Discovering General-purpose Active Learning Strategies

Ksenia Konyushkova *
CVLab, EPFL

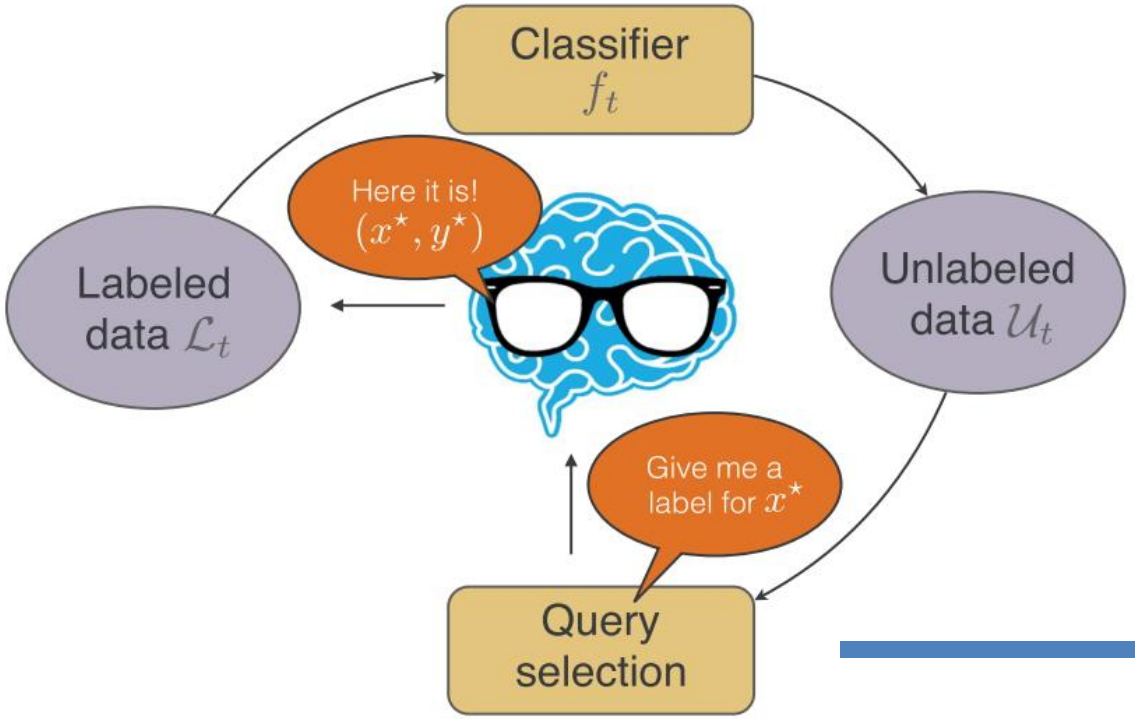
Raphael Sznitman,
University of Bern

Pascal Fua
CVLab, EPFL

Contents

- Introduction
- RLAL
  - Formulating AL as An MDP
  - Policy Learning Using RL
- Experiments

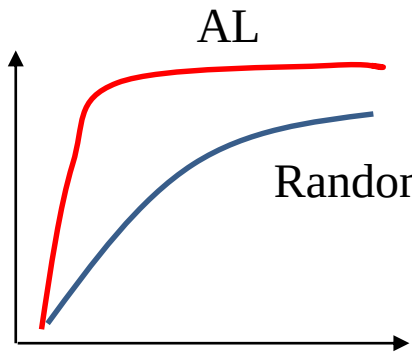
Introduction



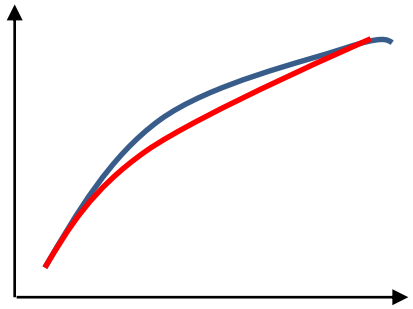
Uncertainty
Density
Diversity
…

hand-designed

Can only be applied to specific applications.



