

# Lifelong and Meta-Learning: Beyond Just More of the Same

Christoph H. Lampert

IST Austria (Institute of Science and Technology Austria)



*Institute of Science and Technology*



- ▶ institute for **basic research**
- ▶ opened in 2009
- ▶ located in outskirts of Vienna

## Research at IST Austria

- ▶ curiosity-driven
- ▶ all natural/formal sciences
  - ▶ machine learning: ELLIS unit
- ▶ focus on interdisciplinarity

## We're hiring! (on all levels)

- ▶ interns, graduate school, postdocs, faculty (tenured or tenure-track), ...

## Machine Learning Theory

- ▶ Transfer Learning
- ▶ Lifelong Learning/ Meta-learning
- ▶ Robust Learning
- ▶ Theory of Deep Learning

## Models/Algorithms

- ▶ Zero-shot Learning
- ▶ Continual Learning
- ▶ Weakly-supervised Learning
- ▶ Trustworthy Learning

## Learning for Computer Vision

- ▶ Scene Understanding
- ▶ Generative Models
- ▶ Abstract Reasoning
- ▶ Semantic Representations

## Machine Learning Theory

- ▶ Transfer Learning
- ▶ Lifelong Learning/ Meta-learning
- ▶ Robust Learning
- ▶ Theory of Deep Learning

## Models/Algorithms

- ▶ Zero-shot Learning
- ▶ Continual Learning
- ▶ Weakly-supervised Learning
- ▶ Trustworthy Learning

## Learning for Computer Vision

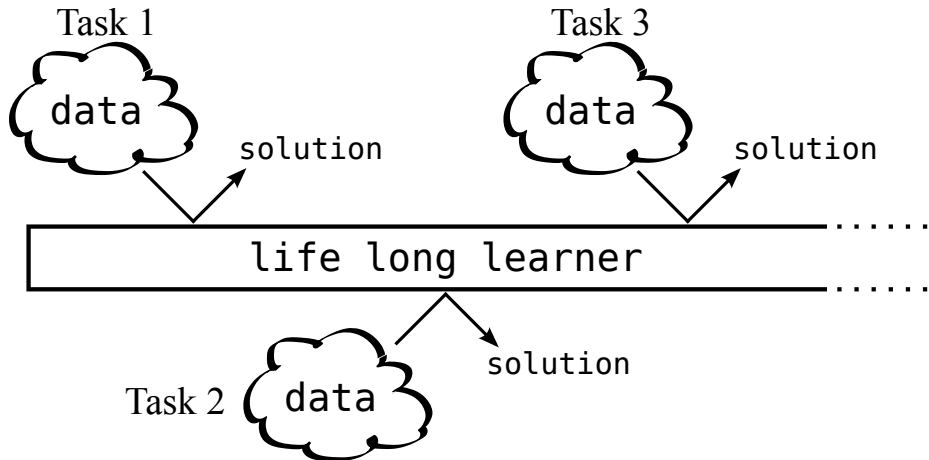
- ▶ Scene Understanding
- ▶ Generative Models
- ▶ Abstract Reasoning
- ▶ Semantic Representations

Overview

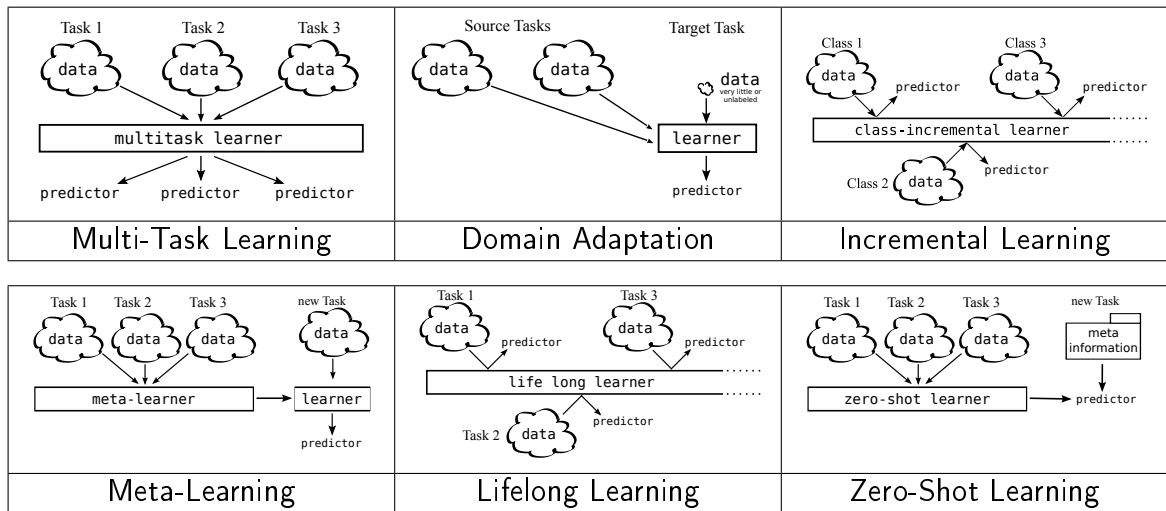
Refresher: Inductive Bias

Meta-Learning Beyond "More of the Same"

Slides available at: <http://cvml.ist.ac.at>



# Transfer Learning – What you see at 2020s conferences



many more: curriculum learning, self-paced learning, weakly-supervised learning, self-supervised learning, . . . , not to mention reinforcement learning etc.

# Refresher: Inductive Bias



## Learning task:

- ▶ input set  $\mathcal{X}$ , output set  $\mathcal{Y}$ , loss function  $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ .
  - ▶ e.g.  $\mathcal{X} = \{\text{set of all images}\}$ ,  $\mathcal{Y} = \{\text{cat, dog, neither}\}$ ,  $\ell(y, y') = \begin{cases} 0 & \text{if } y = y' \\ 1 & \text{otherwise} \end{cases}$
- ▶ probability distribution  $D$  on  $\mathcal{X} \times \mathcal{Y}$

## Goal:

- ▶ find a predictor,  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , that works well on future data, i.e. has small *risk*

$$\text{er}(f) = \mathbb{E}_{(x,y) \sim D}(\ell(y, f(x)))$$

## Learning Algorithm: $\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$

- ▶ for example: minimize training error

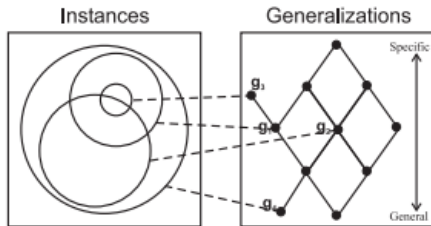
$$\mathcal{L}(f) = \min_{f: \mathcal{X} \rightarrow \mathcal{Y}} \widehat{\text{er}}(f) \quad \text{for} \quad \widehat{\text{er}}(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \ell(y, f(x))$$

# The Need for Biases in Learning Generalizations

Tom M. Mitchell

## 1. Introduction

Learning involves the ability to generalize from past experience in order to deal with new situations that are "related to" this experience. The inductive leap needed to deal with new situations seems to be possible only under certain biases for choosing one generalization of the situation over another. This paper defines precisely the notion of bias in generalization problems, then shows that biases are necessary for the inductive leap. Classes of justifiable biases are considered, and the relationship between bias and domain-independence is considered.



## Traditional Machine Learning

Inductive bias determined by

- ▶ which functions can be represented?  
→ hypothesis class
- ▶ which functions is actually selected?  
→ e.g. regularization

Reasonably well understood

- ▶ lower complexity (e.g. VC-dim)  
means fewer samples are needed

## Deep Machine Learning

Inductive bias determined by

- ▶ which functions can be represented?  
→ model architecture
- ▶ which functions are actually selected?  
→ e.g. regularization, optimization  
algorithm, parameter initialization, ...

