Some Thoughts about Learning Predictions Online

Martha White

TTT, 2019

Online Prediction Learning

- Constant stream of data $(X_1, Y_1), (X_2, Y_2), \ldots, (X_t, Y_t), \ldots$
- **Goal**: Predict target y, given input x
- Standard prediction problem, but
  - input sequence is correlated (e.g., Markov chain, time series)
  - predicting many things
  - might add new predictions as time passes

Why this setting matters

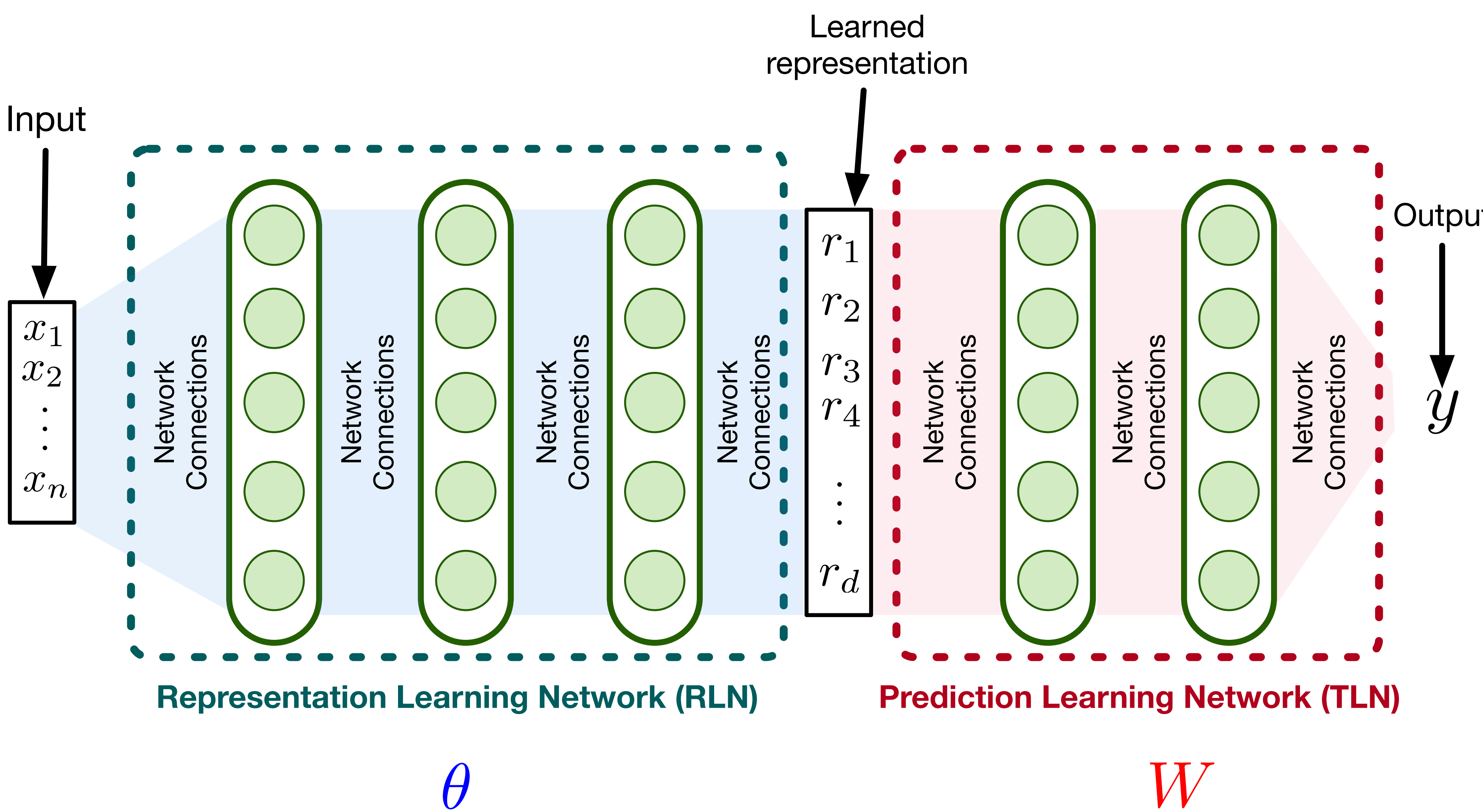
- **It reflects how we really get data**
- Even if you
  - do not want the agent to update online (e.g., safety)
  - or can store and update with all of your data
- You still get data online; it can be good to remember that