Dataset-1



Dataset-2

Data-driven

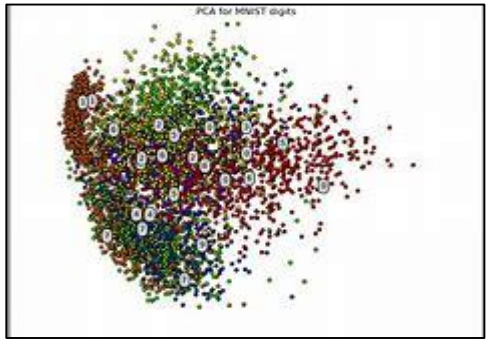
Existing data-driven AL methods have some limitations:

- learn from closely related domains (Bachman et al., 2017)
- use a greedy selection that may be suboptimal (Konyushkova et al., 2017)
- rely on properties of specific classifiers (Konyushkova et al., 2017)

There is still no general-purpose non-myopic methods that depend neither on the kind **data** nor on the specific ML **model** used in training.

Purpose

- **transferable**

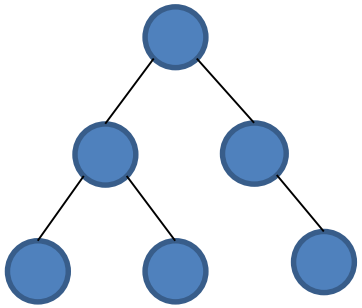


MNIST

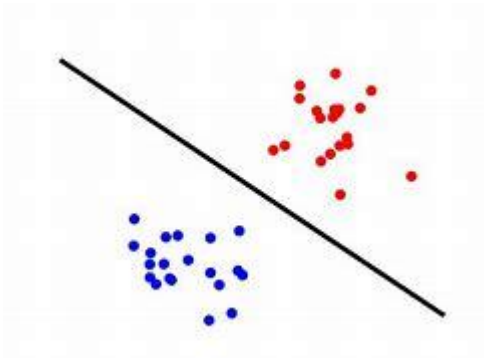
CIFAR-10

Should be **transferable** across unrelated datasets

- **flexibility**



Decision tree



SVM

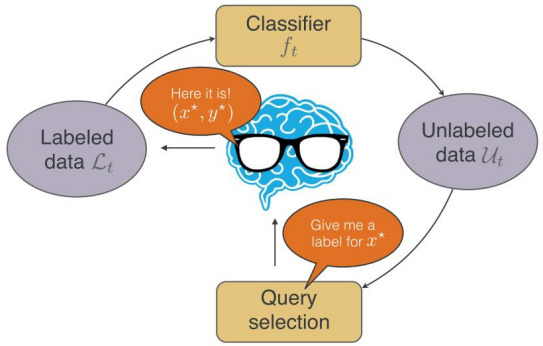
Have sufficient **flexibility** to be applied in different ML models

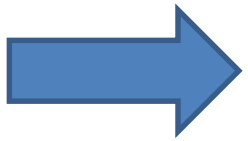
Contents

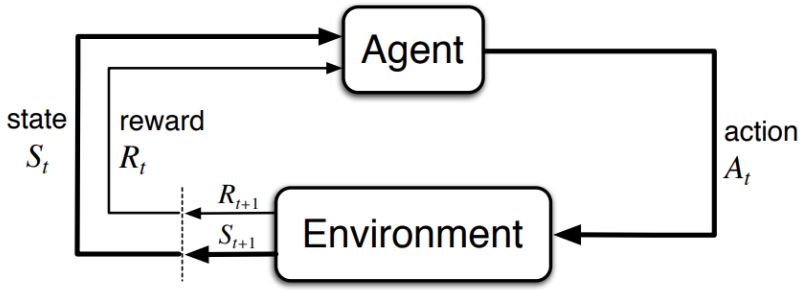
- Introduction
- RLAL
  - Formulating AL as An MDP
  - Policy Learning Using RL
- Experiments

Approach

Reformulate AL as a Markov Decision Process (MDP)

