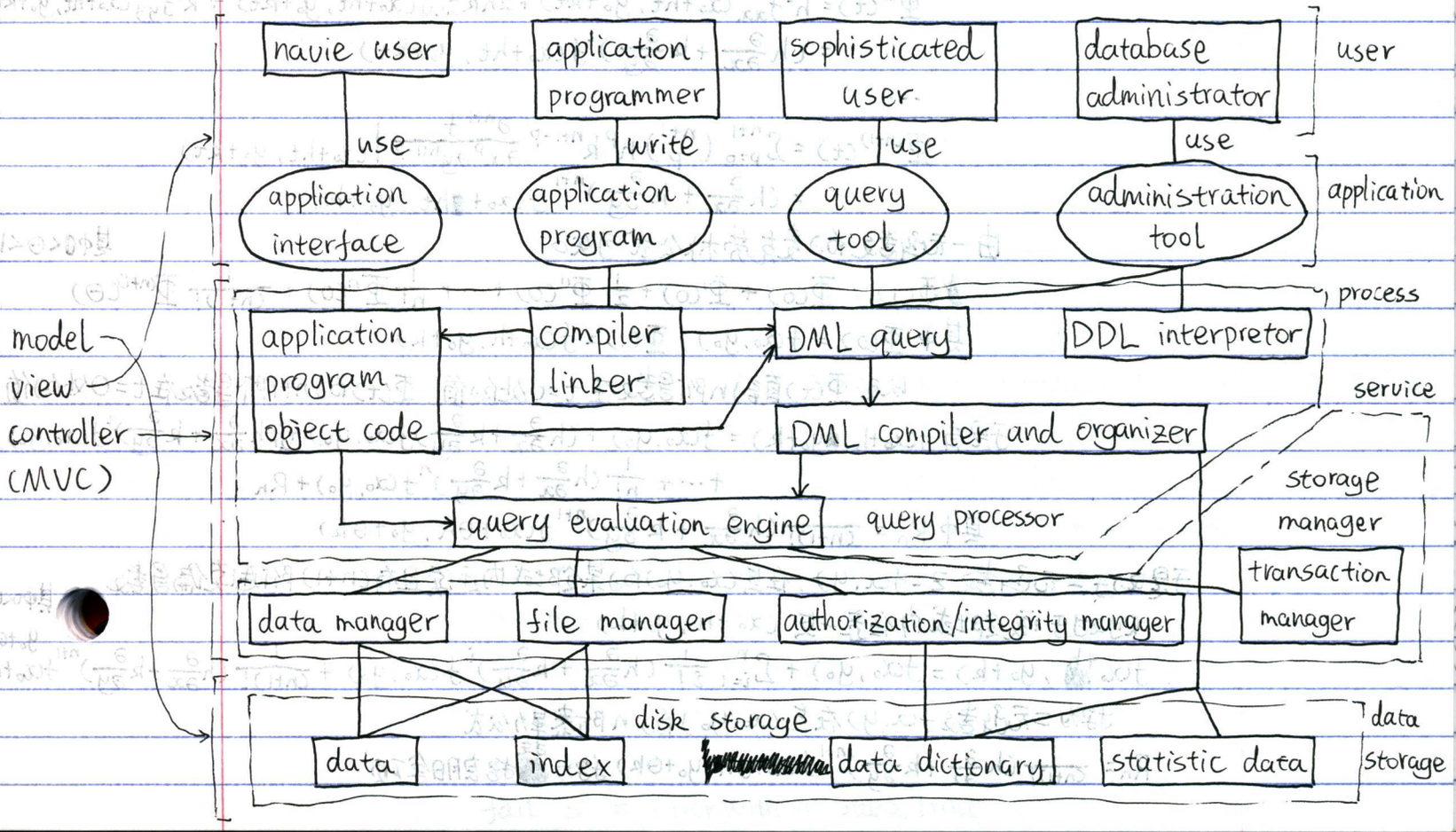
Python - P46

knowledge pattern : characterization : summarizing data
discrimination : comparing data
association : measuring frequency of data occurrence
classification : modeling data as class
clustering : high group similarity, low external similarity

data operation : cleaning : to remove noise and inconsistent data
integration : where multiple data source may be combined
selection : where data relevant to task retrieved from database
transformation : where data transformed into form appropriate for mining
mining : intelligent method applied in order to extract data pattern
evaluation : to identify truly interesting pattern
presentation : where visualization of knowledge is presented

data definition language (DDL) : to specify database schema
data manipulation language (DML) : to express query and update

