WEEKS 5-9

Introduction to Machine Learning

Dr Mykola Gordovskyy

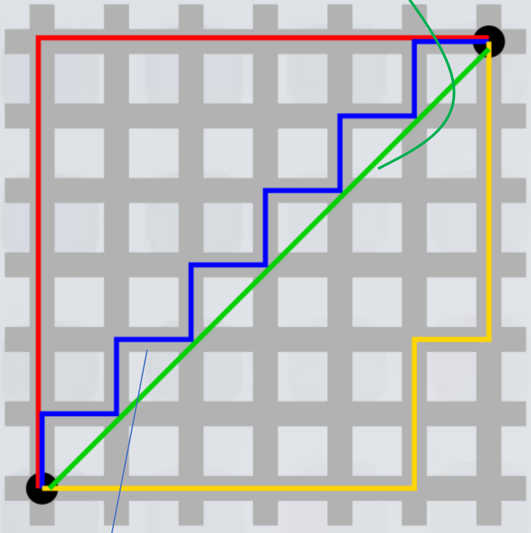
Week 8

- K-means clustering - metrics
- Association rules
- Semi-supervised learning
- Reinforcement learning
- Learning curve (example) -> LAB

K Means - metrics

Euclidean

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$



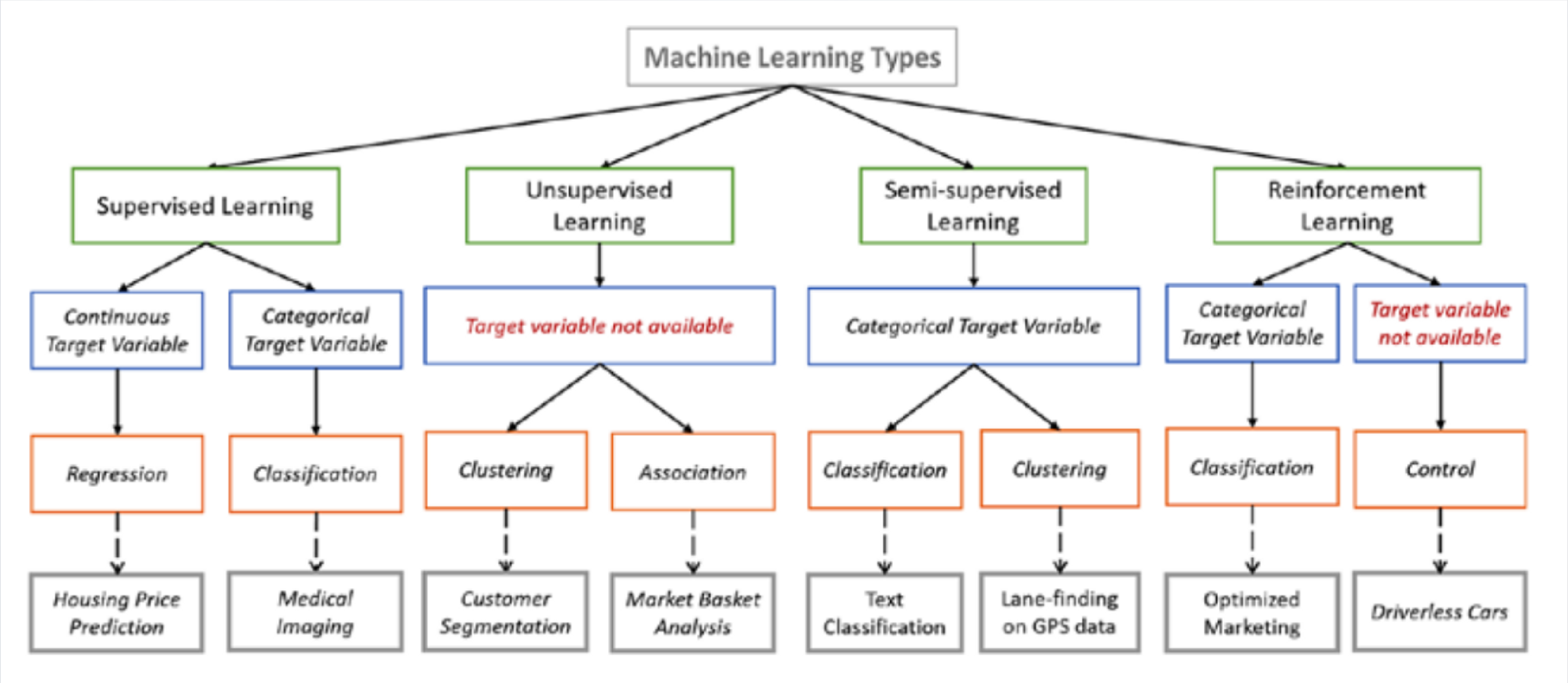
$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

