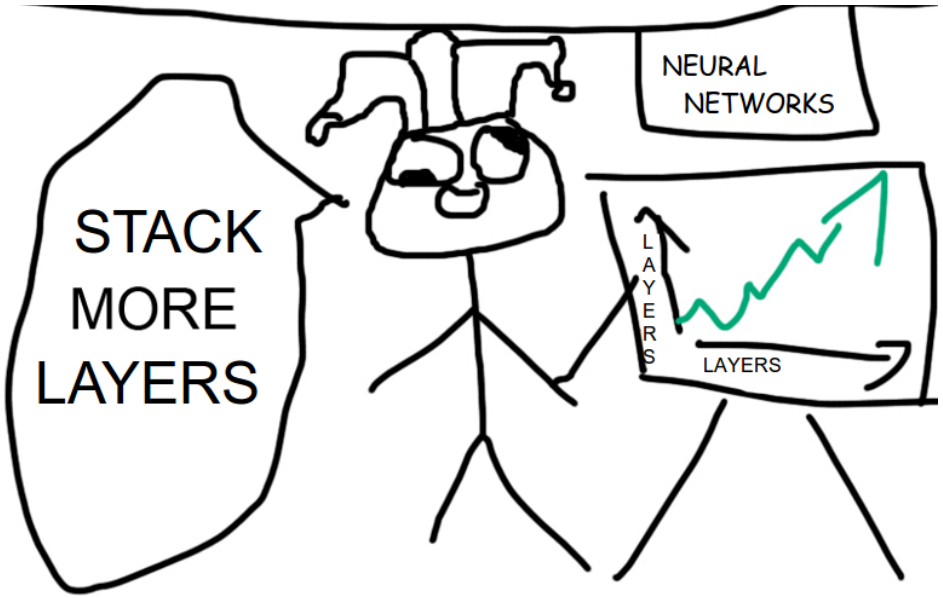
What is clustering?

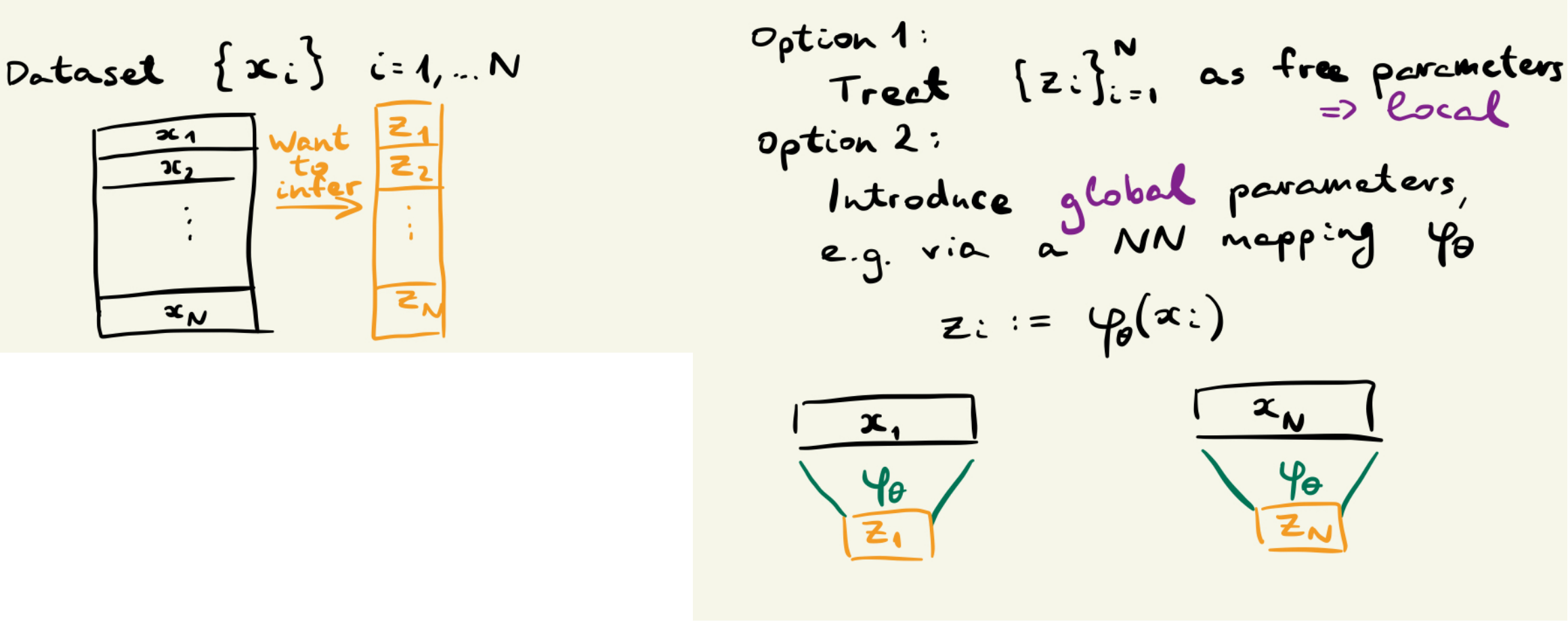
Meta-learning a clustering model across datasets



Background:
amortised inference and meta-learning

Amortised inference

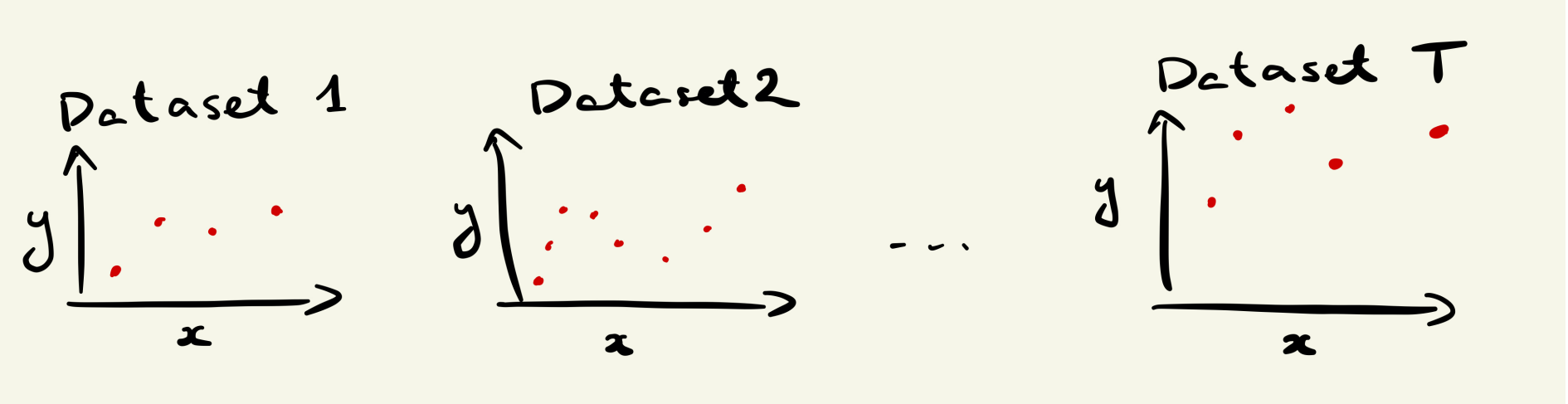
In the context of clustering or dimensionality reduction of a single dataset



Pros and cons of amortisation?

