Data-efficient learning, in general and in LLM preference tuning

Danica Sutherland University of British Columbia (Vancouver) and Amii (she/her)

based on:

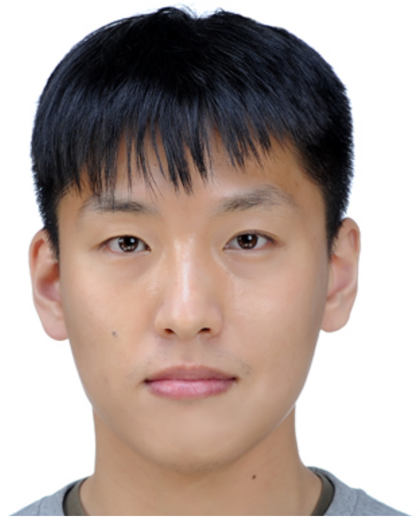
1. **Generalized Coverage for More Robust Low-Budget Active Learning** (ECCV 2024; arXiv:2407.12212)
2. **Uncertainty Herding: One Active Learning Method for All Label Budgets** (ICLR 2025; arXiv:2412.20644)
3. **Rethinking Selective Annotation for In-Context Learning in LLMs** (in submission, not online yet)
4. **Learning Dynamics of LLM Finetuning** (ICLR 2025; arXiv:2407.10490)

with:

Wonho Bae
UBC
(1 2 3)

Junhyug Noh
Ewha Womans University
(1)

Gabriel Oliveira
Borealis AI
(2)



Mingyu Kim
UBC
(3)

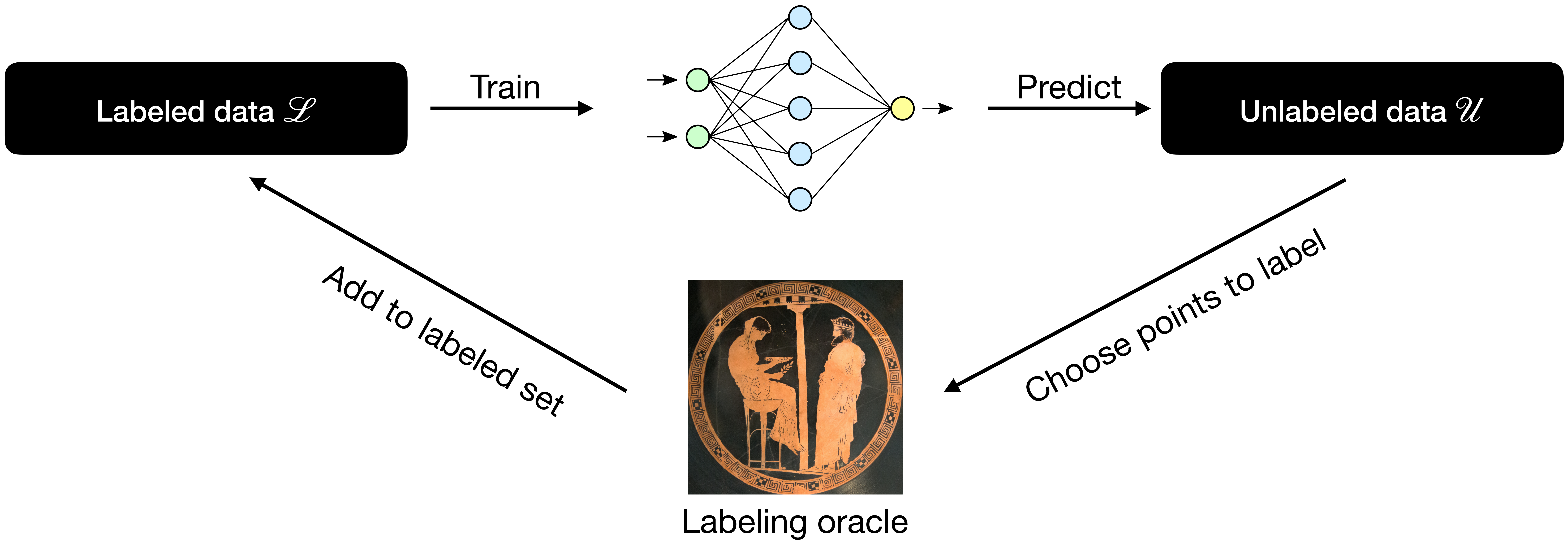
Hamed Shirzad
UBC
(3)