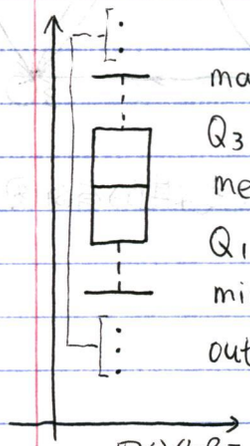


Python - P47

attribute type :
- qualitative
  - nominal : relating to name, no meaningful order, not quantitative
  - binary : nomial attribute with only two categories / states
    - symmetric : equally valuable, carry same weight
    - asymmetric : not equally important
  - ordinal : meaningful order or ranking, no magnitude between successive values
- numeric : quantitative, represented in integer / real value

	interval scale	ratio scale
ratio	unable to calculate ratio	able to calculate ratio
zero point	arbitrary zero point	absolute zero / character of origin
mean	arithmetic mean	geometric/harmonic mean
measurement	size and magnitude as multiple factors of defined unit	size and magnitude as one factor of one defined unit in terms of another

box plot



- max : $Q_3 + 1.5IQR$
- Q_3 (quartile 3) : 75th percentile
- median (Q_2) : 中位数
- Q_1 (quartile 1) : 25th percentile
- min : $Q_1 - 1.5IQR$
- outlier : outlier > $Q_3 + 1.5IQR$ V outlier < $Q_1 - 1.5IQR$

四分位距 (interquartile, IQR) : $Q_3 - Q_1$, 为一种稳健统计 (robust statistic)

四分位差 (quartile deviation, QD) : $\frac{Q_3 - Q_1}{2}$

data issue

- incomplete / missing : lacking attribute value / certain attribute of interest / containing only aggregate data
- inaccurate / noisy : containing error / value deviate from expected
- inconsistent : containing discrepancy, inconsistency in department codes

handling missing data :
- ignore value / tuple
- fill missing value manually
- using global constant
- using measure of central tendency for attribute
- using attribute mean / median for all samples
- using most probable value

Python - P48

data smoothing technique for noisy data
- binning : smooth sorted data value by consulting neighborhood
- regression : conform data value to function
- outlier analysis : require to find out value differ considerably from other values

data integration : match or relate object from different sources
- entity identification : not all entities identified consistently
  - functional dependency and referential constraint in source system may not match in target system
- correlation analysis : measure how attribute vary depending on another

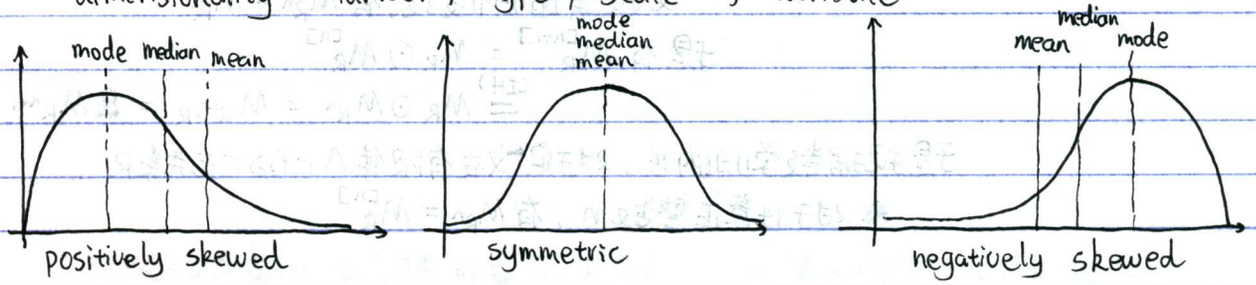
data reduction : dimensionality reduction : attribute subset selection : removing irrelevant attribute
- attribute construction : small set of more useful attribute derived from original set
- numerosity reduction : data replaced by alternative, smaller representation using parametric / nonparametric model

