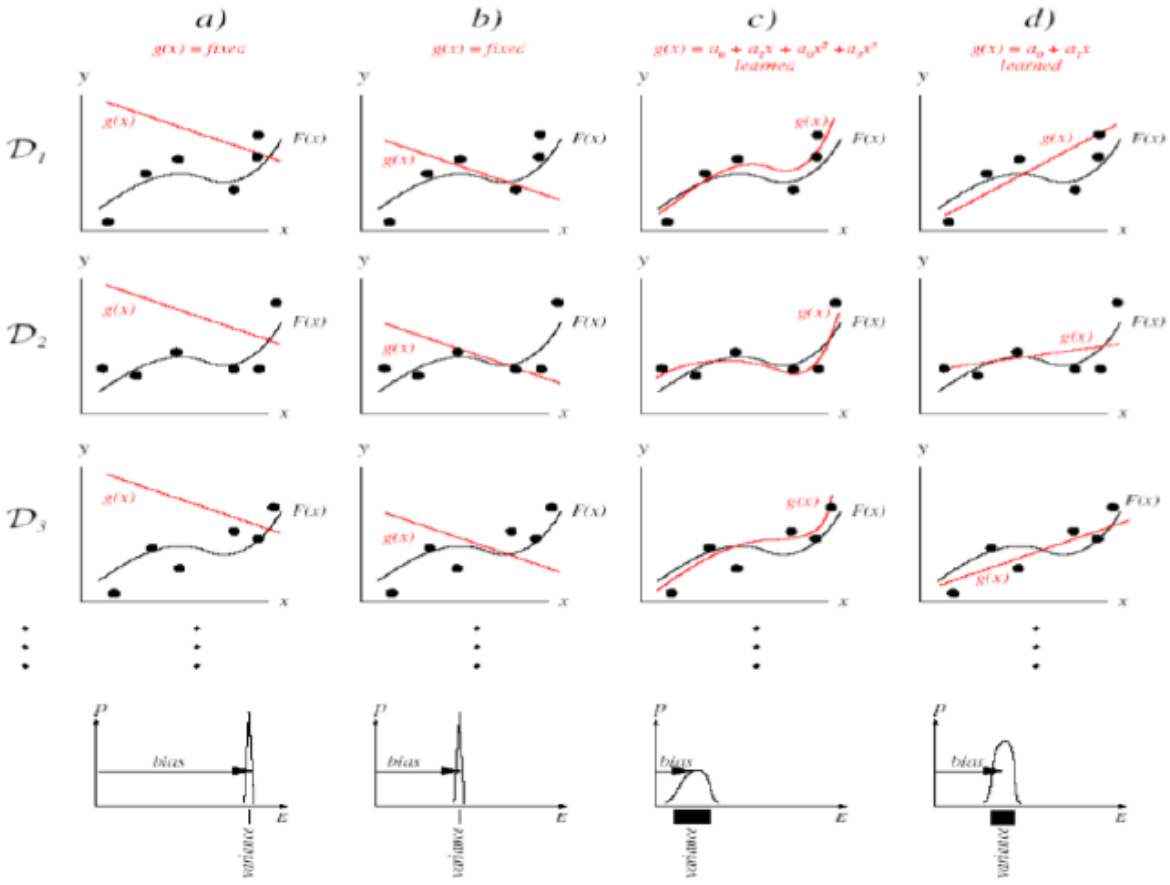
2 0 T 2

Comp9417

Review 3

Comp9417

Bias-Variance Decomposition



The *bias* of a learning scheme is the expected error due to the mismatch between the learner's hypothesis space (class of models) and the space of target concepts

The *variance* of a learning scheme is the expected error due to differences in the training sets used

Comp9417
Stability

- for a given data distribution $\mathcal{D}$
- train algorithm L on training sets S_1, S_2 sampled from $\mathcal{D}$
- expect that the model from L should be the same (or very similar) on both S_1 and S_2
- if so, we say that L is a stable learning algorithm
- otherwise it is unstable
- typical stable algorithm: kNN (for some k)
- typical unstable algorithm: decision-tree learning
- stable algorithms typically have high bias
- unstable algorithms typically have high variance
- BUT: take care to consider effect of parameters on stability, e.g., in kNN:

Comp9417

Bagging

Algorithm Bagging$(D, T, \mathcal{A})$ // train ensemble from bootstrap samples

Input: dataset D; ensemble size T; learning algorithm $\mathcal{A}$.

Output: set of models; predictions to be combined by voting or averaging.

```
for t = 1 to T do
    bootstrap sample D_t from D by sampling |D| examples with replacement
    run A on D_t to produce a model M_t
end
return {M_t | 1 ≤ t ≤ T}
```

Comp9417

Random Forest

Algorithm RandomForest(D, T, d) // train ensemble of randomized trees

Input: data set D; ensemble size T; subspace dimension d.