Yi (Joshua) Ren
UBC
(4)

Snowflake, February 2025

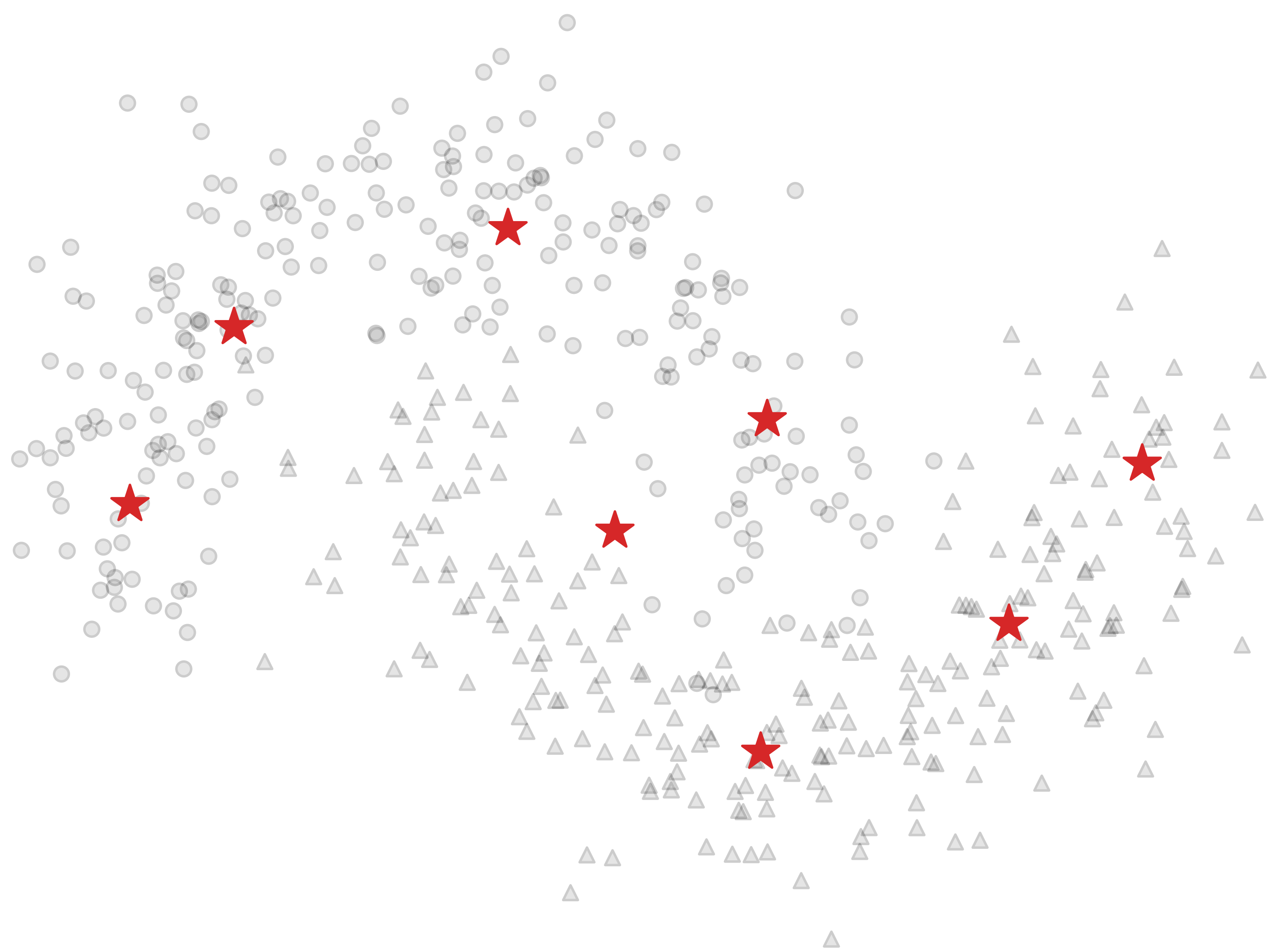
Active learning

- Data is everywhere!
    - …but maybe not cleanly labeled data
    - …that's relevant to the particular task we'd like to learn
- **Pool-based** active learning:

