1\}, \dots, \{\mathbf{x}_N\}\}$
- $t=0$

– Repeat

- $t=t+1$
- Choose (C_i, C_j) in K_{t-1} such that

$$g(C_i, C_j) = \begin{cases} \min_{r,s} g(C_r, C_s), & \text{if } g \text{ is a disim. function} \\ \max_{r,s} g(C_r, C_s), & \text{if } g \text{ is a sim. function} \end{cases}$$

- Define $C_q = C_i \cup C_j$ and produce $K_t = (K_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$

– Until all vectors lie in a single cluster.

Remarks

- If two vectors come together into a single cluster at level t of the hierarchy, they will remain in the same cluster for all subsequent clusterings. As a consequence, there **is no way** to recover a “**poor**” clustering that may have occurred in an earlier level of hierarchy
- Number of operations: $O(N^3)$

Let's continue ...

Definitions of some useful quantities:

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, with $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{il}]^T$.

- **Pattern matrix** ($D(X)$): An $N \times l$ matrix whose i -th row is \mathbf{x}_i (transposed).
- **Proximity (similarity or dissimilarity) matrix** ($P(X)$): An $N \times N$ matrix whose (i, j) element equals the proximity $p(\mathbf{x}_i, \mathbf{x}_j)$ (similarity $s(\mathbf{x}_i, \mathbf{x}_j)$ or dissimilarity $d(\mathbf{x}_i, \mathbf{x}_j)$)

Example 1: Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$, with:

$$\mathbf{x}_1 = [1, 1]^T, \mathbf{x}_2 = [2, 1]^T, \mathbf{x}_3 = [5, 4]^T, \mathbf{x}_4 = [6, 5]^T, \mathbf{x}_5 = [6.5, 6]^T$$

Euclidean distance

$$D(X) = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 5 & 4 \\ 6 & 5 \\ 6.5 & 6 \end{bmatrix}$$

$$P(X) = \begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$$

Tanimoto pattern

$$P'(X) = \begin{bmatrix} 1 & 0.75 & 0.26 & 0.21 & 0.18 \\ 0.75 & 1 & 0.44 & 0.35 & 0.20 \\ 0.26 & 0.44 & 1 & 0.96 & 0.90 \\ 0.21 & 0.35 & 0.96 & 1 & 0.98 \\ 0.18 & 0.20 & 0.90 & 0.98 & 1 \end{bmatrix}$$

Threshold dendrogram

- **Threshold dendrogram** (or **dendrrogram**): It is an effective way of representing the sequence of clusterings which are produced by an agglomerative algorithm.

In the previous example, if $d_{\min}^{ss}(C_i, C_j)$ is employed as the distance measure between two sets and the Euclidean one as the distance measure between two vectors, the following series of clusterings are produced:

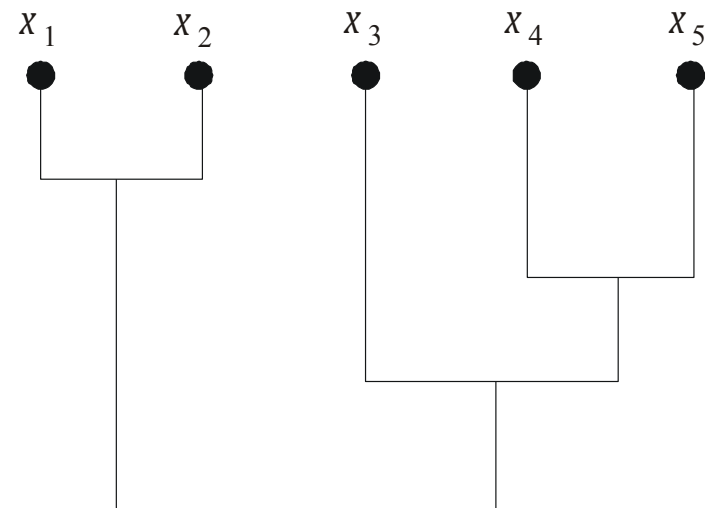
$\{\{X_1\}, \{X_2\}, \{X_3\}, \{X_4\}, \{X_5\}\}$

$\{\{X_1, X_2\}, \{X_3\}, \{X_4\}, \{X_5\}\}$

$\{\{X_1, X_2\}, \{X_3\}, \{X_4, X_5\}\}$

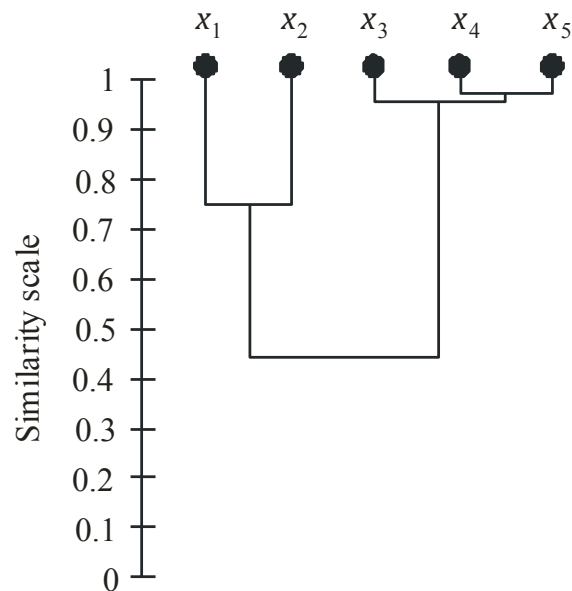
$\{\{X_1, X_2\}, \{X_3, X_4, X_5\}\}$

$\{\{X_1, X_2, X_3, X_4, X_5\}\}$

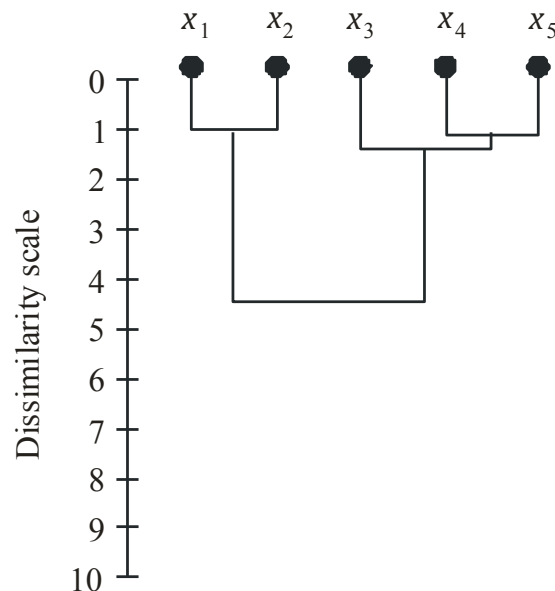


Proximity dendrogram

- **Proximity** (dissimilarity or dissimilarity) **dendrogram**: A dendrogram that takes into account the level of proximity (dissimilarity or similarity) where two clusters are **merged for the first time**.
- **Example 2**: In terms of the previous example, the proximity dendrograms that correspond to $P'(X)$ and $P(X)$ are



(a)



(b)

- **Remark**: One can readily observe the level in which a cluster is formed and the level in which it is absorbed in a larger cluster (indication of the natural clustering).

Algorithms based on matrix theory

Agglomerative algorithms are divided into:

- Algorithms based on **matrix theory**.
- Algorithms based on **graph theory**.

In the sequel we focus only on **dissimilarity measures**.

Algorithms based on matrix theory:

- They take as input the $N \times N$ dissimilarity matrix $P_0 = P(X)$.
- At each level t where two clusters C_i and C_j are merged to C_q , the dissimilarity matrix P_t is extracted from P_{t-1} by:
 - *Deleting the two rows and columns of P_t that correspond to C_i and C_j .*
 - *Adding a new row and a new column that contain the distances of newly formed $C_q = C_i \cup C_j$ from the remaining clusters C_s , via a relation of the form*

$$d(C_q, C_s) = f(d(C_i, C_s), d(C_j, C_s), d(C_i, C_j))$$

- A number of distance functions comply with the following update equation

$$d(C_q, C_s) = a_i d(C_i, C_s) + a_j d(C_j, C_s) + b d(C_i, C_j) + c |d(C_i, C_s) - d(C_j, C_s)|$$

Algorithms that follow the above equation are:

- **Single link (SL) algorithm** ($a_i=1/2, a_j=1/2, b=0, c=-1/2$). In this case

$$d(C_q, C_s) = \min \{d(C_i, C_s), d(C_j, C_s)\}$$

- **Complete link (CL) algorithm** ($a_i=1/2, a_j=1/2, b=0, c=1/2$). In this case

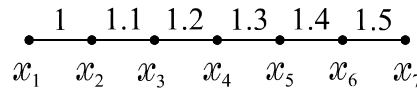
$$d(C_q, C_s) = \max \{d(C_i, C_s), d(C_j, C_s)\}$$

➤ **Remarks:**

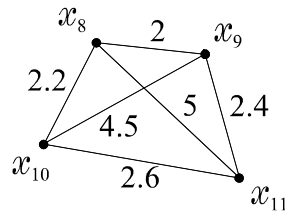
- **Single link** forms clusters at **low dissimilarities** while **complete link** forms clusters at **high dissimilarities**.
- **Single link** tends to form **elongated clusters** (*chaining effect*) while **complete link** tends to form **compact clusters**.
- The rest algorithms are compromises between these two extremes.