data transformation :
- aggregation : summary operation applied to data
- normalization : data scaled to fall within specified range
- generalization : low-level "primitive" data replaced by high-level concept through concept hierarchy

knowledge discovery : extraction of knowledge from data by detecting interesting pattern
- understandable / valid / useful / novel

data characteristic : volume : staggering amount of data being generated
- jerk or jounce - acceleration of acceleration of amount of data
- complexity : structure / spatial / temporal
- dimensionality : number / type / scale of attribute

centrality



Python - P49

max (iterable, * [, key, default])
min (iterable, * [, key, default])
max (arg1, arg2, *args [, key])
min (arg1, arg2, *args [, key])

return largest/smallest item in iterable or of two or more argument
if one positional argument provided, should be iterable
the largest / smallest in the iterable returned
if two or more positional arguments provided
the largest / smallest of positional arguments returned
key : specify one-argument ordering function like used for list.sort()
default: specify object to return if provided iterable empty
ValueError raised if iterable empty and default not provided
if multiple items are maximal / minimal
return the first one encountered
consistent with other sort-stability preserving tool
sorted(iterable, key = keyfunc, reverse = True/False)[0]
heapq.nlargest(1, iterable, key = keyfunc)
heapq.nsmallest(1, iterable, key = keyfunc)

pow (x, y [, z]), return x to the power y pow(x, y) % z
if z present, return x to the power y, modulo z, more efficiently than
two-argument form pow(x,y) equivalent to power operator x ** y
if y negative, z must be omitted
if z present, x and y must be of integer type, and y must be non-negative

repr (object), return string containing printable representation of object eval()
make attempt to return string yield object with same value when passed to
otherwise string enclosed in angle bracket contain name of type of object
together with additional information including name and address
class can control function return by defining __repr__() method

Python - P50

print (*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
print objects to text stream file, seperated by sep, followed by end
sep/end/file/flush must be given as keyword argument if present
all non-keyword argument converted to string like str()
written to stream, seperated by sep, followed by end
sep/end must be string, can be None to use default value
if no objects given, just write end
file must be object with method write(string)
if not present or None, use sys.stdout
print() can not be used with binary mode file object
use file.write(...) instead
output buffered usually determined by file
if flush is True, stream flushed forcibly

str(object='')
str(object=b'', encoding='utf-8', errors='strict')
return string version of object, or empty string if object not provided
if neither encoding nor errors given
str(object) return object.__str__()
"informal" or nicely printable string representation of object
fall back to returning repr(object) if object not have method __str__()
if at least one of encoding or errors given
object should be bytes-like object, 如 bytes / bytearray
if object is bytes / bytearray object
str(object, encoding, errors) 等价于 bytes.decode(encoding, errors)
otherwise bytes object underlying buffer object obtained before calling bytes.decode
passing bytes object to str() without encoding or errors
fall under first case of returning informal string representation
also -b command-line option to Python

reversed(seq), return reverse iterator
seq must be object has __reversed__() method
or support sequence protocol starting at 0
__len__() method and __getitem__() method with integer argument

Python - P51

class property(fget=None, fset=None, fdel=None, doc=None) return property attribute

fget : function for getting attribute value
fset : function for setting attribute value
fdet : function for deleting attribute value
doc : create docstring for attribute
otherwise will copy docstring of fget (if exist)