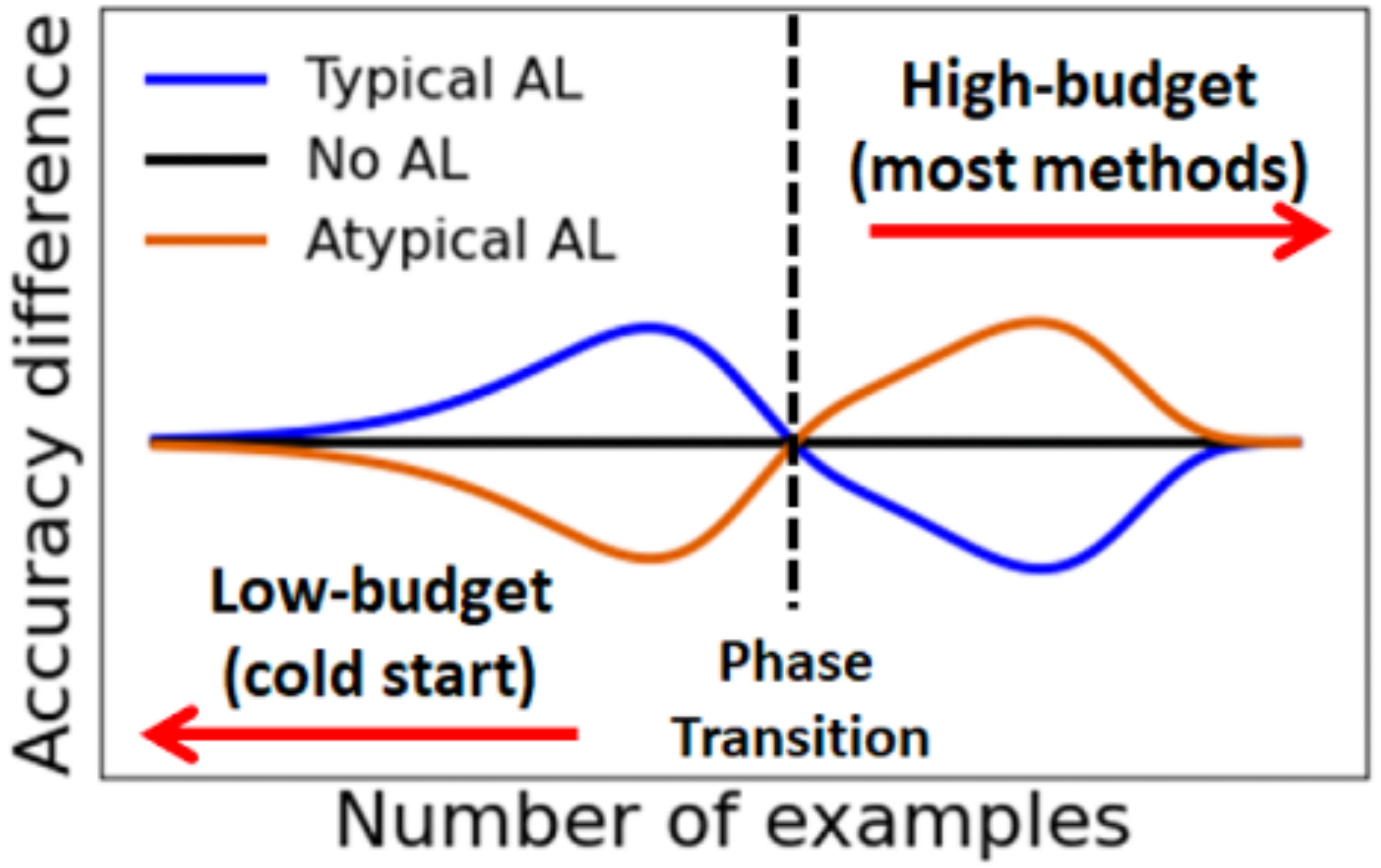


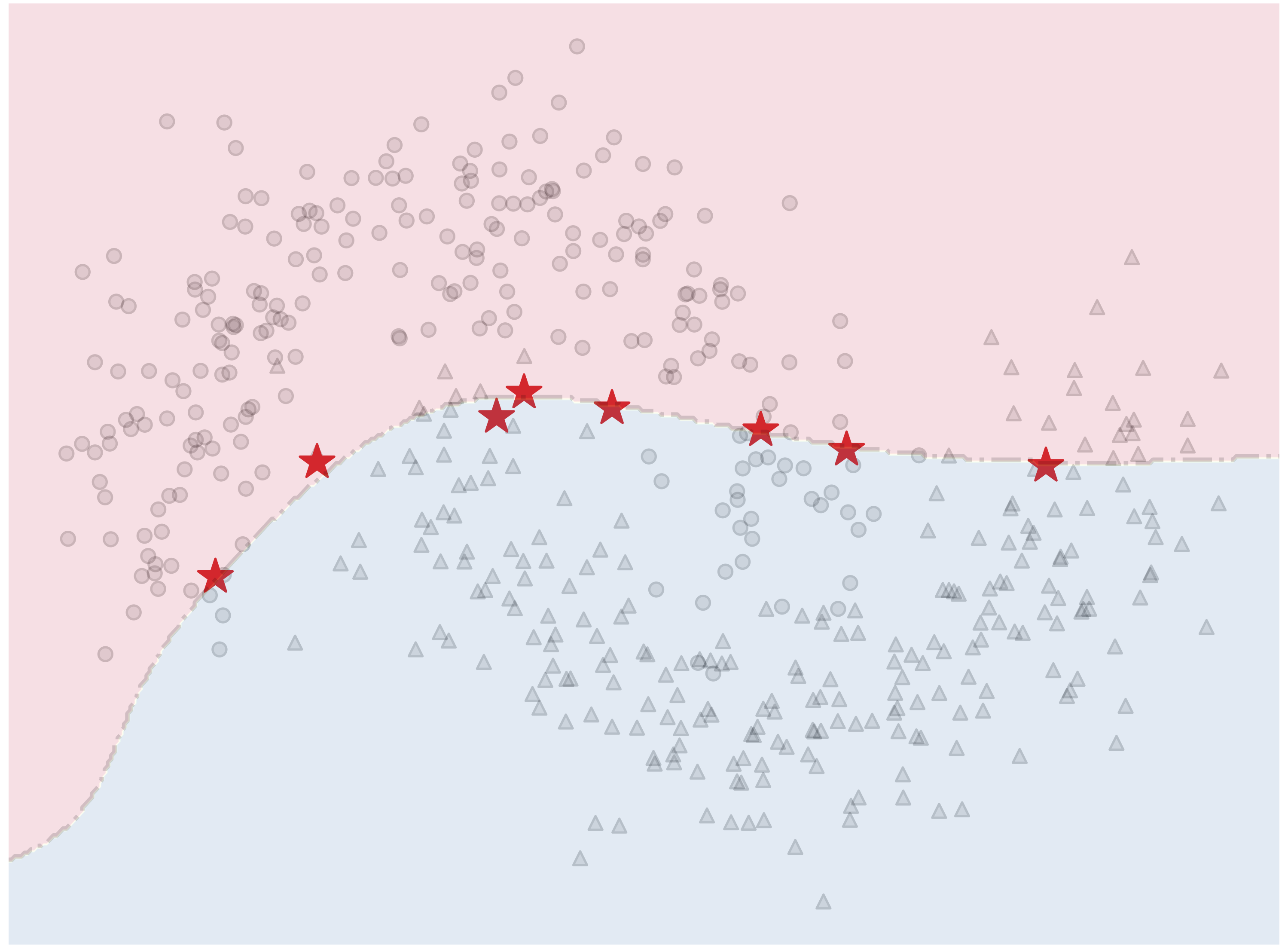
Selection criteria

- The key question: which points should we choose for labeling?