Manhattan

Minkowski: generalisation of Euclidean

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

Hamming: similarity between strings



(from Sengupta et al. 2020)

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Bread	x	x			x	x		x				x			x	x	x		x	
Milk	x			x				x	x			x		x			x		x	x
Cheese	x		x	x				x			x	x			x		x	x	x	
Tomato		x				x		x	x		x				x			x	x	
Cucumbers		x			x	x			x				x		x				x	x
Aubergines	x		x			x	x						x	x		x			x	
Celery	x	x		x		x				x	x		x	x				x	x	
Pasta		x	x				x					x				x	x		x	x
Rice				x			x		x	x						x			x	
Eggs	x		x		x		x			x				x		x			x	x
Jam	x		x	x				x			x	x			x		x		x	
Cookies				x	x				x		x					x	x	x	x	
Wine			x	x		x	x			x	x				x				x	
Washing powder		x	x				x				x					x			x	x
Batteries	x					x							x	x					x	

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Bread	x	x			x	x		x				x			x	x	x		x	
Milk	x			x				x	x			x		x			x		x	x
Cheese	x		x	x				x			x	x			x		x	x	x	
Tomato		x				x		x	x		x				x			x	x	
Cucumbers		x			x	x			x				x		x				x	x
Aubergines	x		x			x	x						x	x		x			x	
Celery	x	x		x		x				x	x		x	x				x	x	
Pasta		x	x				x					x				x	x		x	x
Rice				x			x		x	x						x			x	
Eggs	x		x		x		x			x				x		x			x	x
Jam	x		x	x				x			x	x			x		x		x	
Cookies				x	x				x		x					x	x	x	x	
Wine			x	x		x	x			x	x				x				x	
Washing powder		x	x				x				x					x			x	x
Batteries	x					x							x	x					x	

Association rules

Also, items can characterise those making transactions (customer’s age, gender, car make etc

Item

Transactions

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	2
Bread	x	x			x	x		x				x			x	x	x		x	
Milk	x			x				x	x			x		x			x		x	x
Cheese	x		x	x				x			x	x			x		x	x	x	
Tomato		x				x		x	x		x				x			x	x	
Cucumbers		x			x	x			x				x		x				x	x
Aubergines	x		x			x	x						x	x		x			x	
Celery	x	x		x		x				x	x		x	x				x	x	
Pasta		x	x				x					x				x	x		x	x
Rice				x			x		x	x						x			x	
Eggs	x		x		x		x			x				x		x			x	x
Jam	x		x	x				x			x	x			x		x		x	
Cookies																				

Association rules

Itemset = a set of two or more items

{Aubergine, batteries} is an itemset (more precisely, 2-itemset)

Item's frequency and support = a number and proportion of the item in the dataset

{Aubergine} occurs in 8 out of 20 transactions, its support is 0.4

Itemset frequency and support = a number and proportion of the itemset in the dataset

{Aubergine,Batteries} occurs in 5 out of 20 transactions, its support is 0.25

Association rules

Ultimately, the objective is to find constructions, or **rules**, in form

IF {*ITEMS A(, B, C....)* HAPPEN(S)} THEN {*ITEM Z* HAPPENS}

Antecedent
Condition
Prerequisite
…

Consequent
Outcome
Result
…

Association rules

Itemset = a set of two or more items

{Aubergine, batteries} is an itemset (more precisely, 2-itemset)