如 class C:

```
def __init__(self):
    self._x = None                    c = C()
def getx(self):          ] C.x invoke getter
    return self._x
def setx(self, value):   ] c.x = value invoke setter
    self._x = value
def delx(self):          ] del c.x invoke deleter
    del self._x
x = property(getx, setx, delx, "it's _x")
```

can create read-only property using property() as decorator

@property decorator turn method into getter for read-only attribute with same name
take docstring of method as docstring of property

decorator create copy of property with corresponding accessor function
set to decorated function with same name as original property

如在 class C 中

```
@property                ] 设置 x 的 getter
def x(self):
    """ it's _x """
    return self._x
@.x.setter               ] 设置 x 的 setter
def x(self, value):
    self._x = value
@x.deleter               ] 设置 x 的 deleter
def x(self):
    del self._x
```

Python - P52

round (number [, ndigits]) , return <number> rounded to <ndigits> precision after decimal point
- if ndigits omitted or None , return nearest integer to input
- for built-in type supporting round()
  - value rounded to closest multiple of 10 to power minus ndigits
- if two multiple equally close , rounding done toward even choice
  - ex round(0.5) , round(-0.5) → 0
  - round(1.5) , round(-1.5) → 2
- any integer value (positive/0/negative) valid for ndigits
- if ndigits omitted or None , return value is integer
  - otherwise return value same type as <number>
- for general Python object number
  - delegate to number.__round__

sorted (iterable , * , key = None , reverse = False)
- return new sorted list from item in iterable
- optional argument key , reverse must be specified as keyword argument
- key : specify function of one argument
  - used to extract comparison key from each element in iterable
  - default value None : compare element directly
- reverse : if set to True , list element sorted as if comparison reversed
- use functools.cmp_to_key() to convert old-style cmp function to key function
- built-in sorted() function guaranteed to be stable
  - guarantee not to change relative order of element compare equal

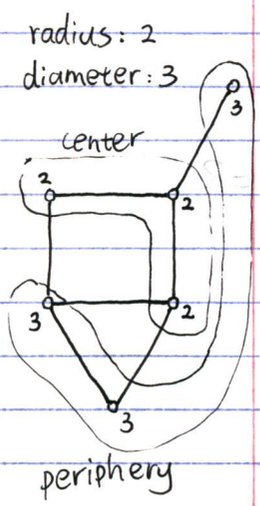
class slice (stop)
class slice (start , stop [, step]) range(start, stop , step)
- return slice object representing set of indices specified by
  - Start , step default to None
- slice object have read-only data attribute start , stop , step
  - return argument values , have no other explicit functionality
  - used by Numerical Python and third party extension
- slice object generated when extended indexing syntax used
  - ex a[start : stop : step] or a[start : stop , i]
- itertools.islice() for alternate version return iterator

Python - P53

graph centrality, eccentricity (偏心率): 对于连通图 $G=(V,E)$ 的点, $v \in V$

记点 v 的偏心率 $\epsilon(v)$ 为 v 到其他顶点的距离最大值

即有 $\epsilon(v) = \max\{d(v,u) \mid u \in V \wedge u \neq v\}$



radius (半径): minimum eccentricity

即 $r(G) = \min\{\epsilon(v) \mid v \in V\}$

diameter (直径): maximum eccentricity

即 $d(G) = \max\{\epsilon(v) \mid v \in V\}$

直径也是连通图中最远的两点间的距离

通常需要求任意两点之间的最短路径长度, 则长度的最大值为图的直径

center node (中心点): 偏心率等于 r 的顶点

即对于 $v \in V$, 有 $\epsilon(v) = r(G)$, 则 v 为 G 的中心点

periphery node (边缘点): 偏心率等于 d 的顶点

即对于 $v \in V$, 有 $\epsilon(v) = d(G)$, 则 v 为 G 的边缘点

eccentricity centrality : minimize maximum distance

closeness centrality : minimize distance to farthest node

betweenness : shortest paths between all pairs

prestige centrality: measure importance or rank of node

recursively measure indegree of node

converge to dominant : eigenvector / eigenvalue

power iteration approach : ratio of max entry in iteration k to max entry of iteration k-1