Not well understood

- ▶ double descent phenomenon,  
adversarial examples, ...

**ARTICLE**  **Communicated by Steven Nowlan**

## **The Lack of A Priori Distinctions Between Learning Algorithms**

**David H. Wolpert**

*The Santa Fe Institute, 1399 Hyde Park Rd.,  
Santa Fe, NM, 87501, USA*

**This is the first of two papers that use off-training set (OTS) error to investigate the assumption-free relationship between learning algorithms. This first paper discusses the senses in which there are no a priori distinctions between learning algorithms.**

# Meta-Learning

- ▶ **Given:** datasets for multiple related tasks
- ▶ **Goal:** find an inductive bias that works well on future tasks
- ▶ **Open questions:**
  - ▶ What do we mean by "related"?
  - ▶ What do we mean by "work well"?
  - ▶ What is a "future task?"

**Assumption: tasks don't just appear arbitrary.**

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on

## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



## Assumption: tasks don't just appear arbitrary.

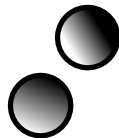
- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on





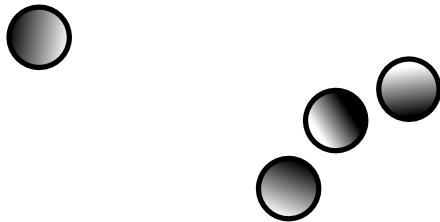
**Assumption: tasks don't just appear arbitrary.**

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



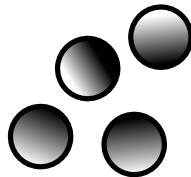
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



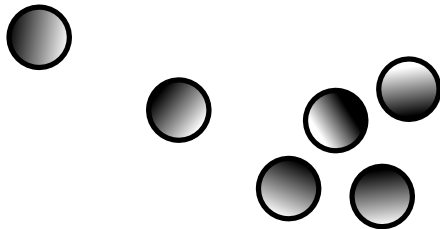
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



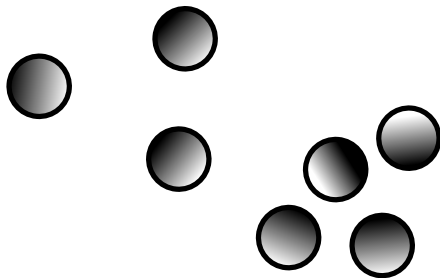
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



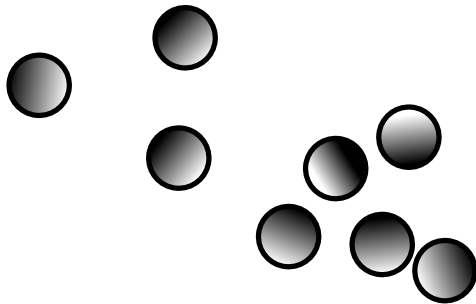
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



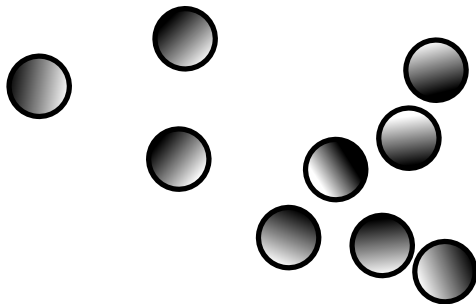
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



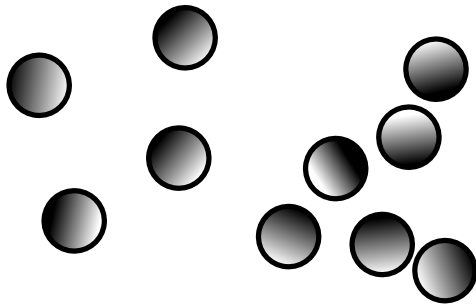
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



## Assumption: tasks don't just appear arbitrary.

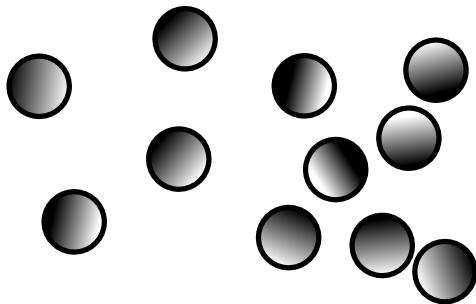
- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on





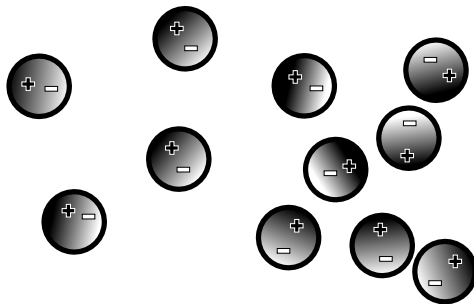
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



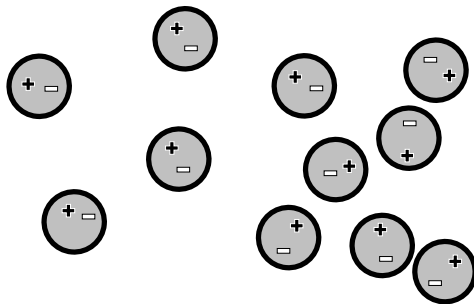
## Assumption: tasks don't just appear arbitrary.

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



**Assumption: tasks don't just appear arbitrary.**

- ▶ there is an **task environment**  $(\mathcal{T}, \tau)$ :
  - ▶  $\mathcal{T}$  is the set of possible tasks,  
 $t = (\mathcal{X}, \mathcal{Y}, \ell, \mathcal{H}, D_t) \in \mathcal{T}$
  - ▶  $\tau$  is a distribution over  $\mathcal{T}$ .
- ▶ observed tasks are samples from the environment,  $t_1, \dots, t_T \stackrel{i.i.d.}{\sim} \tau$ ,
  - ▶ e.g. users who visit a website
- ▶ observed training sets are sampled from task-specific distribution,  $S_t \stackrel{i.i.d.}{\sim} D_t$ .
  - ▶ e.g. which ads the user clicks on



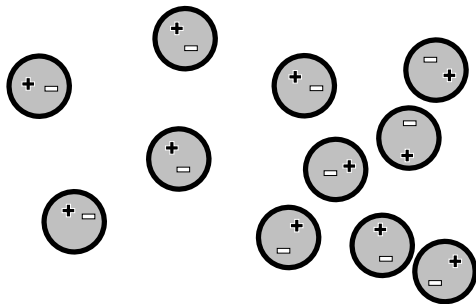
**Meta-learning:**

Use  $S_1, \dots, S_T$  to construct a *learner*

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f : \mathcal{X} \rightarrow \mathcal{Y}\},$$

that is good at learning future tasks from  $(\mathcal{T}, \tau)$ , i.e.

- ▶ new task  $t_* \sim \tau$ , data distribution  $D_*$
- ▶ training set for this task,  $S_* \stackrel{i.i.d.}{\sim} D_*$ ,
- ▶ learner should produce a good predictor, i.e.  $\text{risk}_{\text{er}}(\mathcal{L}(S_*))$  should be small



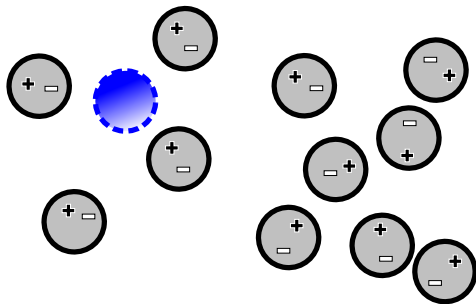
**Meta-learning:**

Use  $S_1, \dots, S_T$  to construct a *learner*

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f : \mathcal{X} \rightarrow \mathcal{Y}\},$$

that is good at learning future tasks from  $(\mathcal{T}, \tau)$ , i.e.