Output: set of models; predictions to be combined by voting or averaging.

```
for t = 1 to T do
    bootstrap sample D_t from D by sampling |D| examples with replacement
    select d features at random and reduce dimensionality of D_t accordingly
    train a tree model M_t on D_t without pruning
end
return {M_t | 1 ≤ t ≤ T}
```

Comp9417

Boosting

1. set $w_i = 1$ for $i = 1, \ldots, n$
2. Repeat until sufficient number of hypothesis
    - Train model L_j using the dataset with weight w
    - Increase w_i for misclassified instances of L_j
3. Ensemble hypothesis is the weighted majority/weighted average of k learners $L_1, \ldots, L_k$ with weight $\lambda_1, \ldots, \lambda_k$ which are proportional to the accuracy of L_j

Comp9417
Boosting

- $w_i = \frac{1}{N} \ \forall\, i$
- For j = 1 to k do
  - Learn L_j with data weight w
  - $\epsilon_j = \sum_{i=1}^{n} w_i^{(j)} 1[L_j(\mathrm{x}_i) \neq y_i] / \sum_{i=1}^{n} w_i^{(j)}$
  - $\lambda_j = \frac{1}{2}\log(\frac{1-\epsilon_j}{\epsilon_j})$
  - $w_i^{(j+1)} = w_i^{(j)} \exp(\lambda_j)$ for misclassified instances
  - $w_i^{(j+1)} = w_i^{(j)} \exp(-\lambda_j)$ for correct instances
- End

- Make prediction using the final model: $y(\mathrm{x}) = sign(\sum_{j=1}^{k} \lambda_i L_j(\mathrm{x}))$

Comp9417

Unsupervised Learning

Supervised learning — classes are *known* and need a "definition", in terms of the data. Methods are known as: classification, discriminant analysis, class prediction, supervised pattern recognition.

Unsupervised learning — classes are initially *unknown* and need to be "discovered" with their definitions from the data. Methods are known as: cluster analysis, class discovery, unsupervised pattern recognition.

So: *unsupervised learning* methods, such as *clustering*, address the problem of assigning instances to classes *given only observations about the instances*, i.e., without being given class "labels" for instances by a "teacher".

Comp9417

Cluster Analysis

Clustering algorithms form two broad categories:

- **hierarchical methods** and **partitioning methods**

Hierarchical algorithms are either **agglomerative** i.e. bottom-up or **divisive** i.e. top-down.

In practice, hierarchical agglomerative methods often used - efficient exact algorithms available, but more importantly to users the *dendrogram*, or tree, can be visualized.

Partitioning methods usually require specification of the number of clusters, then try to construct the clusters and fit objects to them.

Comp9417

K-means

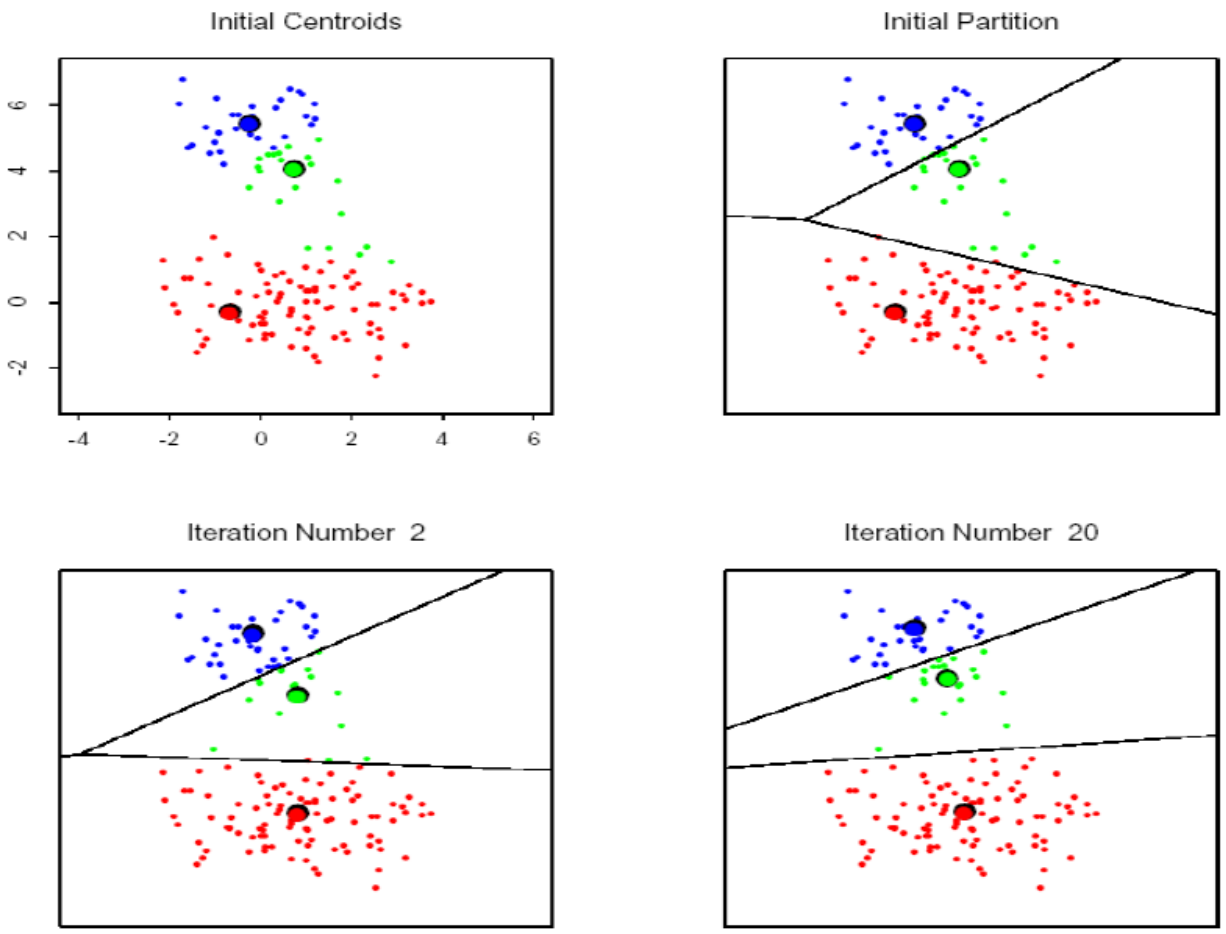
Set value for k, the number of clusters (by prior knowledge or via search)