Meta-learning (regression)

E.g. consider a series of regression problems across multiple (related) datasets



Could try to amortize this *across datasets.*

Could map $\{x_i^{(t)}, y_i^{(t)}\}_{i=1}^{N_t} \longmapsto$ $f^{(t)}$ some functional representation, or $(x_*^{(t)}, y_*^{(t)})$

Challenge: variable sized inputs

Permutation invariant neural networks

Permutation invariance

$$f(\{x_1, \ldots, x_N\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(N)}\})$$

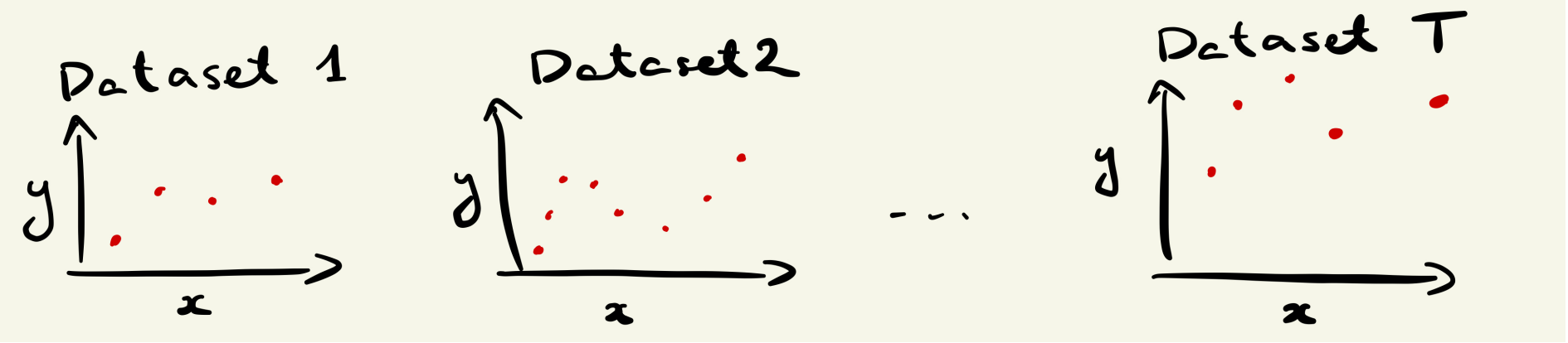
"Deep Sets" (Zaheer et al, 2017)

f is permutation invariant iff

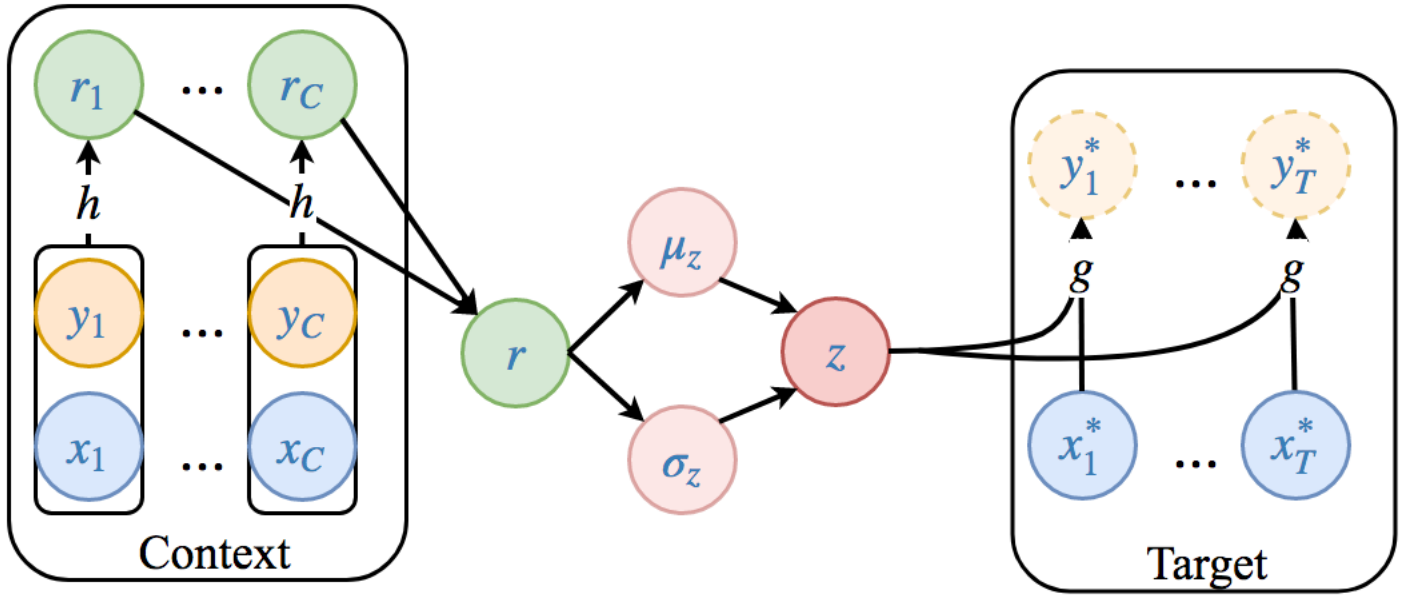
$$f(\{x_1, \ldots, x_N\}) = \rho\left(\sum_{i=1}^{N} \varphi(x_i)\right)$$

Example

A permutation-invariant architecture: Neural Processes

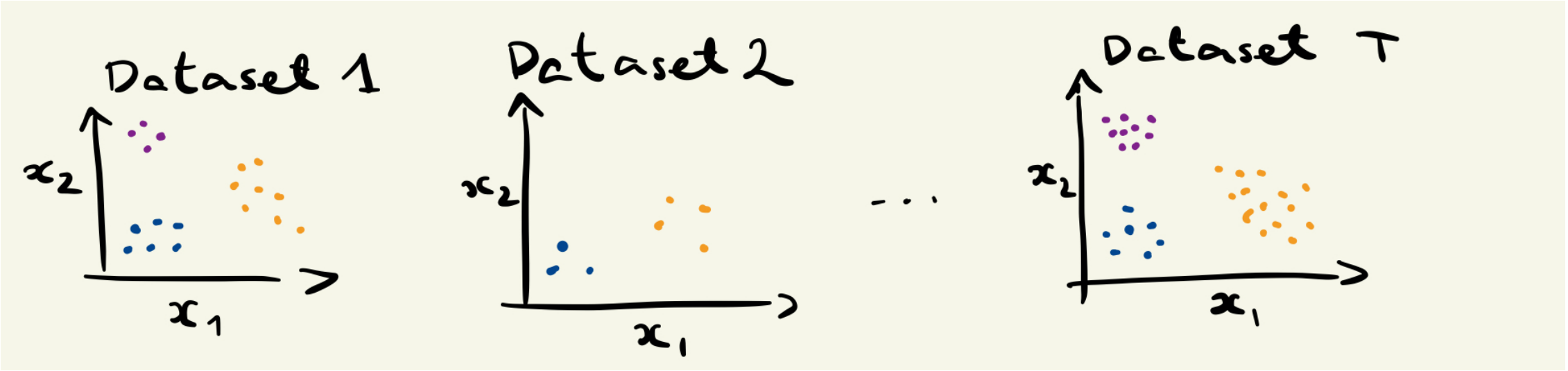


Generative model of the Neural Process



Meta-learning (clustering)

Learning a clustering model from a series of (related) datasets with ***known cluster labels***



At test time, we want to infer cluster labels

Meta-learning (clustering)

Benefits

- High cost at training time, **low cost at test time**
- **“Data-driven” properties** (e.g. the number of clusters, the shapes of clusters) without explicit assumptions

Downsides

- While data-driven, **assumptions** are not explicit (**hard-to-interpret**)
- But do we actually have **loads of training data**??
  - If we decide to generate synthetic data, we still have to make assumptions
- Not clear how **robustly** it will work in a scenario that deviates from training scenarios (e.g. a larger number of clusters)

Background: Attention

What is “attention”?