- ▶ new task  $t_* \sim \tau$ , data distribution  $D_*$
- ▶ training set for this task,  $S_* \stackrel{i.i.d.}{\sim} D_*$ ,
- ▶ learner should produce a good predictor, i.e.  $\text{risk}_{\text{er}}(\mathcal{L}(S_*))$  should be small



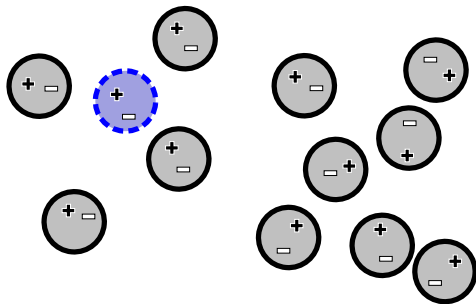
**Meta-learning:**

Use  $S_1, \dots, S_T$  to construct a *learner*

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f : \mathcal{X} \rightarrow \mathcal{Y}\},$$

that is good at learning future tasks from  $(\mathcal{T}, \tau)$ , i.e.

- ▶ new task  $t_* \sim \tau$ , data distribution  $D_*$
- ▶ training set for this task,  $S_* \stackrel{i.i.d.}{\sim} D_*$ ,
- ▶ learner should produce a good predictor, i.e.  $\text{risk}_{\text{er}}(\mathcal{L}(S_*))$  should be small



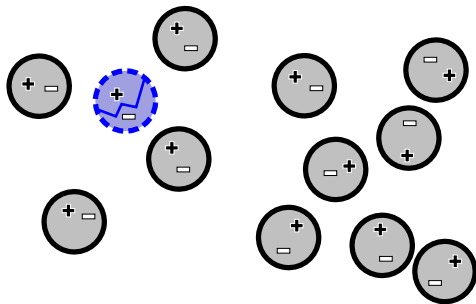
**Meta-learning:**

Use  $S_1, \dots, S_T$  to construct a *learner*

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f : \mathcal{X} \rightarrow \mathcal{Y}\},$$

that is good at learning future tasks from  $(\mathcal{T}, \tau)$ , i.e.

- ▶ new task  $t_* \sim \tau$ , data distribution  $D_*$
- ▶ training set for this task,  $S_* \stackrel{i.i.d.}{\sim} D_*$ ,
- ▶ learner should produce a good predictor, i.e.  $\text{risk}_{\text{er}}(\mathcal{L}(S_*))$  should be small



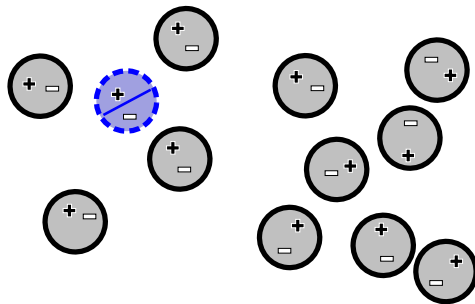
**Meta-learning:**

Use  $S_1, \dots, S_T$  to construct a *learner*

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f : \mathcal{X} \rightarrow \mathcal{Y}\},$$

that is good at learning future tasks from  $(\mathcal{T}, \tau)$ , i.e.

- ▶ new task  $t_* \sim \tau$ , data distribution  $D_*$
- ▶ training set for this task,  $S_* \stackrel{i.i.d.}{\sim} D_*$ ,
- ▶ learner should produce a good predictor, i.e.  $\text{risk}_{\text{er}}(\mathcal{L}(S_*))$  should be small





- ▶ Learners minimize the training error over some hypothesis space:

$$\mathcal{L}_{\mathcal{H}}(S) = \operatorname{argmin}_{f \in \mathcal{H}} \widehat{\mathbf{e}}\mathbf{r}(f; S)$$

- ▶ Learning the learner  $\equiv$  learning the hypothesis space  $\mathcal{H}$

- Learners minimize the training error over some hypothesis space:

$$\mathcal{L}_{\mathcal{H}}(S) = \operatorname{argmin}_{f \in \mathcal{H}} \widehat{\operatorname{er}}(f; S)$$

- Learning the learner  $\equiv$  learning the hypothesis space  $\mathcal{H}$

**Theorem:** Let  $\mathbb{H}$  be a set of hypothesis spaces. If  $\mathbb{H}$  has finite capacity, then it holds uniformly for all  $\mathcal{H} \in \mathbb{H}$ :

$$\mathbb{E}_{S_*}[\operatorname{er}_{D_*}(\mathcal{L}_{\mathcal{H}}(S_*))] \leq \frac{1}{T} \sum_{t=1}^T \widehat{\operatorname{er}}_t(\mathcal{L}_{\mathcal{H}}(S_t)) + O\left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{T}}\right)$$

- ▶ Learners minimize the training error over some hypothesis space:

$$\mathcal{L}_{\mathcal{H}}(S) = \operatorname{argmin}_{f \in \mathcal{H}} \hat{\text{er}}(f; S)$$

- ▶ Learning the learner  $\equiv$  learning the hypothesis space  $\mathcal{H}$

**Theorem:** Let  $\mathbb{H}$  be a set of hypothesis spaces. If  $\mathbb{H}$  has finite capacity, then it holds uniformly for all  $\mathcal{H} \in \mathbb{H}$ :

$$\mathbb{E}_{S_*}[\text{er}_{D_*}(\mathcal{L}_{\mathcal{H}}(S_*))] \leq \frac{1}{T} \sum_{t=1}^T \hat{\text{er}}_t(\mathcal{L}_{\mathcal{H}}(S_t)) + O\left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{T}}\right)$$

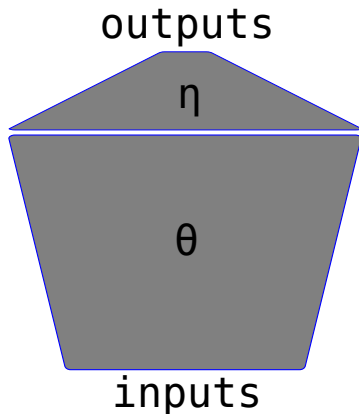
**Theorem provides a principled way to find promising  $\mathcal{H}$**

- ▶ minimize right hand side:  $\operatorname{argmin}_{\mathcal{H} \in \mathbb{H}} \frac{1}{T} \sum_{t=1}^T \min_{f_t \in \mathcal{H}} \hat{\text{er}}(f_t; S_t)$
- ▶  $\mathbb{H}$  encodes a **meta-inductive bias** (or inductive meta-bias?)