Initialise: choose points for centres (means) of k clusters (at random)

Procedure:

1) assign each instance x to the closest of the k points to form k clusters
2) re-assign the k points to be the means of each of the k clusters
3) repeat 1 and 2 until convergence to a reasonably stable clustering

Comp9417
K-means



Comp9417

Hierarchical Clustering

- Bottom up: at each step join the two closest clusters (starting with single-instance clusters)
  - Design decision: distance between clusters
    E.g. two closest instances in clusters vs. distance between means
- Top down: find two clusters and then proceed recursively for the two subsets
  - Can be very fast
- Both methods produce a dendrogram (tree of “clusters”)

Comp9417

Hierarchical Clustering

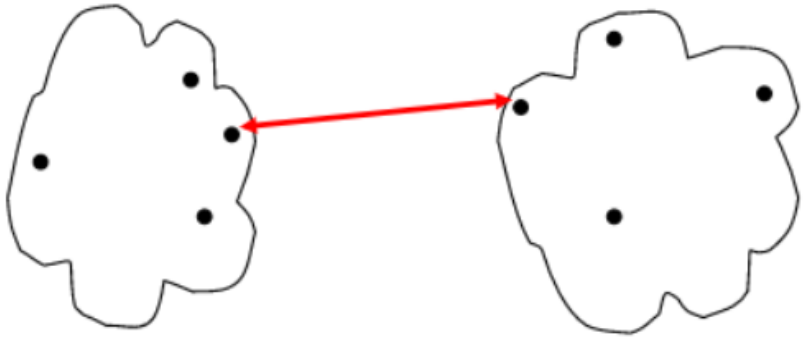
Algorithm Hierarchical agglomerative

/* dissimilarity matrix $D(ij)$ is given */

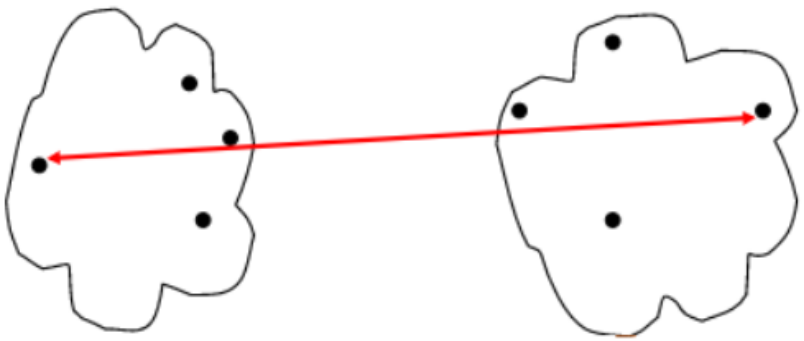
1. Find minimal entry d_{ij} in D and merge clusters i and j
2. Update D by deleting column i and row j, and adding new row and column $i \cup j$
3. Revise entries using
   $d_{k,i\cup j} = d_{i\cup j,k} = \alpha_i d_{ki} + \alpha_j d_{kj} + \gamma |d_{ki} - d_{kj}|$
4. If there is more than one cluster then go to step 1.