From the “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention” paper (Xu et al 2015)

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

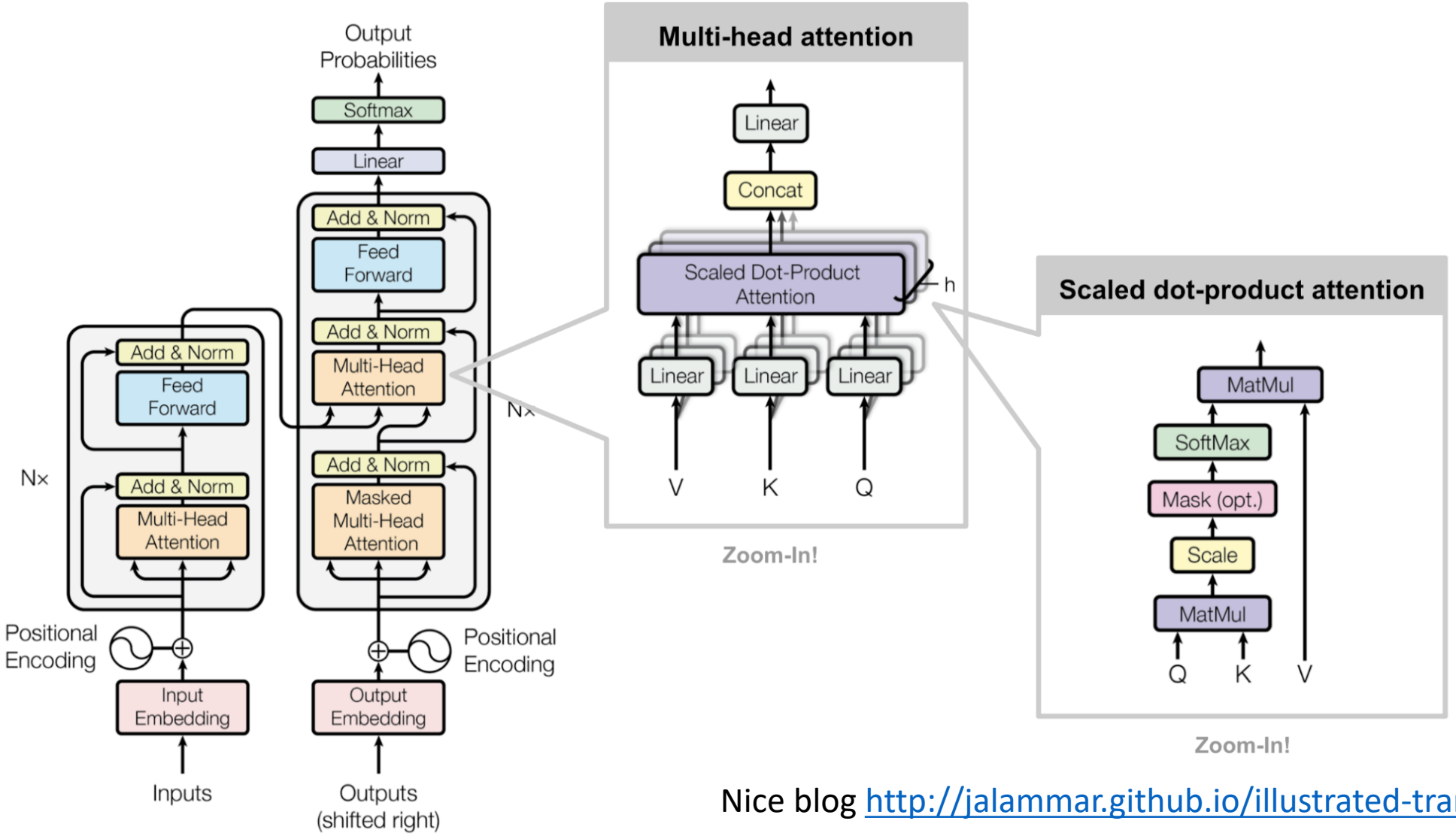
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

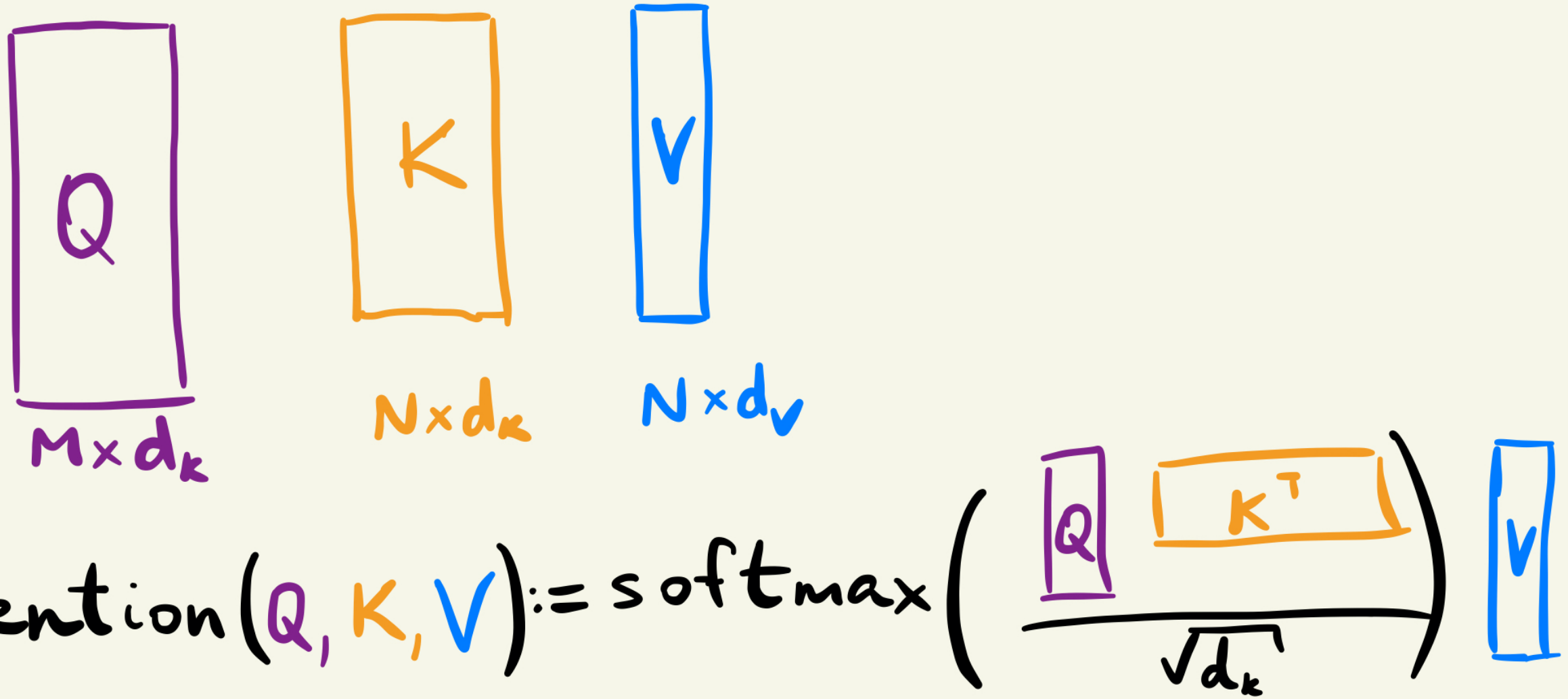
Transformers

"Attention Is All You Need" (Vaswani et al 2017)