Item's frequency and support = a number and proportion of the item in the dataset

{Aubergine} occurs in 8 out of 20 transactions, its support is 0.4

Itemset frequency and support = a number and proportion of the itemset in the dataset

{Aubergine,Batteries} occurs in 5 out of 20 transactions, its support is 0.25

Confidence = support of consequent & antecedent divided by support of antecedent

Support{Aubergine,Batteries}/Support{Aubergine} = 0.25/0.4=0.625

Association rules - thresholds

Support and Confidence

A possible rule $\boldsymbol{R}$: IF $\{A\}$ THEN $\{B\}$ or $\{A\} \rightarrow \{B\}$

Support of $\{A\}$: Occurrence of $\{A\}$ in $\boldsymbol{I}$ (whole dataset)

Support of $\{\boldsymbol{A,B}\}$: Occurrence of $\{\boldsymbol{A,B}\}$ in $\boldsymbol{I}$ (whole dataset)

Confidence of rule $\boldsymbol{R}$ = Support of $\{\boldsymbol{A,B}\}$/Support of $\{A\}$

A possible rule $\boldsymbol{R}$ becomes a rule if the confidence of $\boldsymbol{R}$ and support of $\{\boldsymbol{A,B}\}$ exceed relevant thresholds

Confidence and support thresholds are set by user

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Bread	x	x			x	x		x				x			x	x	x		x	
Milk	x			x				x	x			x		x			x		x	x
Cheese	x		x	x				x			x	x			x		x	x	x	
Tomato		x				x		x	x		x				x			x	x	
Cucumbers		x			x	x			x				x		x				x	x
Aubergines	x		x			x	x						x	x		x			x	
Celery	x	x		x		x				x	x		x	x				x	x	
Pasta		x	x				x					x				x	x		x	x
Rice				x			x		x	x						x			x	
Eggs	x		x		x		x			x				x		x			x	x
Jam	x		x	x				x			x	x			x		x		x	
Cookies				x	x				x		x					x	x	x	x	
Wine			x	x		x	x			x	x				x				x	
Washing powder		x	x				x				x					x			x	x
Batteries	x					x							x	x					x	

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Bread	x	x			x	x		x				x			x	x	x		x	
Milk	x			x				x	x			x		x			x		x	x
Cheese			x	x				x			x	x			x		x	x	x	
Tomato		x				x		x	x		x				x			x	x	
Cucumbers		x			x	x			x				x		x				x	x
Aubergines	x		x			x	x						x	x		x			x	
Celery	x	x		x		x				x	x		x	x				x	x	
Pasta		x	x				x					x				x	x		x	x
Rice				x			x		x	x						x			x	
Eggs	x		x		x		x			x				x		x			x	x
Jam	x		x	x				x			x	x			x		x		x	
Cookies				x	x				x		x					x	x	x	x	
Wine	x		x	x		x	x			x	x				x				x	
Washing powder		x	x				x				x					x			x	x
Batteries	x					x							x	x					x	

Association rules – thresholds - example

Support{Jam,Wine}=6/20 = 0.3

Support{Jam,Wine,Cheese} = 5/20 **= 0.25**

Confidence of {Jam,Wine}→ {Cheese} is

Support{Jam,Wine,Cheese}/Support{Jam,Wine} =0.25/0.3 = **0.83**

Why do we need both, support and confidence?

Confidence = 1.0 but support = 0.05

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Bread	x	x			x	x		x				x			x	x	x		x	
Milk	x			x				x	x			x		x			x		x	x
Cheese	x		x	x				x			x	x			x		x	x	x	
Tomato		x				x		x	x		x				x			x	x	
Cucumbers		x			x	x			x				x		x				x	x
Aubergines	x		x			x	x						x	x		x			x	
Celery	x	x		x		x				x	x		x	x				x	x	
Pasta		x	x				x					x				x	x		x	x
Rice				x			x		x	x						x			x	
Eggs	x		x		x		x			x				x		x			x	x
Jam	x		x	x				x			x	x			x		x		x	
Cookies				x	x				x		x					x	x	x	x	
Kettle				x			x													
Washing powder		x	x				x				x					x			x	x
Sesame oil	x						x													

Association rule mining

Brute force approach

1) Select all possible association rules

2) Calculate their support and confidence values

3) Select those with support and confidence values above thresholds

It will work, but computationally VERY expensive!!!