Comp9417

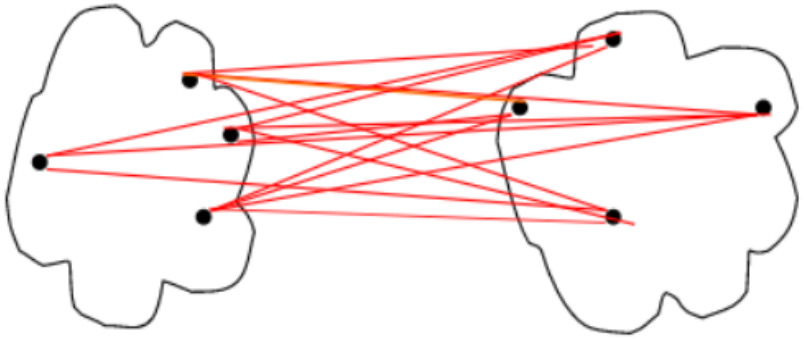
Hierarchical Clustering



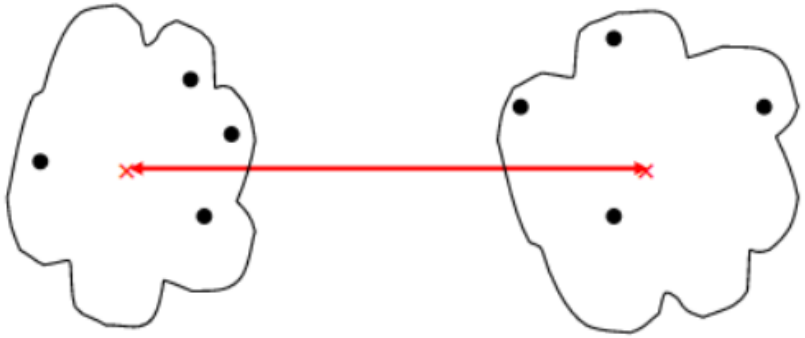
Single link



Complete link



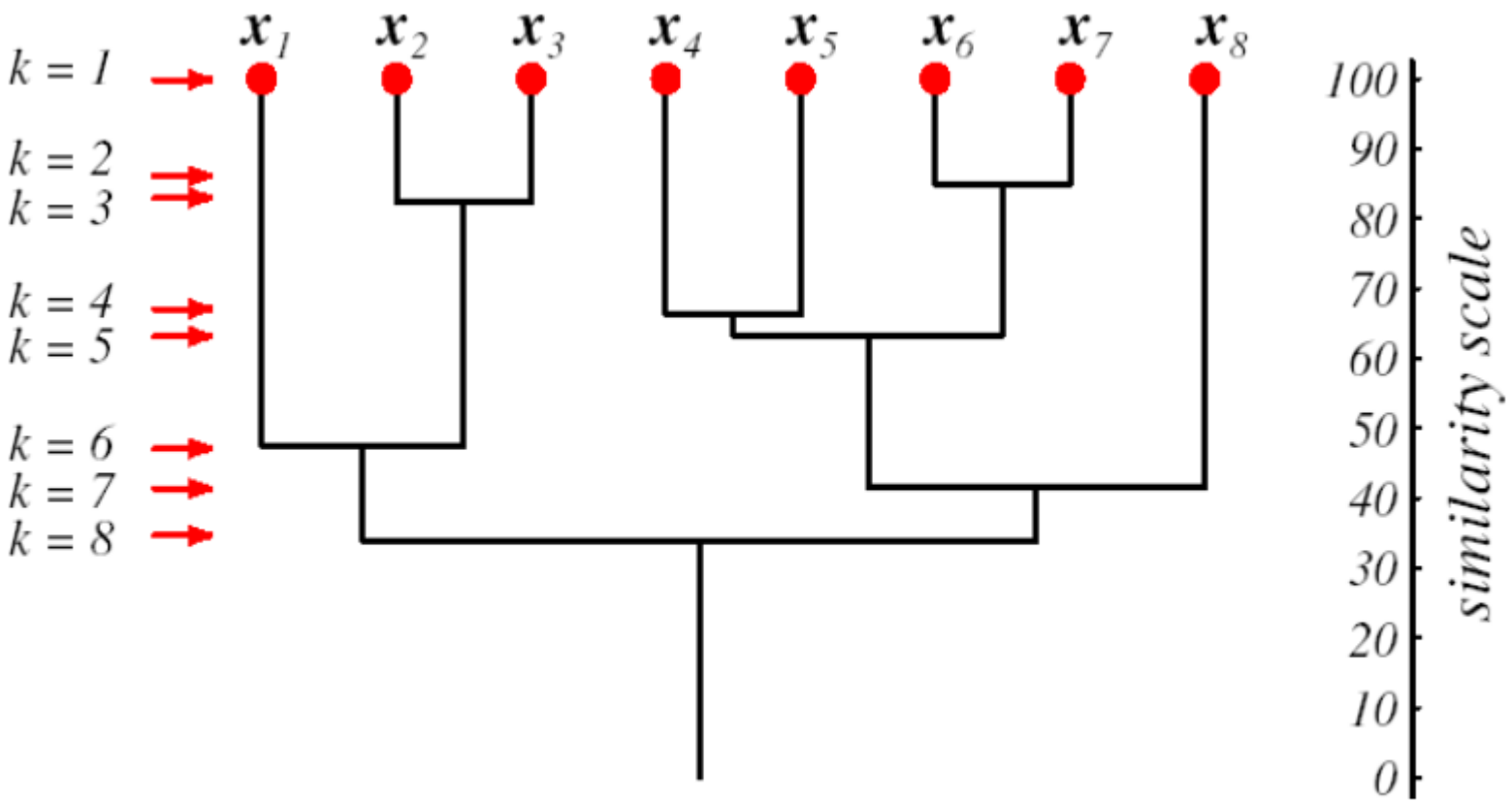
Average link



Centroid distance

Comp9417

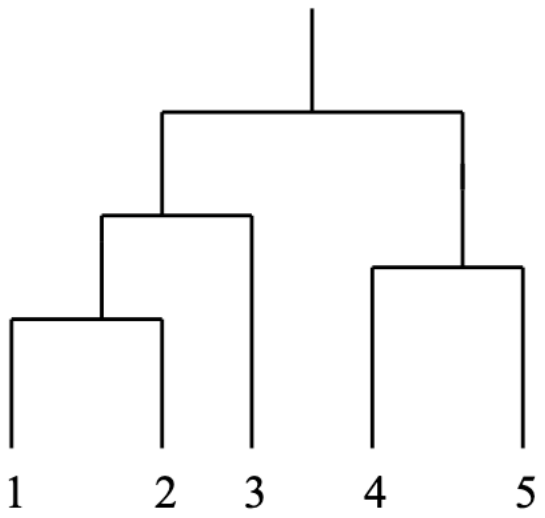
Hierarchical Clustering



Comp9417

Min or Single Link

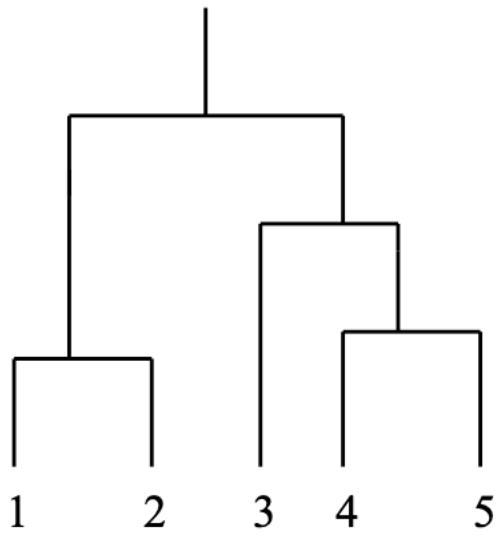
	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



Comp9417

Max or Complete Link

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



Comp9417

Group Average

- Similarity of two clusters is the average of pair-wise similarity between points in the two clusters.

$$\text{similarity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i, p_j \in \text{Cluster}_i \cup \text{Cluster}_j \\ p_j \neq p_i}} \text{similarity}(p_i, p_j)}{(|\text{Cluster}_i| + |\text{Cluster}_j|) * (|\text{Cluster}_i| + |\text{Cluster}_j| - 1)}$$