**Example Setting: feature learning**

- ▶ fixed neural network architecture:
  - ▶ first  $L$  layers: feature extraction  $\phi_\theta$ ,
  - ▶ remaining  $L'$  layers: classifier  $f_\eta$
- ▶  $\mathcal{H}_\theta = \{ \text{all classifiers } f_\eta \text{ for fixed features } \phi_\theta \}$
- ▶  $\mathbb{H} = \{ \mathcal{H}_\theta : \theta \in \Theta \}$

**Result:** given enough tasks, the quality of a feature map for prior tasks approximates its quality on future tasks



## More modern view of generalization:

- ▶ arbitrary learning algorithm, randomized learning, task-incremental updates

## Standard PAC-Bayesian Setting

- ▶  $P, Q$  distributions over hypothesis space  $\mathcal{H}$
- ▶ prior:  $P$  (set without having seen dataset set  $S$ )
- ▶ posterior:  $Q = \mathcal{A}(P, S)$  for some learning algorithm  $\mathcal{A}$

## PAC-Baysian Lifelong Learning: learn the prior distribution

- ▶ hyper-prior:  $\mathcal{P}$  (set without having data for tasks)
- ▶ hyper-posterior:  $Q$
- ▶ prior for new task is sampled  $P \sim Q$

## Applications:

- ▶ feature learning, biased regularization, learning the initialization, ...

**Theorem (over-simplified): PAC-Bayesian Bound for Meta-Learning**

$$\forall \mathcal{Q} \quad \text{er}_{\mathcal{Q}}(Q_{t_*}) \leq \frac{1}{T} \sum_{t=1}^T \left[ \hat{\text{er}}_{\mathcal{Q}}(Q_t) + \frac{1}{\sqrt{n}} \mathbb{E}_{P \sim \mathcal{Q}} \{ \text{KL}(Q_t \| P) \} \right] \\ + \frac{1}{\sqrt{T}} \text{KL}(\mathcal{Q} \| \mathcal{P}) + \dots$$

**Theorem yields:**

- ▶ principled algorithm for learning  $\mathcal{Q}$ : minimize right hand side
  - ▶ first term: training error on observed tasks
  - ▶ second term: per-task regularizer ( $Q_t = \mathcal{A}(S_t, P_t)$  should be close to  $P_t$ )
  - ▶ third term: meta-regularizer ( $\mathcal{Q}$  should stay close to  $\mathcal{P}$ )
- ▶ insight: meta-learning needs meta-regularization, or meta-overfitting might occur!

When can meta-learning guarantee generalization to future tasks?

## When can meta-learning guarantee generalization to future tasks?

**Main assumptions:** a lot of the same kind of tasks

- ▶ tasks (observed and future) are i.i.d. samples from a task environment  $(\mathcal{T}, \tau)$
- ▶ once we've seen enough, we essentially know what can happen in the future



## When can meta-learning guarantee generalization to future tasks?

**Main assumptions:** a lot of the same kind of tasks

- ▶ tasks (observed and future) are i.i.d. samples from a task environment  $(\mathcal{T}, \tau)$
- ▶ once we've seen enough, we essentially know what can happen in the future

**Sometimes, this task-i.i.d. assumption might be okay**

- ▶ e.g., tasks are customers on a shopping website, ...

**Often, it is not "realistic":**

- ▶ tasks might not be independent, e.g. referral programs, repeating tasks, ...
- ▶ tasks might not be identically distributed, e.g. increasing difficulty.

## Dependent Tasks [A. Pentina, CHL, NeurIPS 2015]

## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



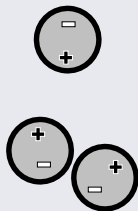
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



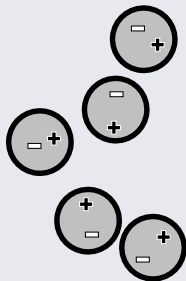
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



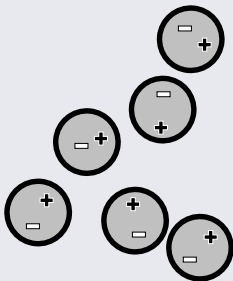
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



## Consistently Changing Tasks [ditto]

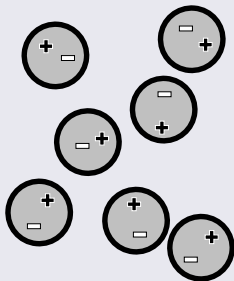
## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



## Consistently Changing Tasks [ditto]

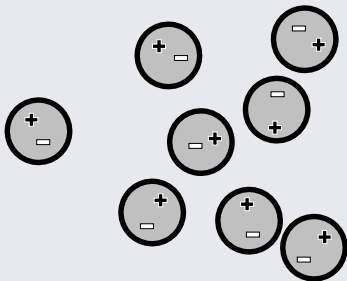


## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



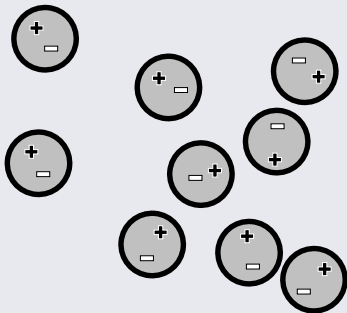
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



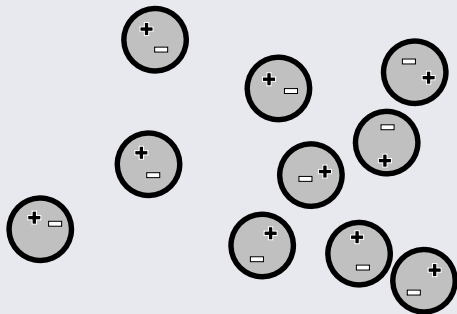
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



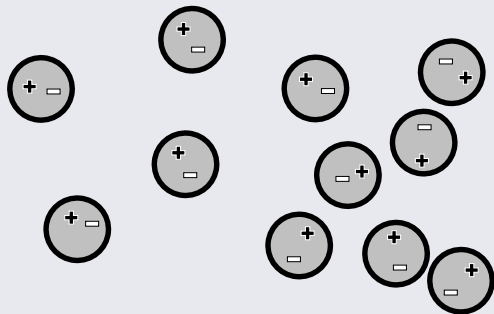
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



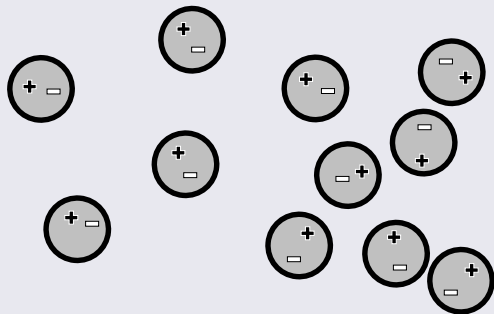
## Consistently Changing Tasks [ditto]

## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



## Consistently Changing Tasks [ditto]

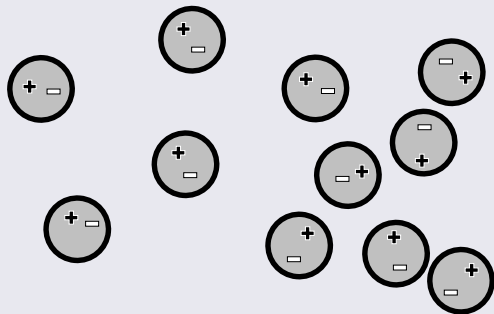
## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



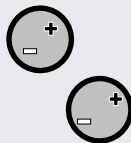
## Consistently Changing Tasks [ditto]



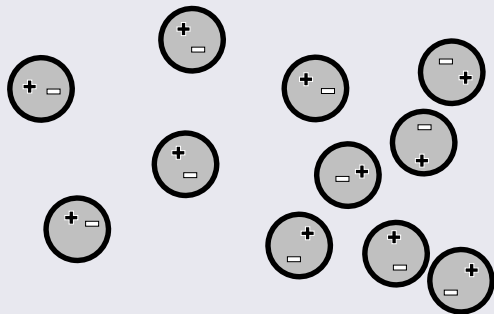
## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