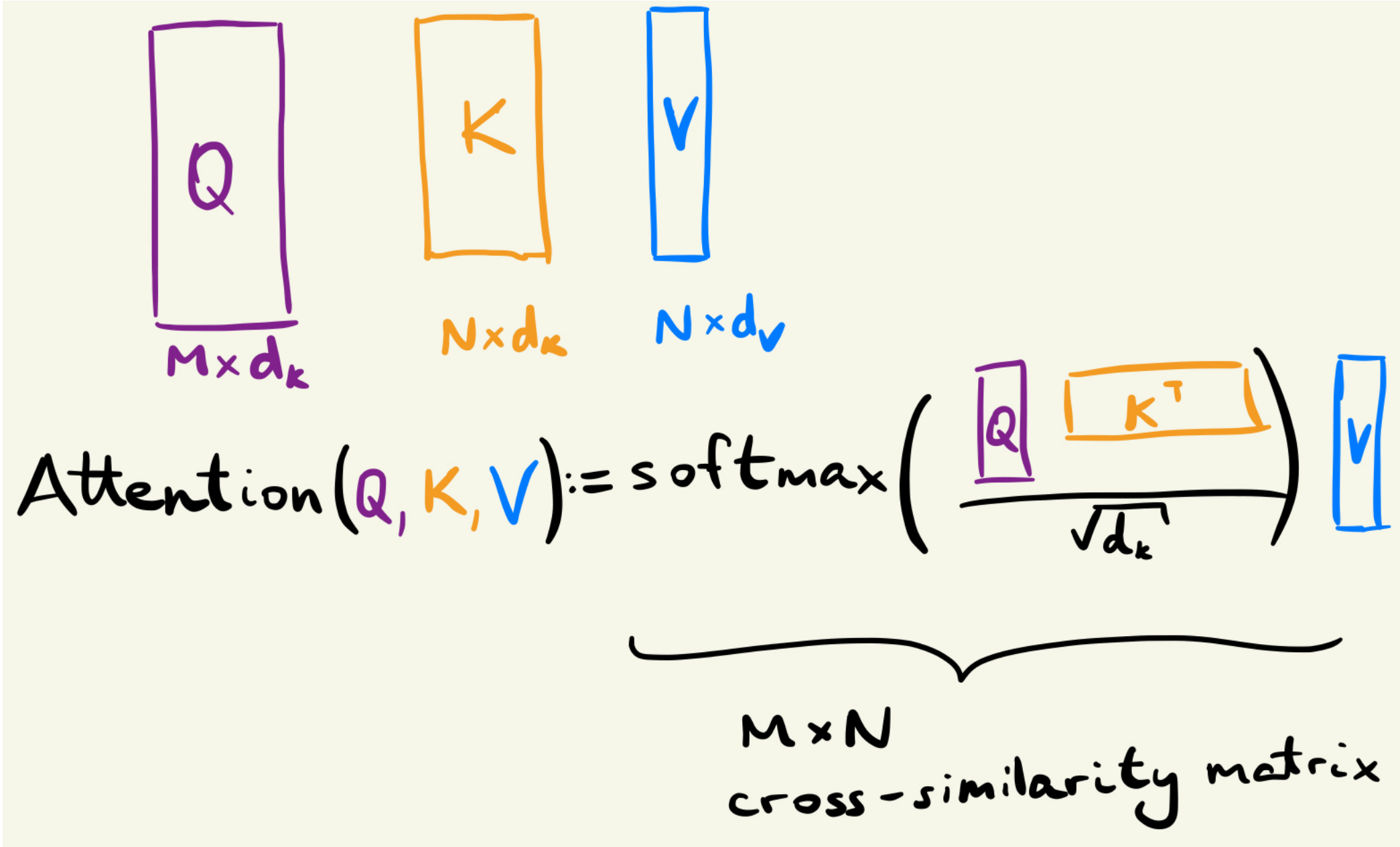


Nice blog http://jalammar.github.io/illustrated-transformer/

Attention



Attention


Q
K
V
M × d_k
N × d_k
N × d_v
Attention(Q, K, V) := softmax(Q K^T / $\sqrt{d_k}$) V
M × N
cross-similarity matrix

Multihead Attention

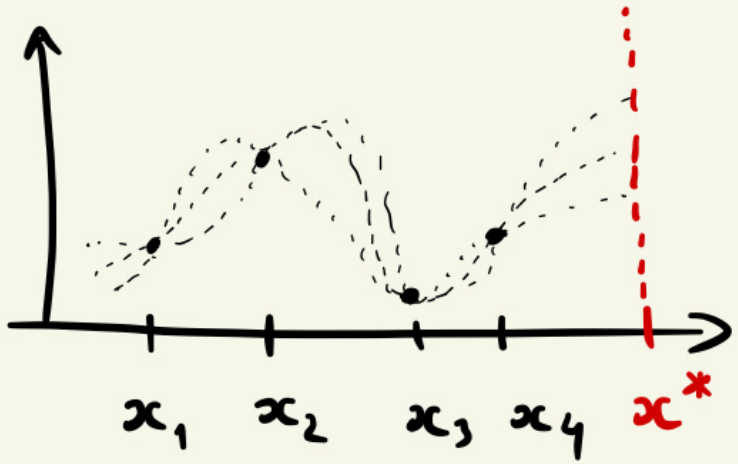
$$\text{MultiHead}_k(Q, K, V) = \text{concat}(O_1, \ldots, O_k) \cdot W^O$$

where $O_j = \text{Attention}(QW_j^Q, KW_j^K, VW_j^K)$

with learnable params $\{W^O, W_j^Q, W_j^K, W_j^K\}$

Analogy to GP regression

GP regression



Prediction at x^*

$$\mathbb{E}\, y^* = \sum_{i=1}^{N} w_i\, y_i$$

where

$$w_{1:N} = K_{*N}\, K_{NN}^{-1}$$

Same, but with attention

$$\mathbb{E}\, y^* = \sum_{i=1}^{N} w_i\, y_i ,$$

where

$$w_{1:N} = \mathrm{softmax}\left(\frac{[x^*][x_1, \dots, x_N]}{\sqrt{d}} \right)$$

Analogy to GP regression

Compute cost wise

Similarity

- Relies on N x N (and N x M) sized kernel/similarity matrices

But

- No matrix inverse!

