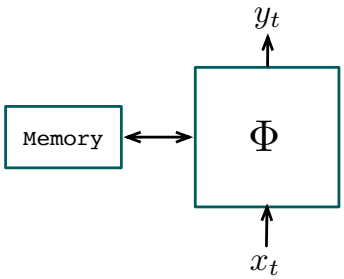
Neural machines: From fruit flies to embedded devices

Sanjoy Dasgupta

University of California, San Diego

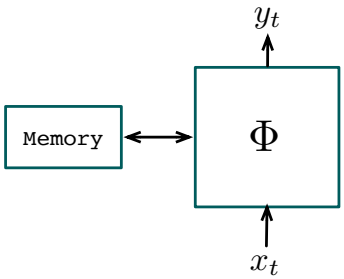
Neural machines



Biological or human-made machines for lifelong learning:

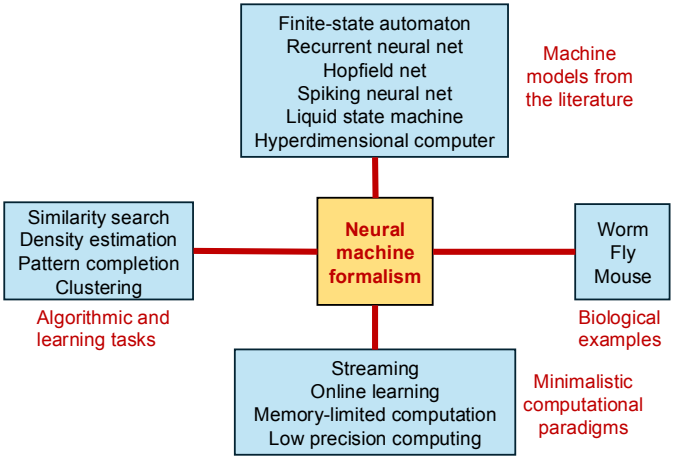
- Handle an endless stream of input demanding constant-time responses.
- Learn from experience.

Neural machines



Biological or human-made machines for lifelong learning:

- Handle an endless stream of input demanding constant-time responses.
- Learn from experience.



Outline

1. Algorithms from the fruit fly
   - **(a)** An expand-and-sparsify representation
   - **(b)** Continual learning without catastrophic forgetting
2. Hyperdimensional computing for embedded devices

Common theme: algorithmic benefits of sparse, randomized, high-dimensional representations.

Algorithms from neuroscience

Study neural circuitry of simple organisms from an algorithmic perspective.

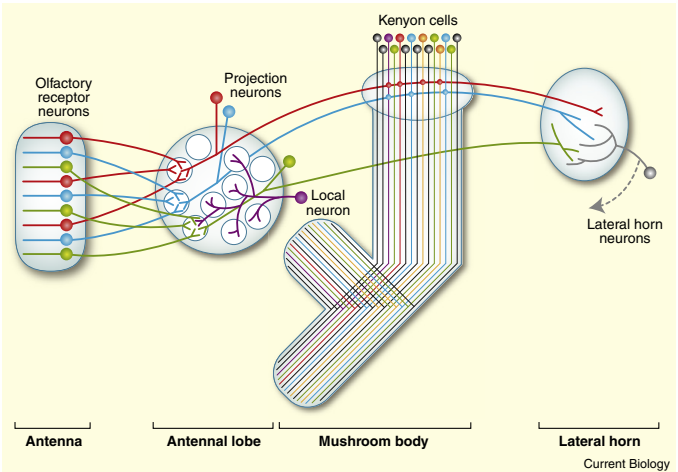
Saket Navlakha
(lead)

Chuck Stevens

Has yielded algorithms for:

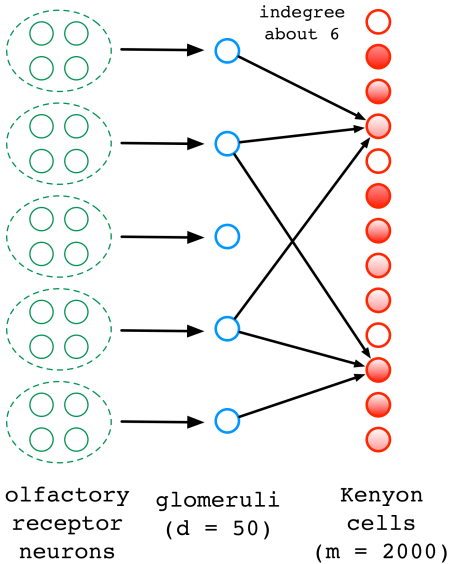
- Similarity search (Science 17)
- Novelty detection (PNAS 18)
- Habituation (PNAS 20)
- Approximate counting (Nat Comm 22)
- Continual learning (Neur Comp 23)
- Bipartite matching (PNAS 24)