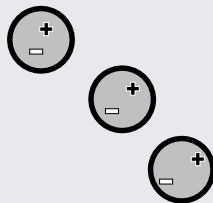
## Consistently Changing Tasks [ditto]



## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]

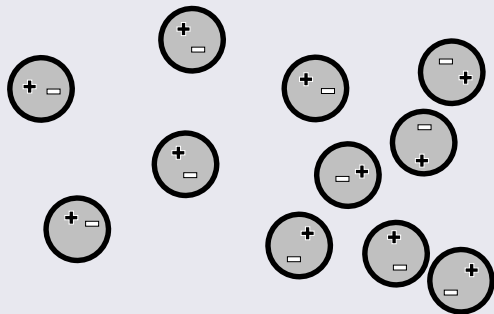


## Consistently Changing Tasks [ditto]

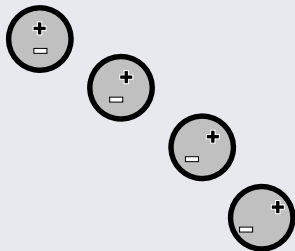




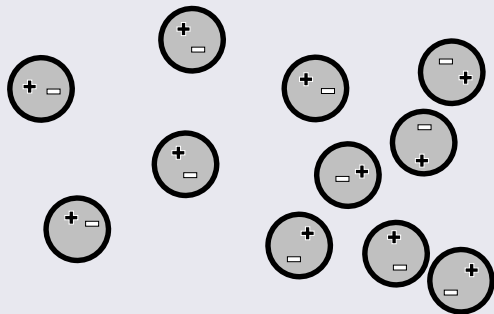
## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



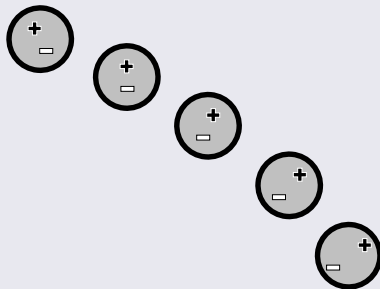
## Consistently Changing Tasks [ditto]



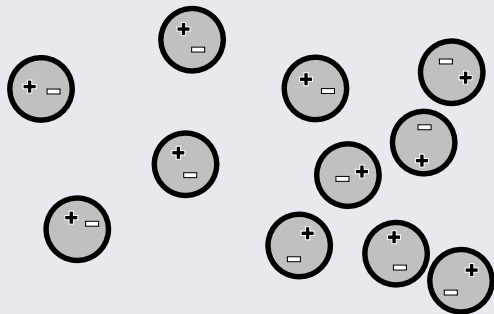
## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



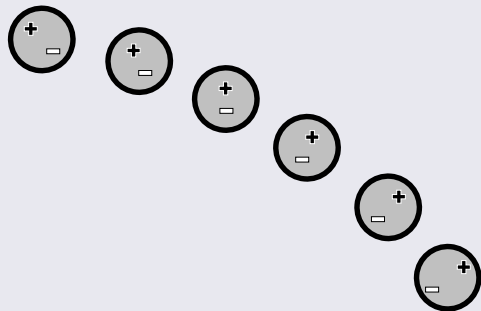
## Consistently Changing Tasks [ditto]



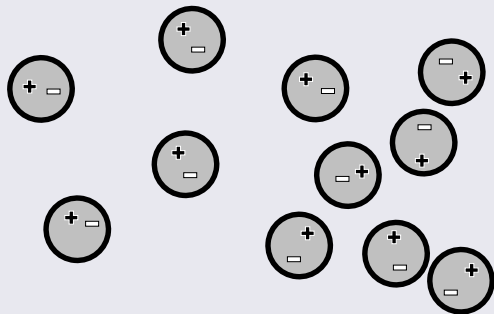
## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



## Consistently Changing Tasks [ditto]



## Dependent Tasks [A. Pentina, CHL. NeurIPS 2015]



## Consistently Changing Tasks [ditto]

But: generalization still based on some *law of large numbers* argument

# Meta-Learning Beyond "More Of The Same"

## Learning a sequence of tasks:

### Given:

- ▶ *sequence of tasks*,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

### Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



### Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

### Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ *sequence of tasks*,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

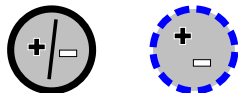
- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$



## Learning a sequence of tasks:



### Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

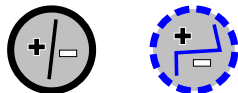
### Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$



## Learning a sequence of tasks:



### Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

### Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

Quantity of interest: average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

Quantity of interest: average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t_t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## Learning a sequence of tasks:



## Given:

- ▶ sequence of tasks,  $t_1, t_2, t_3, \dots$ , with  $t_t = (\mathcal{X}, \mathcal{Y}, \mathcal{H}, \ell, D_t)$ 
  - ▶ potentially open-ended
  - ▶ no assumption how tasks are related to each other

## Goal:

- ▶ learns tasks sequentially: at any time  $t = 1, 2, \dots$ :
  - ▶ learner observes training set  $S_t \stackrel{i.i.d.}{\sim} D_t$  for task  $t$
  - ▶ learner outputs a predictor  $f_t : \mathcal{X} \rightarrow \mathcal{Y}$

**Quantity of interest:** average multi-task risk at any time  $T$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_{D_t}(f_t)$$

## "Tasks without Borders" method

- ▶ train sequentially on all data, ignoring task boundaries
- ▶ continuously update (=finetune) distribution over predictors
- ▶ at task boundaries, construct (randomized) predictor  $Q_t$  by *online-to-batch conversion*

Theorem (simplified) [Zimin, CHL, AMTL 2019]

**Regret-type bound:** for any  $P$  it holds uniformly for any distribution  $Q$  over  $\mathcal{H}$ :

$$\frac{1}{T} \sum_{t=1}^T \text{er}_t(Q_t) - \frac{1}{T} \sum_{t=1}^T \text{er}_t(Q) \leq \frac{1}{\sqrt{nT}} \left( \frac{1}{4} + 2 \text{KL}(Q||P) + \log \frac{2}{\delta} \right)$$

**Insight:** we'll approach or improve over best *fixed* predictor in hindsight.

## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

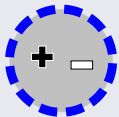
## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

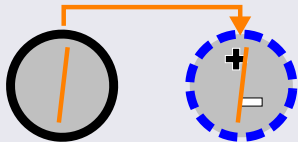
## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

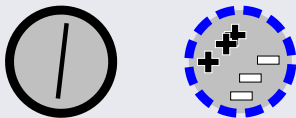
## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]





## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner, NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]





## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Given:

- ▶ sequence of learning tasks (potentially open-ended)
- ▶ for each task, the learner can influence the size of the training set

## Goal:

- ▶ learn classifiers,  $f_t$ , for them in an online fashion

## Quantity of interest:

- ▶ each task's error  $er_t(f_t)$  (not simply the average)

Lifelong learning with weighted majority votes [Pentina, Urner. NeurIPS 2016]



## Theorem (simplified) [Pentina, Urner. NeurIPS 2016]

Assume that each task are *realizable* and the tasks are  $(\gamma, k, \xi)$ -related with  $\gamma < \epsilon/4$  and  $k\xi < \epsilon/8$ . Then the incremental learning algorithm learns classifiers  $f_1, \dots, f_T$  such that

- ▶ the error on each task is bounded:  $\text{er}_t(g_t) \leq \epsilon$  for  $t = 1, \dots, T$ ,
- ▶ the total number of labeled samples is at most  $\tilde{O}\left(\frac{nk + \text{VC}(\mathcal{H})k^3}{\epsilon}\right)$ .