As a result, various ideas like “inducing points” and “random features” become relevant for attention-based models as well

Meta-learning a clustering model across datasets

Papers on amortised clustering

Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks

Juho Lee [1 2] **Yoonho Lee** [3] **Jungtaek Kim** [4] **Adam R. Kosiorek** [1 5] **Seungjin Choi** [4] **Yee Whye Teh** [1]

Neural Clustering Processes

Ari Pakman [1] **Yueqi Wang** [1 2] **Catalin Mitelut** [1] **JinHyung Lee** [1] **Liam Paninski** [1]

Attentive Clustering Processes

Ari Pakman* Columbia University

Yueqi Wang* Google

Yoonho Lee AITRICS

Pallab Basu Witwatersrand University

Juho Lee KAIST

Yee Whye Teh Oxford University

Liam Paninski Columbia University

Set transformer

Considers the setup “input = set”. Proposes an attention-based approach that

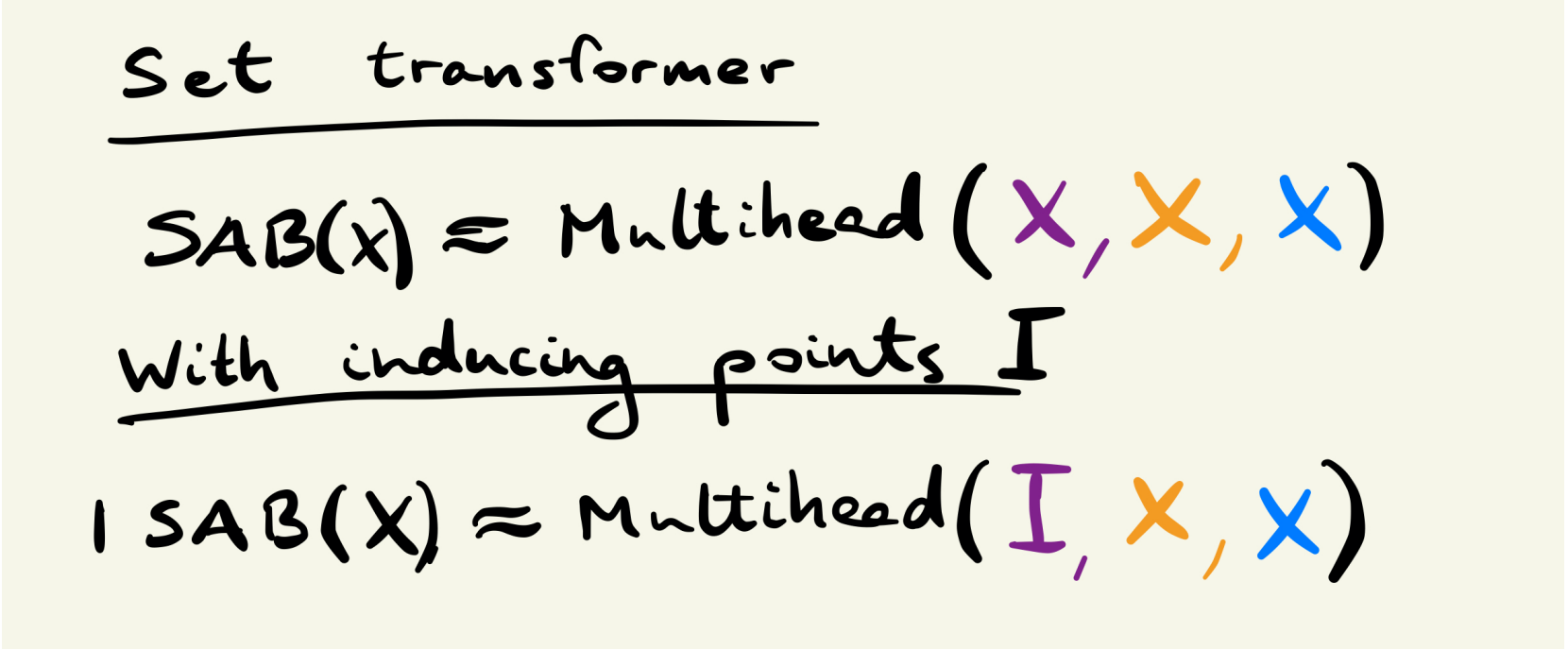
- handles variable size inputs
- is permutation-invariant

Set transformer

Considers the setup "input = set". Proposes an attention-based approach that

- handles variable size inputs
- is permutation-invariant

Proposes an attention-based set operation, the "Set Attention Block (*SAB*)":



Set transformer

Set transformer

$SAB(X) \approx \text{Multihead}(X, X, X)$

With inducing points I

$ISAB(X) \approx \text{Multihead}(I, X, X)$

Given matrices $X, Y \in \mathbb{R}^{n \times d}$ which represent two sets of d-dimensional vectors, we define the Multihead Attention Block (MAB) with parameters ω as follows:

$$\text{MAB}(X, Y) = \text{LayerNorm}(H + \text{rFF}(H)), \tag{6}$$

$$\text{where } H = \text{LayerNorm}(X + \text{Multihead}(X, Y, Y; \omega)), \tag{7}$$

rFF is any row-wise feedforward layer (i.e., it processes each instance independently and identically), and LayerNorm is layer normalization (Ba et al., 2016). The MAB is an adaptation of the encoder block of the Transformer (Vaswani et al., 2017) without positional encoding and dropout. Using the MAB, we define the Set Attention Block (SAB) as

$$\text{SAB}(X) := \text{MAB}(X, X). \tag{8}$$

Set transformer

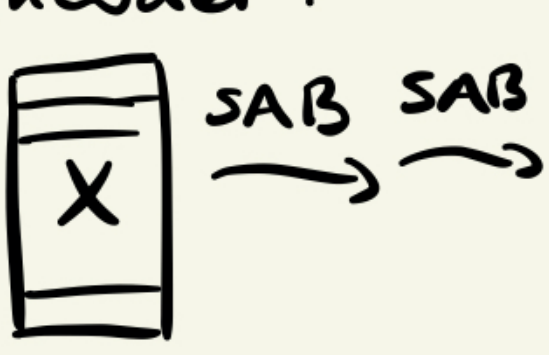
Set transformer

$SAB(X) \approx \text{Multihead}(X, X, X)$

With inducing points I

$ISAB(X) \approx \text{Multihead}(I, X, X)$

Encoder :



aggregate

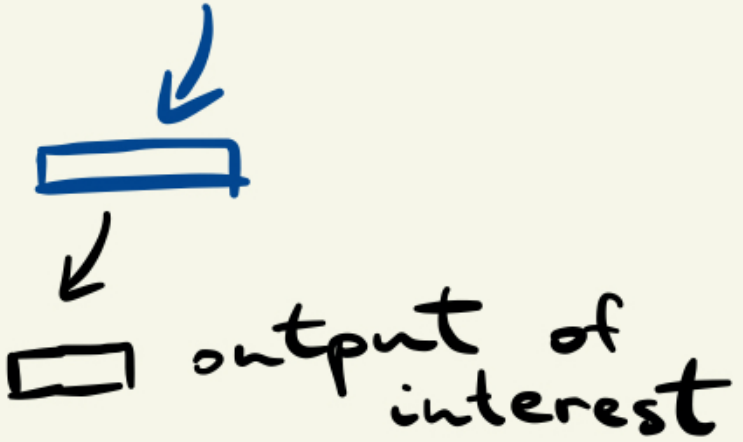
Instead of $\sum_i z_i$,

they do

$\approx \text{MultiHead}(S, Z, Z)$

with a learnable vector S

Decoder :