Association rule mining

More efficient two-step approach: Decouple support and confidence

ID#	
0	Bread, Tomato
1	Bread, Pasta, Olives, Egg
2	Tomato, Pasta, Olives, Cheese
3	Bread, Tomato, Pasta, Olives
4	Bread, Tomato, Pasta, Cheese

Possible rules:

{Tomato, Pasta} → {Olives} (s=0.4, c=0.67)
{Tomato, Olives} → {Pasta} (s=0.4, c=1.0)
{Pasta, Olives} → {Tomato} (s=0.4, c=0.67)
{Olives} → {Tomato, Pasta} (s=0.4, c=0.67)
{Pasta} → {Tomato, Olives} (s=0.4, c=0.5)
{Tomato} → {Pasta, Olives} (s=0.4, c=0.5)

Any combination of items in a rule gives the same support but, generally, different confidence

Association rule mining

More efficient two-step approach: Decouple support and confidence

Any combination of items in a rule gives the same support but, generally, different confidence

Hence, we can use a two-step process with the support and confidence requirements decoupled as follows:

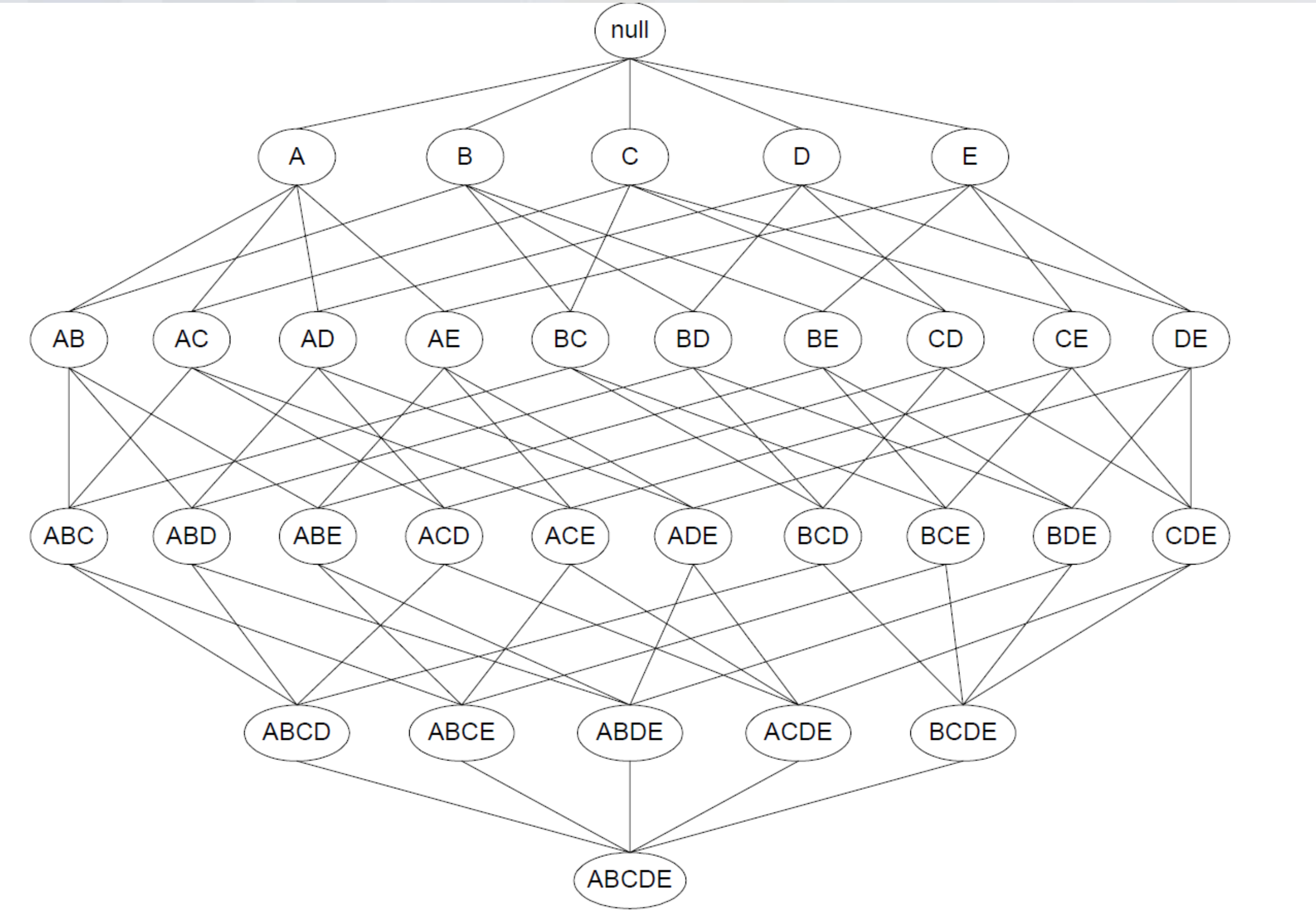
Step #1: Frequent Itemset Generation
– Generate all itemsets whose support exceeds support threshold

