Drosophila hits Machine Learning - A new algorithm for similarity search derived from the olfactory processing of fruit flies

Meetup: Berlin Machine Learning Group
03-12-2018

Dr. Daniela Schmidt

The Fruit Fly Brain Improves Our Search Algorithms

Written by Mike James
Wednesday, 15 November 2017

FRUIT FLY BRAINS INFORM SEARCH ENGINES OF THE FUTURE

November 9, 2017

The way fruit flies identify similarities between odors offers a new approach for search algorithms

REPORT

A neural algorithm for a fundamental computing problem

Sanjoy Dasgupta[1], Charles F. Stevens[2,3], Saket Navlakha[4,*]
+ See all authors and affiliations

Science 10 Nov 2017:
Vol. 358, Issue 6364, pp. 793-796
DOI: 10.1126/science.aam9868

Lessons From the Fly Brain Improve Search Algorithms (3 of 3) (IMAGE)
AMERICAN ASSOCIATION FOR THE ADVANCEMENT OF SCIENCE

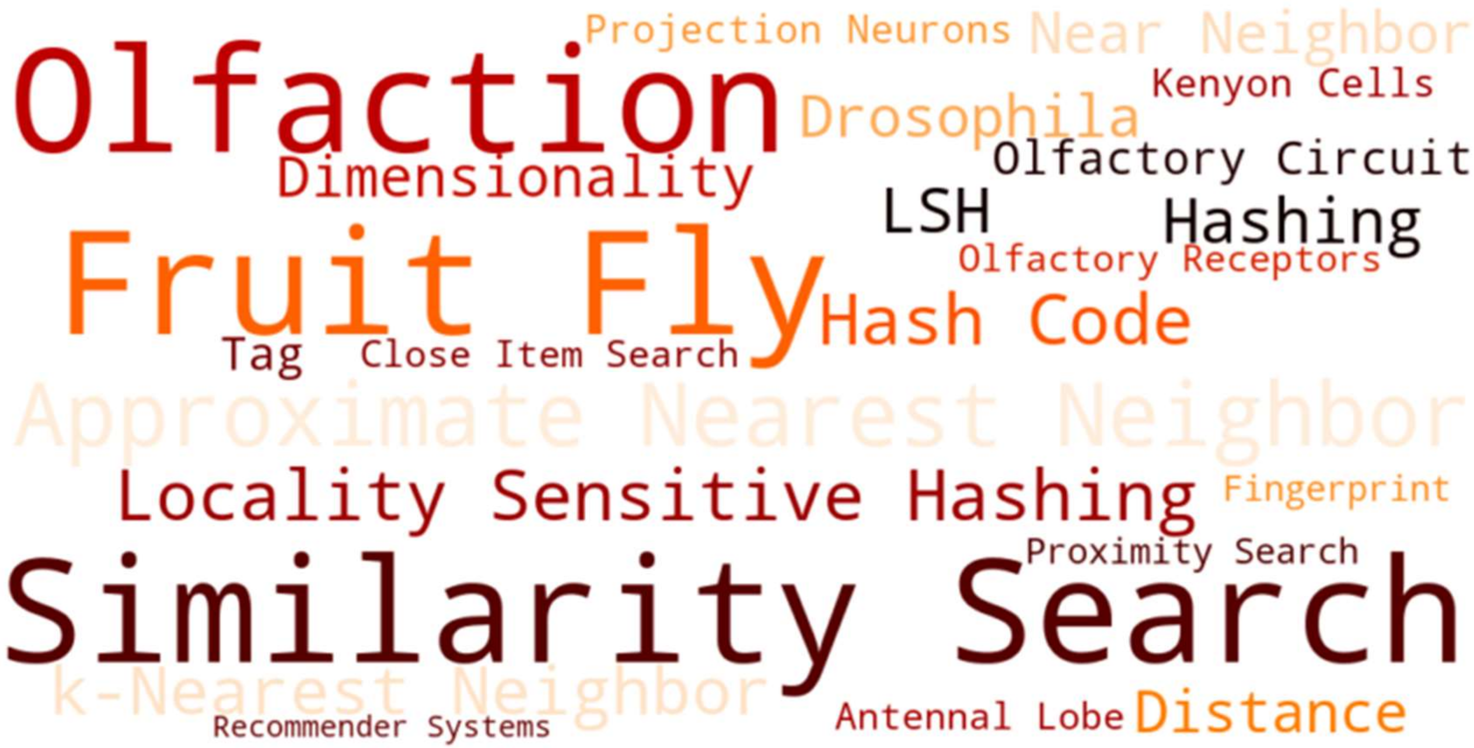
Fly brain inspires computing algorithm

Flies use an algorithmic neuronal strategy to sense and categorize odors. Dasgupta *et al.* applied insights from the fly system to come up with a solution to a computer science problem. On the basis of the algorithm that flies use to tag an odor and categorize similar ones, the authors generated a new solution to the nearest-neighbor search problem that underlies tasks such as searching for similar images on the web.