Comp9417

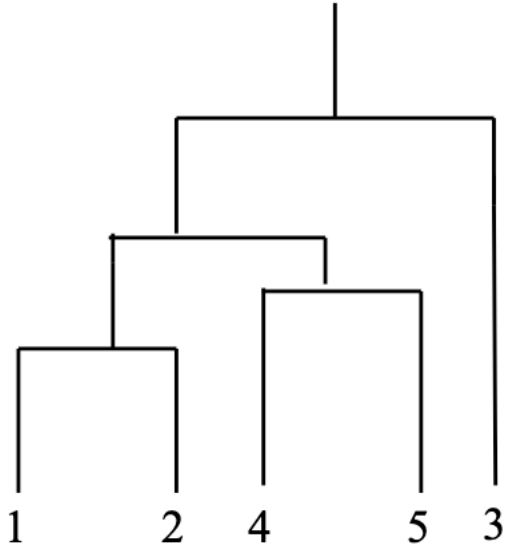
Group Average

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2		1.00	0.70	0.60	0.50
P3			1.00	0.40	0.30
P4				1.00	0.80
P5					1.00

	12	P3	P4	P5
12	1.00	0.567	0.717	0.533
P3		1.00	0.40	0.30
P4			1.00	0.80
P5				1.00

	12	P3	45
12	1.0	0.567	0.608
P3		1.00	0.5
45			1.00



Sim(12,3)=2*(0.1+0.7+0.9)/6 = 0.5666666

Sim(12,45)=2*(0.9+0.65+0.2+0.6+0.5+0.8)/12 = 0.608

Comp9417

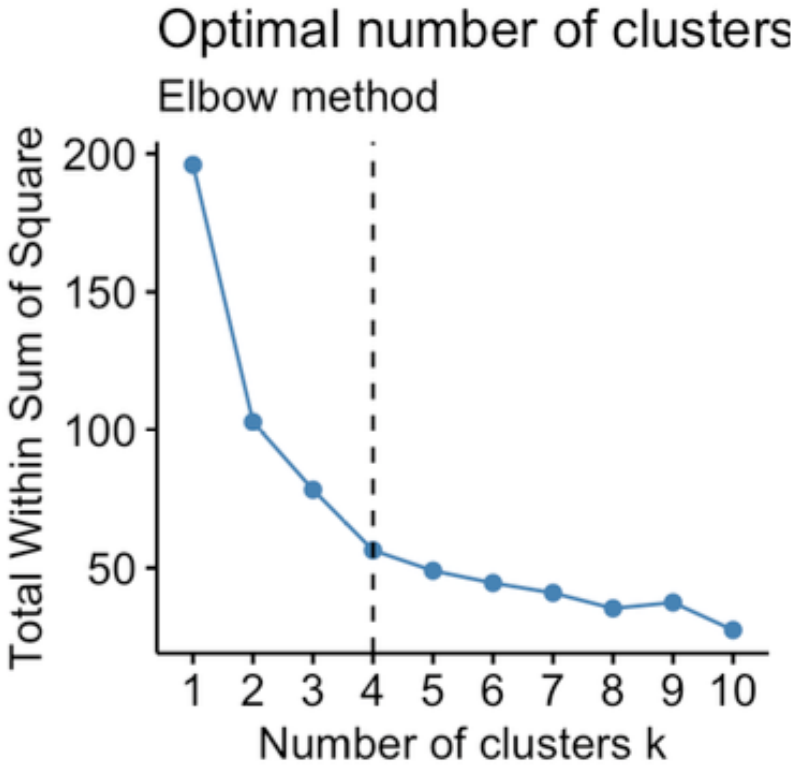
Numbers of Clusters

Many methods of estimating the "correct" number of clusters have been proposed, based on some clustering criterion:

- Elbow method:
  - measure the within-cluster dispersion (total sum of squared distances from each point to the cluster centroid)
  - compute this for various k choices
  - choose the k that doesn't improve the dispersion much