Step #2: Rule Generation
– Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

It will work, but still computationally expensive because of #1

Association rule mining

With N items in the database you can create 2^N itemsets



(from Tan, Steinbach, Kumar 2004)

With items ABCDE, you can create 2^5=32 itemsets

Association rule mining

With M items in the itemset you can create (2^M -2) possible rules

With Blue, Green, Red you can create

{Blue Green} → {Red}
{Blue Red} → {Green}
{Blue} → {Red Green}
{Green} → {Blue Red}
{Red} → {Blue Green}
{Red Green} → {Blue}

Frequent itemset generation

Reduce the number of candidates (M)

- Complete search: M=2d
- Use pruning techniques to reduce M

Reduce the number of transactions (N)

- Reduce size of N as the size of itemset increases
- Used by DHP and vertical-based mining algorithms

Reduce the number of comparisons (NM)

- Use efficient data structures to store the candidates or transactions
- No need to match every candidate against every transaction

Frequent itemset generation

Apriori principle:

– If an itemset is frequent, then all of its subsets must also be frequent

Apriori principle works because

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

… which means that

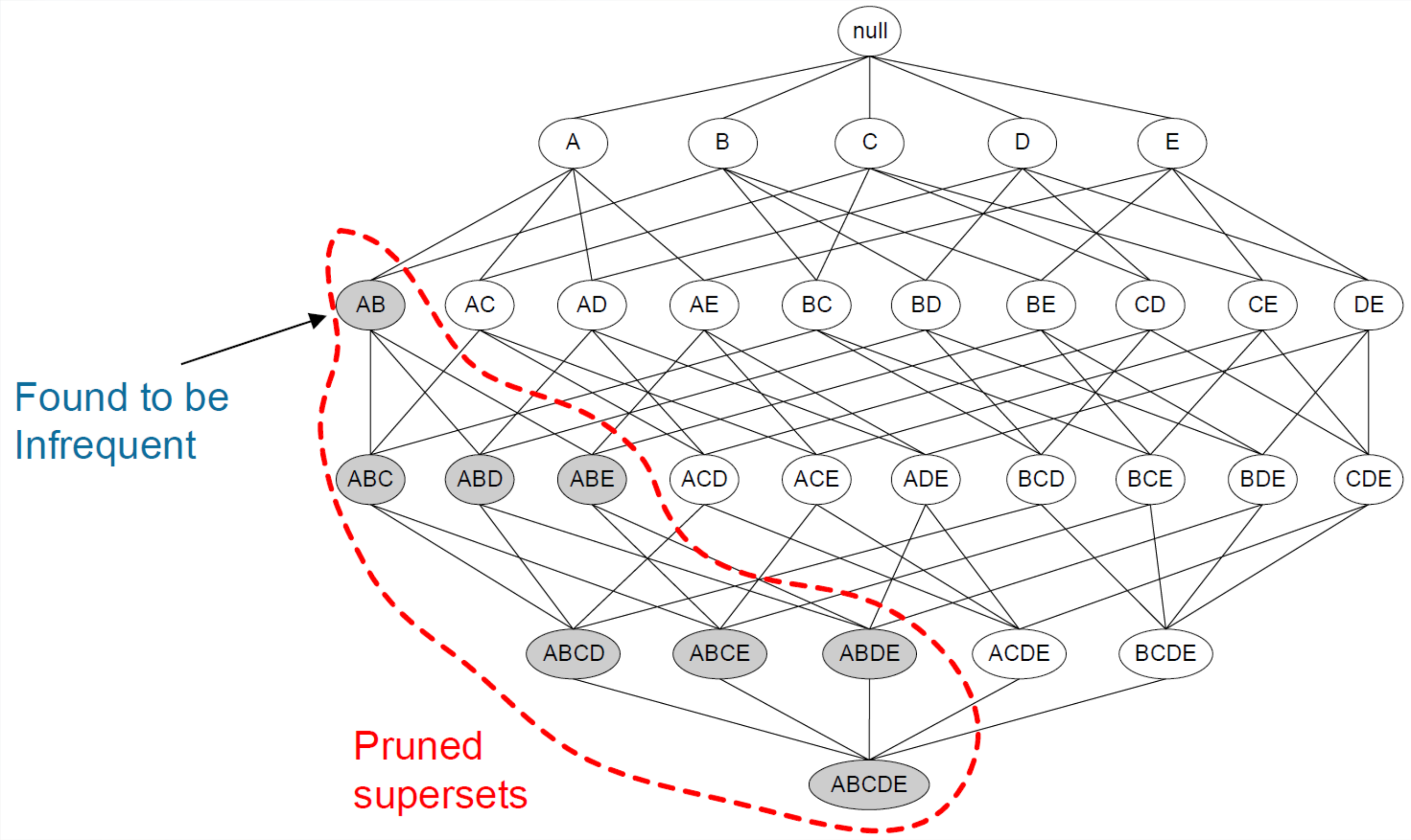
– Support of an itemset never exceeds the support of its subsets

– This is known as the anti-monotone property of support

Frequent itemset generation

Therefore, if a support for an item or itemset *S* is low, then support for any larger itemset containing *S* will be low