Example

(a) The data set X

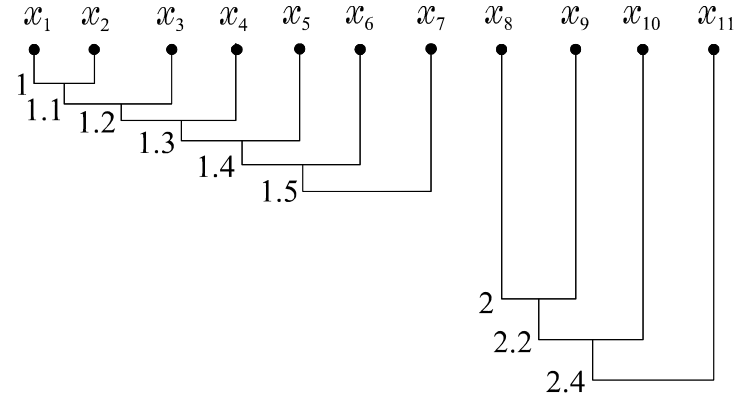


(rest of distances are very large)



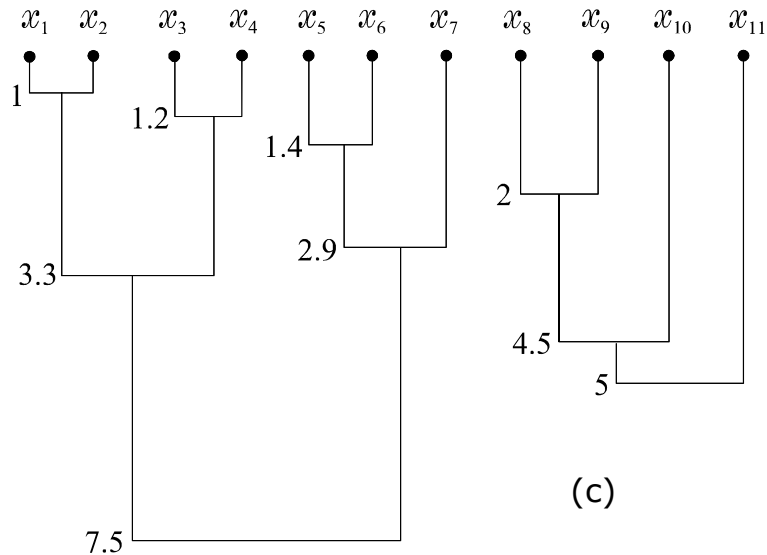
(a)

(b) The single link algorithm dissimilarity dendrogram



(b)

(c) The complete link algorithm dissimilarity dendrogram



(c)

- **Weighted Pair Group Method Average (WPGMA)** ($a_i=1/2$, $a_j=1/2$, $b=0$, $c=0$). In this case:

$$d(C_q, C_s) = (d(C_i, C_s) + d(C_j, C_s)) / 2$$

- **Unweighted Pair Group Method Average (UPGMA)** ($a_i=n_i/(n_i+n_j)$, $a_j=n_j/(n_i+n_j)$, $b=0$, $c=0$, where n_i is the cardinality of C_i). In this case:

$$d(C_q, C_s) = (n_i d(C_i, C_s) + n_j d(C_j, C_s)) / (n_i + n_j)$$

- **Unweighted Pair Group Method Centroid (UPGMC)** ($a_i=n_i/(n_i+n_j)$, $a_j=n_j/(n_i+n_j)$, $b=-n_i n_j / (n_i + n_j)^2$, $c=0$). In this case:

$$d_{qs} = \frac{n_i}{n_i + n_j} d_{is} + \frac{n_j}{n_i + n_j} d_{js} - \frac{n_i n_j}{(n_i + n_j)^2} d_{ij}$$

For the UPGMC, it is true that $d_{qs} = \|\mathbf{m}_q - \mathbf{m}_s\|^2$, where \mathbf{m}_q is the mean of C_q .

- **Weighted Pair Group Method Centroid (WPGMC)** ($a_i=1/2$, $a_j=1/2$, $b=-1/4$, $c=0$). In this case

$$d_{qs} = (d_{is} + d_{js})/2 - d_{ij}/4$$

For WPGMC there are cases where $d_{qs} \leq \max\{d_{is}, d_{js}\}$ (**crossover**)

- **Ward or minimum variance algorithm.** Here the distance d'_{ij} between C_i and C_j is defined as

$$d'_{ij} = (n_i n_j / (n_i + n_j)) \| \mathbf{m}_i - \mathbf{m}_j \|^2$$

d'_{qs} can also be written as

$$d'_{qs} = ((n_i + n_j)d'_{is} + (n_i + n_j)d'_{js} - n_s d'_{ij}) / (n_i + n_j + n_s)$$

- **Remark:** Ward's algorithm forms K_{t+1} by merging the two clusters that **lead to the smallest possible increase of the total variance, i.e.,**

$$E_t = \sum_{r=1}^{N-t} \sum_{\mathbf{x} \in C_r} \| \mathbf{x} - \mathbf{m}_r \|^2$$

Example

- **Example 3:** Consider the following dissimilarity matrix (Euclidean distance)

$$P_0 = \begin{bmatrix} 0 & 1 & 2 & 26 & 37 \\ 1 & 0 & 3 & 25 & 36 \\ 2 & 3 & 0 & 16 & 25 \\ 26 & 25 & 16 & 0 & 1.5 \\ 37 & 36 & 25 & 1.5 & 0 \end{bmatrix}$$

$$K_0 = \{\{\mathbf{x}_1\}, \{\mathbf{x}_2\}, \{\mathbf{x}_3\}, \{\mathbf{x}_4\}, \{\mathbf{x}_5\}\},$$

$$K_1 = \{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_3\}, \{\mathbf{x}_4\}, \{\mathbf{x}_5\}\},$$

$$K_2 = \{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_3\}, \{\mathbf{x}_4, \mathbf{x}_5\}\},$$

$$K_3 = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \{\mathbf{x}_4, \mathbf{x}_5\}\},$$

$$K_4 = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}\}$$

All the algorithms produce the above sequence of clusterings at **different** proximity levels:

	<i>SL</i>	<i>CL</i>	<i>WPGMA</i>	<i>UPGMA</i>	<i>WPGMC</i>	<i>UPGMC</i>	<i>Ward</i>
K_0	0	0	0	0	0	0	0
K_1	1	1	1	1	1	1	0.5
K_2	1.5	1.5	1.5	1.5	1.5	1.5	0.75
K_3	2	3	2.5	2.5	2.25	2.25	1.5
K_4	16	37	25.75	27.5	24.69	26.46	31.75

Hard Clustering Algorithms

Each vector belongs **exclusively** to a single cluster. This implies that:

- $u_{ij} \in \{0, 1\}, \quad j = 1, \dots, m$
- $\sum_{j=1}^m u_{ij} = 1$

The cost function

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^N \sum_{j=1}^m u_{ij} d(\mathbf{x}_i, \boldsymbol{\theta}_j)$$

is **not differentiable** with respect to $\boldsymbol{\theta}_j$.

Taking into account: for fixed $\boldsymbol{\theta}_j$'s, the u_{ij} 's that minimize $J(\boldsymbol{\theta}, U)$ are chosen as

$$u_{ij} = \begin{cases} 1, & \text{if } d(\mathbf{x}_i, \boldsymbol{\theta}_j) = \min_{k=1, \dots, m} d(\mathbf{x}_i, \boldsymbol{\theta}_k) \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \dots, N$$

Hard Clustering Algorithms (cont.)

Generalized Hard Algorithmic Scheme (GHAS)

- Choose $\theta_j(0)$ as initial estimates for $\theta_j, j=1, \dots, m$.
- $t=0$
- Repeat

– *For $i=1$ to N*

- For $j=1$ to m

Determination of the partition:

$$u_{ij}(t) = \begin{cases} 1, & \text{if } d(\mathbf{x}_i, \boldsymbol{\theta}_j(t)) = \min_{k=1, \dots, m} d(\mathbf{x}_i, \boldsymbol{\theta}_k(t)) \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \dots, N$$

- End {For- j }

– *End {For- i }*

– *$t=t+1$*

Hard Clustering Algorithms (cont.)

Generalized Hard Algorithmic Scheme (GHAS) (cont.)

– For $j=1$ to m

o Parameter updating: Solve

$$\sum_{i=1}^N u_{ij}(t-1) \frac{\partial d(\mathbf{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = 0$$

o with respect to $\boldsymbol{\theta}_j$ and set $\boldsymbol{\theta}_j(t)$ equal to the computed solution

– End {For- j }

- Until a termination criterion is met

Remarks:

- In the update of each $\boldsymbol{\theta}_j$, only the vectors \mathbf{x}_i for which $u_{ij}(t-1)=1$ are used.
- GHAS may terminate when either
 - $\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)\| < \varepsilon$ or
 - U remains unchanged for two successive iterations.