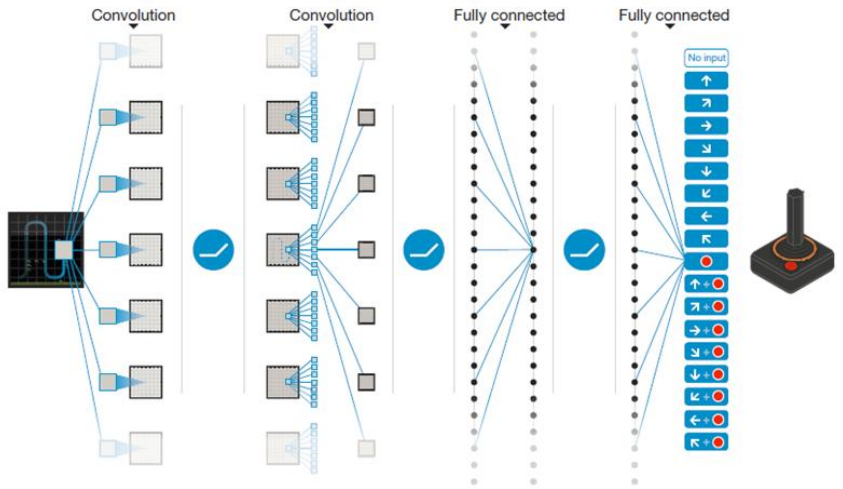






Use reinforcement learning (RL) to find AL strategy as an optimal MDP policy.

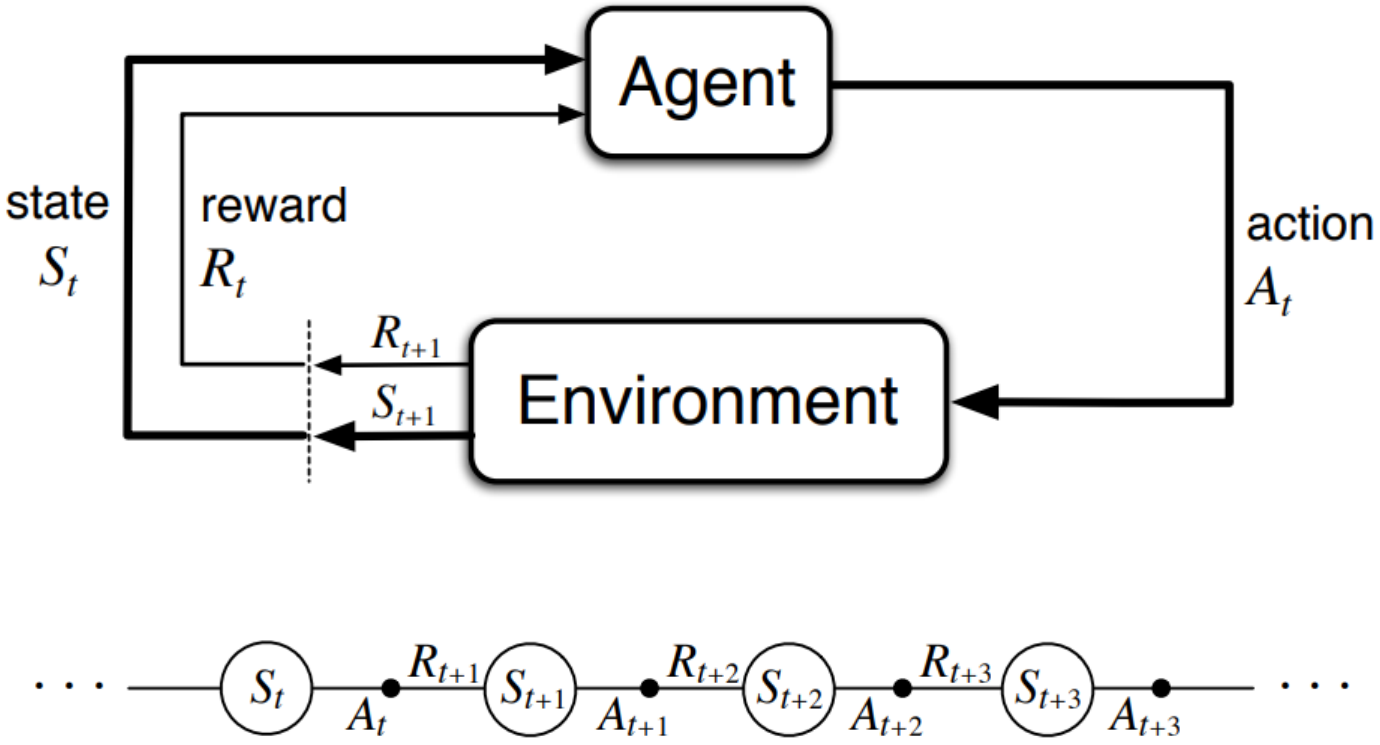
DQN



Contents

- Introduction
- RLAL
  - Formulating AL as An MDP
  - Policy Learning Using RL
- Experiments