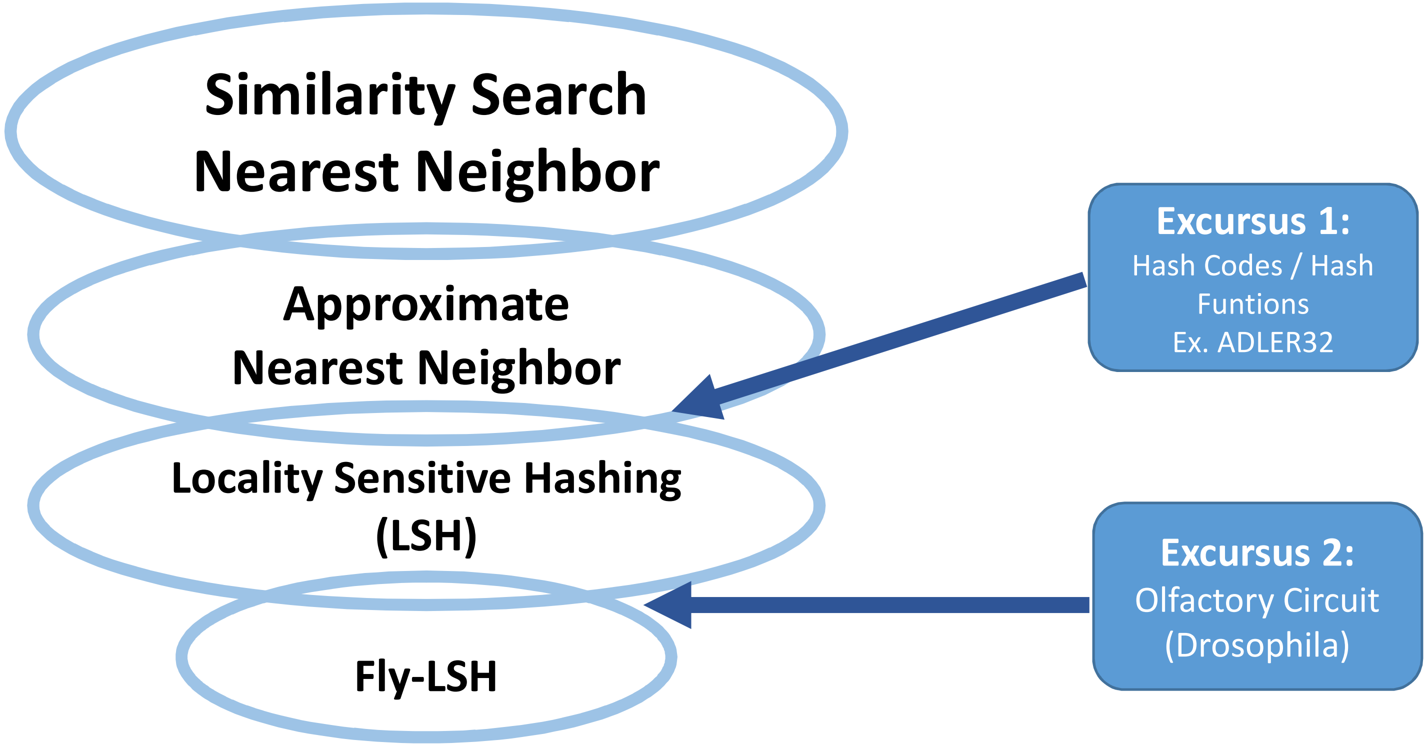
COMPUTING 60-SECOND SCIENCE

Insect Brain System Knows What You Want

By Christopher Intagliata on November 10, 2017

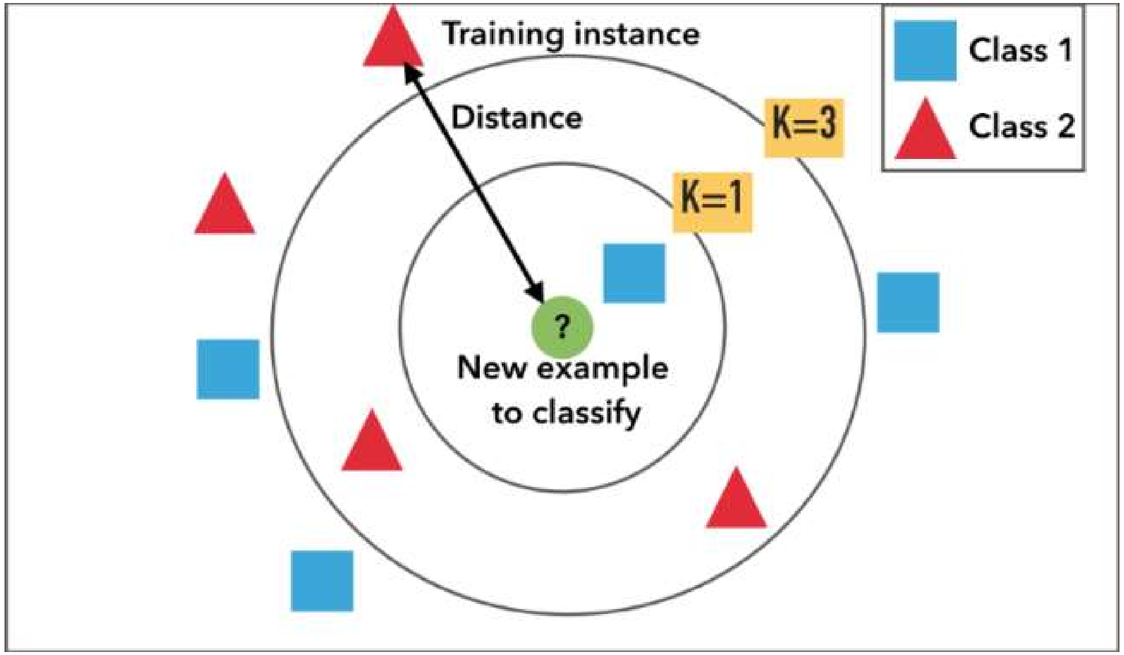

Projection Neurons
Near Neighbor
Olfaction
Kenyon Cells
Drosophila
Olfactory Circuit
Dimensionality
LSH
Hashing
Fruit Fly
Olfactory Receptors
Hash Code
Tag
Close Item Search
Approximate Nearest Neighbor
Locality Sensitive Hashing
Fingerprint
Proximity Search
Similarity Search
k-Nearest Neighbor
Recommender Systems
Antennal Lobe
Distance

OVERVIEW



Similarity Search
Nearest Neighbor
Approximate
Nearest Neighbor
Locality Sensitive Hashing
(LSH)
Fly-LSH
Excursus 1:
Hash Codes / Hash
Funtions
Ex. ADLER32
Excursus 2:
Olfactory Circuit
(Drosophila)

SIMILARITY SEARCH / *k*-NEAREST NEIGHBOR



Applications:

- Classification
- Recommendation
- Near-Duplicates
- …

Source: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

SIMILARITY SEARCH / *k*-NEAREST NEIGHBOR

Advantages

- No assumptions about data
- Simple
- Quite high accuracy

Drawbacks

- Computationally expensive
- All training data need to be saved
  → high memory requirements
- Bad scalability
- Sensitive to irrelevant features and scaling of data

SIMILARITY SEARCH / *k*-NEAREST NEIGHBOR

Advantages

- No assumptions about data
- Simple
- Quite high accuracy

Drawbacks

- Computationally expensive
- All training data need to be saved
  → high memory requirements
- Bad scalability
- Sensitive to irrelevant features and scaling of data

Approximate Nearest Neighbor

APPROXIMATE NEAREST NEIGHBOR

- **Idea:** Estimation of Nearest Neighbors is sufficient

  → no guarantee to identify real nearest neighbor
  → faster and less memory expensive

- **Hashing:** most popular solution for Approximate Nearest Neighbor

Excursus 1

HASH CODE / HASH FUNCTIONS

WHAT IS A HASH?

- **Synonyms:** fingerprint, checksum, signature, tag
- **Hashing:**
  - Mapping of data into a low-dimensional representation
  - short code = bit sequence (e.g. 00111001)
- **Applications:**
  - Data bases
  - Checksum for data integrity
  - Cryptography

HASH FUNCTION ADLER32

- Aim: 32-bit code for document/text

- Example: Drosophila hits Machine Learning - A new algorithm for similarity search derived from the olfactory processing of fruit flies

		D	r	o	s	o	p		s
ASCII-Code		**68**	**114**	**111**	**115**	**111**	**112**		**115**
CumSum A	1	69	183	294	409	520	632	…	11‘888
CumSum B	0	69	252	546	955	1475	2107		740‘229

HASH FUNCTION ADLER 32

- Division of each CumSum by **highest 16-bit prime number** (i.e. 65‘521)
  → **modulo** as binary number

	final sum modulo 65‘521	binary number (16-bit)
CumSum A	11‘888 % 65‘521 = 11‘888	0010 1110 0111 0000
CumSum B	740‘229 % 65‘521 = 19‘498	0100 1100 0010 1010

Combination of both 16-bit codes to 32-bit code:
→ Adler32 hash code: 0100 1100 0010 1010 0010 1110 0111 0000
(as hexadecimal number: 4c2a2e70)

HASH FUNCTION ADLER 32

- Drosophila hits Machine Learning - A new algorithm for similarity search derived from the olfactory processing of fruit flies

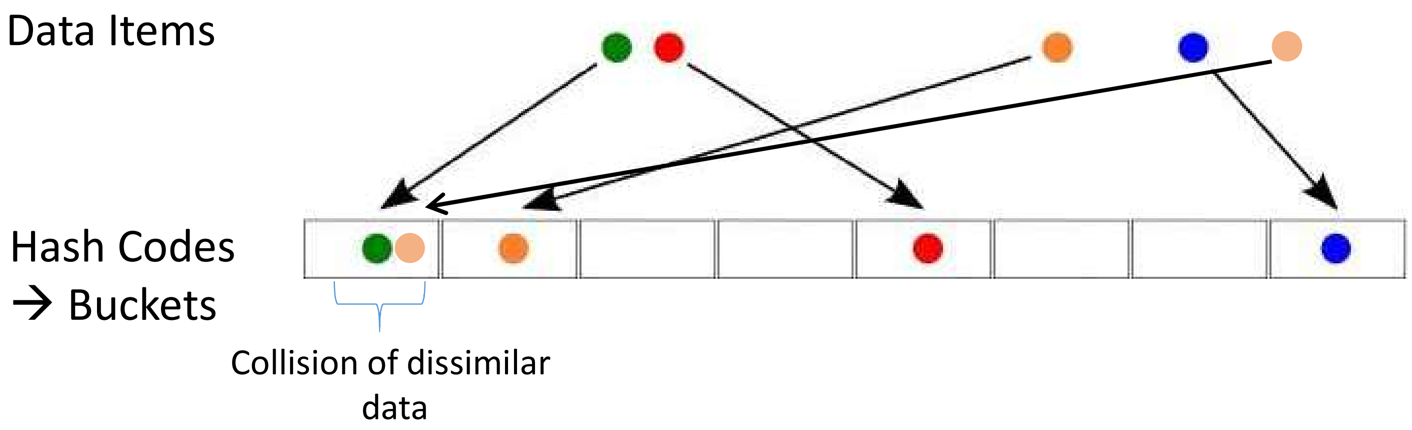
  0100 1100 0010 1010 0010 1110 0111 0000 (4c2a2e70)

- Drosophila hits **m**achine **l**earning - A new algorithm for similarity search derived from the olfactory processing of fruit flies

  0110 0110 0110 1010 0010 1110 1011 0000 (666a2eb0)

Small differences lead to a completly different hash!!!