Desired Outcomes

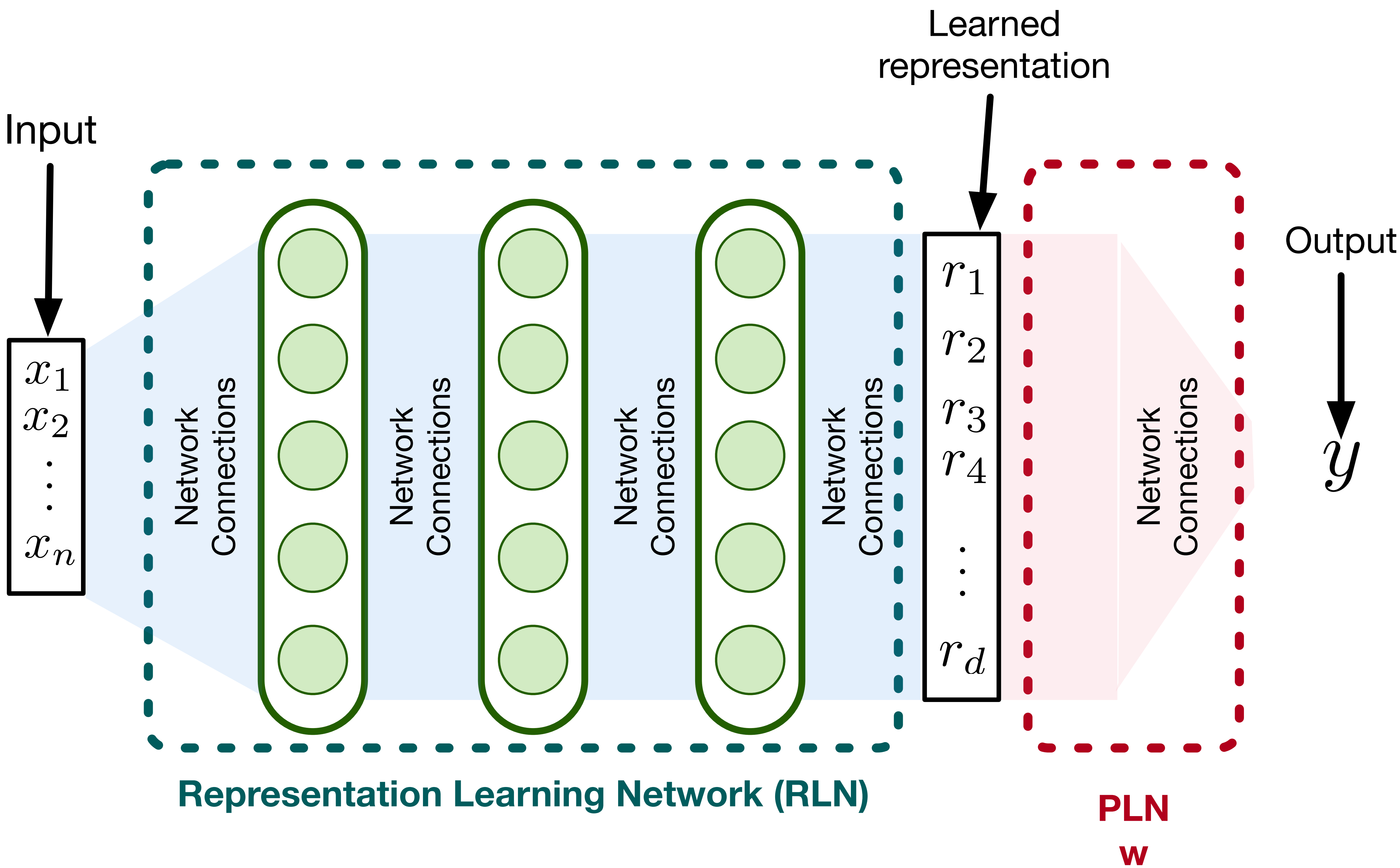
- **Generalization**: learning on observed samples enables accurate predictions on unobserved (but related) samples
- **Faster learning**: learning on observed samples enables you to learn faster on new samples
- **Minimal forgetting**: maintain learning on all observed data
  - updating on recent samples does not ruin accuracy on older samples

How can we achieve this?