yield eigenvalue estimate

pagerank centrality: prestige computation for web search

probability of random surfer landing on page

recursively depend on inbound pagerank

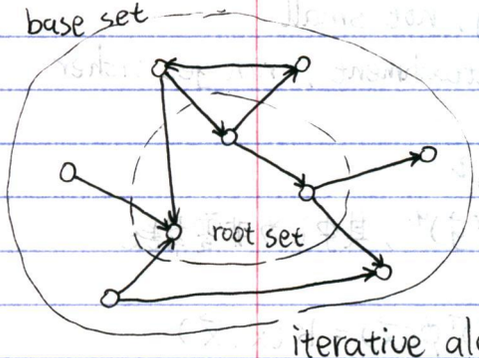
fully defined as combination : normalized prestige + random jump

probability of jump to any other page any means

Python - P54

HITS (hyperlink-induced topic search，超文本敏感标题搜索)

- 2种 hub and authority
- 对于用户输入关键词后，算法对返回的匹配页面计算两种值
  - 枢纽值(hub score)：所有导出链接指向页面的权威值之和
  - 权威值(authority score)：所有导入链接所在页面的枢纽值之和
  - 两种值相互依存，相互影响
- to begin ranking, $\forall p$ auth(p) = 1 and hub(p) = 1
- in order to calculate hub/authority score of each node
  - repeated iteration of Authority Update / Hub Update applied
  - k-step application of Hub-Authority algorithm
    - entail applying for k times first Authority Update and then Hub Update
- Authority Update Rule : authority score is sum of hub score of page point to p
  - $\forall p$ $auth(p) = \sum_{q \in P_{to}} hub(q)$, P_{to} : all page link to page p
- Hub Update Rule : hub score is sum of authority score of page p point to
  - $\forall p$ $hub(p) = \sum_{q \in P_{from}} auth(q)$, P_{from} : all page p link to
- normalization : final hub-authority score of node determined
  - after infinite repetition of algorithm
  - as directly and iteratively applying Update Rule lead to diverging value
  - necessary to normalize matrix after every iteration
  - value obtained from this process eventually converge
  - $\forall p$ $auth(p) = auth(p) / \sqrt{\sum_q [auth(q)]^2}$
  - $\forall p$ $hub(p) = hub(p) / \sqrt{\sum_q [auth(q)]^2}$



- iterative algorithm based on linkage of document on the web
  - like Page, Brin's PageRank
  - by search item
  - query depedent : hub/authority score resulting from link analysis influenced
  - as corollary, executed at query time, not at indexing time
    - with associated hit on performance accompany query-time processing
  - not commonly used by search engine
  - compute two scores per document (hub/authority) as opposed to single score
  - processed on small subset of 'relevant' document not all document (PageRank)
    - 'focused subgraph' or base set

Python - P55

real graph model : small world property : short path between node pair
scale logarithmically with number node
ultra-small-world much less than $\lg(n)$
scale-free property : scale-free degree distribution
follow power law
intuitively : vast majority of node - very small degree
few "hub node" with high degree
use cumulative degree distribution to smooth
clustering effect : two node likely to be connected if share neighbour
hierarchical clustering of node
sparsely connected node : smaller degree
part of higher clustering area
few hub node : higher degree
connect higher clustering area

random graph model : Erdös-Rényi : small-world , no clustering
Watts-Strogatz : high clustering , not small
Barabási-Albert : preferential attachment , rich get richer

polynomial kernel : homogeneous : $K_q(\vec{x},\vec{y}) = \phi(\vec{x})^T\phi(\vec{y}) = (\vec{x}^T\vec{y})^q$
inhomogeneous : $K_q(\vec{x},\vec{y}) = \phi(\vec{x})^T\phi(\vec{y}) = (c+\vec{x}^T\vec{y})^q$, 其中c为非零常数