GENERAL HASHING

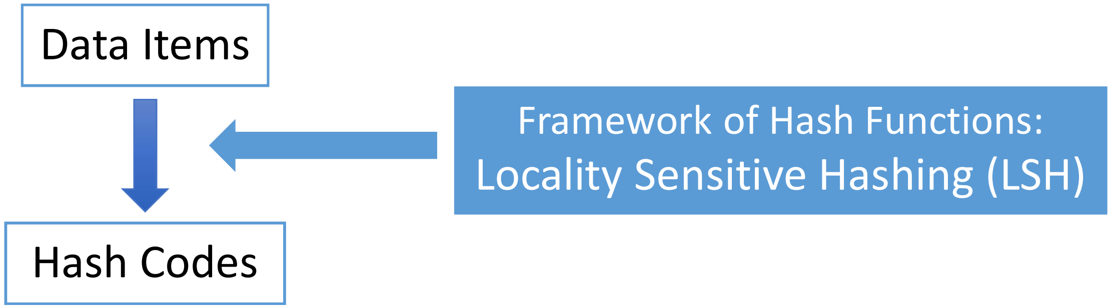


Source: https://0110.be/Software

BACK TO

APPROXIMATE NEAREST NEIGHBOR

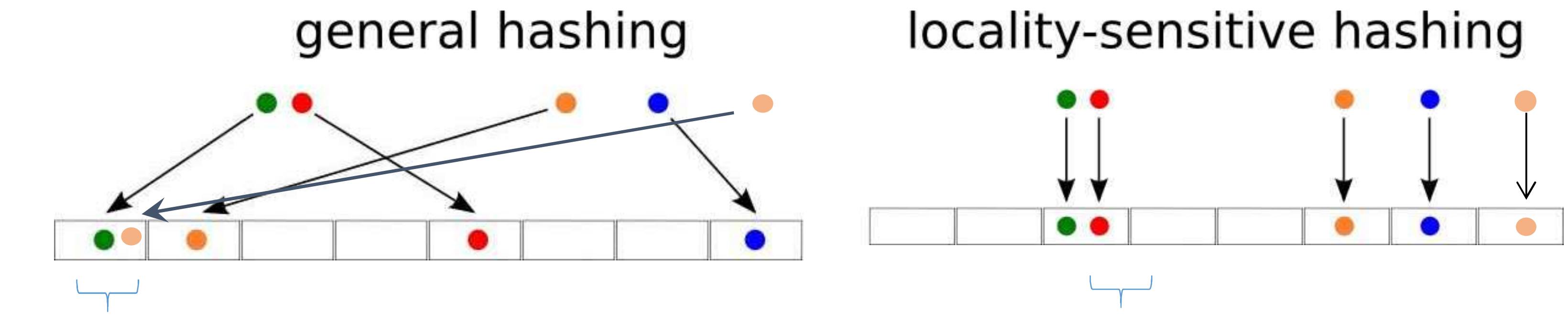
HASHING IN APPROXIMATE NEAREST NEIGHBOR



LOCALITY SENSITIVE HASHING

Hash functions fulfilling the “**locality sensitive property**”:

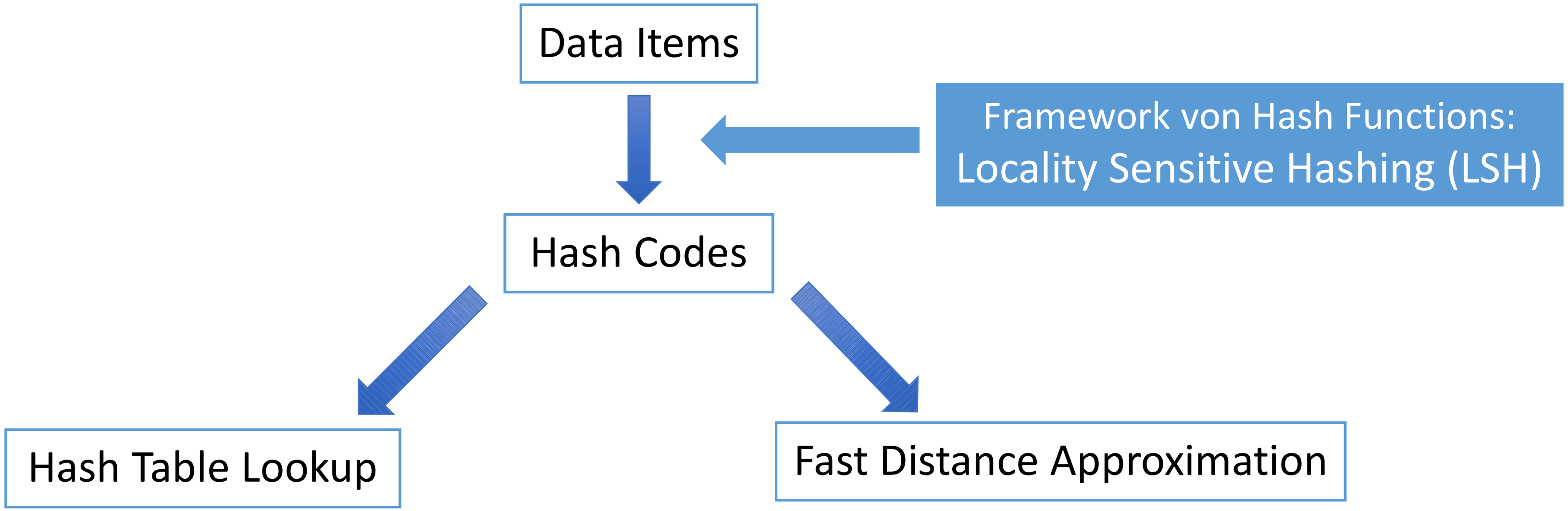
Higher probability that similar items are mapped to the same hash code than dissimilar items.



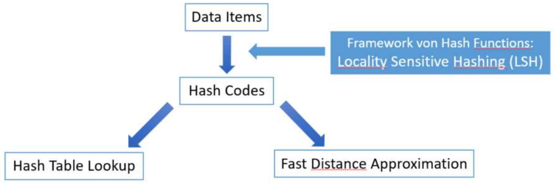
Collision of
von **dissimilar**
data items

collision
of **similar**
data items

HASHING IN APPROXIMATE NEAREST NEIGHBOR

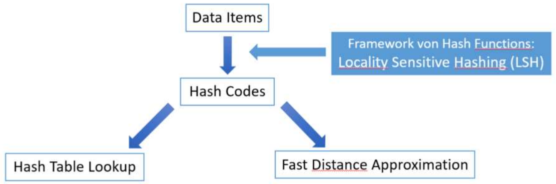


LSH – HASH TABLE LOOKUP



1. Training data: Generation of hash table with LSH
   →Similar items have same hash code
2. Calculate hashcode for query item
3. Extract near neighbors of query item (same hash code)
4. Calculate **exact distance** (query item – near neighbors)

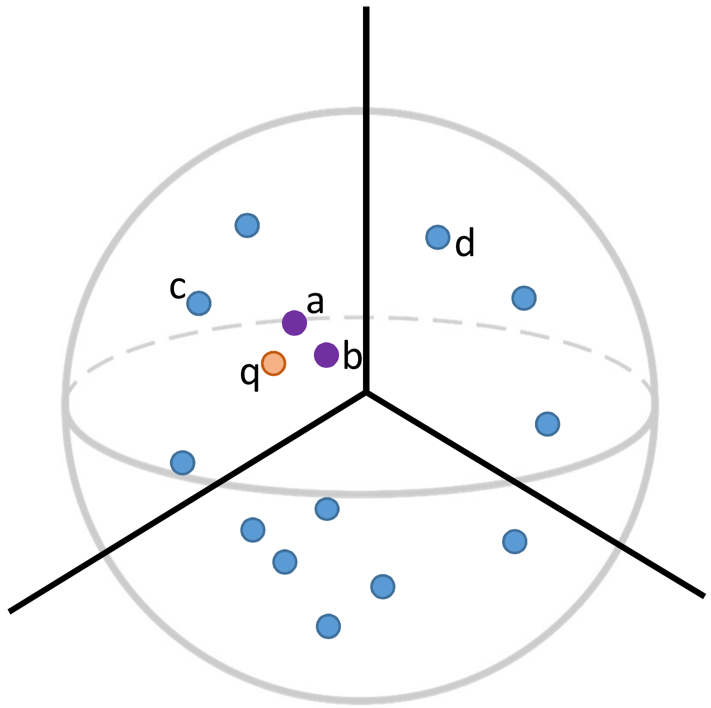
LSH – FAST DISTANCE APPROXIMATION



1. Training data and query item: generate hash code with LSH
   → Similar data items have similar hash code
2. **Hash code distances** between query item and training data