Learn a representation that makes it easier
to learn a function with these properties



Or more simply for this talk



Learning functions or representations?

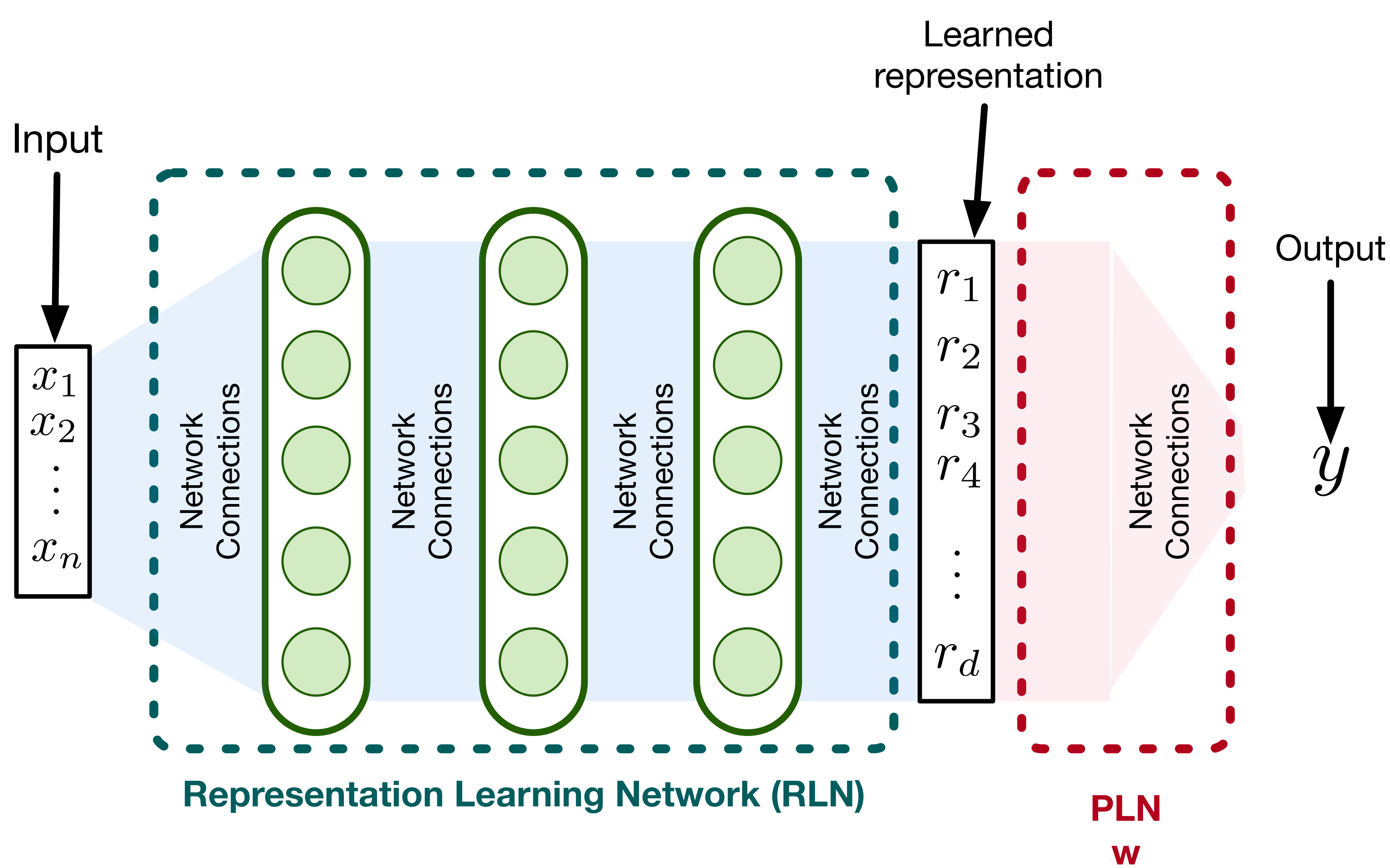
- Why do we talk about learning a representation?
- These three goals can be achieved just by thinking about the function itself directly
  - Recall goals: Generalization, Faster Learning, Minimize Interference
- NN could implicitly learn a representation anyway