**The theorem yields:** insight which conditions should be fulfilled

- ▶ realizability: each task can -in principle- be solved perfectly given enough data
- ▶ a measure of relatedness between tasks
- ▶ trade-off between task-relatedness, complexity of the hypothesis class, and required number of labeled samples

## Can we choose a beneficial order in which to learn tasks?

## Given:

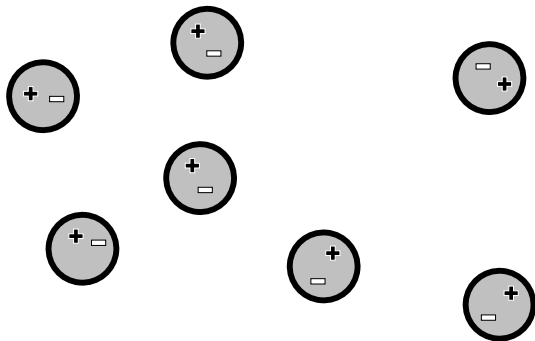
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

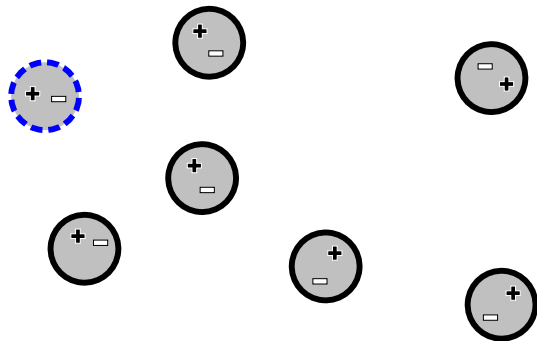
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

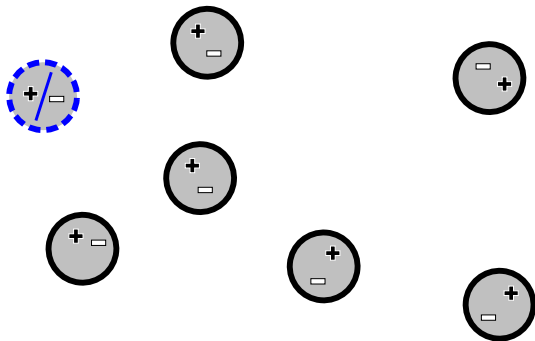
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

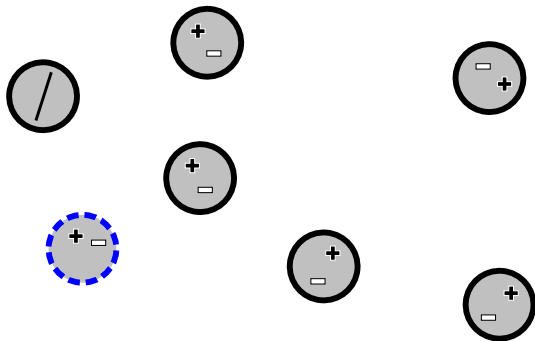
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$





Can we chose a beneficial order in which to learn tasks?

Given:

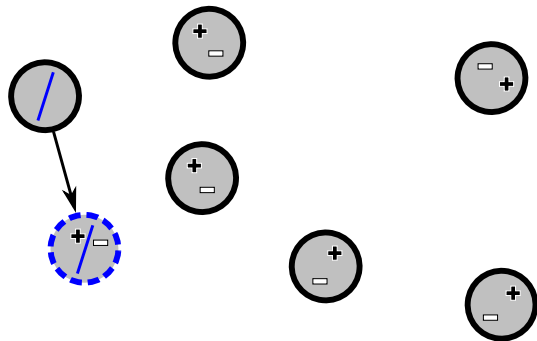
- ▶ datasets for  $T$  tasks

Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

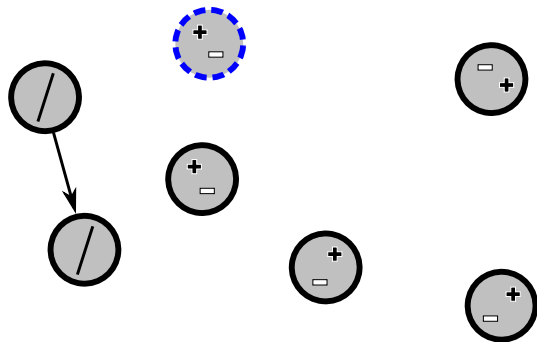
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



Can we chose a beneficial order in which to learn tasks?

Given:

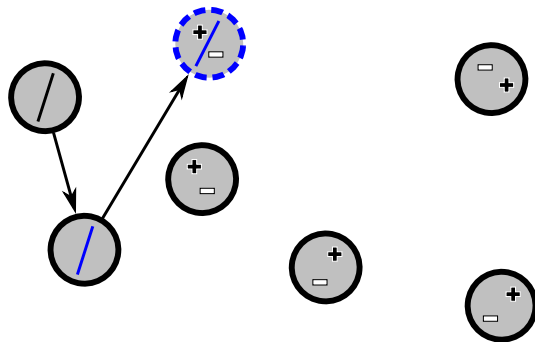
- ▶ datasets for  $T$  tasks

Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

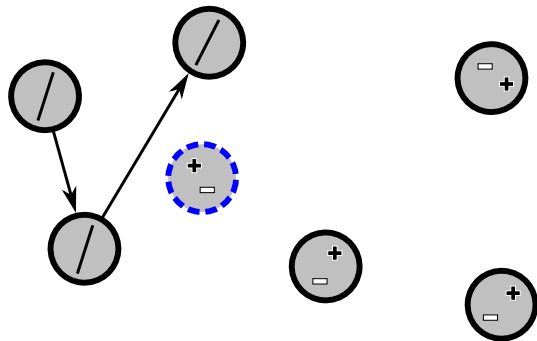
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

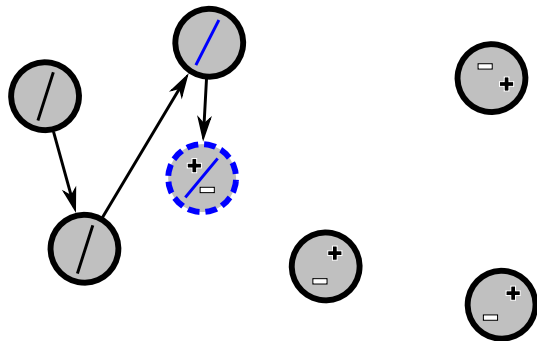
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

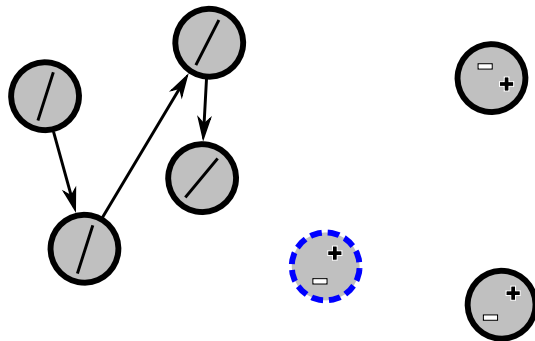
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

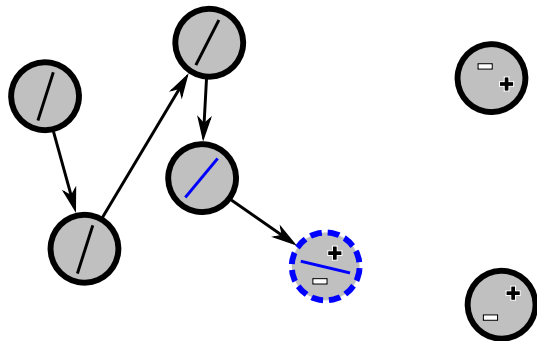
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