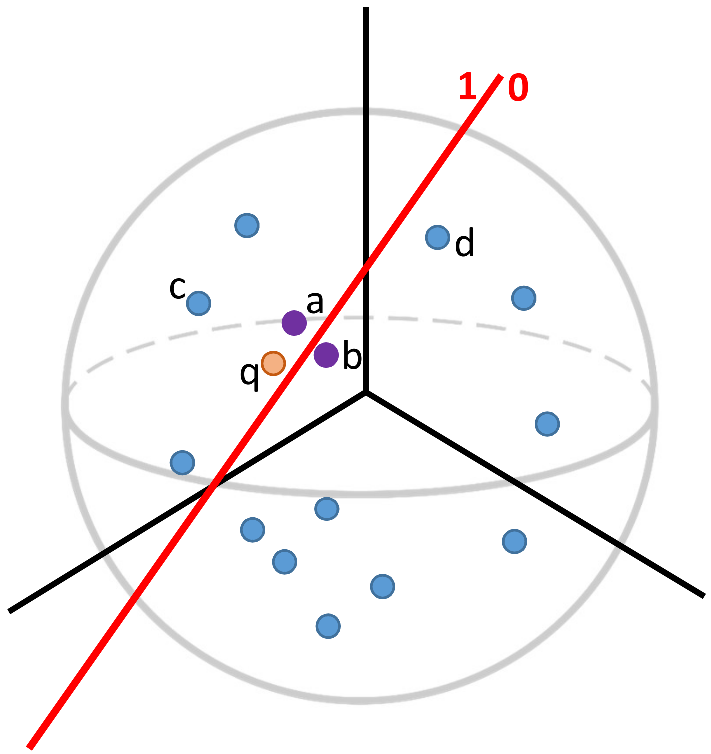
TRADITIONAL LSH

- Unit Sphere



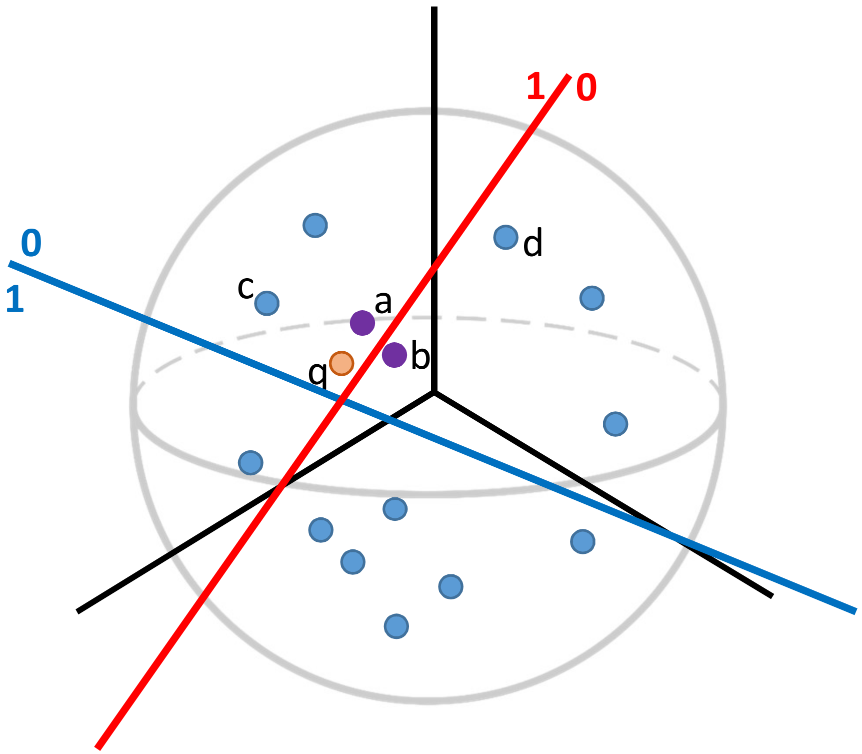
TRADITIONAL LSH

- Unit Sphere
- Generate random hyperplanes
  h1, h2, h3



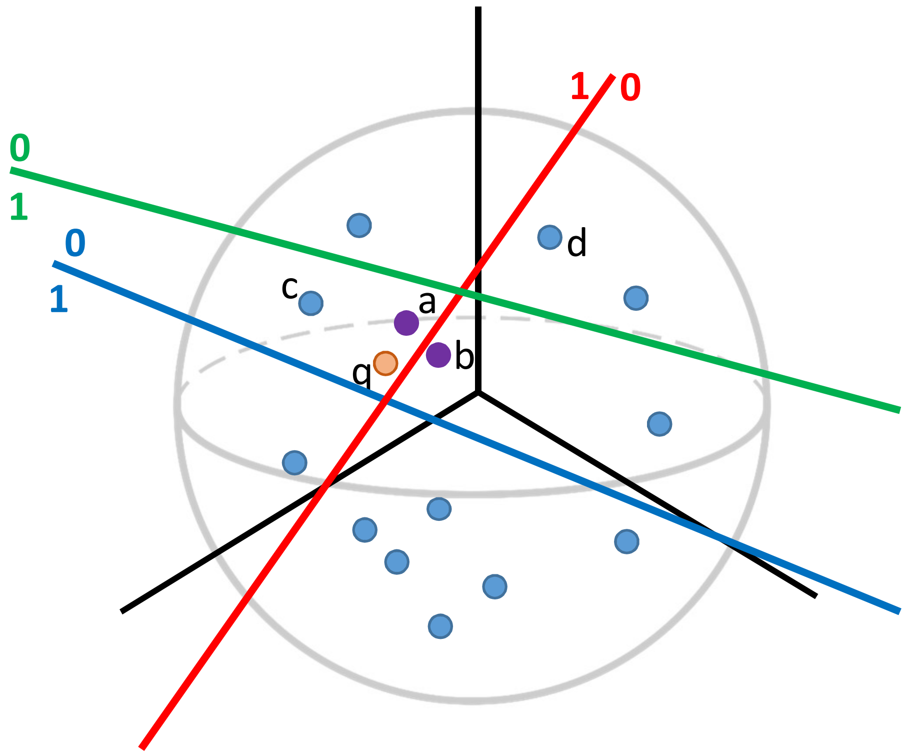
TRADITIONAL LSH

- Unit Sphere
- Generate random hyperplanes **h1**, **h2**, **h3**



TRADITIONAL LSH

- Unit Sphere
- Generate random hyperplanes **h1**, **h2**, **h3**



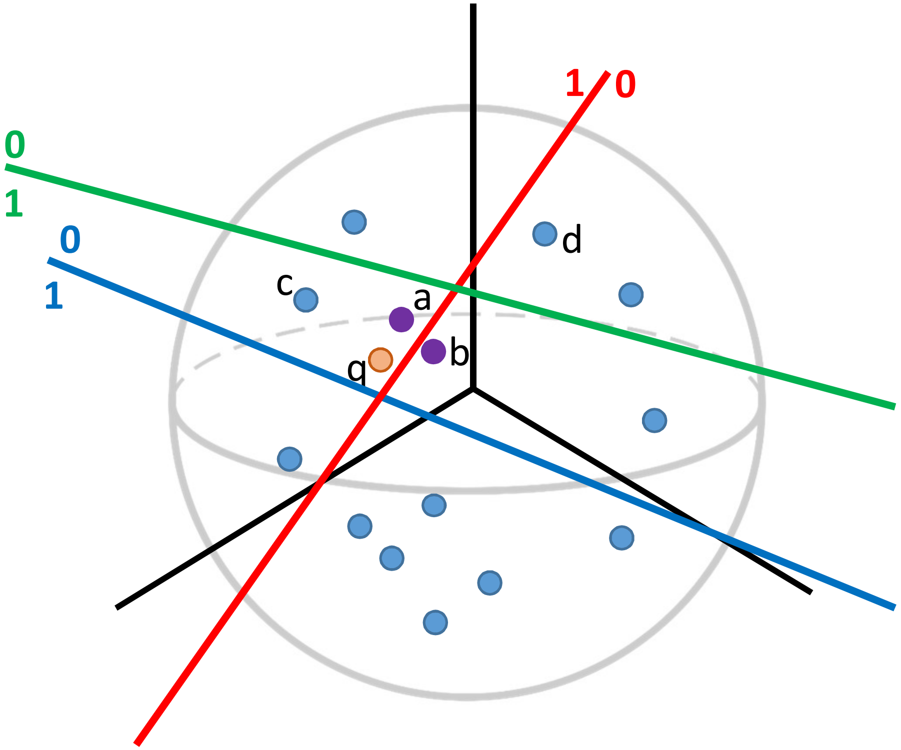
TRADITIONAL LSH

- Unit Sphere
- Generate random hyperplanes **h1, h2, h3**

Hash Table

Hash (Bucket)	Data Items
...	
101	q, a, c
000	d
100	
001	b
011	
...	

Near neighbors of q



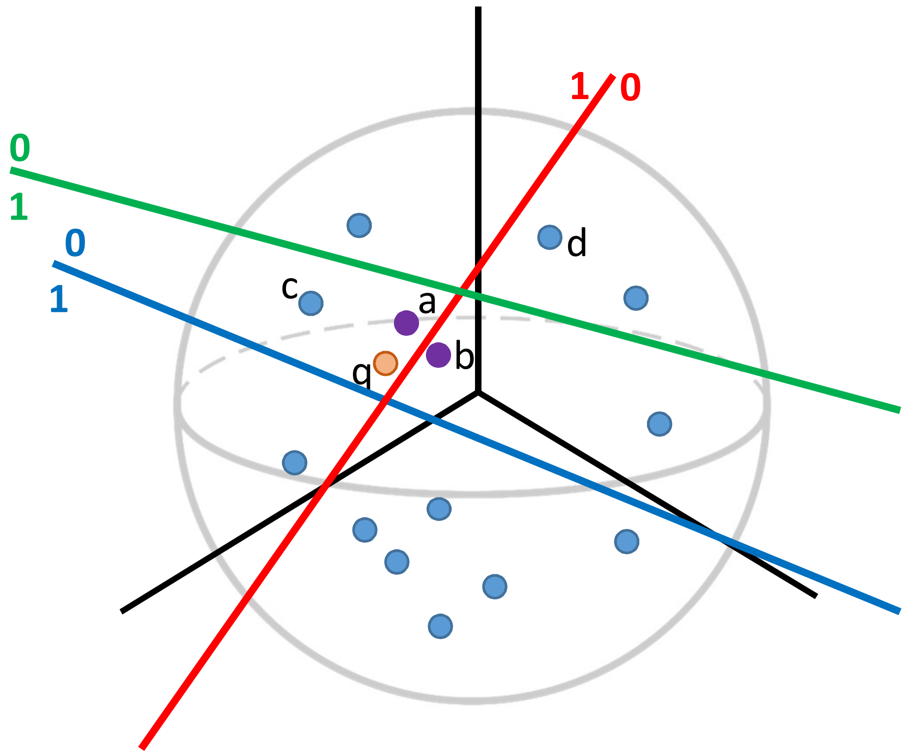
TRADITIONAL LSH

- Unit Sphere
- Generate random hyperplanes
  h1, h2, h3

Hash Table

Hash (Bucket)	Data Items
. . .	
101	q, a, c
000	d
100	
001	b
011	
. . .	

Near neighbors of q



Several hash tables with other random hyperplanes

RANDOM GAUSSIAN PROJECTION

- s = Input vector (feature-vector, zero mean) with d dimensions
- R = projection matrix
  - Number of columns: length of input vector (input dimension d)
  - Number of rows: number of hyperplanes (m = hash length k)