The fly's sense of smell



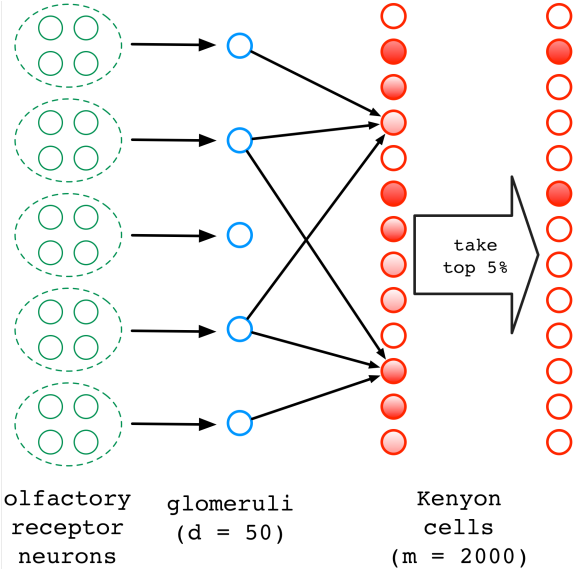
Reproduced without permission from: Masse, Turner, Jefferis. Olfactory information processing in Drosophila. Current Biology, 2009.

Step 1: Linearly map to higher dimension



The expansive map:
Multiplication by a sparse binary random matrix (of size 2000×50).

Step 2: Sparsify

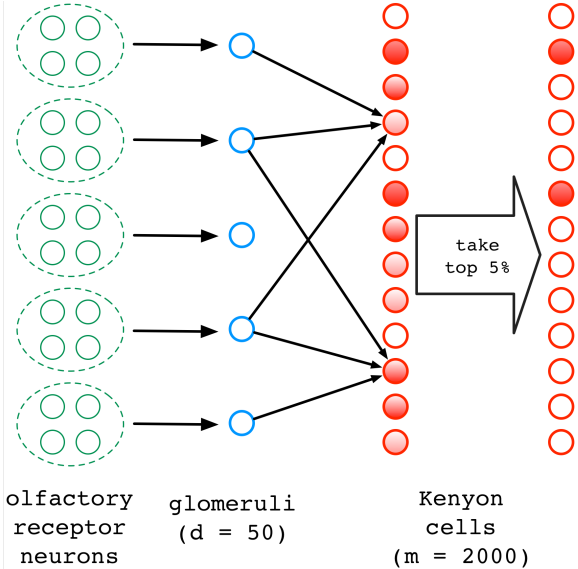


Result: a sparse binary vector, marking the locations of the k largest entries.

This is the **tag** used for:

- similarity search
- subsequent learning

Step 2: Sparsify



Result: a sparse binary vector, marking the locations of the k largest entries.

This is the **tag** used for:

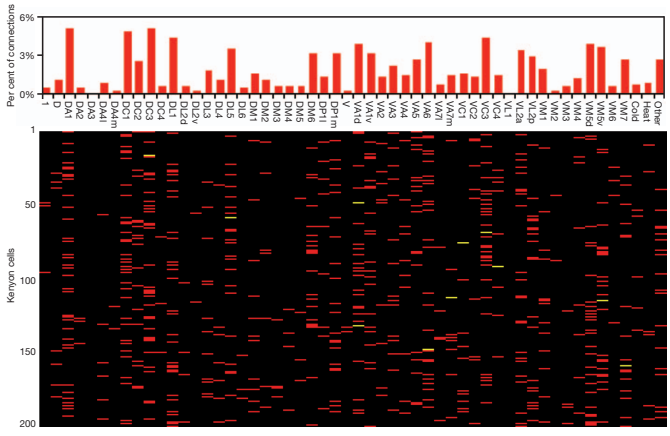
- similarity search
- subsequent learning

1. How to model the random expansive map?
2. What is the geometry of tag space?
3. What are beneficial properties of the tag?

