Class 4

Wednesday, 4 February 2015 4:00 AM

– Homomorphism : $A, B \xrightarrow{+} A+B$

$$\begin{array}{ccc} A, B & \xrightarrow{+} & A+B \\ g \downarrow & & \downarrow \\ g(A), g(B) & \xrightarrow{+} & g(A+B) \end{array}$$

→ we can study relationships between different spaces of dimensions m and n

if $g(A+B) = g(A) + g(B)$ then g is a homomophism

example: g = boundary operator (•—• + —— + —— = ——————)

⊛ homology groups: example sphere and torus

+ simplex : 0-simplex ·
  1-simplex ——
  2-simplex (interior)
  3-simplex

face of a simplex : remove one vertex
example: $(v_0, \ldots v_n)$ has n faces
$(v_0, \ldots \hat{v}_k, \ldots v_n)$ ← removed
remove vertex again
$(v_0, \ldots \hat{v}_i, \ldots \hat{v}_j, \ldots v_n)$

+ complex: $\sigma \in C$ then faces$(\sigma) \in C$
$\sigma_1 \cap \sigma_2 = \text{Faces}(\sigma_1) \cap \text{Faces}(\sigma_2)$ (so is not a complex, it must be)

+ p-chains: $C = a_1\sigma_1^p + a_2\sigma_2^p + a_3\sigma_3^{(p)} + \ldots$ → dimension
a_i: either 0 or 1; σ: a complex

a p-chain is normally not a complex since the simplexes must have the same dimension

sum of chains is the sum of indices
$C_1 + C_2 = \sum (a_i + b_i)\sigma_i$
↓ module 0
$0+1 = 1$
$1+0 = 1$
$0+0 = 0$
$1+1 = 0$

this is 1
this is 0

a complex can be broken into groups of chains: C_{p+1} C_p C_{p-1}

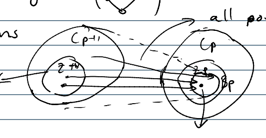
+ boundary operator : simplex σ. boundary of σ =
$\partial\sigma^4 = \sum \sigma^{p-1}$ or $\partial \triangle$ = set of m-1 faces
tetrahedra → 4 triangles; simplex, m

$\partial C = \sum a_i \partial \sigma_i$ (cycle)

example ◁▷ → boundary = ⟨ ⟩ (4 edges)

+ cycle ; o—o—o—o = chain
◇ = cycle

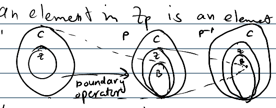
boundary of (◇) = 0 (since each vertices are counted twice)



(C_{p+1} = set of chains of dimension p+1)
Z_{p+1} = set of all cycles of dimension p+1

→ all possible chains map down to B_p through the boundary operator

an element in Z_p is an element in B_p + something



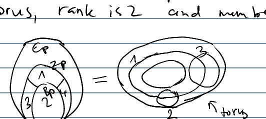
depending on where a cycle lives, it can be a boundary or not

8 → boundary

homology $H_p = Z_p / B_p$
any cycle can be write as $C = \alpha_1 c_1 + \ldots + \alpha_k c_k + B$ (distinct cycles)
if we have k of the distinct cycles C the Betti number is k

Rank (H_p) = Betti number

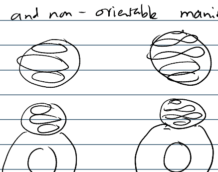
⊛ Example : on a torus, rank is 2 and number of distinct cycles is 4



B_0 = number of connected components
B_1 = number of cycles

B_0
B_1 0 2 4
B_2
B_3 ↘ number of bits to count the number of distinct cycles

⊛ Orientable and non-orientable manifold



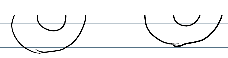
→ projected plane

→ klein bottle, $\text{betti}_1 = 2$
same as the orientable 2-manifold

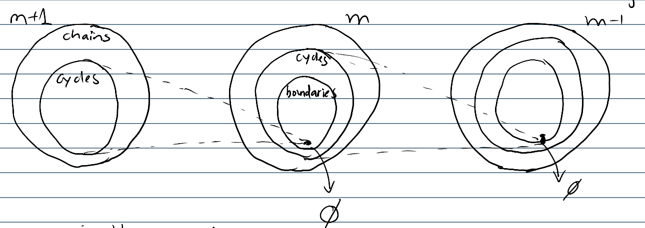
$\partial\partial S = \emptyset$ (boundary of boundary)

m+1 chains m m-1

same as the orientable 2-manifold



$\partial\partial S = \emptyset$ (boundary of boundary)



boundaries are cycles

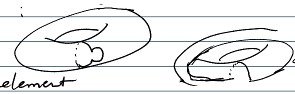
are all cycles boundaries?

depends on the space

this cycle does not bound anything on the surface of the torus
(if you cut a legitle cycle, you separate the space)

m-1 cycles is the kernel of the ∂ operator
(eg: the kernel of 3D projection is a set of lines that map to the same point)

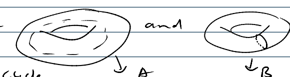
take a cycle and add a boundary to it:
is in the same class



this is a cycle

so the boundaries are the same as the $\emptyset$ element

the intersection of

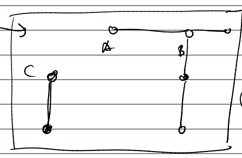


and ... is another class of cycle

so you can have a cycle
by adding $aA + bB + c\partial$
1/0 1/0 → Betty1 = 2

when m = 0 this counts connected components

our space

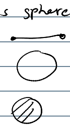


B can be obtained from A by adding AB
but C cannot be obtained from A by adding a boundary
because there is no chain for which A and C are the boundary
(x) adding boundaries really mean stretching and shrinking

B0 = # connected components
B1 = # of 1-cycles
B2 = # of 2-cycles

this is counting basis spheres

0 - sphere (2 points)
1 - sphere
2 - sphere



(x) Computing Betty numbers (we always need 2 spaces)
to form a graph (vertex + edge)

naive algorithm needs $O(n^2)$ time
we can also do it in $O(n)$ time if we fix the graph

Union - find

CREATE(V) → add vertices
UNION (V_1, V_2) → add edges
FIND(V) → find the set of V