- r = Output vector with m dimensions (m = hash length k)

$$r = s \times R$$

- r → Hash f(x) = 1 if x > 0 else 0

RANDOM GAUSSIAN PROJECTION - EXAMPLE

s (d= 7)	x	R (d x m)							=	r (m = 3)	> 0 → 1	Hash (k=3)
-0.24												
-0.14	h1	0.756	0.945	0.675	0.488	0.105	0.694	0.326		-0.087		0
-0.04	h2	0.063	0.797	0.562	0.961	0.753	0.181	0.913		0.230		1
0.56	h3	0.778	0.734	0.912	0.303	0.595	0.749	0.547		-0.225		0
-0.04												
0.06												
-0.14												

LSH - FRAMEWORK

- For different distance and similarity measures
  - z.B. Jaccard coefficient (→ Min Hash), Hamming distance, angle based distance, $\chi 2-$distance, …
- Variants: learning LSH from data

- Overview: Hashing for Similarity Search (Wang et al. 2014): https://arxiv.org/pdf/1408.2927.pdf
- Introduction to LSH:

  https://www.youtube.com/watch?v=356GoYkmYKg
  (sequence of 12 movies made by Victor Lavrenko)

SIMILARITY SEARCH IN BIOLOGY

Our brain is confronted with similarity search all the time!

X looks like Y

X sounds like Y

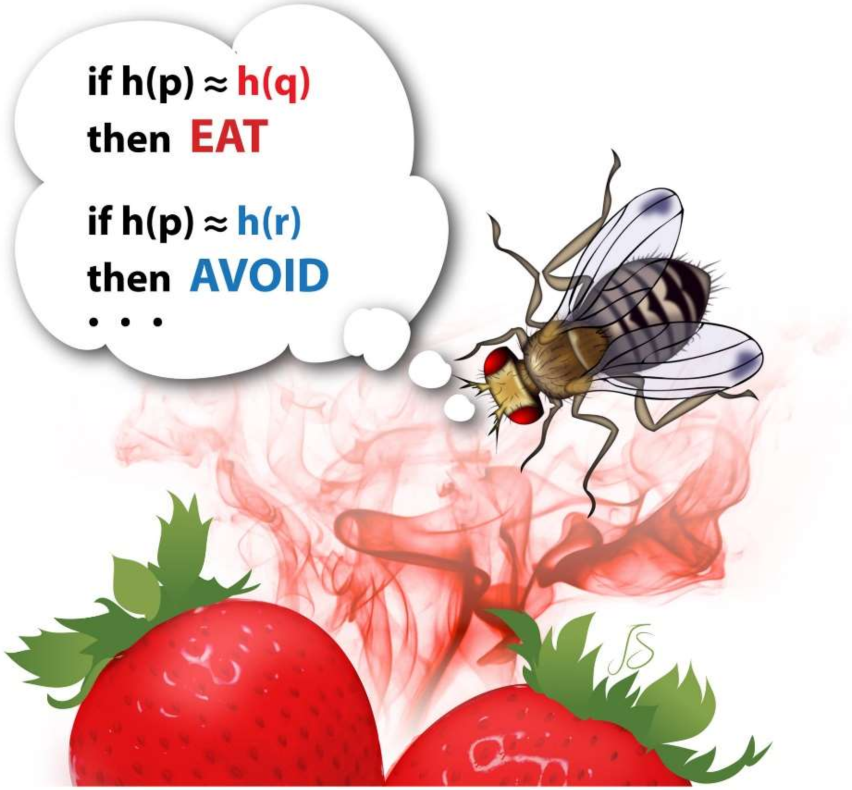
X tastes like Y

X feels like Y

X **smells** like Y

How does the brain do similarity search?