MDP


Agent
state
S_t
reward
R_t
action
A_t
R_{t+1}
S_{t+1}
Environment
S_t
A_t
R_{t+1}
S_{t+1}
A_{t+1}
R_{t+2}
S_{t+2}
A_{t+2}
R_{t+3}
S_{t+3}
A_{t+3}

Formulating AL as An MDP

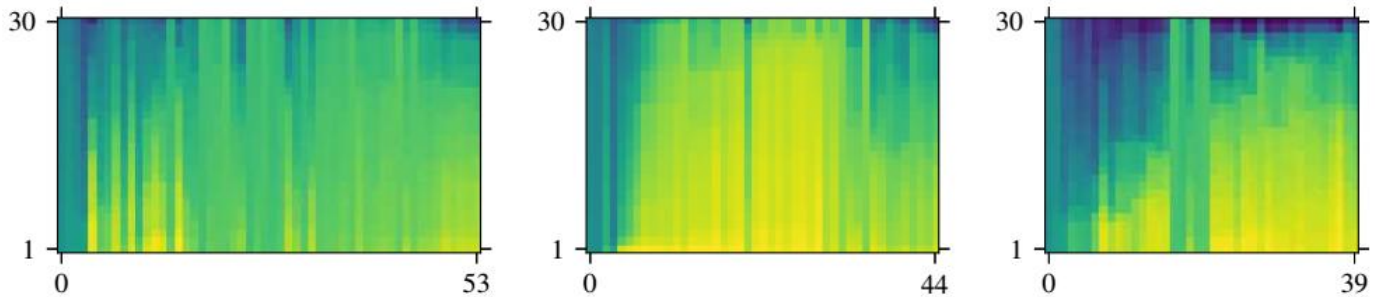
1. Train a classifier f_t using $\mathcal{L}_t$.
2. A *state* s_t is characterised by f_t, $\mathcal{L}_t$, and $\mathcal{U}_t$.
3. The AL *agent* selects an *action* $a_t \in \mathcal{A}_k$ by following a policy $\pi : s_t \mapsto a_t$ that defines a datapoint $\boldsymbol{x}^{(t)} \in \mathcal{U}_t$ to be annotated.
4. Look up the label $y^{(t)}$ of $\boldsymbol{x}^{(t)}$ in $\mathcal{D}$ and set $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{(\boldsymbol{x}^{(t)}, y^{(t)})\}, \mathcal{U}_{t+1} = \mathcal{U}_t \setminus \{\boldsymbol{x}^{(t)}\}$.
5. Give the agent the *reward* r_{t+1} linked to empirical performance value ℓ_t.

How to define ***state***, ***action*** and ***reward*** ?

State

States It only makes sense to perform AL when there is a lot of unlabelled data. Without loss of generality, we can therefore set aside at the start of each AL run a subset $\mathcal{V} \subset \mathcal{U}_0$ and replace $\mathcal{U}_0$ by $\mathcal{U}_0 \setminus \mathcal{V}$. We use the classifier's score $\hat{y}_t$ on $\mathcal{V}$ as a means to keep track of the state of the learning procedure. Then, we take the state representation to be a vector $\boldsymbol{s}_t$ of sorted values $\hat{y}_t(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i$ in $\mathcal{V}$.

$$s_t = sort(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_v)$$



(a) Random (b) Uncertainty (c) Learnt strategy

Figure 1: The evolution of the learning state vector $\boldsymbol{s}_t$ during an annotation episode starting from the same state for (a) random sampling, (b) uncertainty sampling, and (c) our learnt strategy. Every column represents $\boldsymbol{s}_t$ at iteration t, with $|\mathcal{V}| = 30$. Yellow corresponds to values of $\hat{y}_t$ that predict class 1 and blue – class 0.

Action

taking an action a_t ⟺ selecting a data point $x^{(t)}$ to be annotated

the current classifier f_t on x_i

$$\boldsymbol{a}_i = [\hat{y}_t(\boldsymbol{x}_i), g(\boldsymbol{x}_i, \mathcal{L}_t), g(\boldsymbol{x}_i, \mathcal{U}_t)] \qquad \boldsymbol{x}_i \in \mathcal{U}_t$$

$$g(\boldsymbol{x}_i, \mathcal{L}_t) = \sum_{x_j \in \mathcal{L}_t} d(\boldsymbol{x}_i, x_j)/|\mathcal{L}_t|$$

$$g(\boldsymbol{x}_i, \mathcal{U}_t) = \sum_{x_j \in \mathcal{U}_t} d(\boldsymbol{x}_i, x_j)]/|\mathcal{U}_t|$$

d is a distance measure

Reward

Goal of AL:

reach the quality q in as few iterations as possible

Goal of RL:

maximize cumulative reward

Reward for every iteration: $r_t = -1$

Cumulative reward R: $r_1 + \ldots + r_{T-1} = -T + 1$

The fewer iterations, the larger the reward.