Set transformer

- This model can be used for clustering, when we train it to output a fixed number of cluster components

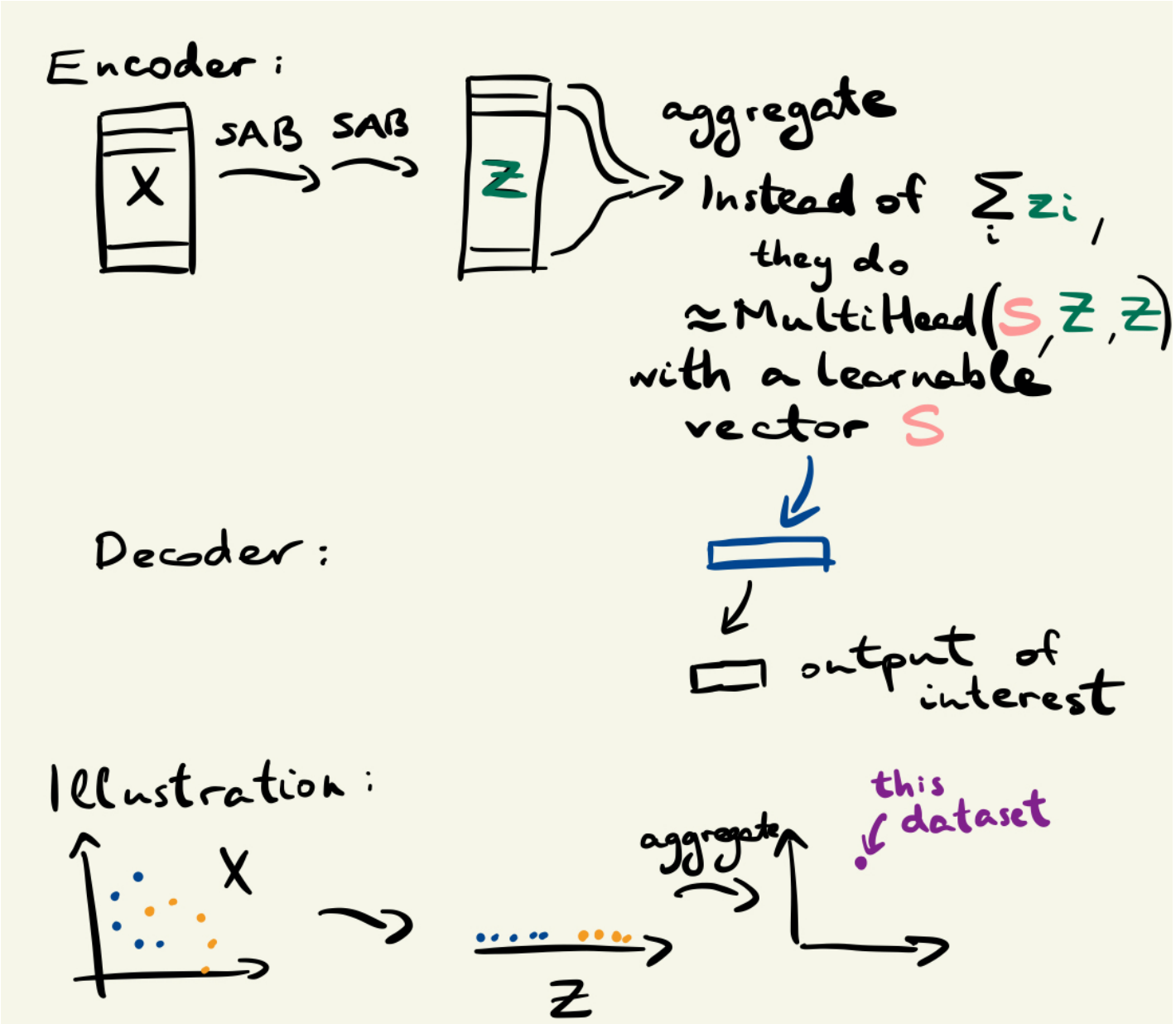
input set X to $\theta^*(X)$. One can also view this as amortized maximum likelihood learning. Specifically, given a dataset X, we train a neural network to output parameters $f(X;\lambda) = \{\pi(X), \{\mu_j(X), \sigma_j(X)\}_{j=1}^k\}$ which maximize

$$\mathbb{E}_X \left[\sum_{i=1}^{|X|} \log \sum_{j=1}^{k} \pi_j(X) \mathcal{N}(x_i; \mu_j(X), \mathrm{diag}(\sigma_j^2(X))) \right]. \quad (18)$$

We structured $f(\cdot;\lambda)$ as a set-input neural network and learned its parameters λ using stochastic gradient ascent, where we approximate gradients using minibatches of *datasets*.

Note: this does not explicitly assign data points to clusters

Set transformer



Questions / thoughts about the Set Transformer?

Papers on amortised clustering

Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks

Juho Lee [1 2] **Yoonho Lee** [3] **Jungtaek Kim** [4] **Adam R. Kosiorek** [1 5] **Seungjin Choi** [4] **Yee Whye Teh** [1]

Neural Clustering Processes

Ari Pakman [1] **Yueqi Wang** [1 2] **Catalin Mitelut** [1] **JinHyung Lee** [1] **Liam Paninski** [1]

Attentive Clustering Processes

Ari Pakman* Columbia University

Yueqi Wang* Google

Yoonho Lee AITRICS

Pallab Basu Witwatersrand University

Juho Lee KAIST

Yee Whye Teh Oxford University

Liam Paninski Columbia University

Neural Clustering Process

Consider

$$p(c_{1:N} \mid x_{1:N}) = p(c_1 \mid x_{1:N})\, p(c_2 \mid c_1, x_{1:N}) \cdots p(c_N \mid c_{1:(N-1)}, x_{1:N})$$