SIMILARITY SEARCH – OLFACTORY PERCEPTION



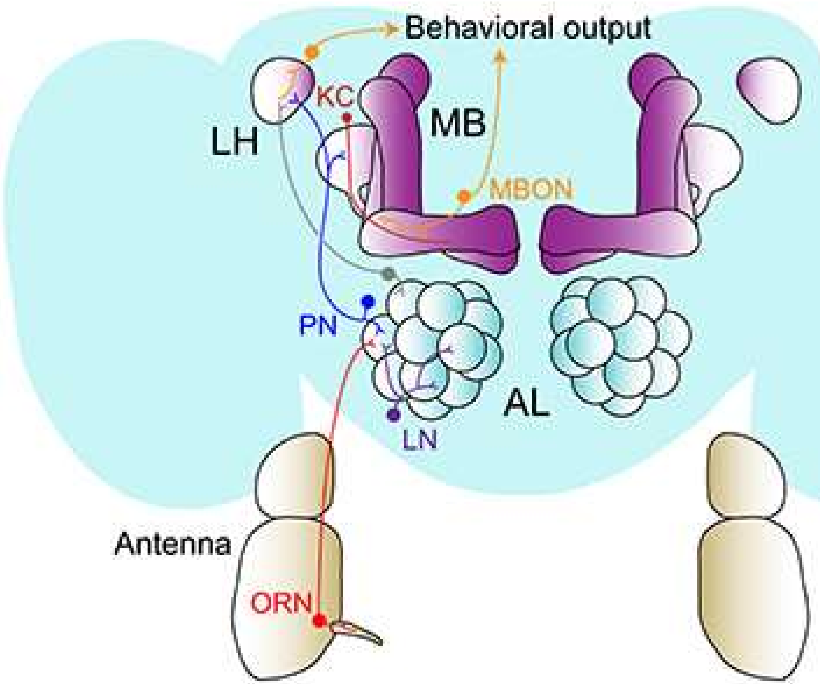
Source: https://3c1703fe8d.site.internapcdn.net/newman/gfx/news/hires/2017/1-fruitflybrai.jpg

Excursus 2

OLFACTORY CIRCUIT

DROSOPHILA

INSECT OLFACTORY SYSTEM



ORNs Olfactory Receptor Neurons

AL Antennal Lobe
Glomeruli

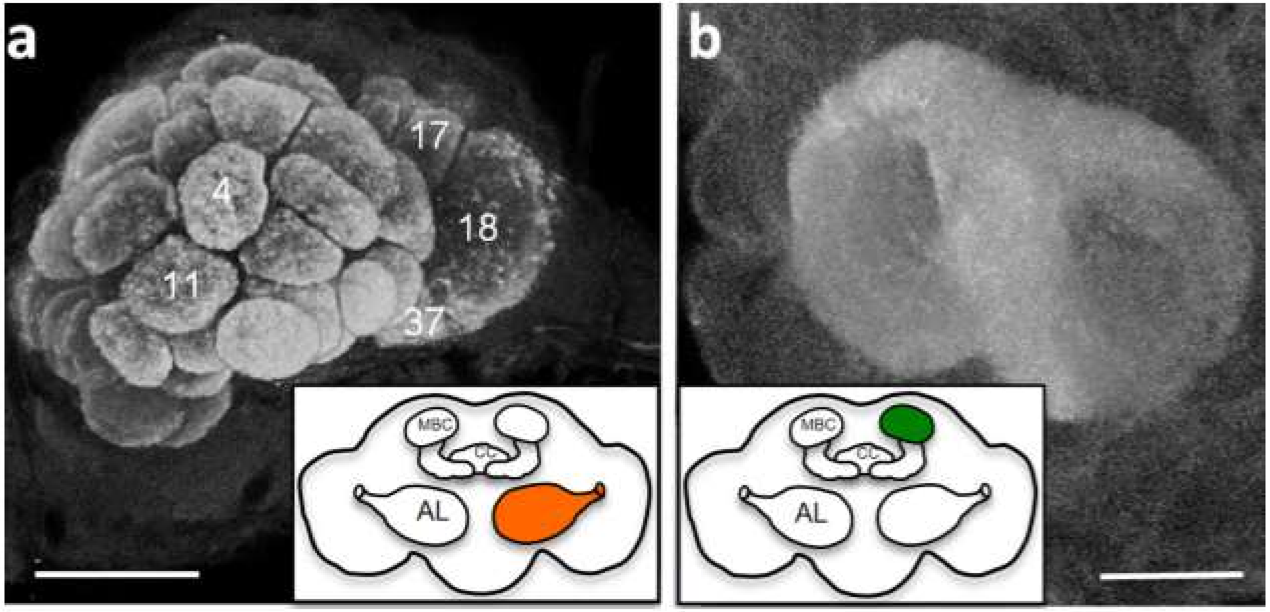
PNs Projection Neurons

MB Mushroom Body
Microglomeruli

KCs Kenyon Cells

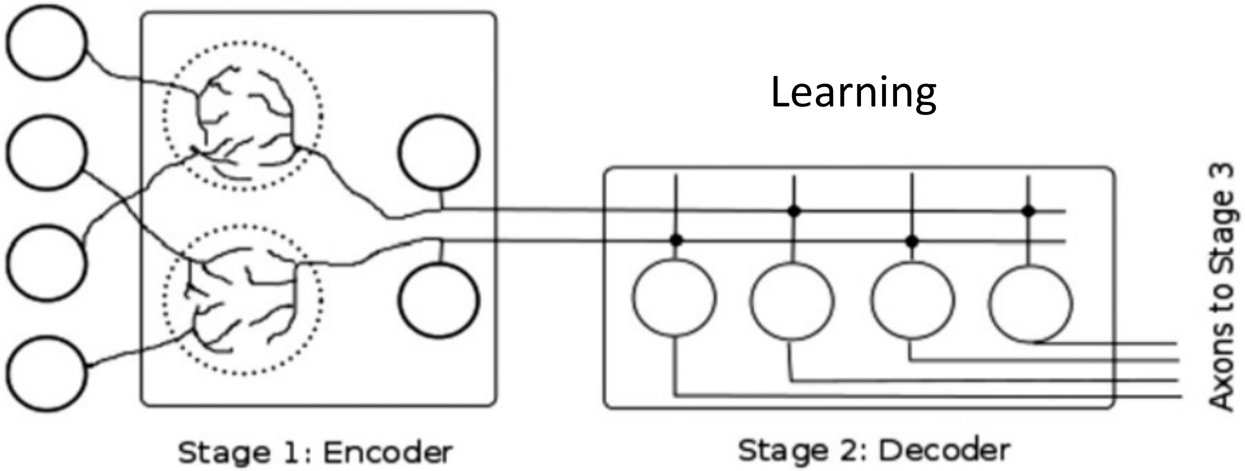
Source: https://www.frontiersin.org/articles/10.3389/fncel.2018.00011/full

ANTENNAL LOBE (WITH GLOMERULI) AND MUSHROOM BODY



Source: https://www.researchgate.net/publication/282876817_Brief_sensory_experience_differentially_affects_the_volume_of_olfactory_brain_centres_in_a_moth

MARR MOTIV



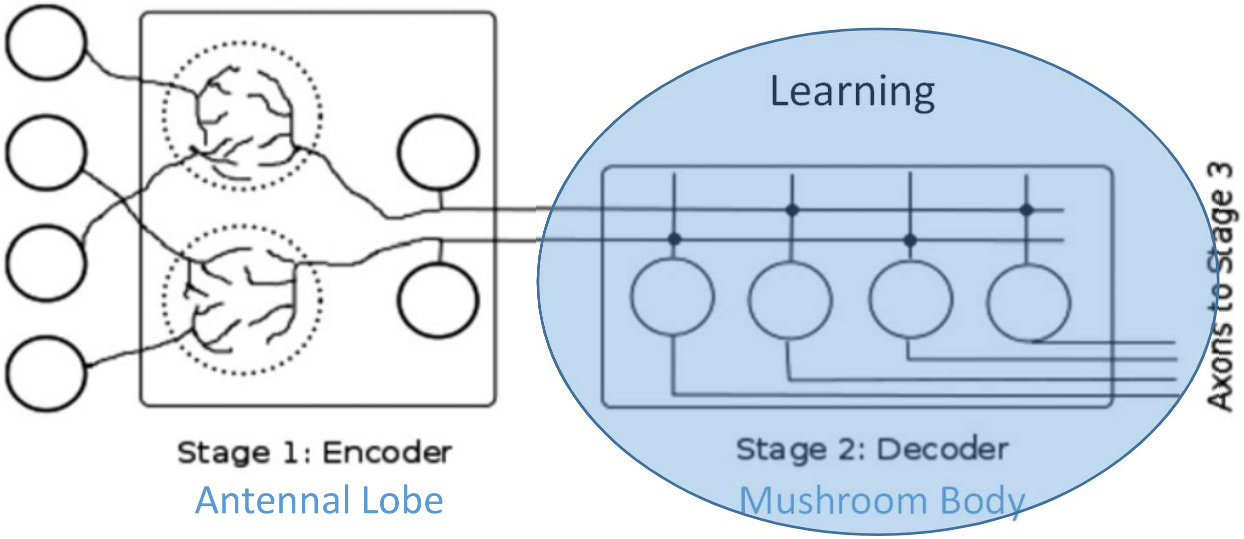
Antennal Lobe

Mushroom Body

1‘500 ORNs → **50 Glomeruli** → **50 PNs** → **2‘000 Microglomeruli** → **2‘000 Kenyon Cells** →
50 Olfactory receptors