Comp9417

Numbers of Clusters



Comp9417
Sample

In these two questions you will apply the k-MEANS algorithm. You will use a univariate (one-variable) dataset containing the following 12 instances:

$$\text{Dataset} = \{\ 2.01,\ 3.49,\ 4.58,\ 4.91,\ 4.99,\ 5.01,\ 5.32,\ 5.78,\ 5.99,\ 6.21,\ 7.26,\ 8.00\ \}$$

Use the *Manhattan* or *city-block* distance, i.e., the distance between two instances x_i and x_j is the absolute value of the difference $x_i - x_j$. For example, if $x_i = 2$ and $x_j = 3$ then the distance between x_i and x_j is $|2 - 3| = 1$. Use the arithmetic mean to compute the centroids.

Apply the k-MEANS algorithm to the above dataset of examples. Let $k = 2$. Let the two centroids (means) be initialised to $\{3.33, 6.67\}$. On each iteration of the algorithm record the centroids.

After two iterations of the algorithm you should have recorded two sets of two centroids.

Centroids	**After 1 iteration**	**After 2 iterations**
Centroids 1	$\{\ 2.75,\ 5.81\ \}$	$\{\ 3.24,\ 6.01\ \}$
Centroids 2	$\{\ 4.00,\ 6.22\ \}$	$\{\ 4.17,\ 6.43\ \}$
Centroids 3	$\{\ 4.51,\ 6.87\ \}$	$\{\ 4.33,\ 6.65\ \}$
Centroids 4	$\{\ 4.67,\ 7.16\ \}$	$\{\ 4.51,\ 6.87\ \}$
Centroids 5	$\{\ 4.83,\ 7.03\ \}$	$\{\ 4.28,\ 6.79\ \}$

Comp9417
Sample

(a) After applying your algorithm to the dataset for two iterations, which of the sets of centroids in the table above has been learned ?
(select the row of the table with values closest to your centroids)

(a) Centroids 1
(b) Centroids 2
(c) Centroids 3
(d) Centroids 4
(e) Centroids 5

Now apply the algorithm for one more iteration. Record the new centroids after iteration 3 and answer the following question.

(b) After 3 iterations it is clear that:
(a) due to randomness in the data, the centroids could change on further iterations
(b) due to randomness in the algorithm, the centroids could change on further iterations
(c) k-MEANS converges in probability to the true centroids
(d) the algorithm has converged and the clustering will not change on further iterations
(e) the algorithm has not converged and the clustering will change on further iterations

Comp9417

Hypothesis space

Instance space

$$\begin{aligned} \mathit{Sky} \times \mathit{AirTemp} \times \ldots \times \mathit{Forecast} &= 3 \times 2 \times 2 \times 2 \times 2 \times 2 \\ &= 96 \end{aligned}$$

Hypothesis space

$$\begin{aligned} \mathit{Sky} \times \mathit{AirTemp} \times \ldots \times \mathit{Forecast} &= 5 \times 4 \times 4 \times 4 \times 4 \times 4 \\ &= 5120 \\ \text{(semantically distinct}^1\text{ only)} &= 1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3) \\ &= 973 \end{aligned}$$

Comp9417

Version Space

A hypothesis h is **consistent** *with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D.*

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x)$$

The **version space**, *$VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.*

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

Comp9417

Exhausting

Definition: *The version space $VS_{H,D}$ is said to be ϵ-**exhausted** with respect to c and $\mathcal{D}$, if every hypothesis h in $VS_{H,D}$ has error less than ϵ with respect to c and $\mathcal{D}$.*

$$(\forall h \in VS_{H,D})\ error_{\mathcal{D}}(h) < \epsilon$$

If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent random examples of some target concept c, then for any $0 \leq \epsilon \leq 1$, the probability that the version space with respect to H and D is not ϵ-exhausted (with respect to c) is less than

$$|H|e^{-\epsilon m}$$

If we want this probability to be below δ

$$|H|e^{-\epsilon m} \leq \delta$$

then

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

Comp9417

VC Dimension

Definition: *The* **Vapnik-Chervonenkis dimension**, $VC(H)$, *of hypothesis space* H *defined over instance space* X *is the size of the largest finite subset of* X *shattered by* H. *If arbitrarily large finite sets of* X *can be shattered by* H, *then* $VC(H) \equiv \infty$.

From the earlier slide on shattering a set of instances by a conjunctive hypothesis, if we have an instance space X where each instance is described by d Boolean features, and a hypothesis space H of conjunctions of up to d Boolean literals, then the VC Dimension $VC(H) = d$.

In general, for linear classifiers in d dimensions the VC dimension is $d + 1$.

Comp9417
Sample

Question 3 Refer to the VC dimension example for linear classifiers in the 2-dimensional x, y plane of slides 39–41 of the lecture notes. Answer the following:

1. give an intuitive argument for why the VC dimension must be at least 3;
2. suppose you have a set of 3 points that are collinear – does that change your argument ?
3. can the VC dimension be 4 ?