Idea: model $q(c_n = k \mid c_{1:(n-1)}, x_{1:N})$

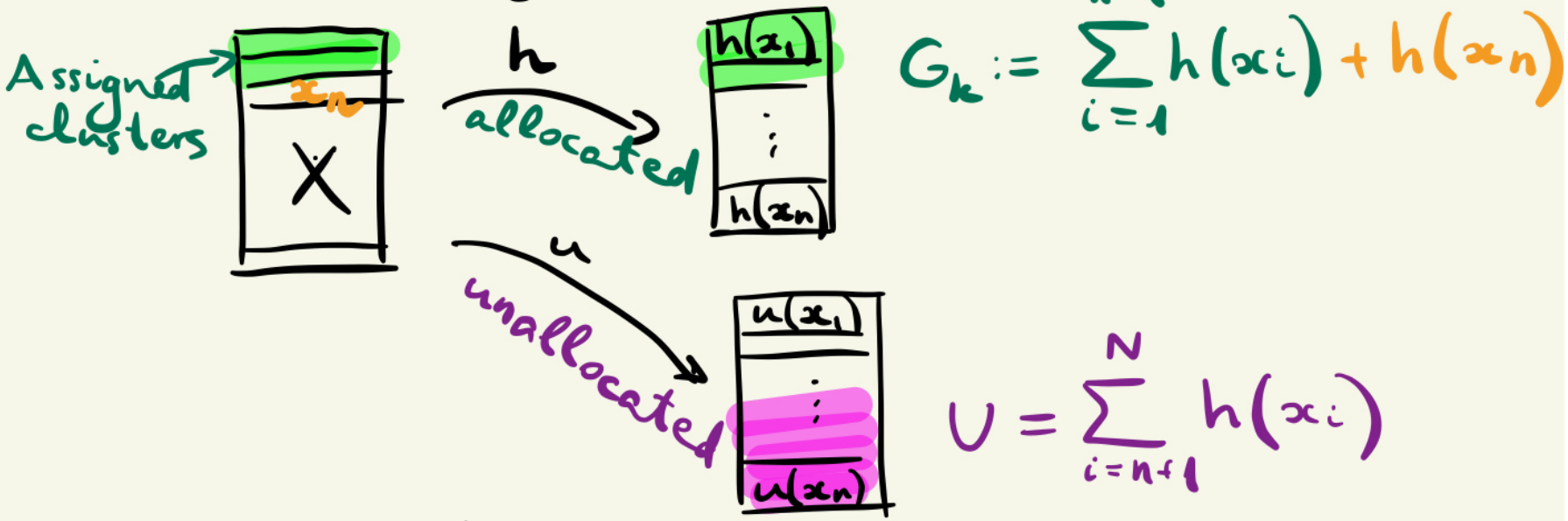
Neural Clustering Process

Consider

$$p(c_{1:N} \mid x_{1:N}) = p(c_1 \mid x_{1:N})\, p(c_2 \mid c_1, x_{1:N}) \cdots p(c_N \mid c_{1:(N-1)}, x_{1:N})$$

Idea: model $q(c_n = k \mid c_{1:(n-1)}, x_{1:N})$

Iterate through the dataset:



$$G_k := \sum_{i=1}^{n-1} h(x_i) + h(x_n)$$

$$U = \sum_{i=n+1}^{N} h(x_i)$$

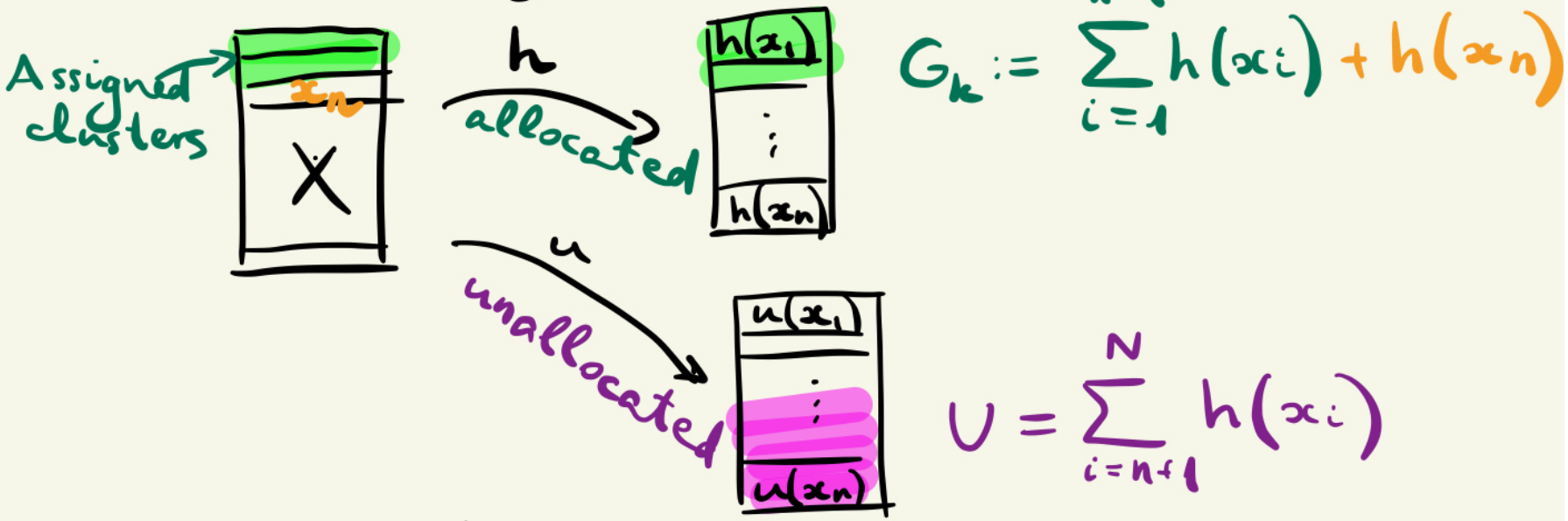
Neural Clustering Process

Consider

$$p(c_{1:N} \mid x_{1:N}) = p(c_1 \mid x_{1:N})\, p(c_2 \mid c_1, x_{1:N}) \cdots p(c_N \mid c_{1:(N-1)}, x_{1:N})$$

Idea: model $q(c_n = k \mid c_{1:(n-1)}, x_{1:N})$

Iterate through the dataset:



$$G_k := \sum_{i=1}^{n-1} h(x_i) + h(x_n)$$

$$U = \sum_{i=n+1}^{N} h(x_i)$$

Define $q(c_n = k \mid \cdots) \propto \exp(f(G_k, U))$

Neural Clustering Process – now with Clusterwise Sampling

Neural Clustering Process – now with Clusterwise Sampling

Two representations:

$$c_{1:N} = (1, 1, 2, 2, 3, 1)$$

or alternatively

$$s_1 = (1, 2, 6)$$

$$s_2 = (3, 4)$$

$$s_3 = (5)$$

Neural Clustering Process – now with Clusterwise Sampling

Two representations:

$c_{1:N} = (1, 1, 2, 2, 3, 1)$

or alternatively

$s_1 = (1, 2, 6)$

$s_2 = (3, 4)$

$s_3 = (5)$

While the NCP algorithm is good enough for small datasets, $O(N)$ forward calls might be too many for large datasets. We consider now an $O(K)$ alternative, based on modeling the factors in the clusterwise expansion (4),

$$p(\mathbf{s}_{1:K}|\mathbf{x}) = p(\mathbf{s}_1|\mathbf{x})p(\mathbf{s}_2|\mathbf{s}_1, \mathbf{x}) \dots p(\mathbf{s}_K|\mathbf{s}_{1:K-1}, \mathbf{x}) \,. \quad (12)$$