The Isodata or k-Means or c-Means algorithm

General comments

- It is a special case of GHAS where
 - *Point representatives are used.*
 - *The squared Euclidean distance is employed.*

- The cost function $J(\theta, U)$ becomes now

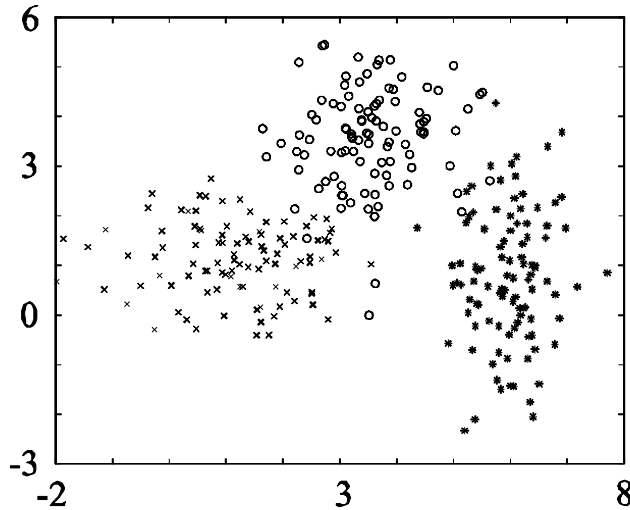
$$J(\theta, U) = \sum_{i=1}^N \sum_{j=1}^m u_{ij} \| \mathbf{x}_i - \boldsymbol{\theta}_j \|^2$$

- Applying GHAS in this case, it turns out that it **converges to a minimum** of the cost function
- Isodata recovers clusters that are as compact as possible
- For other choices of the distance (including the Euclidean), the algorithm converges but not necessarily to a minimum of $J(\theta, U)$

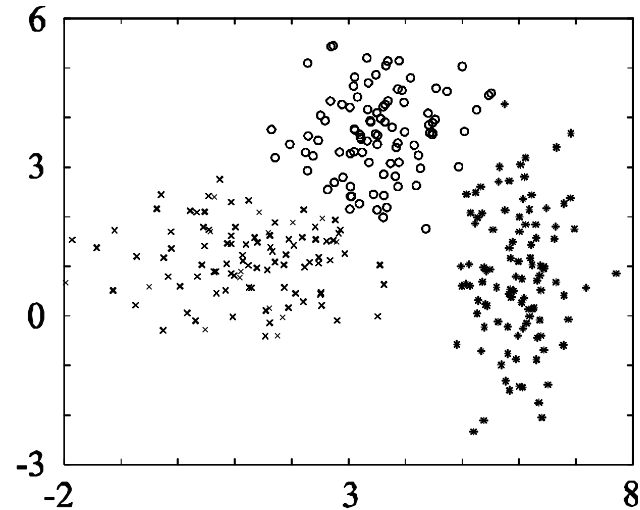
The Isodata or k-Means or c-Means algorithm

- Choose arbitrary initial estimates $\theta_j(0)$ for the θ_j 's, $j=1, \dots, m$.
- Repeat
 - *For $i=1$ to N*
 - o Determine the closest representative, say θ_j , for x_i
 - o Set $b(i)=j$.
 - *End {For}*
 - *For $j=1$ to m*
 - o *Parameter updating:* Determine θ_j as the mean of the vectors $x_i \in X$ with $b(i)=j$.
 - *End {For}*
- Until no change in θ_j 's occurs between two successive iterations

Example



(a)



(b)

Confusion matrix:

	<i>Cluster 1</i>	<i>Cluster 2</i>	<i>Cluster 3</i>
<i>1st distribution</i>	99	0	1
<i>2nd distribution</i>	0	100	0
<i>3rd distribution</i>	3	4	93

The k-means algorithm with $m=3$ identifies successfully the clusters in the data set of previous example. The confusion matrix is now:

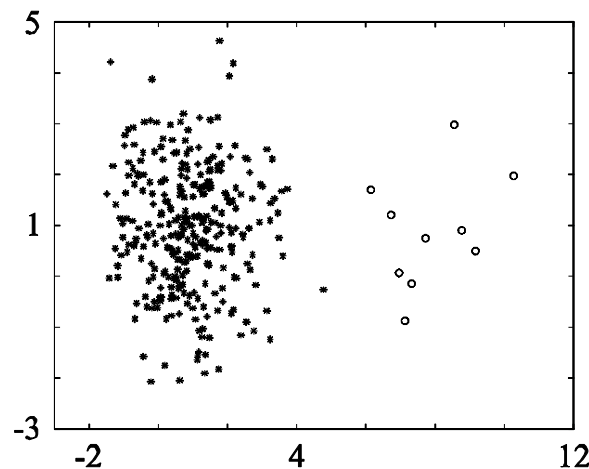
$$A = \begin{bmatrix} 94 & 3 & 3 \\ 0 & 100 & 0 \\ 9 & 0 & 91 \end{bmatrix}$$

Another example

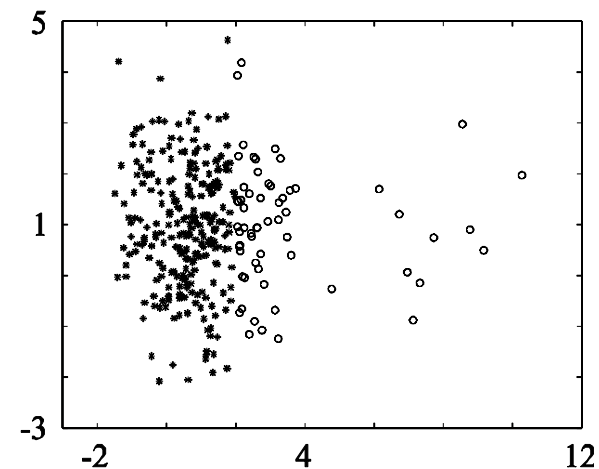
Example: (i) Consider two 2-dimensional Gaussian distributions $N(\boldsymbol{\mu}_1, \Sigma_1)$, $N(\boldsymbol{\mu}_2, \Sigma_2)$, with $\boldsymbol{\mu}_1 = [1, 1]^T$, $\boldsymbol{\mu}_2 = [8, 1]^T$, $\Sigma_1 = 1.5I$ and $\Sigma_2 = I$. (ii) Generate 300 points from the 1st distribution and 10 points from the 2nd distribution. (iii) Set $m=2$ and initialize randomly $\boldsymbol{\theta}_j$'s

After convergence the large group has been split into two clusters. Its right part has been assigned to the same cluster with the points of the small group (see figure below)

This indicates that **k-means cannot deal accurately with clusters having significantly different sizes.**



(a)



(b)

k-Means algorithm remarks

- k-means recovers compact clusters
- Sequential versions of the k-means, where the updating of the representatives takes place immediately after the identification of the representative that lies closer to the current input vector \mathbf{x}_i , have also been proposed
- A variant of the k-means results if the number of vectors in each cluster is constrained *a priori*
- The computational complexity of the k-means is $O(Nmq)$, where q is the number of iterations required for convergence. In practice, m and q are significantly less than N , thus, k-means becomes eligible for processing large data sets
- Some drawbacks of the original k-means accompanied with the variants of the k-means that deal with them are discussed next.

k-Means algorithm's Drawbacks

Drawback 1: *Different initial partitions may lead k-means to produces different final clusterings, each corresponding to a different local minimum*

Strategies for facing Drawback 1:

- Single run methods
 - *Use a sequential algorithm (discussed previously) to produce initial estimates for θ_j 's.*
 - *Partition randomly the data set into m subsets and use their means as initial estimates for θ_j ' s.*
- Multiple run methods
 - *Create different partitions of X , run k-means for each one of them and select the best result.*
 - *Compute the representatives iteratively, one at a time, by running k-means mN times. It is claimed that convergence is independent of the initial estimates of θ_j ' s.*
- Utilization of tools from stochastic optimization techniques (simulated annealing, genetic algorithms etc).

k-Means algorithm's Drawbacks

Drawback 2: *Knowledge of the number of clusters m is required a priori*

Strategies for facing Drawback 2:

- Employ splitting, merging and discarding operations of the clusters resulting from k-means
- Estimate m as follows:
 - *Run a sequential algorithm many times for different thresholds of dissimilarity Θ .*
 - *Plot Θ versus the number of clusters and identify the largest plateau in the graph and set m equal to the value that corresponds to this plateau.*

k-Means algorithm's Drawbacks

Drawback 3: *k-means is sensitive to outliers and noise*

Strategies for facing Drawback 3:

- Discard all “small” clusters (they are likely to be formed by outliers)
- Use a k-medoids algorithm (see below), where a cluster is represented by one of its points

Drawback 4: *k-means is not suitable for data with nominal (categorical) coordinates*

Strategies for facing Drawback 4:

- Use a k-medoids algorithm.

k-Medoids Algorithms

- Each cluster is represented by a vector selected **among** the elements of X (**medoid**)
- A cluster contains
 - *Its medoid*
 - *All vectors in X that*
 - o Are not used as medoids in other clusters
 - o Lie closer to its medoid than the medoids representing other clusters

Let Θ be the set of medoids of all clusters, I_Θ the set of indices of the points in X that constitute Θ and $I_{X-\Theta}$ the set of indices of the points that are not medoids.

k-Medoids Algorithms (cont.)

- Obtaining the set of medoids Θ that best represents the data set, X is equivalent to minimizing the following cost function:

$$J(\Theta, U) = \sum_{i \in I_{X-\Theta}} \sum_{j \in I_{\Theta}} u_{ij} d(\underline{x}_i, \underline{x}_j)$$

$$u_{ij} = \begin{cases} 1, & \text{if } d(\underline{x}_i, \underline{x}_j) = \min_{q \in I_{\Theta}} d(\underline{x}_i, \underline{x}_q) \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \dots, N$$

Mean values vs. medoids

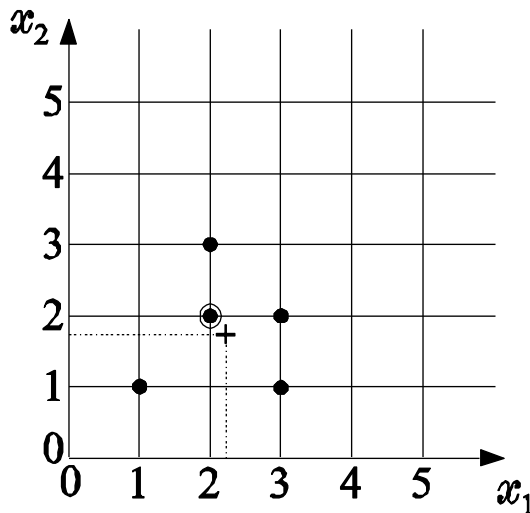
Mean Values	Medoids
1. Suited only for continuous domains	1. Suited for either cont. or discrete domains
2. Algorithms using means are sensitive to outliers	2. Algorithms using medoids are less sensitive to outliers
3. The mean possess a clear geometrical and statistical meaning	3. The medoid has not a clear geometrical meaning
4. Algorithms using means are not computationally demanding	4. Algorithms using medoids are more computationally demanding