Random linear mappings in the brain?

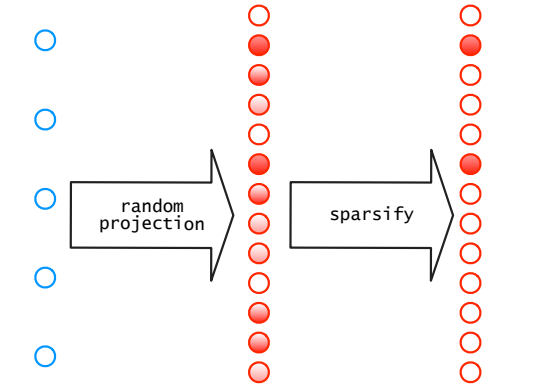
Reprinted without permission from:

Caron, Ruta, Abbott, Axel. *Random convergence of olfactory inputs in the Drosophila mushroom body.* Nature, 2013.



Effect of sparse coding: Boolean case

Data space: $\mathcal{X} = \{x \in \{0,1\}^d : \|x\|_1 = b\}$



Expansive map given by $m \times d$ random binary matrix Θ, with c ones per row.

Given an input x:

- $y = \Theta x$
- $z_i = 1$ if y_i is one of the k largest entries of y

For any $x, x' \in \mathcal{X}$, we have

$$\mathbb{E}_\Theta[\langle z, z' \rangle] \approx k \left(\frac{\langle x, x' \rangle}{b} \right)^c .$$

Algorithmic benefits of sparse code

1. It is a locality-sensitive map.

2. It facilitates many types of downstream computations.
   - A simple novelty detector in the next layer (MBONs).
   - A simple mechanism for approximately counting similar items.

3. Universal approximation.
   Pick any continuous function of x, say $f : \mathcal{X} \to \mathbb{R}$. Then f is almost-linear in the sparse tag z (that is, $f(x) \approx w \cdot z(x) + b$), if the tag size is large enough. This holds even for fixed sparsity level (# of nonzero entries).

Outline

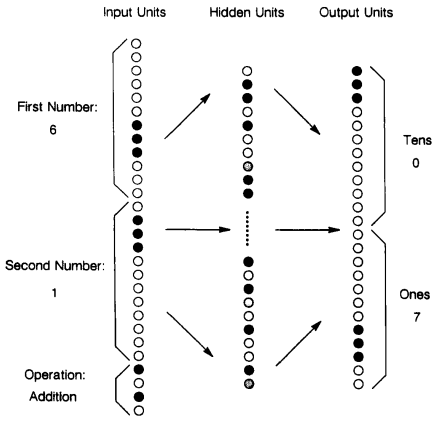
1. Algorithms from the fruit fly
   - **(a)** An expand-and-sparsify representation
   - **(b)** Continual learning without catastrophic forgetting
2. Hyperdimensional computing for embedded devices

Common theme: algorithmic benefits of sparse, randomized, high-dimensional representations.

Catastrophic forgetting

French (1993): Catastrophic forgetting is the inability of a neural network to retain old information in the presence of the new. New information destroys old unless the old information is continually relearned by the net.

Experiment of McCloskey and Cohen (1989):



Continual learning without catastrophic forgetting

Shen, Dasgupta, Navlakha (2023). *Reducing catastrophic forgetting with associative learning: a lesson from fruit flies.*

(a) Catastrophic forgetting and the Perceptron algorithm

(b) A fly-inspired associative variant of Perceptron

The multiclass Perceptron

Setting: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, 2, \ldots, k\}$

Model: Each class j has a linear function $w_j \cdot x + b_j$

Prediction: On instance x, predict label $\arg\max_j (w_j \cdot x + b_j)$

- Initialize $w_1 = \cdots = w_k = 0$ and $b_1 = \cdots = b_k = 0$
- Repeat forever:
  - Get next point x, predict label $\widehat{y} = \arg\max_j (w_j \cdot x + b_j)$
  - Get correct label y. If $y \neq \widehat{y}$:

$$\begin{aligned} \text{for correct label } y: \quad & w_y = w_y + x \\ & b_y = b_y + 1 \\ \text{for predicted label } \widehat{y}: \quad & w_{\widehat{y}} = w_{\widehat{y}} - x \\ & b_{\widehat{y}} = b_{\widehat{y}} - 1 \end{aligned}$$