Answer

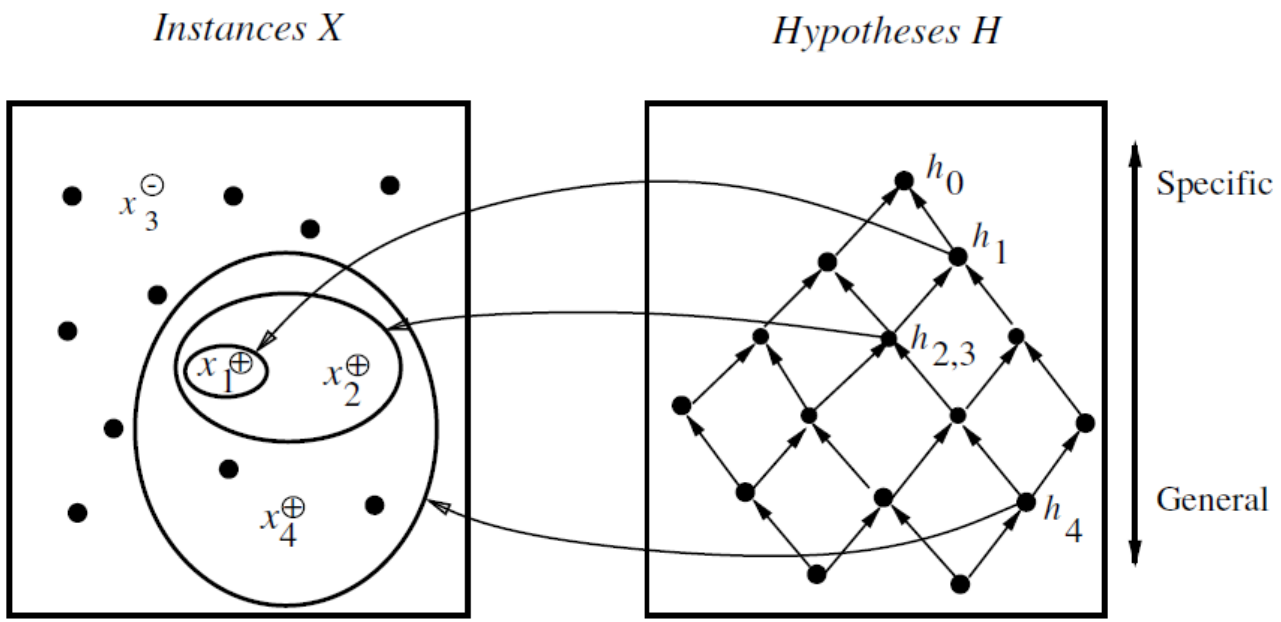
1. in the first 3 diagrams on slide 39 are shown sets of 3 points that can be shattered, i.e., a linear decision surface can be drawn for each of the 2^3 subsets of the points;
2. no – although a set of 3 collinear points cannot be shattered, as long as *at least one* set of 3 points can be shattered, the VC dimension must be at least 3;
3. to show that $VC(H) < d$, we must show that *no* set of size d can be shattered, and in this setting no sets of size four can be shattered, so $VC(H) = 3$ (there is always an XOR-type problem).

Find-S

An online, specific-to-general, concept learning algorithm:

- Initialize h to the most specific hypothesis in H
- For each positive training instance x
  - For each attribute constraint a_i in h
    - If the constraint a_i in h is satisfied by x
    - Then do nothing
    - Else replace a_i in h by the next more general constraint satisfied by x

Comp9417
Find-S



x_1 = <*Sunny Warm Normal Strong Warm Same*>, +

x_2 = <*Sunny Warm High Strong Warm Same*>, +

x_3 = <*Rainy Cold High Strong Warm Change*>, -

x_4 = <*Sunny Warm High Strong Cool Change*>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

h_1 = <*Sunny Warm Normal Strong Warm Same*>

h_2 = <*Sunny Warm ? Strong Warm Same*>

h_3 = <*Sunny Warm ? Strong Warm Same*>

h_4 = <*Sunny Warm ? Strong ? ?*>

Find-S

- $2n$ terms in initial hypothesis
- first mistake, remove half of these terms, leaving n
- each further mistake, remove at least 1 term
- in worst case, will have to remove all n remaining terms
  - would be most general concept - everything is positive
- worst case number of mistakes would be $n + 1$
- worst case sequence of learning steps, removing only one literal per step

Comp9417
Sample

Question 1 ([Blum et al., 2019]) To get a sense of how learning theory characterises sample complexity we start by formulating a simple *consistent learner* to learn *disjunctions*, i.e., Boolean OR functions, of d variables. Recall that a consistent learner is just one that makes no mistake on the training data. The target concept c is assumed to be expressed as a disjunction of literals, where a literal is defined as some feature x_i being true (having value 1).

So an instance is just a set of literals, such as $\{\mathbf{x}|x_1 = 1 \vee x_3 = 1 \vee x_8 = 1\}$. For example, if the target concept was to distinguish between spam and non-spam emails, the presence in an email of any of the features x_1, x_3 or x_8 would be enough to classify it as spam, whereas the absence of all of them would mean non-spam.

Question 1a) The hypothesis space H is the set of all disjunctions of d features. What is the size of this hypothesis space ?

Answer

Since we can represent any disjunctive hypothesis as specified above simply as the set of the d features that are true in the hypothesis, the hypothesis space is the power set of literals, so it has size 2^d.

Comp9417
Sample

Question 1b) Give an algorithm for a consistent learner for such disjunctive concepts from a set of labelled noise-free training examples S. *HINT*: try adapting the basic approach of the FIND-S algorithm to learn conjunctive concepts shown on slide 44 of the lecture notes.