And, hence, we can do ‘pruning’, i.e. remove some of our 2^N itemsets



(from Tan, Steinbach, Kumar 2004)

Itemset generation using apriori principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
6 + 6 + 1 = 13

(from Tan, Steinbach, Kumar 2004)

Itemset generation using apriori principle

Algorithm

- Begin with *k=1*, generate frequent itemsets of length k(*=1*), i.e. select items with *support> support threshold*

- Repeat until no new itemsets with *support>support threshold* can be found:
    1) k=k+1
    2) Generate itemstes of lengths k+1, which contain frequent itemsets of length k identified during the previous iteration
    3) Evaluate *support* values for newly generated itemsets of length k+1
    4) Prune (i.e. remove) newly generated itemsets with *support< support threshold*

- Once all frequent (i.e. with *support> support threshold*) itemsets identified, generate all possible rules (or candidates) based on the frequent itemsets

- Evaluate confidence values for the possible rules (or candidates) generated

- Select possible rules with *confidence>confidence threshold*

Job done

Semi-supervised learning

<u>Why</u>

- I may have lots of training data, but ~~too lazy~~ it is too time/resource-consuming to label it
- The data may be coming from different sources and labels for part of the data may not be considered reliable enough
- There might be a mix of labelled data and unlabelled data clustering around labelled data
- Etc etc