Mistake bound: $2k/\gamma^2$ when data is linearly separable with margin γ

Catastrophic forgetting and the Perceptron

Say dimension d is a power of two. Pick $x_1, \ldots, x_{d-1} \in \{0,1\}^d$ so that:

- Each x_i has exactly $d/2$ ones
- Any pair of vectors has dot product exactly $d/4$

(E.g. start from $d \times d$ Hadamard matrix.)

Consider $k = d - 1$ classes, with class j supported entirely on x_j.
Ideal classifiers: $w_j = x_j$.

Introduce classes one at a time, in order:

- Class 1 introduced: $w_1 = x_1$
- Class 2 introduced: $w_2 = x_2$ but $w_1 = x_1 - x_2$ corrupted
- Class 3 introduced: $w_3 = x_3$ but $w_2 = x_2 - x_3$ corrupted
- $\vdots$

Associative variant of the Perceptron – 1

Setting: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, 2, \ldots, k\}$

Model: $w_1, \ldots, w_k \in \mathbb{R}^m$

- Initialize $w_1 = \cdots = w_k = 0$
- Repeat forever:
  - Get next point x, predict label $\widehat{y} = \arg\max_j w_j \cdot \phi(x)$
  - Get correct label y
  - Update:

$$w_y = w_y + \phi(x)$$

Associative variant of the Perceptron – 2

Setting: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, 2, \ldots, k\}$

Model: $w_1, \ldots, w_k \in \mathbb{R}^m$

- Initialize $w_1 = \cdots = w_k = 0$
- Repeat forever:
  - Get next point x, predict label $\widehat{y} = \arg\max_j w_j \cdot \phi(x)$
  - Get correct label y
  - Update:

$$w_y = (1 - \alpha) w_y + \beta \phi(x)$$

Analyzing the associative Perceptron

Definition

Let $\pi_1, \ldots, \pi_k$ be distributions over $\mathbb{R}^d$, corresponding to k classes of data points. We say the classes are (γ, δ)-separated, for some $\gamma, \delta > 0$, if for any pair of classes $j \neq j'$, at least $1 - \delta$ fraction of the points x_o from class j satisfy

$$\mathbb{E}_{X \sim \pi_j}[\phi(x_o) \cdot \phi(X)] \geq \gamma + \mathbb{E}_{X' \sim \pi_{j'}}[\phi(x_o) \cdot \phi(X')].$$

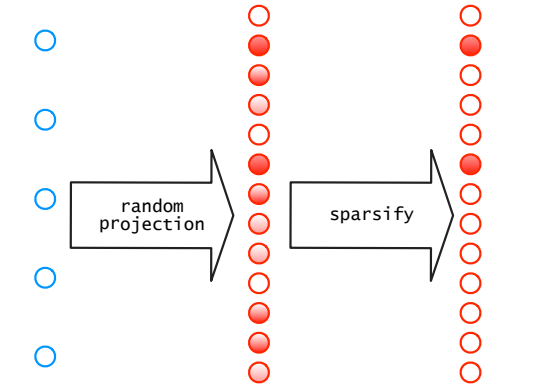
Theorem

Suppose there are k classes that are (γ, δ)-separated and that each $\|\phi(x)\| \leq R$. Let $\mu_1, \ldots, \mu_k$ denote the class means, and suppose their empirical estimates $\widehat{\mu}_j$ satisfy $\|\widehat{\mu}_j - c\mu_j\| < \frac{c\gamma}{2R}$ for $c > 0$. Then the classification rule

$$x \mapsto \arg\max_j \widehat{\mu}_j \cdot \phi(x)$$

has error $\leq (k-1)\delta$ on each class individually.

Some open problems: expand-and-sparsify representations



1. Feedback connections may allow the fly's representation $\phi(x)$ to change over time.
   If this representation changes over time, what happens to previously learned linear functions of it?