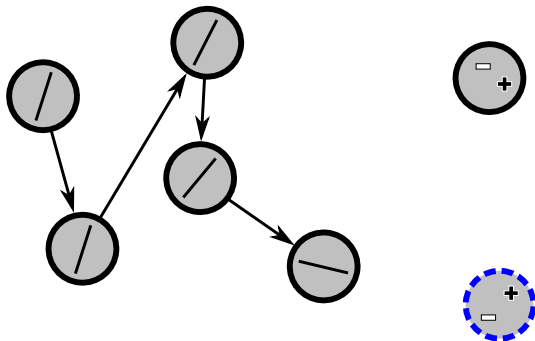
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$





## Can we choose a beneficial order in which to learn tasks?

## Given:

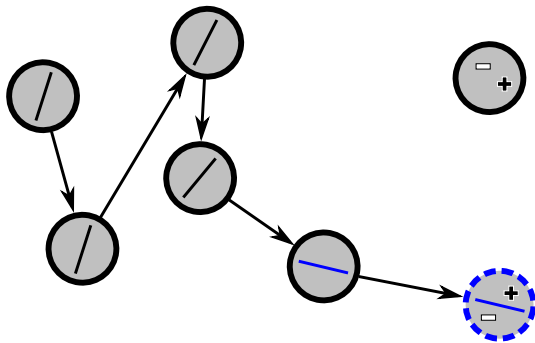
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



## Can we choose a beneficial order in which to learn tasks?

## Given:

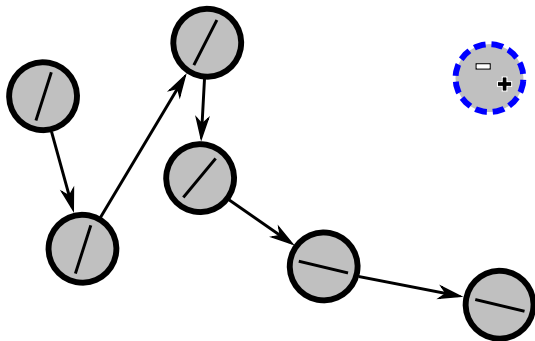
- ▶ datasets for  $T$  tasks

## Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

## Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



Can we chose a beneficial order in which to learn tasks?

Given:

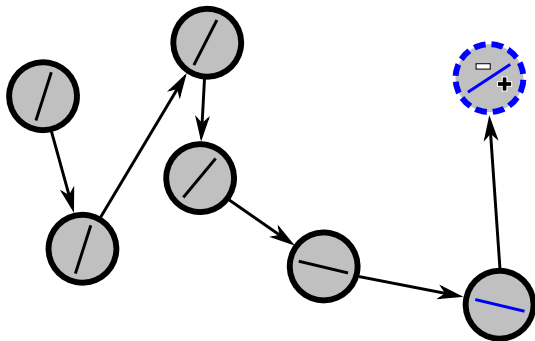
- ▶ datasets for  $T$  tasks

Goal:

- ▶ learn classifiers  $f_t$  for them  
in an arbitrary sequential order

Quantity of interest:

- ▶ average multi-task error  $\frac{1}{T} \sum_{t=1}^T \text{er}_t(f_t)$



Theorem (simplified, for linear classifiers  $f_t = \text{sign}[w_t^\top x]$ )

It holds uniformly over all weight vectors  $w_1, \dots, w_T$  any task orders,  $\pi$ , that:

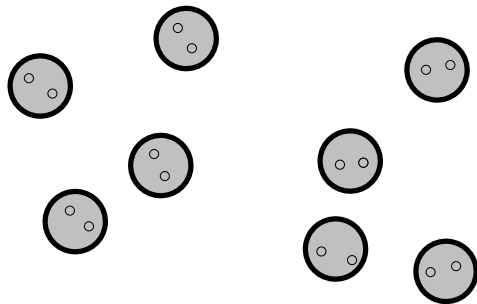
$$\frac{1}{T} \sum_{t=1}^T \text{er}_t(w_t) \leq \frac{1}{T} \sum_{t=1}^T \tilde{\text{er}}(w_t) + \frac{1}{\sqrt{m}} \sum_{t=1}^T \|w_{\pi(t)} - w_{\pi(t-1)}\|^2 + O\left(\frac{\log T}{\sqrt{m}}\right)$$

**The theorem yields:**

- a principled learning algorithm how to choose the order and learn classifiers  
→ minimize right hand side w.r.t.  $\pi$  and  $w_1, \dots, w_T$

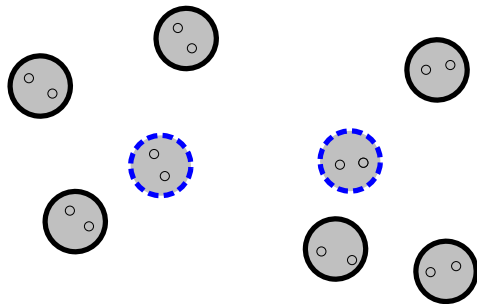
## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)



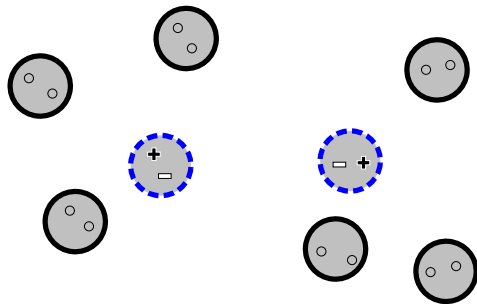
## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)



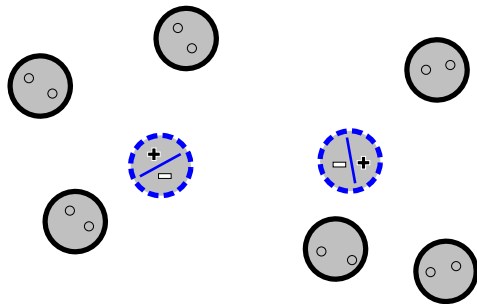
## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)



## Multi-task setting:

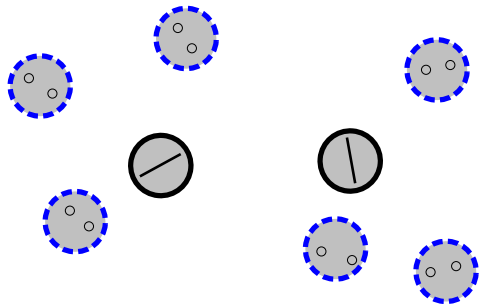
- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)





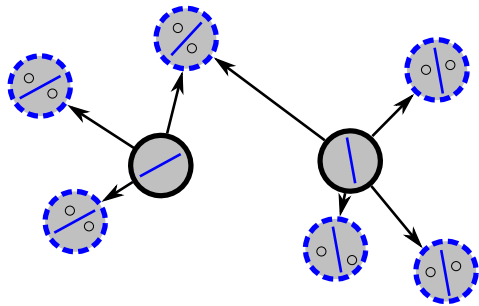
## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)



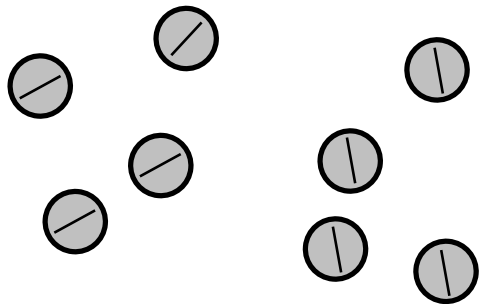
## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)



## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

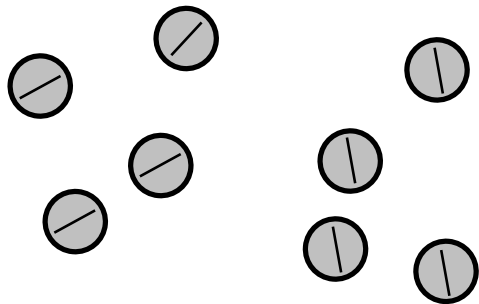


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.