Example

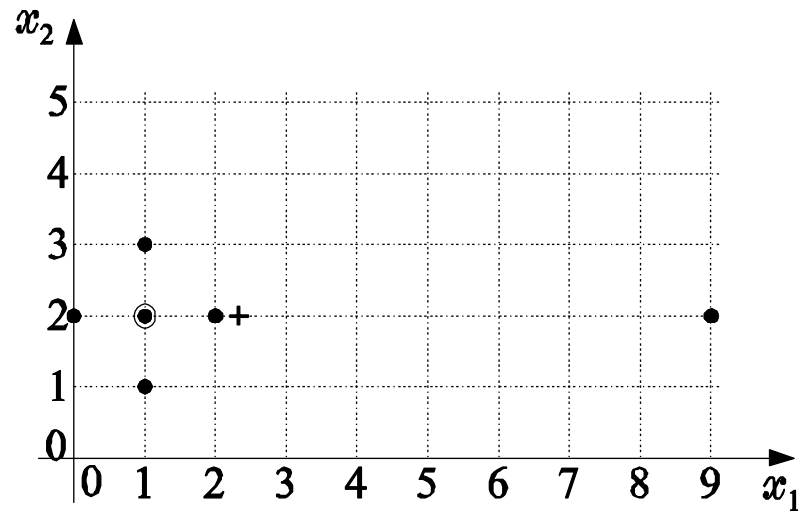
Example: (Illustrates the first two points in the above comparison)

(a) The five-point two-dimensional set stems from the discrete domain $D = \{1, 2, 3, 4, \dots\} \times \{1, 2, 3, 4, \dots\}$. Its medoid is the circled point and **its mean** is the “+” point, which **does not belong to D** .

(b) In the six-point two-dimensional set, the point (9,2) can be considered as an outlier. While **the outlier affects significantly the mean** of the set, **it does not affect its medoid**.



(a)



(b)

Popular k-Medoids Algorithms

Algorithms to consider:

- PAM (Partitioning Around Medoids)
- CLARA (Clustering LARge Applications)
- CLARANS (Clustering Large Applications based on RANdomized Search)

Cluster tendency and validity

Cluster validity

- **Cluster validity**: a task that evaluates **quantitatively** the results of a clustering algorithm
- A clustering structure **C**, resulting from an algorithm may be either
 - A hierarchy of clusterings or
 - A single clustering

Possible approaches

Cluster validity may be approached in three possible directions:

- \mathbf{C} is evaluated in terms of an independently drawn structure, imposed on X *a priori*. The criteria used in this case are called **external criteria**
- \mathbf{C} is evaluated in terms of quantities that involve the vectors of X themselves (e.g., proximity matrix). The criteria used in this case are called **internal criteria**
- \mathbf{C} is evaluated by comparing it with other clustering structures, resulting from the application of the same clustering algorithm but with different parameter values, or other clustering algorithms, on X . Criteria of this kind are called **relative criteria**

Cluster validity for the cases of external and internal criteria

- Hypothesis testing is employed
- The null hypothesis H_0 , which is a statement of randomness concerning the structure of X , is defined
- The generation of a reference data population under the random hypothesis takes place
- An appropriate statistic, q , whose values are indicative of the structure of a data set, is defined. The value of q that results from our data set X is compared against the values obtained for q when the elements of the reference (random) population are considered

Ways for generating reference populations under the null hypothesis (each one used in different situations):

- Random position hypothesis: all arrangements of vectors in a specific region are equally likely
- Random graph hypothesis: adopted with only internal information
- Random label hypothesis: all possible label mappings are equally likely

Statistics suitable for external criteria

- For the comparison of \mathbf{C} with an independently drawn partition \mathbf{P} of X
 - Rand statistics
 - Jaccard statistics
 - Fowlkes-Mallows index
 - Hubert's Γ statistics
 - Normalized Γ statistics
- For assessing the agreement between \mathbf{P} and the proximity matrix P
 - Γ statistics

Statistics suitable for internal criteria

- Validation of hierarchy of clusterings
 - Cophenetic correlation coefficient (*CPCC*)
 - γ statistics
 - Kudall's τ statistics
- Validation of individual clusterings
 - Γ statistics
 - Normalized Γ statistics

Cluster validity for the cases of relative criteria

Let \mathbf{A} denote the set of parameters of a clustering algorithm.

Statement of the problem:

- “Among the clusterings produced by a specific clustering algorithm, for different values of the parameters in \mathbf{A} , choose the one that best fits the data set X ”.

We consider two cases

(a) \mathbf{A} *does not* contain the number of clusters m .

The estimation of the best set of parameter values is carried out as follows:

- Run the algorithm for a wide range of values of its parameters.
- Plot the number of clusters, m , versus the parameters of \mathbf{A} .
- Choose the widest range for which m remains constant.
- Adopt the clustering that corresponds to the values of the parameters in \mathbf{A} that lie in the **middle** of this range.

The cases of relative criteria (contd.)

(b) **A** *does* contain the number of clusters m .

The estimation of the best set of parameter values is carried out as follows:

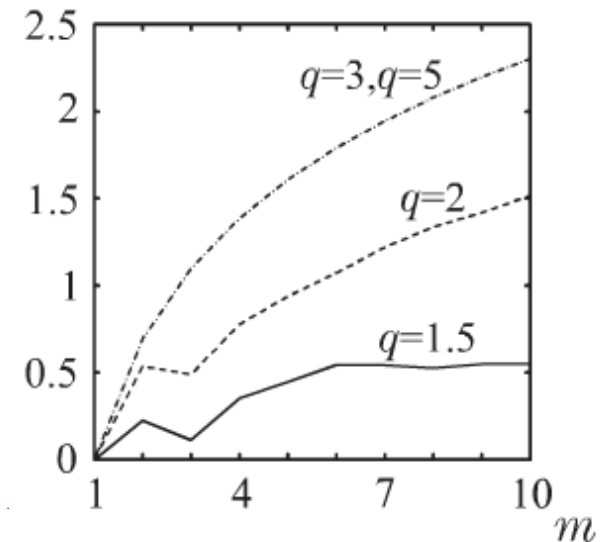
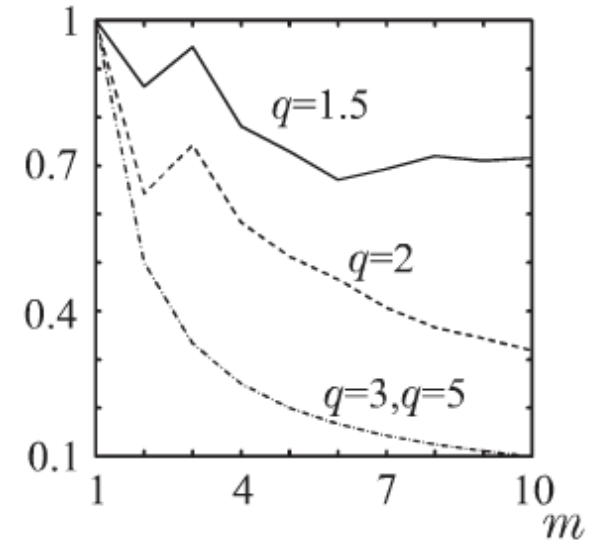
- Select a suitable performance index q (the best clustering is identified in terms of q)
- For $m=m_{min}$ to m_{max} (chosen *a priori*)
 - Run the algorithm r times using different sets of values for the other parameters of **A** and each time compute q
 - Choose the clustering that corresponds to the best q

End for

- Plot the best values of q (for each m) versus m
- Seek the maximum/minimum of the plot , according to whether large/ small values of q indicate good clustering
- The procedure works well if q exhibits no trend with growth of m

The cases of relative criteria (contd.)

- Often q , however, exhibits an increasing/decreasing trend as m grows
- Thus, we no longer can adopt the previous strategy
- Instead, we seek for values of m at which a significant local change in value q occurs
- The presence of a **significant knee** indicates the number of clusters underlying X . **Adopt the clustering that corresponds to that knee**
- The **absence** of such a knee indicates that X possesses **no** clustering structure.



Supervised learning: Vocabulary, notations and basic concepts

A learning problem

Definition of well-posed learning problem (Tom Mitchell, 1998):

A computer program is said to learn from experience E w.r.t some task T and some performance measure P if its performance on T , measured by P , is improved with experience E .

A learning problem

Definition of well-posed learning problem (Tom Mitchell, 1998):

A computer program is said to learn from experience E w.r.t some task T and some performance measure P if its performance on T , measured by P , is improved with experience E .

Regression Problem: Continuous value output

A learning problem

Definition of well-posed learning problem (Tom Mitchell, 1998):

A computer program is said to learn from experience E w.r.t some task T and some performance measure P if its performance on T , measured by P , is improved with experience E .

Regression Problem: Continuous value output

Classification Problem: Discrete value output

A learning problem

Definition of well-posed learning problem (Tom Mitchell, 1998):
A computer program is said to learn from experience E w.r.t some task T and some performance measure P if its performance on T , measured by P , is improved with experience E .

Regression Problem: Continuous value output

Classification Problem: Discrete value output

Example 1: Predicting whether a tumor is benign or cancerous

A learning problem

Definition of well-posed learning problem (Tom Mitchell, 1998):

A computer program is said to learn from experience E w.r.t some task T and some performance measure P if its performance on T , measured by P , is improved with experience E .