l2 norm : for $\phi(\vec{x})$ in feature space , $\|\phi(\vec{x})\|_2^2 = \phi(\vec{x})^T\phi(\vec{x}) = K(\vec{x},\vec{x})$
则有 $\|\phi(\vec{x})\| = \sqrt{K(\vec{x},\vec{x})}$
distance : for $\phi(\vec{x}_i)$, $\phi(\vec{x}_j)$ in feature space,
$$\|\phi(\vec{x}_i)-\phi(\vec{x}_j)\|_2^2 = \|\phi(\vec{x}_i)\|_2^2 + \|\phi(\vec{x}_j)\|_2^2 - 2\phi(\vec{x}_i)^T\phi(\vec{x}_j)$$
$$= K(\vec{x}_i,\vec{x}_i) + K(\vec{x}_j,\vec{x}_j) - 2K(\vec{x}_i,\vec{x}_j)$$
则 $\delta(\phi(\vec{x}_i),\phi(\vec{x}_j)) = \|\phi(\vec{x}_i)-\phi(\vec{x}_j)\|_2 = \sqrt{K(\vec{x}_i,\vec{x}_i)+K(\vec{x}_j,\vec{x}_j)-2K(\vec{x}_i,\vec{x}_j)}$
且有 $\frac{1}{2}(\|\phi(\vec{x}_i)\|_2^2 + \|\phi(\vec{x}_j)\|_2^2 - \|\phi(\vec{x}_i)-\phi(\vec{x}_j)\|_2^2) = \phi(\vec{x}_i)\phi(\vec{x}_j) = K(\vec{x}_i,\vec{x}_j)$
mean : $\|\vec{\mu}_\phi\|_2^2 = \vec{\mu}_\phi^T\vec{\mu}_\phi = [\frac{1}{n}\sum_{i=1}^n\phi(\vec{x}_i)]^T[\frac{1}{n}\sum_{i=1}^n\phi(\vec{x}_i)]$
$$= \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n\phi(\vec{x}_i)^T\phi(\vec{x}_j) = \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n K(\vec{x}_i,\vec{x}_j)$$
variance : $\sigma_\phi^2 = \frac{1}{n}\sum_{i=1}^n\|\phi(\vec{x}_i)-\vec{\mu}_\phi\|_2^2 = \frac{1}{n}\sum_{i=1}^n[K(\vec{x}_i,\vec{x}_i) - \frac{2}{n}\sum_{j=1}^n K(\vec{x}_i,\vec{x}_j) + \frac{1}{n^2}\sum_{a=1}^n\sum_{b=1}^n K(\vec{x}_a,\vec{x}_b)]$
$$= \frac{1}{n}\sum_{i=1}^n K(\vec{x}_i,\vec{x}_i) - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n K(\vec{x}_i,\vec{x}_j)$$

Python - P56

@staticmethod , transform method into static method
@staticmethod form is function decorator using @wrapper syntax
to class C:
@staticmethod
def f():
print("call C.f()")
注意 static method not receive implicit first argument
static method can be called either on class or on instance
to C.f() → 'call C().f()'
C().f() → 'call C().f()'
static method in Python similar to Java / C++
possible to call staticmethod() as regular function
needed where need reference to function from class body
want to avoid automatic transformation to instance method
to class C:
builtin_open = staticmethod(open)

sum (iterable, /, start=0), sum start and item of iterable from left to right
iterable's item : normally number
start value not allowed to be string
good alternative to sum() for some use cases
preferred fast way to concatenate sequence of string by calling ''.join(seq)
add floating value with extended precision by calling math.fsum()
concatenate series of iterable by using itertools.chain()

vars ([object]), return __dict__ attribute for module, class, instance
or any other object with __dict__ attribute
module / instance have updateable __dict__ attribute
other object may have write restriction on __dict__ attribute
class use types.MappingProxyType to prevent direct dictionary update
without argument vars() act like locals()
locals dictionary only useful for read since update to locals dictionary ignored