2. Formalize regularities in data, beyond clusters and manifolds, that might be captured by sparse expansive maps.

Outline

1. Algorithms from the fruit fly
   - **(a)** An expand-and-sparsify representation
   - **(b)** Continual learning without catastrophic forgetting
2. Hyperdimensional computing for embedded devices

Common theme: algorithmic benefits of sparse, randomized, high-dimensional representations.

Hyperdimensional computing

Two trends in computer architecture:

1. Massive amounts of compute needed for training and running deep learning models.
2. Pushing increasing computation onto tiny low-power embedded devices.

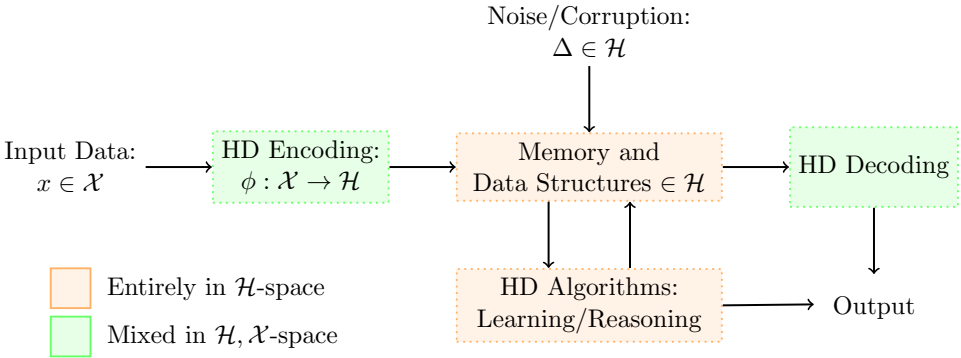
One approach to the latter: *hyperdimensional computing* or *vector-space architectures*.

Tajana Rosing

Anthony Thomas

Hyperdimensional computing

- All data is mapped into a space of very high dimension.
- In that space, only simple operations are allowed: **bundling** (elementwise addition, $\oplus$), **binding** (elementwise multiplication, $\otimes$), and scalar multiplication.



Examples

1. **Set membership.** To store a set of items $S = \{x_1, \ldots, x_m\} \subset \mathcal{X}$: use

$$\mu = \bigoplus_{i=1}^{m} \phi(x_i).$$

To test if $x \in S$, use $\langle \mu, \phi(x) \rangle \geq \theta$.

Examples

1. **Set membership.** To store a set of items $S = \{x_1, \ldots, x_m\} \subset \mathcal{X}$: use

$$\mu = \bigoplus_{i=1}^{m} \phi(x_i).$$

   To test if $x \in S$, use $\langle \mu, \phi(x) \rangle \geq \theta$.

2. **Classification.** Given k classes with examples $S_j \subset \mathcal{X}$ of each, use:

$$\mu_j = \frac{1}{|S_j|} \bigoplus_{x \in S_j} \phi(x), \quad j = 1, 2, \ldots, k.$$

   To classify x, use $\arg\max_j \langle \mu_j, \phi(x) \rangle$.

Examples

1. **Set membership.** To store a set of items $S = \{x_1, \ldots, x_m\} \subset \mathcal{X}$: use

$$\mu = \bigoplus_{i=1}^{m} \phi(x_i).$$

To test if $x \in S$, use $\langle \mu, \phi(x) \rangle \geq \theta$.

2. **Classification.** Given k classes with examples $S_j \subset \mathcal{X}$ of each, use:

$$\mu_j = \frac{1}{|S_j|} \bigoplus_{x \in S_j} \phi(x), \quad j = 1, 2, \ldots, k.$$

To classify x, use $\arg\max_j \langle \mu_j, \phi(x) \rangle$.

Q: Under what kinds of encodings $\phi(\cdot)$ do these (and a host of other basic data processing routines) provably behave correctly?

Encodings with good properties

Discrete data $\mathcal{A}$

Various guarantees (e.g. unique decoding) for encodings that are incoherent:

$$\max_{a \neq a'} \langle \phi(a), \phi(a') \rangle \leq c \min(\|\phi(a)\|^2, \|\phi(a')\|^2).$$

Can be achieved, e.g., by choosing each $\phi(a) \in_R \{-1, +1\}^d$ uniformly at random.