Regression Problem: Continuous value output

Classification Problem: Discrete value output

Example 1: Predicting whether a tumor is benign or cancerous

Example 2: Predicting the amount of snow (inches) in Worcester, MA based on the average summer temperature

A learning problem

Definition of well-posed learning problem (Tom Mitchell, 1998):

A computer program is said to learn from experience E w.r.t some task T and some performance measure P if its performance on T , measured by P , is improved with experience E .

Regression Problem: Continuous value output

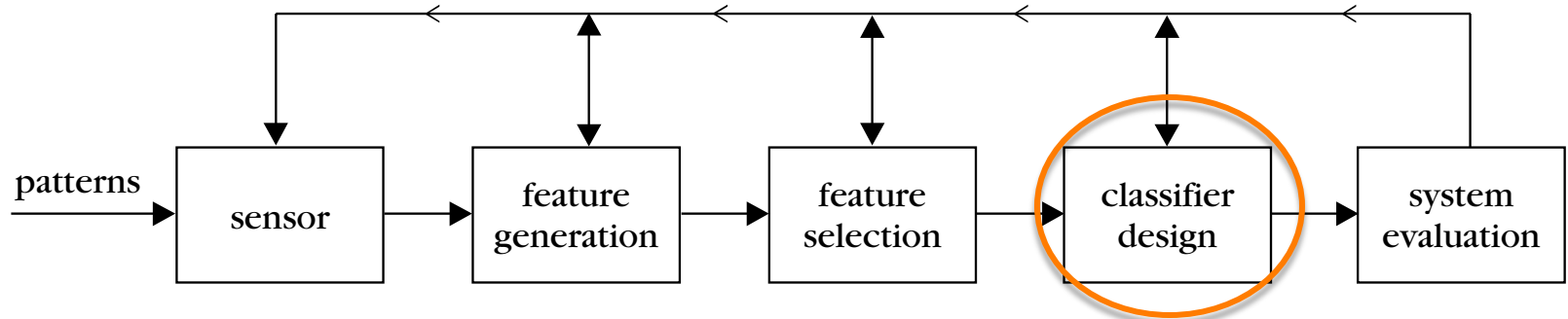
Classification Problem: Discrete value output

Example 1: Predicting whether a tumor is benign or cancerous

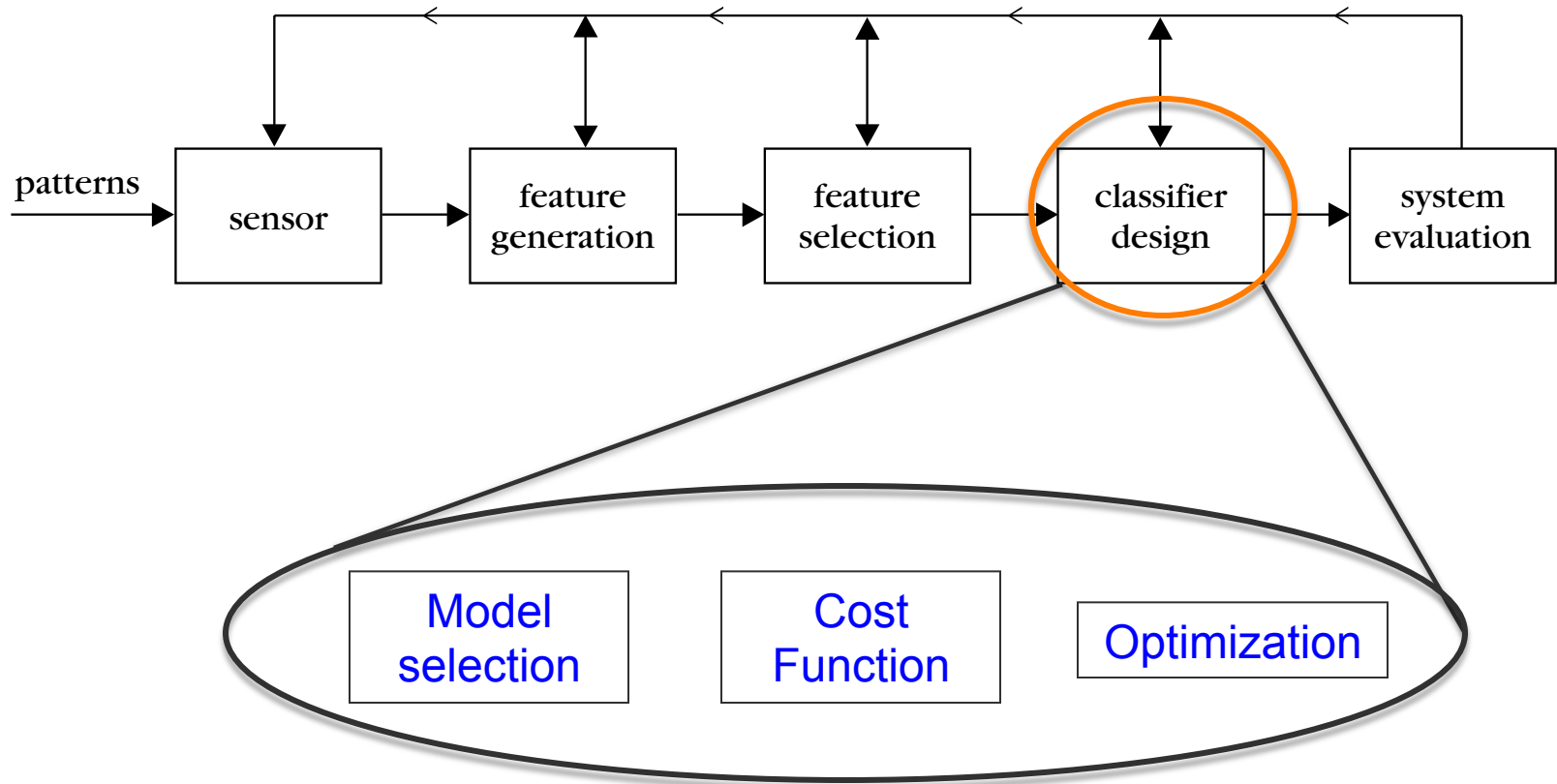
Example 2: Predicting the amount of snow (inches) in Worcester, MA based on the average summer temperature

Example 3: Classification of geographical locations on a satellite image

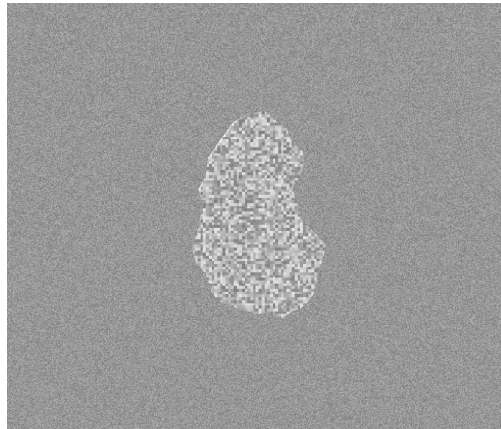
Features, Feature vectors and Classifiers



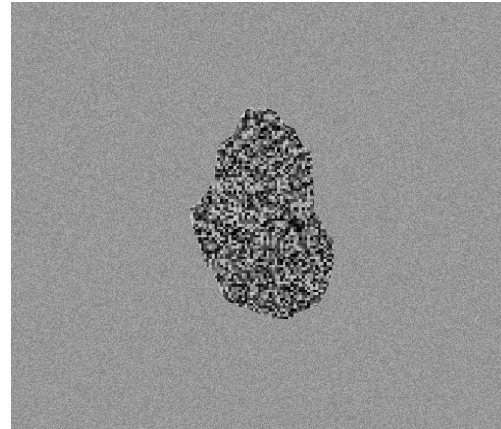
Features, Feature vectors and Classifiers



Features and Feature vectors

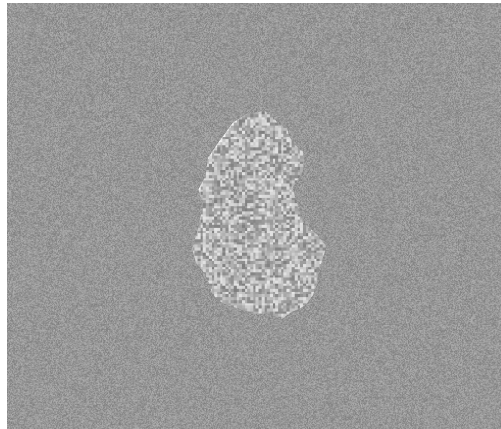


Class A: Benign Tumor

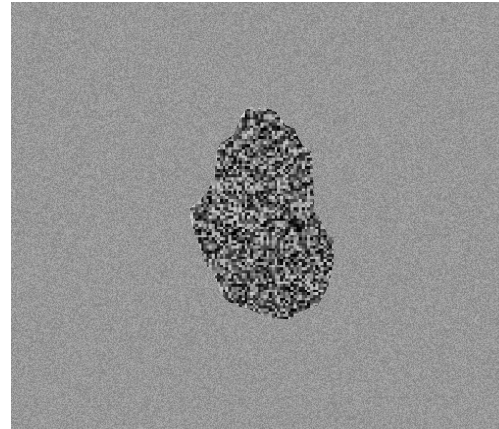


Class B: Cancer

Features and Feature vectors



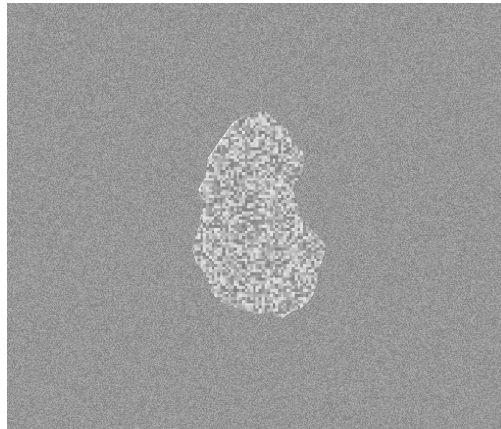
Class A: Benign Tumor



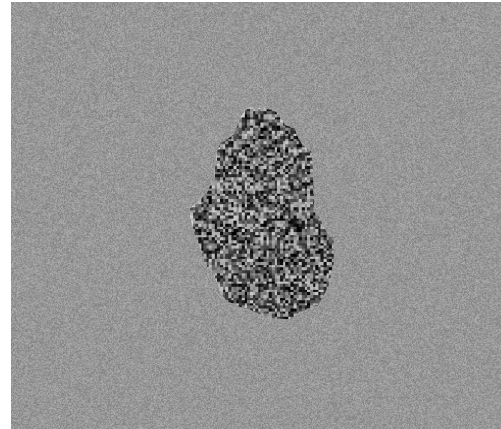
Class B: Cancer

- We need to identify the measurable quantities that make those two images (specifically, the regions) distinct
- Consider the following simple set of two features:
 - x_1 : Mean value of intensity in a region of image
 - x_2 : Standard deviation of around the mean