Source: http://www.pnas.org/content/112/30/9460

MARR MOTIV

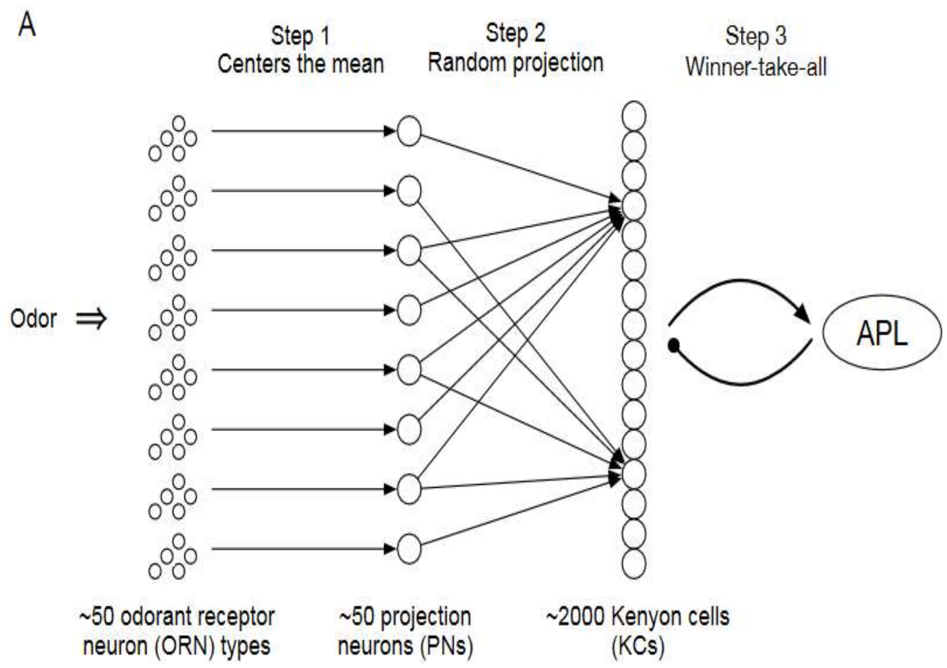


1‘500 ORNs → **50 Glomeruli** → **50 PNs** → **2‘000 Microglomeruli** → **2‘000 Kenyon Cells** →
50 Olfactory receptors

Source: http://www.pnas.org/content/112/30/9460

SIGNAL DECODING

- Each PN transmits signal to about 10 KCs
  1 PN → 10 KCs
- Each KC receives and sums up signal of 6 PNs
  6 PN ← 1 KCs



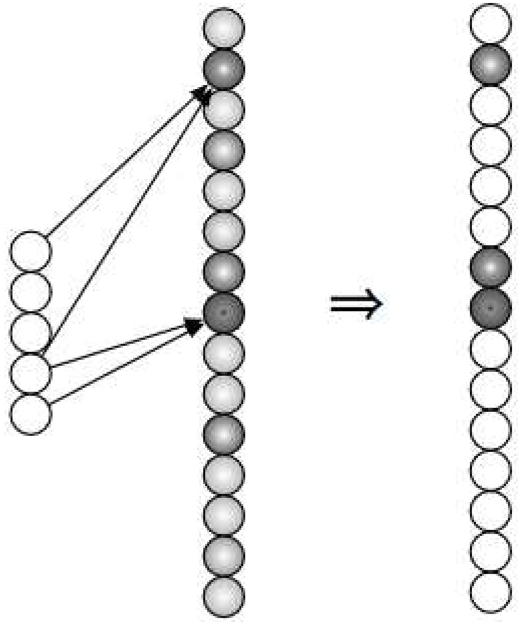
Source: https://www.biorxiv.org/content/biorxiv/early/2017/08/25/180471.full.pdf?%3Fcollection=

WTA – WINNER-TAKE-ALL (SPARSIFICATION)

- 1 inhibitory neuron inhibits all Kenyon cells

  → Tag: only 5% of KCs with highest firing rates transmit signal further

ARGSORT(5%)



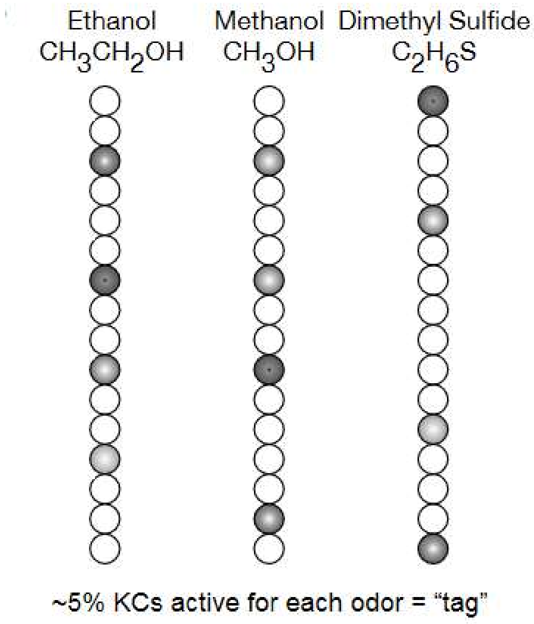
Source: https://www.biorxiv.org/content/biorxiv/early/2017/08/25/180471.full.pdf?%3Fcollection=

WTA – WINNER-TAKE-ALL (SPARSIFICATION)

- 1 inhibitory neuron inhibits all Kenyon cells

  → Tag: only 5% of KCs with highest firing rates transmit signal further

ARGSORT(5%)



Source: https://www.biorxiv.org/content/biorxiv/early/2017/08/25/180471.full.pdf?%3Fcollection=

SPARSE RANDOM PROJECTION

s: Input vector with d dimensions (50 PNs)

R: Projection matrix with d columns und m rows (2000 KCs)
(random matrix, sensing matrix)

r: output vector with m dimensions

$$r = s \times R$$

r → Hash $f(x) = 1$ if $x \in$ top 5% of $\{xi\}$ else 0
= binary vector (length m mit k non-zeros)

SPARSE RANDOM PROJECTION - EXAMPLE

s (d= 7) x **R (2 PNs per KC)** = **r (m = 14)** $\xrightarrow{\text{WTA Top 3}}$ **Hash (k=3)**

s (d= 7)
-0.24
-0.14
-0.04
0.56
-0.04
0.06
-0.14

R (2 PNs per KC)								r (m = 14)	Hash (k=3)
0	1	0	0	0	1	0		-0.086	0
1	0	0	1	0	0	0		0.314	0
0	1	0	1	0	0	0		0.414	1
0	0	1	0	0	1	0		0.014	0
0	0	0	0	1	0	1		-0.186	0
0	1	0	0	1	0	0		-0.186	0
0	0	1	0	0	0	1		-0.186	0
1	0	0	0	0	0	1		-0.386	0
0	0	0	0	1	1	0		0.014	0
0	0	1	1	0	0	0		0.514	1
0	1	0	0	0	1	0		-0.086	0
0	0	1	0	1	0	0		-0.086	0
0	0	0	1	0	1	0		0.614	1
0	0	1	0	0	0	1		-0.186	0

RANDOM GAUSSIAN PROJECTION - EXAMPLE

s (d= 7) x R (d x m) = r (m = 3) $\xrightarrow{> 0 \rightarrow 1}$ Hash (k=3)

s (d= 7)
-0.24
-0.14
-0.04
0.56
-0.04
0.06
-0.14

R (d x m)								r (m = 3)	Hash (k=3)
h1	0.756	0.945	0.675	0.488	0.105	0.694	0.326	-0.087	0
h2	0.063	0.797	0.562	0.961	0.753	0.181	0.913	0.230	1
h3	0.778	0.734	0.912	0.303	0.595	0.749	0.547	-0.225	0