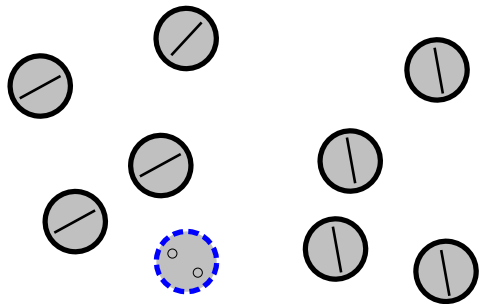


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.

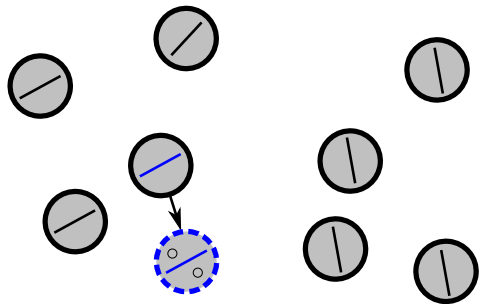


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.

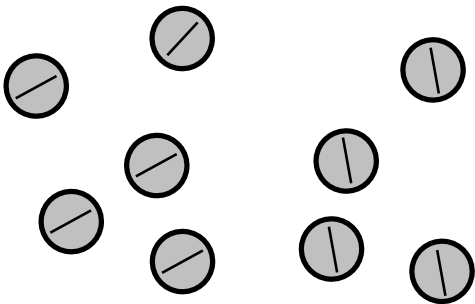


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.

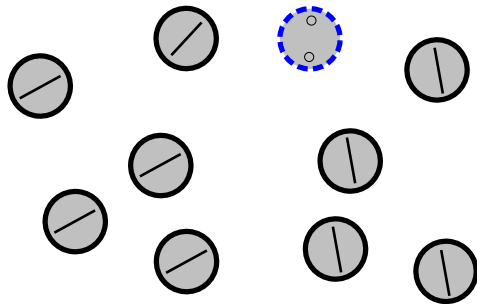


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.



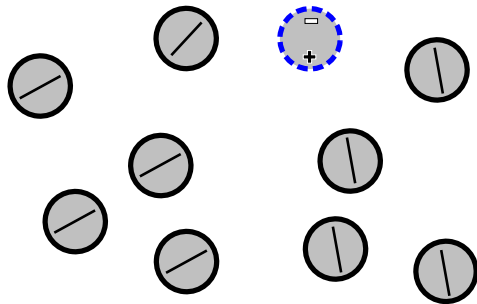


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.

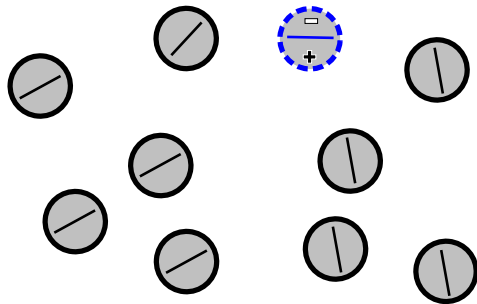


## Multi-task setting:

- ▶ initially given: **unlabeled datasets**
- ▶ the learning algorithm can pick a subset of tasks to be labeled
- ▶ classifiers for other tasks must be inferred (e.g. domain adaptation)

## Lifelong setting:

- ▶ new unlabeled tasks appear sequentially
- ▶ in each case: the learner can either
  - ▶ infer a classifier based on the previous tasks,
  - ▶ or, it must ask for labels.



## Theorem (variant of [Pentina, CHL. 2016]; notation simplified)

Let  $\alpha = (\alpha_s^t)_{s,t=1,\dots,T} \geq 0$  denote the amount of influence of task  $s$  (=source) on task  $t$  (=target). Provided that the choice of  $\alpha$  depends only on the unlabeled data, then the following inequality holds uniformly for all possible choices of  $\alpha$  and  $f_1, \dots, f_T \in \mathcal{H}$ :

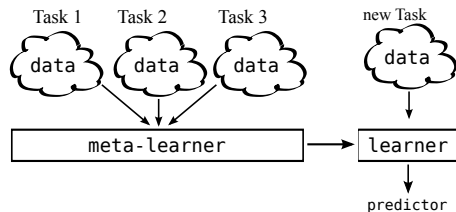
$$\sum_{t=1}^T \text{er}_t(f_t) \leq \sum_{t=1}^T \sum_{s=1}^T \alpha_s^t \widehat{\text{er}}_s(f_t) + \sum_{t=1}^T \sum_{s=1}^T \alpha_s^t \text{disc}(S_s, S_t) + c_1 \|\alpha\|_{2,1} + c_2 \|\alpha\|_{1,2} + \dots$$

Note: tasks  $s$  with  $\alpha_s^* = (0, \dots, 0)$  need no labeled data.

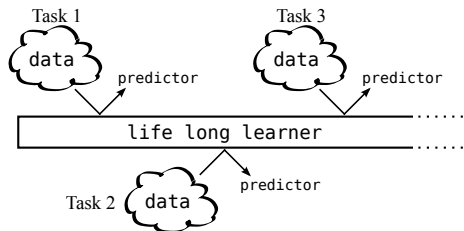
## The theorem yields:

- ▶ insight which similarity measure to use between tasks
- ▶ a principled learning algorithm, i.e. how to choose  $\alpha$  and  $f_1, \dots, f_T$
- ▶ insight which conditions should be fulfilled for the algorithm to work

We looked at two scenarios for continual/lifelong learning:



Meta-Learning



Lifelong Learning

Outcome:

- ▶ a clearer picture what one actually wants, what is possible under what assumptions,
  - ▶ meta-learning: task are sampled from an environment, strong generalization possible
  - ▶ online multi-task learning: tasks can be arbitrary, typically weaker guarantees
- ▶ some algorithms with some theoretical guarantees,
- ▶ many open questions.

## My research group and alumni:



Paul Henderson



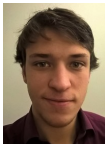
Niko Konstantinov



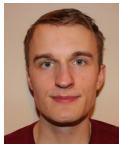
Alex Peste



Mary Phuong



Bernd Prach



Jonny Scott



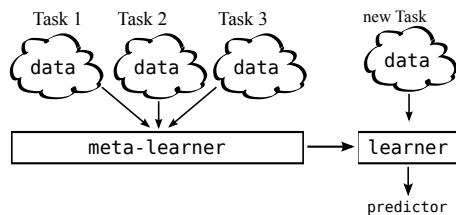
Asya Pentina



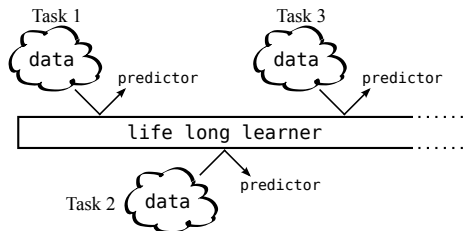
Alex Zimin

## Funding Sources:





Meta-Learning



Lifelong Learning

Questions? → chl@ist.ac.at