Features and Feature vectors

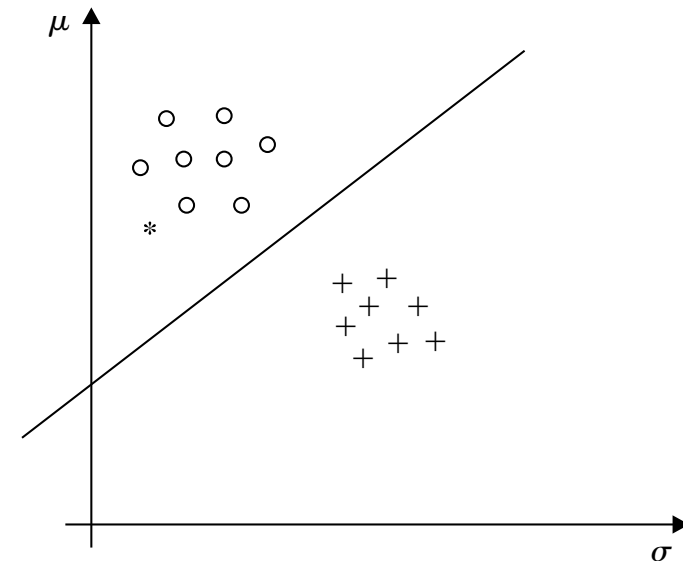


Class A: Benign Tumor



Class B: Cancer

- We need to identify the measurable quantities that make those two images (specifically, the regions) distinct
- Consider the following simple set of two features:
 - x_1 : Mean value of intensity in a region of image
 - x_2 : Standard deviation of around the mean



Notation

- m = number of training examples
- n = dimensionality of feature vector space
- x = input feature vector
- y = output variable
- (x, y) = a single training example
- $(x^{(i)}, y^{(i)})$ = i -th training example
- x_j = the j -th feature (coordinate) of x
- h is the hypothesis function (in regression or classification)

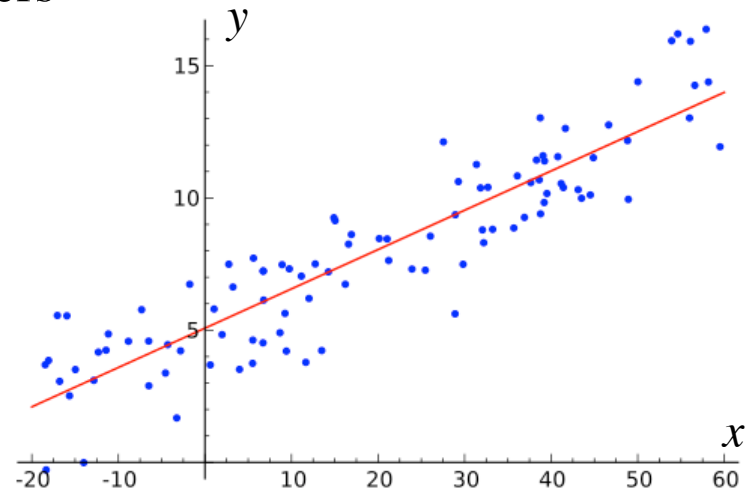
Regression basics

- Linear regression: data can be modeled as a linear function

- $h_w(x) = w_0 + w_1x$, where w_i are parameters

Sometimes, different notation is used

- $h_\theta(x) = \theta_0 + \theta_1x$



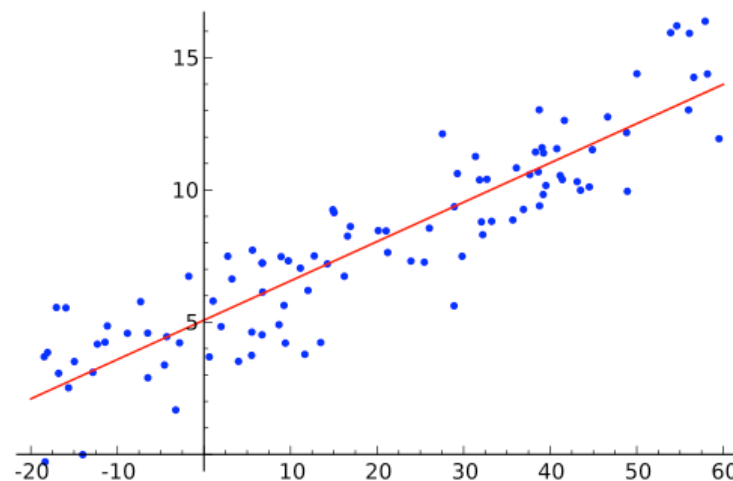
Cost function

Q: How do we know that we perform well for the classification or regression problem?

A: Cost function!

Specifically: Choose parameters w_i such that $h_w(x)$ is close to y for the training set values:

$$\min_{w_0, w_1} \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$



Cost function

Q: How do we know that we perform well for the classification or regression problem?

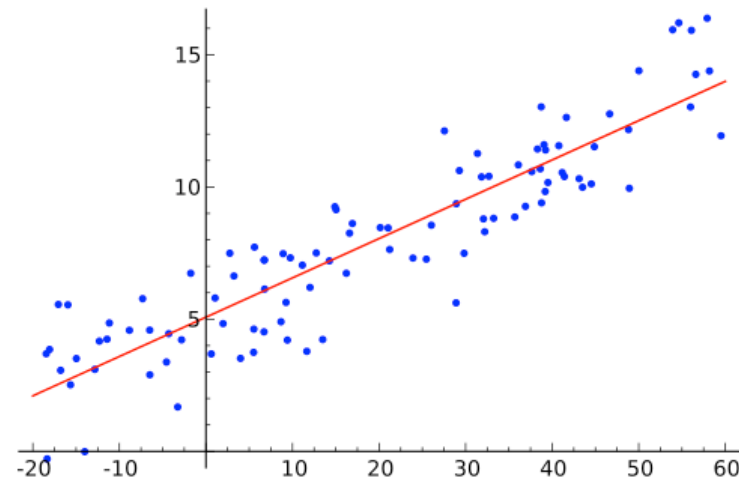
A: Cost function!

Specifically: Choose parameters w_i such that $h_w(x)$ is close to y for the training set values:

$$\min_{w_0, w_1} \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$

Therefore, our cost function is defined as:

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$



need to minimize it!

Optimization of cost function

- A popular technique: **Gradient descent algorithm**
- Given $J(w_0, w_1, \dots, w_n)$, a general case, minimize it over w_i
- Idea 1:
 - Start with some initial guess for w_0, w_1, \dots
 - Keep changing the values of w_0 trying to reduce J
- Idea 2:
 - Any 2-dimensional surface (works for 3- or even n-dimensions) will have valleys and hills
 - What is the fastest way going down with a small step? Following the gradient vector!

Gradient descent

Repeat until convergence:

$$\left\{ \begin{array}{l} w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, \dots), j = 0, 1, \dots, n \end{array} \right\}$$

- α is called a learning rate

- $\frac{\partial}{\partial w_j} J$ is called a derivative term

- How to update w_i ? Simultaneously!