Contents

- Introduction
- RLAL
  - Formulating AL as An MDP
  - Policy Learning Using RL
- Experiments

Policy Learning Using RL

$Q^{\pi}(s_t, a_i)$ aims to predict $-(T - t)$ # remaining steps before reaching the target quality

Procedure To account for the diversity of AL experiences we use a collection of Z annotated datasets $\{\mathcal{Z}_i\}_{1 \leq i \leq Z}$ to simulate AL *episodes*. We start from a random policy π. Then, learning is performed by repeating the following steps:

1. Pick a labelled dataset $\mathcal{Z} \in \{\mathcal{Z}_i\}$ and split it into subsets $\mathcal{D}$ and $\mathcal{D}'$.
2. Use π to simulate AL episodes on $\mathcal{Z}$ by initially hiding the labels in $\mathcal{D}$ and following an MDP as described in Sec. 3.2. Keep the *experience* in the form of transitions $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_{t+1}, \boldsymbol{s}_{t+1})$.
3. Update policy π according to the *experience* with the DQN update rule.

DQN Implementation

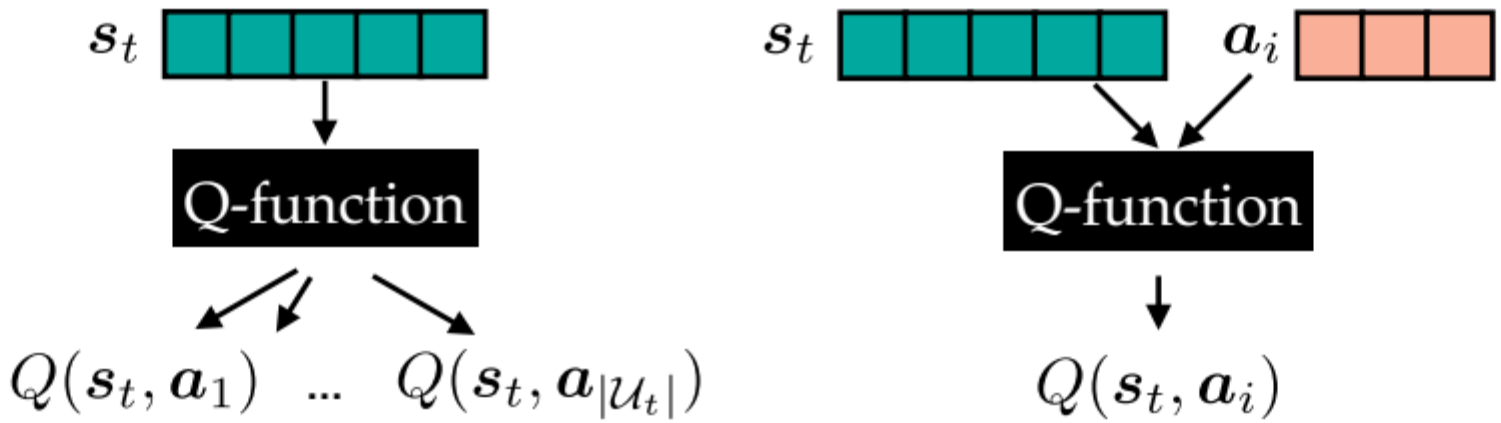


Figure 2: Adapting the DQN architecture. Left: In standard DQN, the Q-function takes the state vector as input and yields an output for each discrete action. Right: In our version, actions are represented by vectors. The Q-function takes action and the state as input and returns a single value.