Recent approaches:
points that are representative
of the distribution

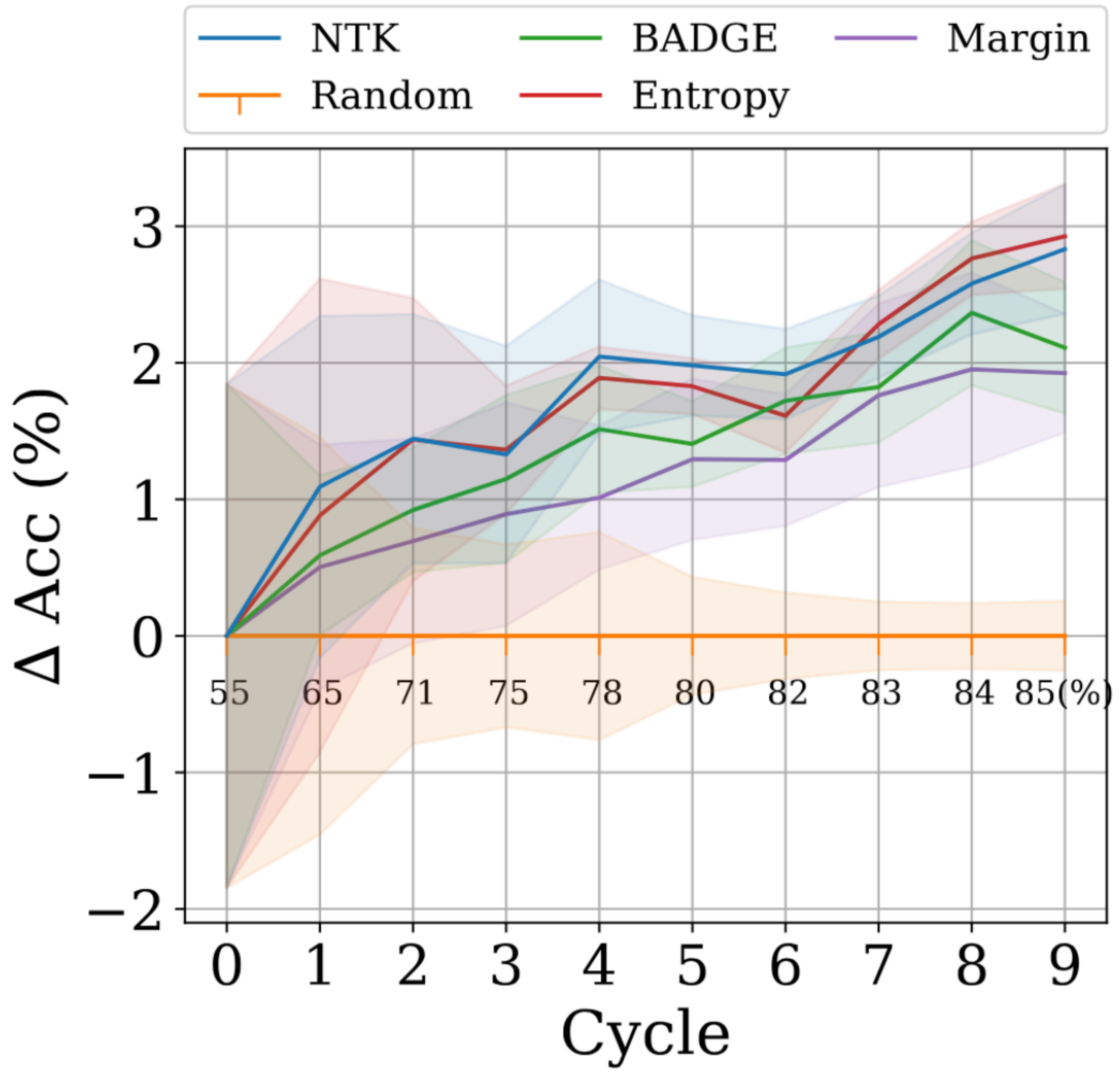




Most approaches:
points that are uncertain
for the current predictor

Uncertainty-based selection

- Myopic selection: $\arg\max_{\tilde{x}\in\mathcal{U}} U(\tilde{x}; f_{\text{current}})$
  - Margin selection: simple baseline that's usually almost best
    $U(\tilde{x}; f) = p_f(\text{most likely class for } \tilde{x}) - p_f(\text{second most likely class for } \tilde{x})$