Answer

Given the specification for learning disjunctive concepts, there is a straightforward algorithm.

Disjunctive Concept Learner:

- Initial hypothesis h is the set of all literals $x_i = 1$, $1 \leq i \leq d$
- For each negative instance x in S
  - Remove from h any literal $x_i = 1$
- Output concept that is the logical OR of the features remaining in h

Comp9417
Sample

Question 1c) Outline the steps in a proof that your disjunctive concept learning algorithm will find a consistent hypothesis h, i.e., that the error on sample S $error_S(h) = 0$.

Answer
Assume that the target concept c is in fact a disjunction. Then for any literal $x_i = 1$ in c, x_i will not be set to 1 in any negative example in S. So h will include $x_i = 1$. Since h will contain all such literals, h will correctly predict all positive examples in S. Furthermore, h will correctly predict negative on all negative examples in S since by design all features set to 1 in any negative example were discarded. Therefore, h is correct on all examples in S.

Question 1d) Analyse the sample complexity of the consistent learner for disjunctive concepts in the PAC learning setting, i.e., use the formula from slide 23 in the lecture notes.

Answer

$$m \geq \frac{1}{\epsilon}(\ln |H| + \ln(1/\delta))$$

Since we know from above that the size of the hypothesis space $|H|$ is 2^d we obtain

$$m \geq \frac{1}{\epsilon}(d \ln 2 + \ln(1/\delta))$$

Comp9417
Halving

Without loss of generality, suppose that the hypothesis space H is a set of Boolean functions.

HALVING ALGORITHM:

- **Initialise** the set of consistent hypotheses $C = H$
- **Repeat** get new instance x
  - let $\pi_0(C, x)$ be subset of C that predict 0
  - let $\pi_1(C, x)$ be subset of C that predict 1
  - **if** $|\pi_0(C, x)| > |\pi_1(C, x)|$ predict 0 **else** 1
  - **if** $\text{class}(x) = 0$ **then** $C = \pi_0(C, x)$ **else** $C = \pi_1(C, x)$

Worst-case mistake bound: on every example x the majority vote prediction is the opposite of the actual $\text{class}(x)$, so this is a mistake. However, on each prediction for x, the majority $\pi_0(C, x)$ (respectively $|\pi_1(C, x)|$) of functions are eliminated. Therefore the size of the set C will be (at least) reduced by half on every example x. That is, the number of mistakes $M_{Halving} \leq \log_2 |H|$.
However, in the best case the situation is reversed! On every example x the majority vote prediction is correct, so the number of mistakes is zero!

Comp9417
Winnow2

```
While some instances are misclassified
  For each instance x
    classify x using current weights w
    If predicted class is incorrect
      If x has class 1
        For each x_i = 1, w_i ← α w_i          # Promotion
        (if x_i = 0, leave w_i unchanged)
      Otherwise
        For each x_i = 1, w_i ← w_i / α         # Demotion
        (if x_i = 0, leave w_i unchanged)
```

Here x and w are vectors of features and weights, respectively.

- typically, the worst-case mistake-bound is something like $\mathcal{O}(r \log n)$

Comp9417

Sample

This dataset has 6 binary features, $x_1, x_2, \ldots x_6$. The class variable y can be either 1, denoting a positive example of the concept to be learned, or 0, denoting a negative example.

Example	x_1	x_2	x_3	x_4	x_5	x_6	Class
1)	0	0	0	0	1	1	1
2)	1	0	1	1	0	1	1
3)	0	1	0	1	0	1	0
4)	0	1	1	0	0	1	0
5)	1	1	0	0	0	0	1

Apply the WINNOW2 algorithm to the above dataset of examples **in the order in which they appear**. Use the following values for the WINNOW2 parameters: threshold $t = 2$, $\alpha = 2$. Initialise all weights to have the value 1.

Learned Weights	w_1	w_2	w_3	w_4	w_5	w_6
Weight vector 1	2.000	1.000	1.000	0.000	2.000	1.000
Weight vector 2	3.000	0.000	1.000	1.000	2.000	1.000
Weight vector 3	2.000	2.000	2.000	2.000	2.000	2.000
Weight vector 4	2.000	0.500	0.500	0.500	2.000	0.500
Weight vector 5	2.000	0.250	0.500	0.500	4.000	0.125

Comp9417
Sample

(a) After one epoch, i.e., one pass through the dataset, which of the above weight configurations has been learned ?

(a) Weight vector 1
(b) Weight vector 2
(c) Weight vector 3
(d) Weight vector 4
(e) Weight vector 5

ANSWER

(d)

(b) On which of the examples did the algorithm ***not*** make a mistake ?

(a) Examples 1), 2) and 5)
(b) Example 5)
(c) Example 4)
(d) Examples 4) and 5)
(e) None of the above

ANSWER

(e)

Comp9417
Sample

(c) The algorithm has learned a consistent concept on the training data:

(a) True
(b) False
(c) It is not possible to determine this

ANSWER

(a)

(d) Assume the target concept from which this dataset was generated is defined by exactly two features. The worst-case mistake bound for the algorithm on this dataset is approximately:

(a) 1.79
(b) 2.58
(c) 3.58
(d) 4.67
(e) 10.75

ANSWER

(c)