Contents

- Introduction
- RLAL
  - Formulating AL as An MDP
  - Policy Learning Using RL
- Experiments

Experiments

Baselines

Rs, random sampling
Us, uncertainty sampling
QUIRE, informativeness and representativeness
ALBE, a recent meta-AL algorithm that adaptively combines strategies, including Us, Rs and QUIRE.
LAL-ind, formulates AL as a regression task and learns a greedy strategy that is transferable between datasets.
LAL-iter, a variation of LAL-ind that tries to better account for the bias caused by AL selection.
MLP-GAL, learns a strategy from multiple datasets with a policy gradient RL method

Transferability

Datasets:
0-*adult*, 1-*australian*, 2-*breast cancer*, 3-*diabetes*, 4-*flare solar*, 5-*heart*, 6-*german*, 7-*mushrooms*, 8-*waveform*, 9-*wdbc*

- Evaluation: the average number of annotations required to achieve the desired target accuracy
- Use LogReg and ran 500 trials where AL episodes run up to 100 iterations
- Leave-one-out: training on 8 of the datasets selected from number 1 to number 9, and evaluating on the remaining one

Scenario / **Dataset**	test 0	leave-one-out 1	2	3	4	5	6	7	8	9
Rs	*50.78*	25.31	25.65	30.33	15.57	44.83	20.80	42.81	45.28	19.36
Us	41.83	**13.53**	27.07	*27.84*	15.50	37.1	**15.60**	**15.6**	23.83	**7.25**
QUIRE	58.33	30.02	33.33	37.12	**9.02**	57.58	20.30	42.9	*36.49*	15.45
ALBE	55.66	29.79	31.84	33.62	10.91	50.71	21.02	39.12	41.23	16.16
LAL-ind	59.39	*20.88*	20.85	**26.63**	15.31	44.14	*18.16*	*24.15*	39.13	*11.22*
LAL-iter	63.29	*20.24*	21.79	28.03	14.84	*40.38*	19.90	25.2	36.97	*10.39*
OURS	**37.52**	**14.15**	**18.79**	**26.77**	*14.67*	**32.16**	**15.06**	21.94	**20.91**	**7.09**
notransf	—	15.01	16.14	24.40	—	23.26	14.65	16.47	18.06	7.14

Table 1: Average number of annotations required to reach a predefined quality level.

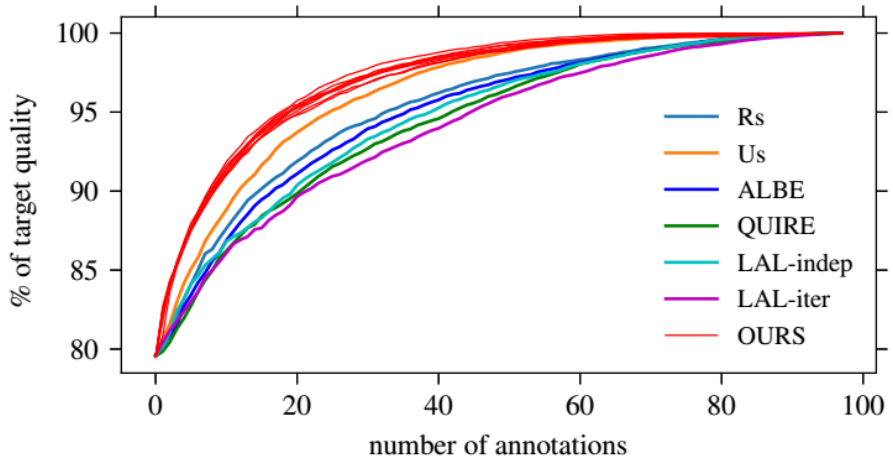


Figure 5: Performance of all the strategies on 0-*adult* dataset.

target quality q: 98% of the maximum quality of the classifier trained on 100 randomly drawn data points

Flexibility

Use an SVM instead of LogReg

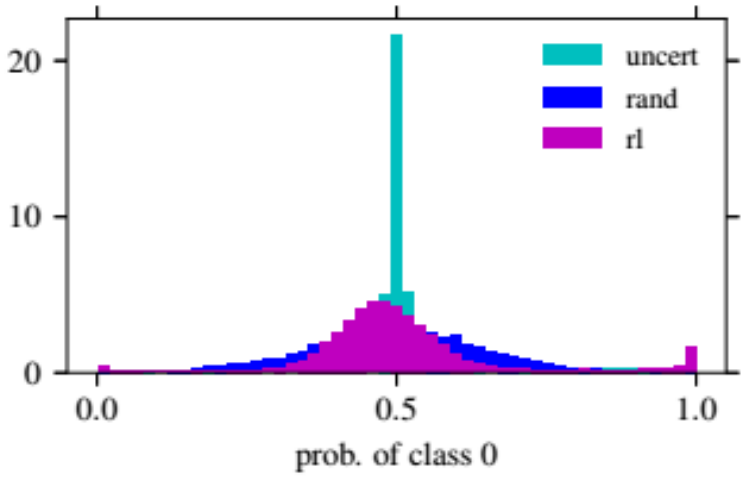
Baseline	Rs	Us	OURS
LogReg-100	32.07	−28.80%	**−34.71%**
LogReg-200	80.06	−29.61%	**−39.96%**
LogReg-500	51.59	−31.49%	**−37.75%**
SVM	30.87	−7.81%	**−28.35%**