Comment 1

- Neural network solutions are likely **under-constrained**
- Learning a function to minimize a loss could
    - produce an “interesting” representation (implicitly)
    - OR it could produce features that mean very little

Hypothesis 1

If we are going to talk about representation learning, then we should **learn representations explicitly**



Consequences

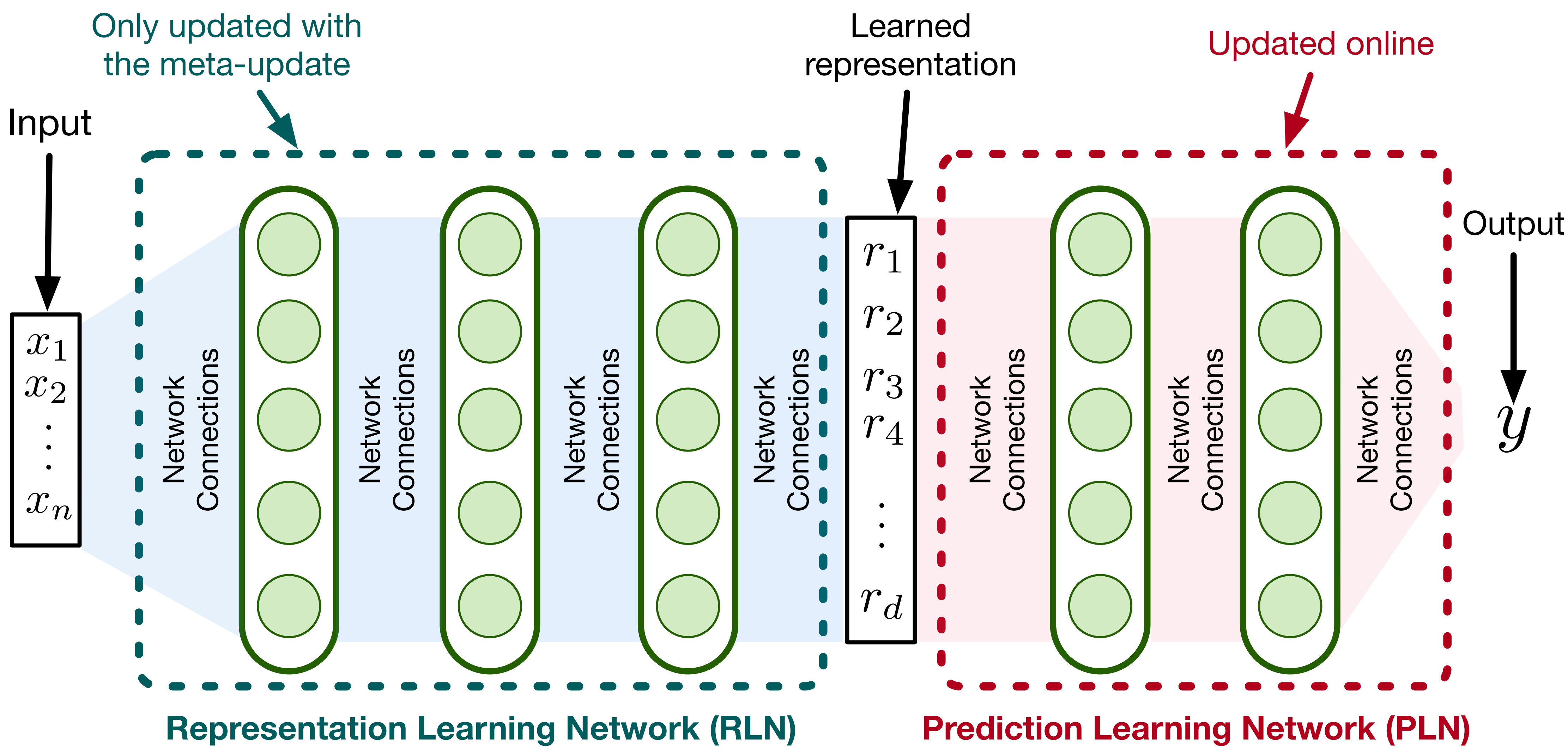
- Consider **different strategies for training representations**
- Representations can be learned slowly, as a background process
- Representations could be learned using generate-and-test
- Representations can be learned using different objectives than the primary objective to minimize the error