Neural Clustering Process – now with Clusterwise Sampling

Consider $p(s_1|x) \cdots p(s_K|s_{1:(K-1)}, x)$

Neural Clustering Process – now with Clusterwise Sampling

Consider $p(s_1|x) \cdots p(s_K | s_{1:(k-1)}, x)$

Sampling illustrated

$s_1 = (1, 4, 5)$

c=1 c=1

Neural Clustering Process – now with Clusterwise Sampling

Consider $p(s_1|x) \cdots p(s_K | s_{1:(K-1)}, x)$

Sampling illustrated

$s_1 = (1, 4, 5)$

Pick index d_2 uniformly from 

and then a binary vector $b_2 | d_2$

Neural Clustering Process – now with Clusterwise Sampling

Consider $p(s_1|x) \cdots p(s_K | s_{1:(K-1)}, x)$

Sampling illustrated

$s_1 = (1, 4, 5)$

Pick index d_2 uniformly from 

and then a binary vector $b_2 | d_2$

b_2: □□ □□

Neural Clustering Process – now with Clusterwise Sampling

So we have $p(d_k, b_k \mid s_{1:(k-1)}, x_{1:N})$.
We can marginalise out d_k:

$$p(b_k \mid s_{1:(k-1)}, x_{1:N})$$

- **Conditional exchangeability:** the distribution $p(\mathbf{b}_k|\mathbf{x}_k)$ satisfies

$$p(b_1 \dots b_{m_k}|x_{a_1}, \dots, x_{a_{m_k}}, x_{d_k}, \mathbf{x_s}) = \quad (13)$$
$$p(b_{\sigma_1} \dots b_{\sigma_{m_k}}|x_{\sigma_{a_1}} \dots x_{\sigma_{a_{m_k}}}, x_{d_k}, \mathbf{x_s}),$$

where σ is an arbitrary permutation of the elements of $\mathbf{b}_k$ and $\mathbf{x_a}$.

CCP model proposes[1]

$$p(\mathbf{b}_k|\mathbf{x}_k) \simeq \int d\mathbf{z}_k \prod_{i=1}^{m_k} p_{\theta,i}(b_i|\mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k|\mathbf{x}_k), \quad (15)$$

with integrands

$$p_\theta(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k|\mathbf{x}_k) \quad (16)$$
$$p_{\theta,i}(b_i = 1|\mathbf{z}_k, \mathbf{x}_k) = \text{sigmoid}[\rho_i(\mathbf{z}_k, \mathbf{x}_k)]. \quad (17)$$

Neural Clustering Process – now with Clusterwise Sampling

So we have $p(d_k, b_k | s_{1:(k-1)}, x_{1:N})$.
We can marginalise out d_k:

$$p(b_k | s_{1:(k-1)}, x_{1:N})$$

- **Conditional exchangeability:** the distribution $p(\mathbf{b}_k|\mathbf{x}_k)$ satisfies

$$p(b_1 \dots b_{m_k}|x_{a_1}, \dots, x_{a_{m_k}}, x_{d_k}, \mathbf{x_s}) = \quad (13)$$
$$p(b_{\sigma_1} \dots b_{\sigma_{m_k}}|x_{\sigma_{a_1}} \dots x_{\sigma_{a_{m_k}}}, x_{d_k}, \mathbf{x_s}),$$

where σ is an arbitrary permutation of the elements of $\mathbf{b}_k$ and $\mathbf{x_a}$.

CCP model proposes[1]

$$p(\mathbf{b}_k|\mathbf{x}_k) \simeq \int d\mathbf{z}_k \prod_{i=1}^{m_k} p_{\theta,i}(b_i|\mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k|\mathbf{x}_k), \quad (15)$$

with integrands

$$p_\theta(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k|\mathbf{x}_k) \quad (16)$$
$$p_{\theta,i}(b_i = 1|\mathbf{z}_k, \mathbf{x}_k) = \text{sigmoid}[\rho_i(\mathbf{z}_k, \mathbf{x}_k)]. \quad (17)$$

Since from eq (15) the posteriors of the b_i's are conditionally independent, after sampling $p_\theta(\mathbf{z}_k|\mathbf{x}_k)$, all the b_i's can be sampled in parallel. This in turn drastically lowers the computational cost: while a full sample of course has cost $O(N)$, the heaviest computational burden, from network evaluations, scales as $O(K)$, since each factor $p(\mathbf{s}_k|\mathbf{s}_{1:k-1}, \mathbf{x})$ in (8) needs $O(1)$ forward calls.

Neural Clustering Process – now with Clusterwise Sampling

To summarize, each new cluster $\mathbf{s}_k$ is sampled from $p(\mathbf{s}_k|\mathbf{s}_{1:k-1}, \mathbf{x})$ in a process with latent variables $d_k, \mathbf{z}_k$ and joint distribution

$$p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k|\mathbf{s}_{1:k-1}, \mathbf{x}) = \tag{18}$$
$$p_\theta(\mathbf{b}_k|\mathbf{z}_k, \mathbf{x}_k)p_\theta(\mathbf{z}_k|\mathbf{x}_k)p(d_k|\mathbf{s}_{1:k-1})$$

where

$$p_\theta(\mathbf{b}_k|\mathbf{z}_k, \mathbf{x}_k) = \prod_{i=1}^{m_k} p_{\theta,i}(b_i|\mathbf{z}_k, \mathbf{x}_k)\,. \tag{19}$$

Note that b_i have variable lengths

Papers on amortised clustering

Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks

Juho Lee [1 2] **Yoonho Lee** [3] **Jungtaek Kim** [4] **Adam R. Kosiorek** [1 5] **Seungjin Choi** [4] **Yee Whye Teh** [1]

Neural Clustering Processes

Ari Pakman [1] **Yueqi Wang** [1 2] **Catalin Mitelut** [1] **JinHyung Lee** [1] **Liam Paninski** [1]

Attentive Clustering Processes

Ari Pakman* Columbia University

Yueqi Wang* Google

Yoonho Lee AITRICS

Pallab Basu Witwatersrand University

Juho Lee KAIST

Yee Whye Teh Oxford University

Liam Paninski Columbia University

Attentive Clustering Process

Attentive Clustering Process = Neural Clustering Process + Set transformer

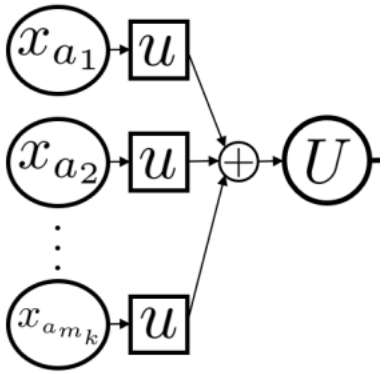
Attentive Clustering Process

Replace $U = \sum_{i \text{ unallocated}} u(x_i)$

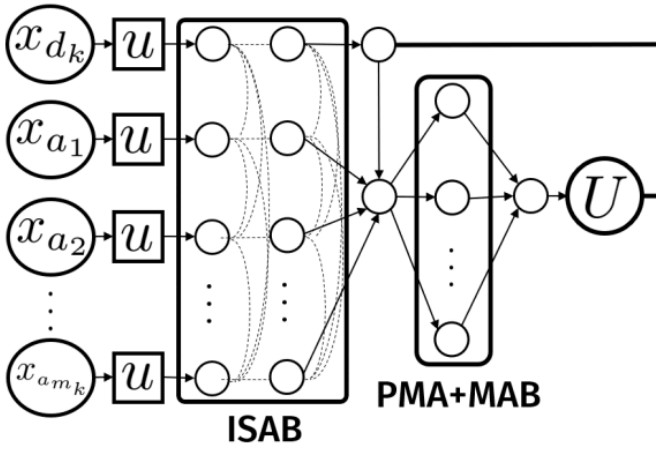
with $\text{ISAB}\left(u(x_{d_k}), u(x_1), \dots, u(x_{m_k})\right)$

followed by attentive aggregation

Without attention



With attention



Over to Dom