Table 2: Increasing the number of annotations still using logistic regression (first three rows) and using SVM instead of logistic regression as the base classifier (fourth row). We report the average number of annotations required using **Rs** and the percentage saved by either **Us** or **OURS**.

Results reaching 98% of the quality of a classifier trained with 200 and 500 random data points

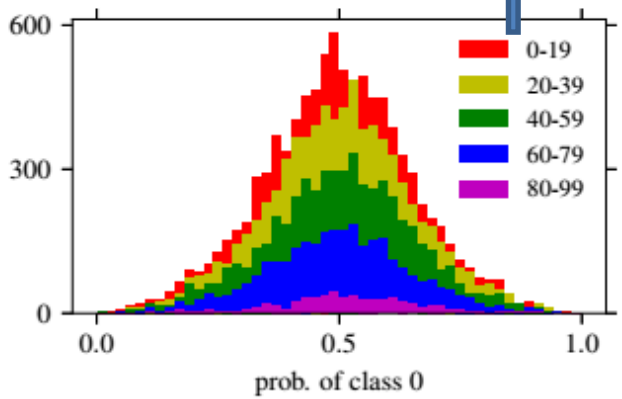
What do we select?

$$p_t = p(y_t = 0|x_t)$$

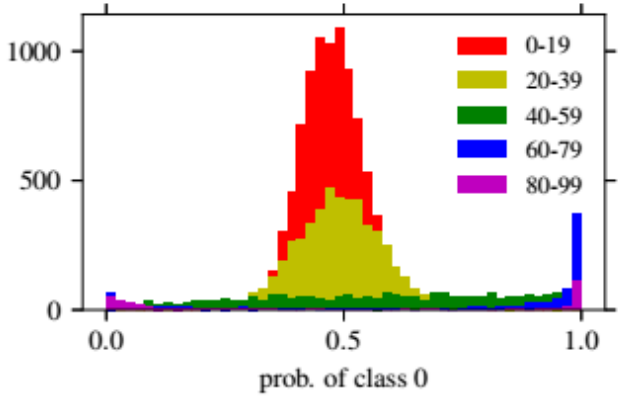


time intervals

Structured behavior



(b) **Rs**



(c) **OURS**

(b) Evolution over time for random.
(c) Evolution over time for **OURS**.

Transfer or not?

OURS-notransfer: learn on one-half of a dataset and transfer to the other half

Scenario	test	leave-one-out								
Dataset	0	1	2	3	4	5	6	7	8	9
Rs	*50.78*	25.31	25.65	30.33	15.57	44.83	20.80	42.81	45.28	19.36
Us	41.83	**13.53**	27.07	*27.84*	15.50	37.1	**15.60**	**15.6**	23.83	**7.25**
QUIRE	58.33	30.02	33.33	37.12	**9.02**	57.58	20.30	42.9	*36.49*	15.45
ALBE	55.66	29.79	31.84	33.62	10.91	50.71	21.02	39.12	41.23	16.16
LAL-ind	59.39	*20.88*	20.85	**26.63**	15.31	44.14	*18.16*	*24.15*	39.13	*11.22*
LAL-iter	63.29	*20.24*	21.79	28.03	14.84	*40.38*	19.90	25.2	36.97	*10.39*
OURS	**37.52**	**14.15**	**18.79**	**26.77**	*14.67*	**32.16**	**15.06**	21.94	**20.91**	**7.09**
notransf	—	15.01	16.14	24.40	—	23.26	14.65	16.47	18.06	7.14

Table 1: Average number of annotations required to reach a predefined quality level.

Learn the strategy on dataset 1 and test it on datasets 2-9. The success rate drops to around 40% on average.
Using multiple datasets is important !

Conclusion

- Presented a data-driven approach to AL that is transferable and flexible.
- Reformulate AL as a Markov Decision Process (MDP) and use reinforcement learning (RL) to find AL strategy as an optimal MDP policy.
- The resulting AL strategies outperform state-of-the-art approaches.