Some of our work

- Meta-learned Representations for Continual Learning, or MRCL (with Khurram)

- Two-timescale Networks (with Wes, Somjit, Ajin)

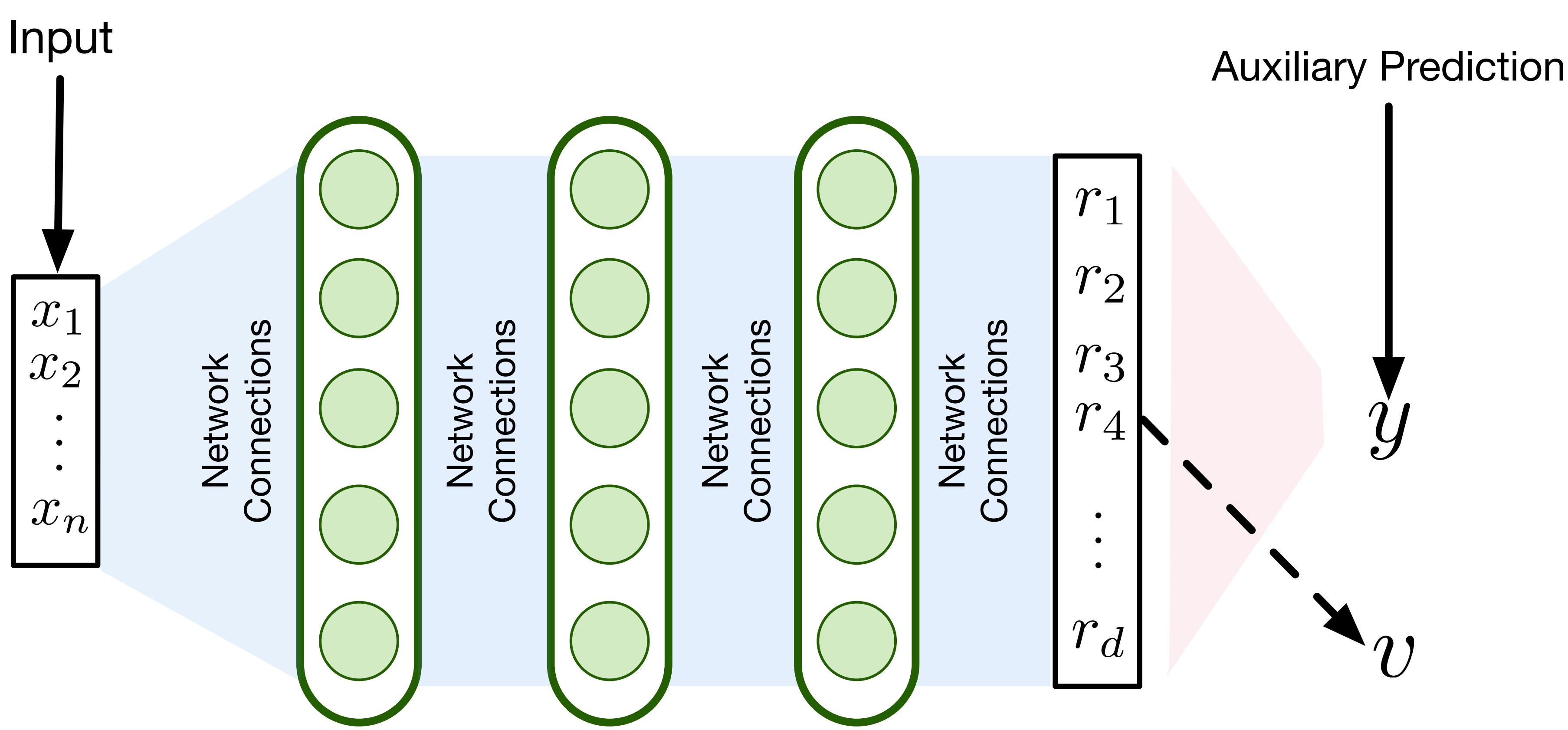
MRCL



*Paper on arXiv, Meta-learning representations for continual learning

Two-timescale Networks

- Train representation with related prediction problems



*Paper at ICLR 2019, Two-Timescale Networks for Nonlinear Value Function Approximation

Comment 2

- Representation learning only makes sense if you will be **learning more in the future**
  - Conversely, it usually does not make sense for a single prediction problem on a batch of data
- Representation learning is a **second-order problem**

Consequence