Encodings with good properties

Discrete data $\mathcal{A}$

Various guarantees (e.g. unique decoding) for encodings that are incoherent:

$$\max_{a \neq a'} \langle \phi(a), \phi(a') \rangle \leq c \min(\|\phi(a)\|^2, \|\phi(a')\|^2).$$

Can be achieved, e.g., by choosing each $\phi(a) \in_R \{-1, +1\}^d$ uniformly at random.

Clarkson et al (2024): Formalizes close connection to *linear sketching*.

Encodings with good properties

Discrete data $\mathcal{A}$

Various guarantees (e.g. unique decoding) for encodings that are incoherent:

$$\max_{a \neq a'} \langle \phi(a), \phi(a') \rangle \leq c \min(\|\phi(a)\|^2, \|\phi(a')\|^2).$$

Can be achieved, e.g., by choosing each $\phi(a) \in_R \{-1, +1\}^d$ uniformly at random.

Clarkson et al (2024): Formalizes close connection to *linear sketching*.

Continuous vector data $\mathcal{X}$

Various guarantees for thresholded linear maps:

$$\phi(x) = \text{sign}(Ax) \in \{-1, +1\}^d$$

E.g. creates sparse linear separability for smooth classification problems, if d is sufficiently large.

Similar to work on expand-and-sparsify representations.

Encodings with good properties

Discrete data $\mathcal{A}$

Various guarantees (e.g. unique decoding) for encodings that are incoherent:

$$\max_{a \neq a'} \langle \phi(a), \phi(a') \rangle \leq c \min(\|\phi(a)\|^2, \|\phi(a')\|^2).$$

Can be achieved, e.g., by choosing each $\phi(a) \in_R \{-1, +1\}^d$ uniformly at random.

Clarkson et al (2024): Formalizes close connection to *linear sketching*.

Continuous vector data $\mathcal{X}$

Various guarantees for thresholded linear maps:

$$\phi(x) = \text{sign}(Ax) \in \{-1, +1\}^d$$

E.g. creates sparse linear separability for smooth classification problems, if d is sufficiently large.

Similar to work on expand-and-sparsify representations.

These applications and results use only **bundling** (elementwise addition, $\oplus$) and scalar multiplication.

Understanding the binding operator

Binding (elementwise multiplication, $\otimes$) can create more complex data structures.

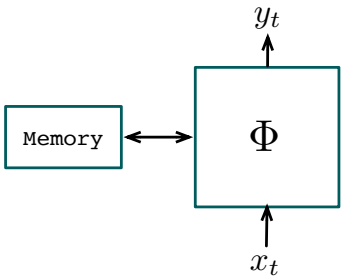
E.g. `Sanjoy: (City: San Diego), (Sign: Taurus)`

$$\phi(\text{Sanjoy}) \otimes ((\phi(\text{City}) \otimes \phi(\text{San Diego})) \oplus (\phi(\text{Sign}) \otimes \phi(\text{Taurus})))$$

Questions:

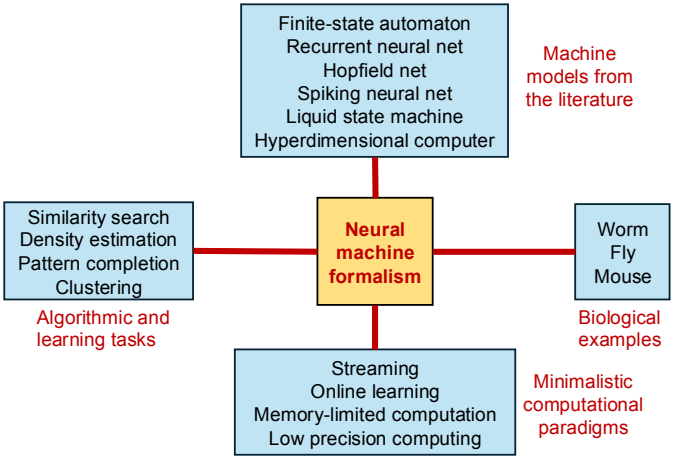
- How far does this go beyond linear sketching?
- How can this be used for simple reasoning?

Neural machines



Biological or human-made machines for lifelong learning:

- Handle an endless stream of input demanding constant-time responses.
- Learn from experience.



Thanks to my collaborators

Saket Navlakha

Chuck Stevens

Tajana Rosing

Anthony Thomas