Intuition

- Consider a simple example of a single variable cost function $J(w_1)$
- $$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1) =$$
$$w_1 - \alpha * \text{PosNumber} \quad \text{or} \quad w_1 - \alpha * \text{NegNumber}$$
- If α is too small, then the convergence is too slow
- If α is too big, the the gradient descent can “overshoot”
- If we got exactly the local minimum, next step will be unchanged

More intuition

- Gradient descent can converge to a minimum even when α is fixed
- As we approach a local minimum, we will automatically make smaller steps
- If there are multiple local minima net to each other, we could get the wrong one

Gradient descent for linear regression

Goal: Apply gradient descent to minimize $J(w_0, w_1)$ (where)

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$

- Need to find $\frac{\partial}{\partial w_j} J(w_0, w_1)$

- After taking derivatives:

$$\frac{\partial}{\partial w_0} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial w_1} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

Algorithm

Repeat until convergence:

$$\left\{ \begin{array}{l} w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right) \\ w_1 := w_1 - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right) x^{(i)} \end{array} \right\}$$

Note: For linear regression, the cost function is always a *convex* function – bowl shape

Remarks

- The reviewed algorithm is so-called “Batch” gradient descent
 - Batch: Using **all** training examples for **each** step of gradient descent
- Gradient descent is known to scale better than the analytical solutions (such as normal equations)

A case of multi-dimensional feature space

- Our hypothesis is generalized:

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n, \text{ where } w_i \text{ are parameters}$$

- Let's define another coordinate $x_0^{(i)}=1$ for each example i . Our feature vector $x^{(i)}$ is now in a $(n+1)$ -dimensional space!
- Our parameter vector $w^{(i)}$ is also in a $(n+1)$ -dimensional space
- Therefore, to vectorize our hypothesis:

$$h_w(x) = w^T x$$

- Cost function:

$$J(w) = J(w_0, w_1, \dots, w_n) = \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$

Modified algorithm

Repeat until convergence:

$$\left\{ w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w) \right.$$

(Simultaneously update for every $j = 0, 1, \dots, n$)

}

Convergence criterion:

$$\|w^{New} - w^{Old}\| = \sqrt{\sum_{i=0}^n (w_i^{New} - w_i^{Old})^2} \leq \varepsilon_0 - \text{a predefined threshold}$$

Modified algorithm: implementation-ready

Repeat until convergence:

{

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

(Simultaneously update for every $j = 0, 1, \dots, n$)

}

How to improve the method: Feature scaling

- In general, many optimization methods perform better (e.g., faster) if the features have comparable (same order of magnitude) values
- Gradient descent is no exception
- Solution: Scaling/Normalization to [0;1] or [-1;1]
- A simple way--mean normalization:
$$x_i \leftarrow x_i - \mu, \text{ where } \mu \text{ is the mean over the values for the } i\text{-th coordinate}$$

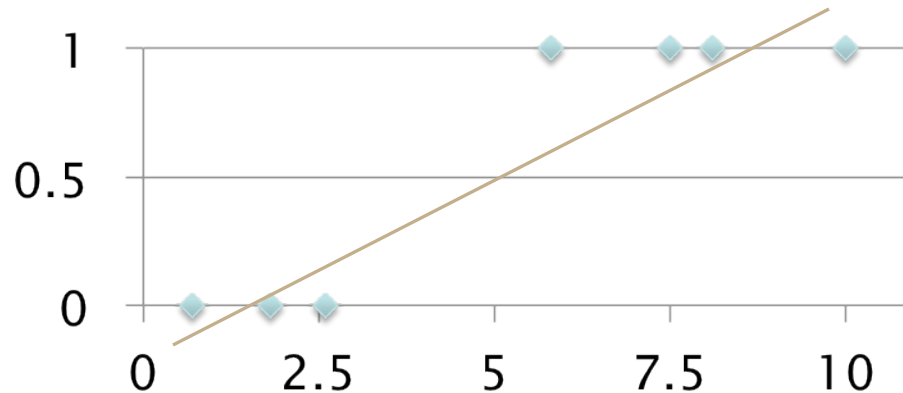
- A better way--standardization:
$$x_i \leftarrow \frac{x_i - \mu}{\sigma}, \text{ where } \mu \text{ is the mean and } \sigma \text{ is standard deviation}$$

How to improve the method: Learning rate (α)

- Can we choose α ?
- If we plot $J(w)$ against the number of iterations, $J(w)$ should decrease after each iteration
- If $J(w)$ increase, we need to choose a different (smaller) learning rate
- If $J(w)$ oscillates, we need to choose a smaller learning rate
- How to choose? A 10-fold increase: 0.001, 0.01, 0.1, ...
Could be a ~ 3 -fold increase: 0.001, 0.003, 0.01, ...

Moving to classification problem: Logistic regression

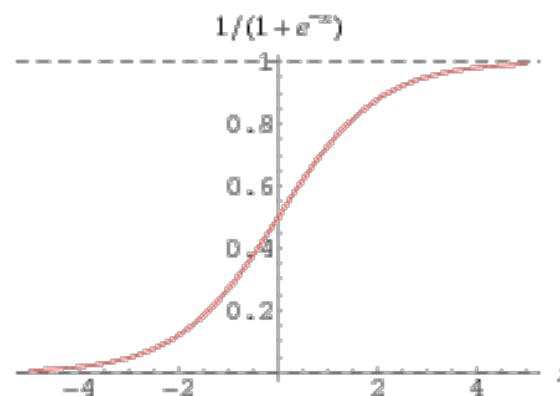
- Consider a problem of classifying a certain type of tumor as benign or cancerous, depending on its size:



- A possible solution:
 - Apply a linear regression: $h_w(x) = w^T x$
 - Threshold of $h_w(x) = 0.5$ would be used to classify output as benign (≤ 0.5) or cancerous (> 0.5)
 - Problem: If we add another “cancerous” data point that corresponds to a large size, the regression line will change although it shouldn't!

Basics of logistic regression

- Intuition: We want $0 \leq h_w(x) \leq 1$
- Solution: $h_w(x) = g(w^T x)$, where $g(z) = \frac{1}{1 + e^{-z}}$ sigmoid (logistic) function



- We need to fit the parameters from w
- Interpretation: $h_w(x) = \text{estimated probability that } y=1, \text{ on input } x$
- Formally: $h_w(x) = P(y=1 \mid x; w)$, therefore $P(y=0 \mid x; w) = 1 - P(y=1 \mid x; w)$

Decision boundary

- Prediction follows the same idea:
 - “ $y=1$ ” when $h_w(x) > 0.5$
 - “ $y=0$ ” otherwise
- We know that for the exponential function:
 - $g(z) > 0.5$ when $z > 0$
- Therefore:
 - $h_w(x) > 0.5$ when $w^T x > 0$
 - $h_w(x) \leq 0.5$ when $w^T x \leq 0$

Decision boundary: Example

- Consider $h_w(x) = g(w^T x) = g(w_0 x_0 + w_1 x_1 + w_2 x_2) =$
 - Let's choose $w_0 = -3, w_1 = 1, w_2 = 1$
- Therefore:
 - “y=1” when $w^T x > 0 : -3 + x_1 + x_2 > 0$
 - “y=0” when $w^T x \leq 0 : -3 + x_1 + x_2 \leq 0$
- Line $x_1 + x_2 = 3$ is called *decision boundary*
- The decision boundary is the property of hypothesis (including our parameter vector w), but NOT the dataset!
- Decision boundaries could be non-linear:
 - “y=1” when: $-1 + x_1^2 + x_2^2 > 0$
 - “y=0” when $w^T x \leq 0.5 : -1 + x_1^2 + x_2^2 \leq 0$

Cost function

- To choose the parameters, let's go to the linear regression cost function and modify it:

$$J(w) = J(w_0, w_1, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \underbrace{\frac{1}{2} (h_w(x^{(i)}) - y^{(i)})^2}_{\text{Replace by } \textit{cost}(h_w(x^{(i)}), y)}$$

- Problem: If we leave the same formula (theoretically possible), our cost function becomes non-convex
- Gradient descent will most like get stucked at a local minimum
- Solution: Finding a cost function that is convex

$$\textit{cost}(h_w(x), y) = \begin{cases} -\log(h_w(x)), & \text{if } y = 1 \\ -\log(1 - h_w(x)), & \text{if } y = 0 \end{cases}$$

Property of the new cost function

- It could be shown that the function is convex
- For $y=1$:
 - if $h_w(x)=1$, $cost = 0$
 - with $h_w(x)$ approaching 0, $cost$ will approach ∞
 - That means penalizing learning algorithm by a huge number
- For $y=0$:
 - if $h_w(x)=0$, $cost = 0$
 - with $h_w(x)$ approaching 1, $cost$ will approach ∞
- A more compact way to write the same cost function:

$$cost(h_w(x), y) = -y \log(h_w(x)) - (1 - y) \log(1 - h_w(x))$$

Finalized cost function

$$\begin{aligned} J(w) &= J(w_0, w_1, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \text{cost}(h_w(x), y) \\ &= \frac{1}{m} \left[\sum_{i=1}^m -y \log(h_w(x)) - (1-y) \log(1-h_w(x)) \right] \end{aligned}$$

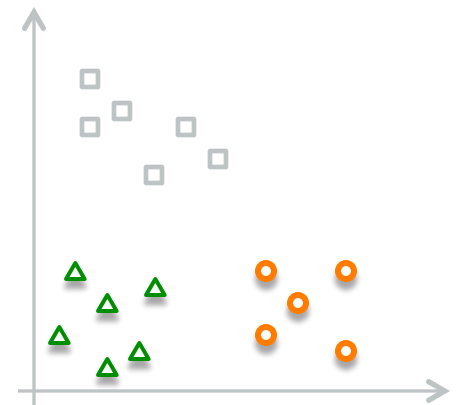
- Fitting parameters w_0, w_1, \dots, w_n : use Gradient Descent (algorithm looks identical to linear regression, but it is different in $h_w(x)$)

- To make a prediction given new x :

$$h_w(x) = g(w^T x), \text{ where } g(z) = \frac{1}{1 + e^{-z}}$$

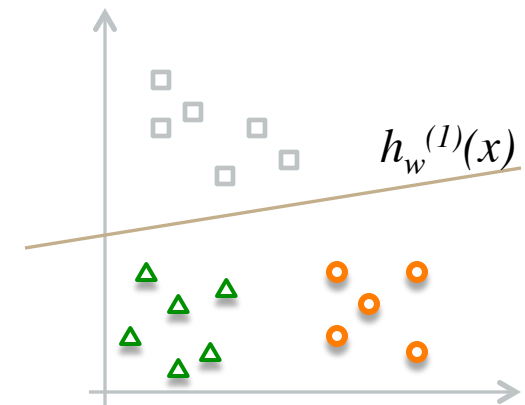
Multiclass classification

- Emails: Spam detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
Idea: convert multi-class to several binary classifications