- **Experimental design** to test representation learning needs to account for learning the representation
  - e.g., design environment where more predictions are added as time passes
  - e.g., introduce non-stationarity
  - e.g., allow for a pre-training phase, to simulate using previous learning for new learning

Comment 3

- Online prediction is a **problem setting** not a solution approach
- Batch is not the opposite of Online

Consequence

- We should be open to appropriate batch approaches
- Batch updating (say by storing data) could be part of the solution to the Online Prediction Problem
- Experience replay could be part of the solution (or some variant of it)

Comment 4

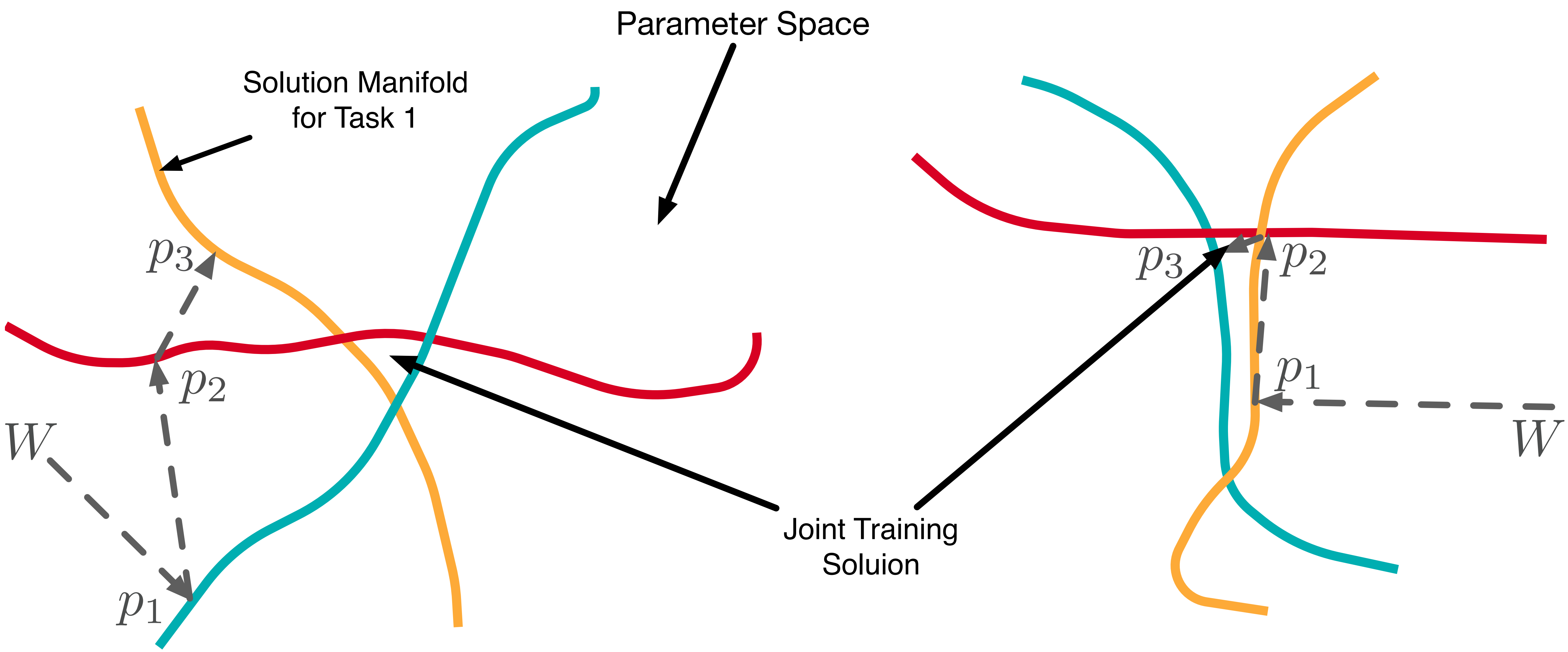
- **Sample efficiency and minimizing interference** are linked