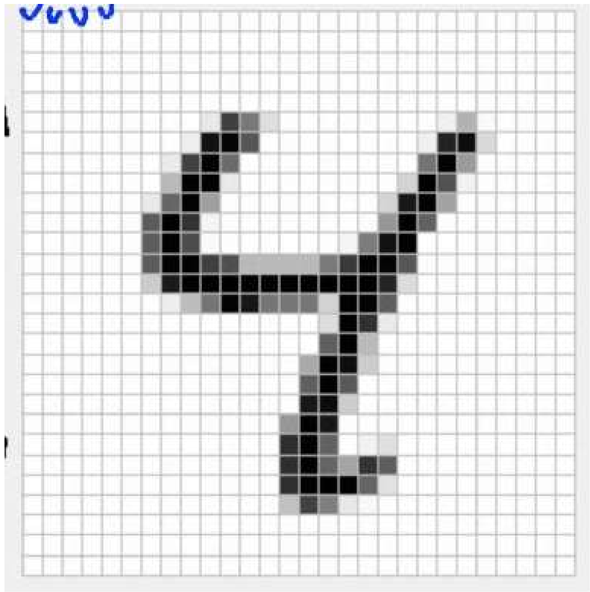
FLY-LSH

- „Sparse connected network“
- Random **sparse** projection
- Enhanced dimensionality after projection: **d << m**
- Activation function:
  ARGSORT() (WTA)
  $f(x) = 1$ if $x \in$ top 5% of $\{x_i\}$ else 0

TRADITIONAL LSH

- „Dense connected network“
- Random **gaussian** projection
- Reduced dimensionality after projection: **d >> m**
- Activation function:
  Step function
  $f(x) = 1$ if $x>0$ else 0

COMPARISON TRADITIONAL LSH – FLY-LSH



Datasets:

MNIST: handwritten digits (d=784)

SIFT (d = 128)

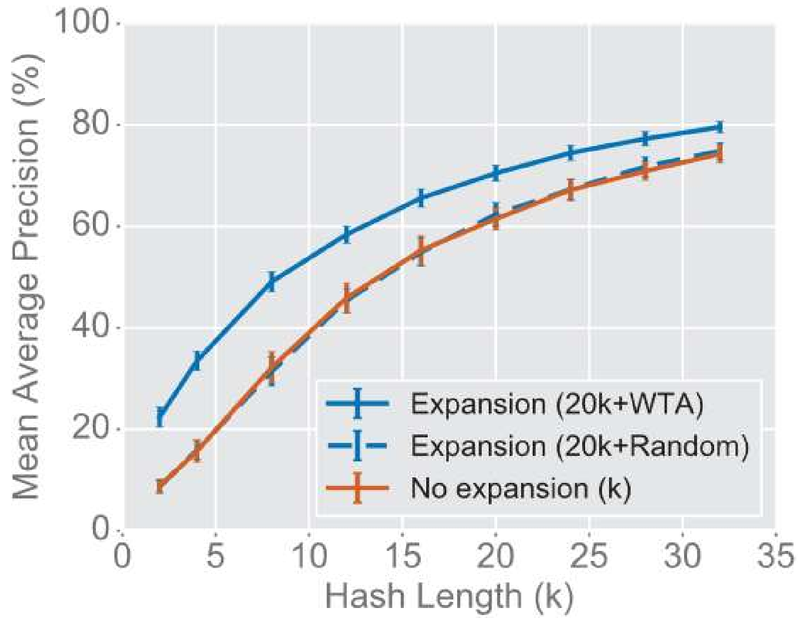
GLOVE (d = 300)

COMPARISON TRADITIONAL LSH – FLY-LSH

- Same computational costs (same number of mathematical operations)
  - Fly-LSH: 20x faster

- Better performance by WTA
  → WTA: best preserves relative distances between inputs

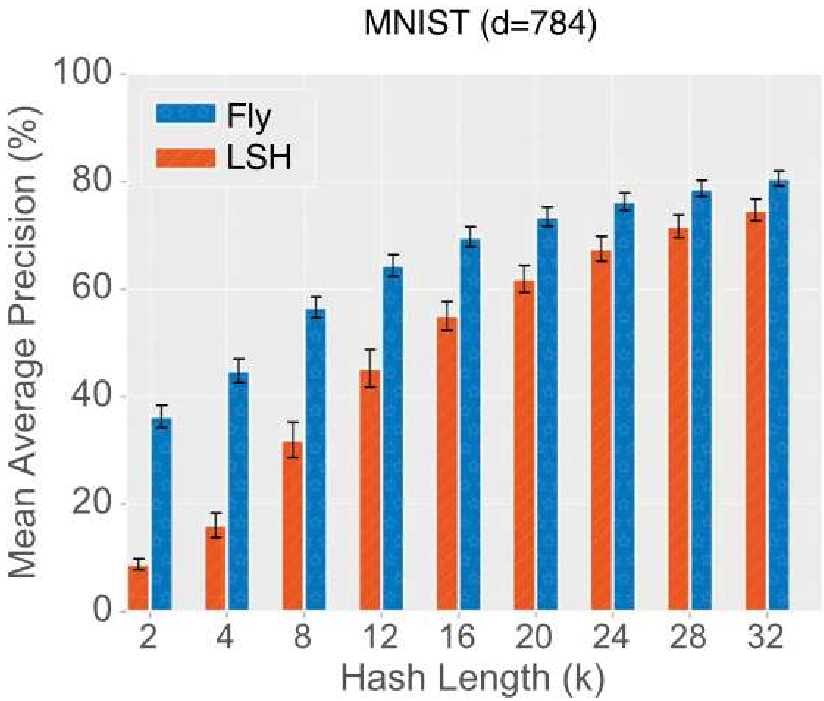
MNIST (d=784)



Source: https://www.biorxiv.org/content/biorxiv/early/2017/08/25/180471.full.pdf?%3Fcollection=

SIMILAR CONDITIONS LIKE FRUIT FLY

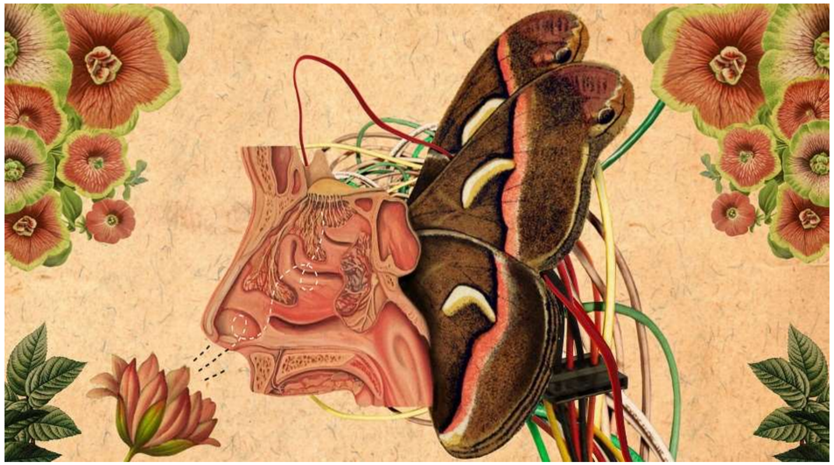
- Dimensionality expansion (~fruit fly)
  10*784 (d << m)
- Improved performance especially for small hash lengths



Source: https://www.biorxiv.org/content/biorxiv/early/2017/08/25/180471.full.pdf?%3Fcollection=

Further Information

- Preliminary Python-Implementation:
  https://github.com/dataplayer12/Fly-LSH
- New AI strategy mimics how brains learn to smell:
  https://www.quantamagazine.org/new-ai-strategy-mimics-how-brains-learn-to-smell-20180918



THANK YOU


Projection Neurons
Near Neighbor
Olfaction
Kenyon Cells
Hashing
Receptors
Tag
Close Item Search
Approximate Nearest Neighbor
Fingerprint
Proximity Search
Recommender Systems
Antennal Lobe
Distance