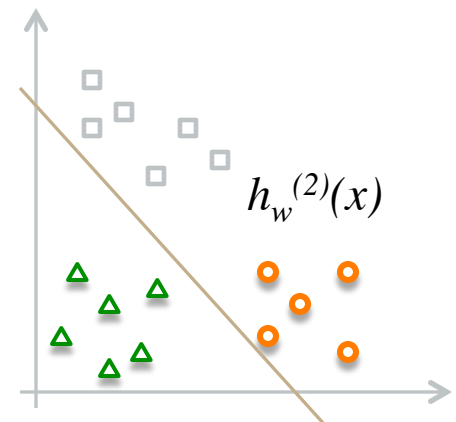
Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
Idea: convert multi-class to several binary classifications



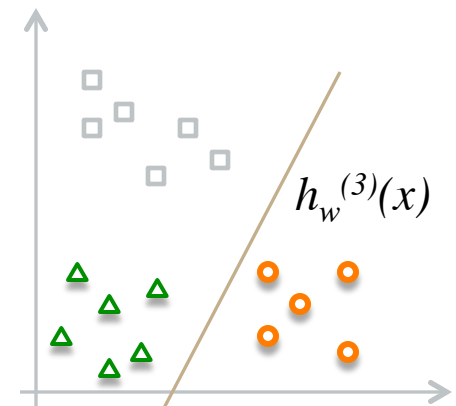
Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
Idea: convert multi-class to several binary classifications



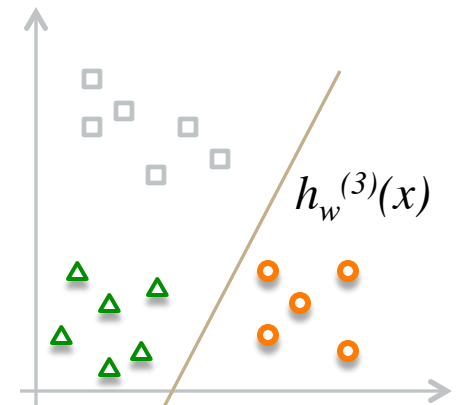
Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
Idea: convert multi-class to several binary classifications

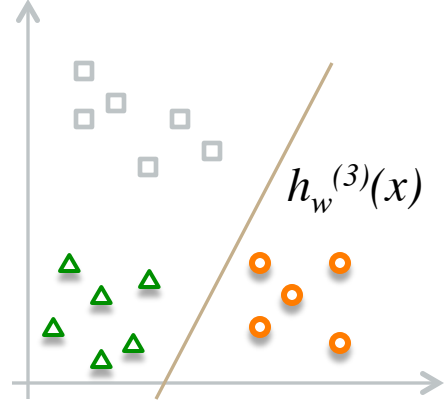


Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
Idea: convert multi-class to several binary classifications
- As a result we have 3 binary classifiers, each one is trained to recognize one class

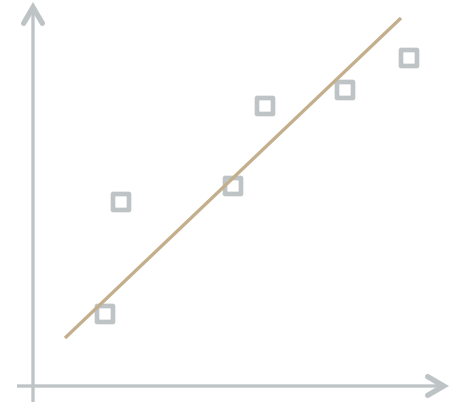


Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
 - Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
 - One-vs-all classification (one-vs-rest)
Idea: convert multi-class to several binary classifications
 - As a result we have 3 binary classifiers, each one is trained to recognize one class
- 
- For a new element: Apply all and select the most optimistic one: $\max(h_w^{(i)}(x))$

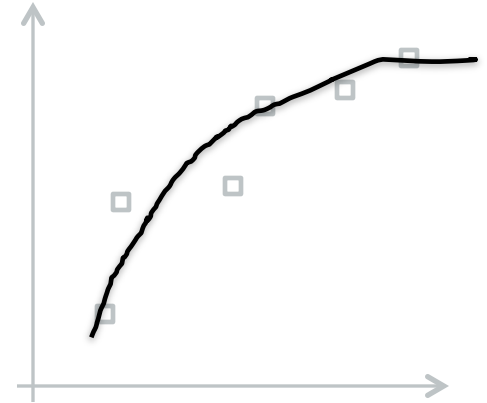
Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
 - But the data “plateau”
 - Underfitting or “high bias”



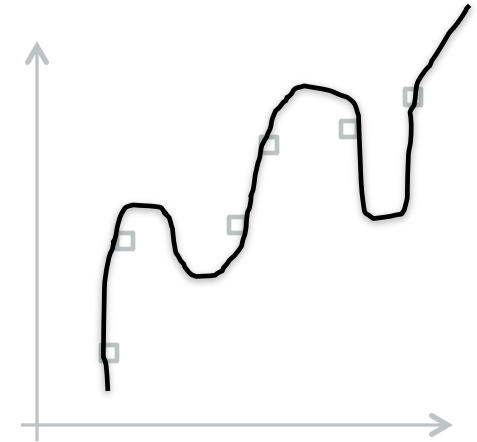
Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
 - But the data “plateau”
 - Underfitting or “high bias”
- Fitting a quadratic function
 - Seems to be a good fit
 - “Just right”



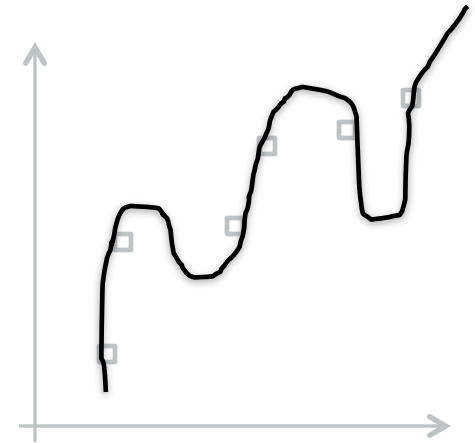
Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
 - But the data “plateau”
 - Underfitting or “high bias”
- Fitting a quadratic function
 - Seems to be a good fit
 - “Just right”
- Fitting a 4-degree polynomial
 - While the fit is perfect , the function does not look as a plausible solution
 - Overfitting or “high variance”: not enough data for the hypothesis



Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
 - But the data “plateau”
 - Underfitting or “high bias”
- Fitting a quadratic function
 - Seems to be a good fit
 - “Just right”
- Fitting a 4-degree polynomial
 - While the fit is perfect , the function does not look as a plausible solution
 - Overfitting or “high variance”: not enough data for the hypothesis
- Same thing happens with the classification (for instance with linear regression)



Addressing overfitting

- If we have many features, the learned hypothesis may fit the data very well but **fail to generalize** (that is predict new examples)
- It is very difficult to address overfitting by visualization (especially when we have any features)

Addressing overfitting

- If we have many features, the learned hypothesis may fit the data very well but **fail to generalize** (that is predict new examples)
- It is very difficult to address overfitting by visualization (especially when we have any features)
- Solutions:
 - Reduce number of features
 - Manually select which features to keep
 - Use a model selection algorithm (will cover later in the course)

Addressing overfitting

- If we have many features, the learned hypothesis may fit the data very well but **fail to generalize** (that is predict new examples)
- It is very difficult to address overfitting by visualization (especially when we have any features)
- Solutions:
 - Reduce number of features
 - Manually select which features to keep
 - Use a model selection algorithm (will cover later in the course)
 - Regularization
 - Keep all the features but reduce the magnitude/values of parameters w_i
 - Works well when we have many features each of which contributes a little bit to the value of y

Intuition: Cost function

- Consider 2 hypothesis for our linear regression:
 - $w_0 + w_1x + w_2x^2$: Just right
 - $w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$: Overfitting

Intuition: Cost function

- Consider 2 hypothesis for our linear regression:

- $w_0 + w_1x + w_2x^2$: Just right

- $w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$: Overfitting

- Lets penalize w_3 and make w_3 and w_4 very small :

$$\min_{w_i} \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2 + 1000w_3 + 1000w_4$$

- With this cost function we do need to make w_3 and w_4 very small. Thus, our 4-degree polynomial function will behave more like a quadratic one

Intuition: Cost function

- Consider 2 hypothesis for our linear regression:

- $w_0 + w_1x + w_2x^2$: Just right
- $w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$: Overfitting

- Lets penalize and make w_3 and w_4 very small :

$$\min_{w_i} \frac{1}{2m} \sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2 + 1000w_3 + 1000w_4$$

- With this cost function we do need to make w_3 and w_4 very small. Thus, our 4-degree polynomial function will behave more like a quadratic one
- Overall idea: small values for the parameters, which leads to simpler hypotheses, which are less prone to overfitting

Modified cost function

- The updated cost function to minimize:

$$J(w_0, w_1) = \frac{1}{2m} \left[\sum_{i=1}^m \left(h_w(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{i=1}^n w_i^2 \right]$$

- Note that we have no w_0
 - This is by convention
 - In practice – little or no difference
- Parameter λ controls the trade-off between 2 terms
 - If λ is very large, then we will end up penalizing our parameters very heavily $\Rightarrow w_i \approx 0$ and $h_w(x) = w_0$, which is underfitting
 - If λ is very small, then no effect on the parameters \Rightarrow overfitting

Logistic regression is similar

$$J(w) = -\frac{1}{m} \left[\sum_{i=1}^m y \log(h_w(x)) + (1-y) \log(1-h_w(x)) \right] + \frac{\lambda}{2m} \sum_{i=1}^n w_i^2$$

- Parameter lambda regulates how overcomplexified is the decision boundary
- GD is slightly different

Acknowledgements

- Lecture materials are based on:
 - The textbook
 - Lecture materials by Stuart Russell (the co-author of the textbook) in Berkeley