- A small mini-batch is not representative of the whole space, even in an iid setting

- If a representation minimizes interference, each mini-batch update should mostly improve estimate

- If a representation does not minimize interference, improvement happens across (more) mini-batch updates

Visualization



Solution manifolds for an unconstrained representation

Solution manifolds for an ideal representation for continual learning

Hypothesis

Might want to consider strategies used to mitigate interference for online updating even for iid data.

Comment 5

- **Mitigating interference** in updates relates to **orthogonality between feature vectors**

$$\nabla \ell_i(\beta)^\top \nabla \ell_j(\beta) \approx 0, \quad \beta = [\theta, w]$$

Comment 5

- **Mitigating interference** in updates relates to **orthogonality between feature vectors**

$$\nabla \ell_i(\beta)^\top \nabla \ell_j(\beta) \approx 0, \quad \beta = [\theta, w]$$

$$\nabla \ell_i(\beta) = \delta_i \nabla f_\beta(x_i)$$

$$\nabla f_\beta(x_i) = [\phi_\theta(x_i), \nabla \phi_\theta(x_i)]$$

Comment 5

- **Mitigating interference** in updates relates to **orthogonality between feature vectors**

$$\nabla \ell_i(\beta)^\top \nabla \ell_j(\beta) \approx 0, \quad \beta = [\theta, w]$$

$$\nabla \ell_i(\beta) = \delta_i \nabla f_\beta(x_i)$$

$$\nabla f_\beta(x_i) = [\phi_\theta(x_i), \nabla \phi_\theta(x_i)]$$

$$\nabla \ell_i(\beta)^\top \nabla \ell_j(\beta) = \delta_i \delta_j \nabla f_\beta(x_i)^\top \nabla f_\beta(x_j) \approx 0$$

$$\phi_\theta(x_i)^\top \phi_\theta(x_j) \approx 0$$

Comment 6

- Finding **nearly orthogonal features** is equivalent to finding **nearly orthogonal feature vectors**

$$\arg\min_{\theta} \sum_{j,k} \left(\mathbb{E}[\phi_{\theta,j}(X)\phi_{\theta,k}(X)] - \delta_{j,k} \right)^2$$

$$= \arg\min_{\theta} \mathbb{E}\left[(\phi_\theta(X)^\top \phi_\theta(U))^2 - \|\phi_\theta(X)\|_2^2 - \|\phi_\theta(U)\|_2^2 \right]$$

$$\delta_{j,k} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}$$

Comment 7

- **Orthogonal non-negative features are likely sparse**
- If $\phi(x)$ is non-negative,
  - $\mathbb{E}[\phi_j(X)\phi_k(X)]$ is near zero for any j != k, only if a small number of features are active (instance sparsity)
  - $\phi(x)^\top \phi(u)$ is small only if there is little overlap in activation between vectors (lifetime sparsity)

Question: How do we get good generalization?

- Do we build-in constraints onto our networks?
- Do we use lots of data/predictions? How much is enough?