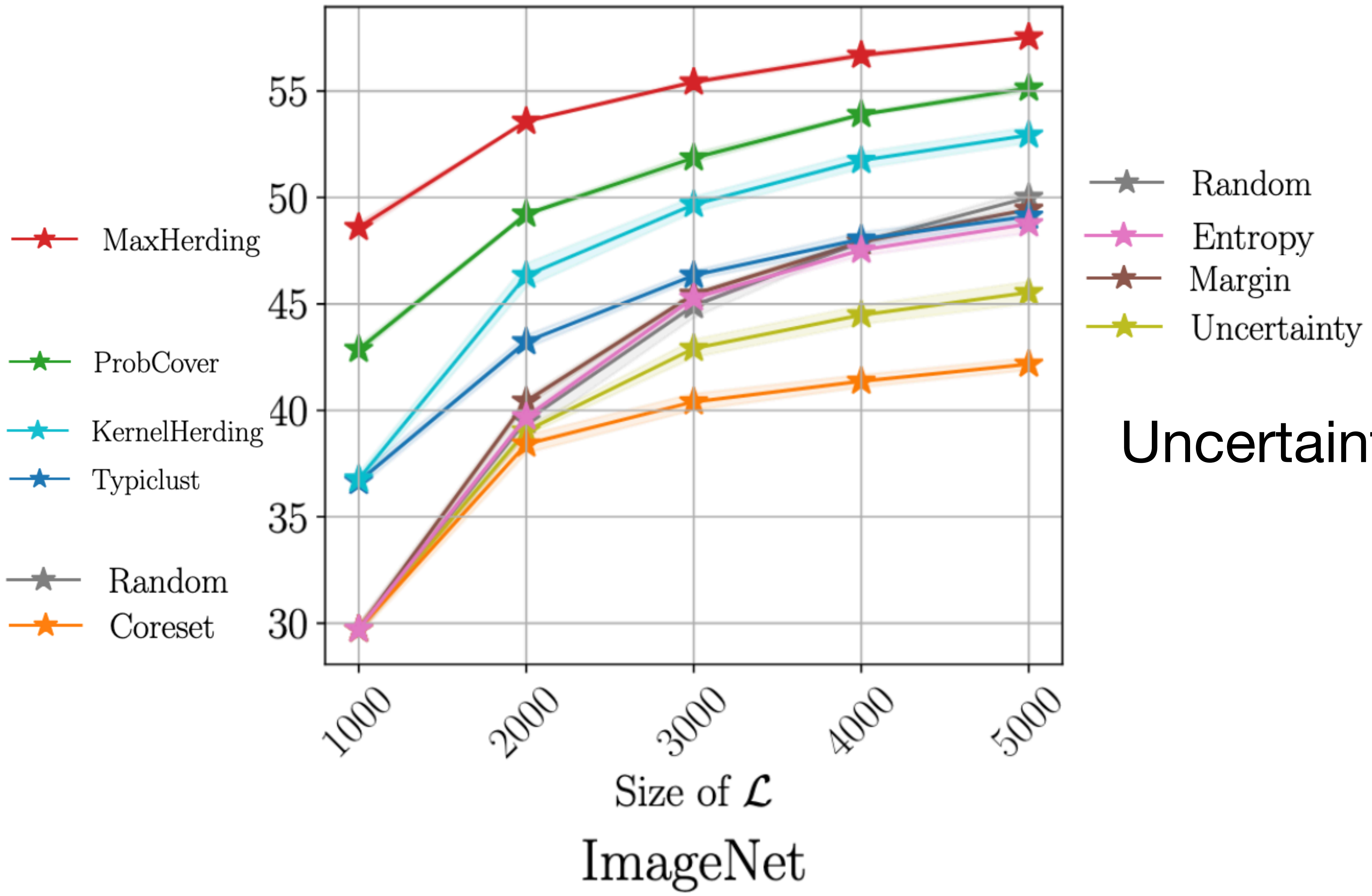
(b) CIFAR10: 2-layer WideResNet

(from our NeurIPS-22 paper)

Low-budget setting

- Very early in training, predictor f_{current} is useless
  - Most active learning papers start with a big batch of random points
  - Early on, uncertainty selection $\leq$ random selection

Representation-based methods



Uncertainty-based methods

Representation methods: ProbCover

Active Learning Through a Covering Lens

Ofer Yehuda[†], Avihu Dekel[†], Guy Hacohen[†‡], Daphna Weinshall[†]

- Motivation: accuracy of a nearest-neighbour classifier on $\mathscr{L}$

$$\Pr_x\left(\hat{f}_{\mathscr{L}}(x) \text{ is wrong}\right)$$

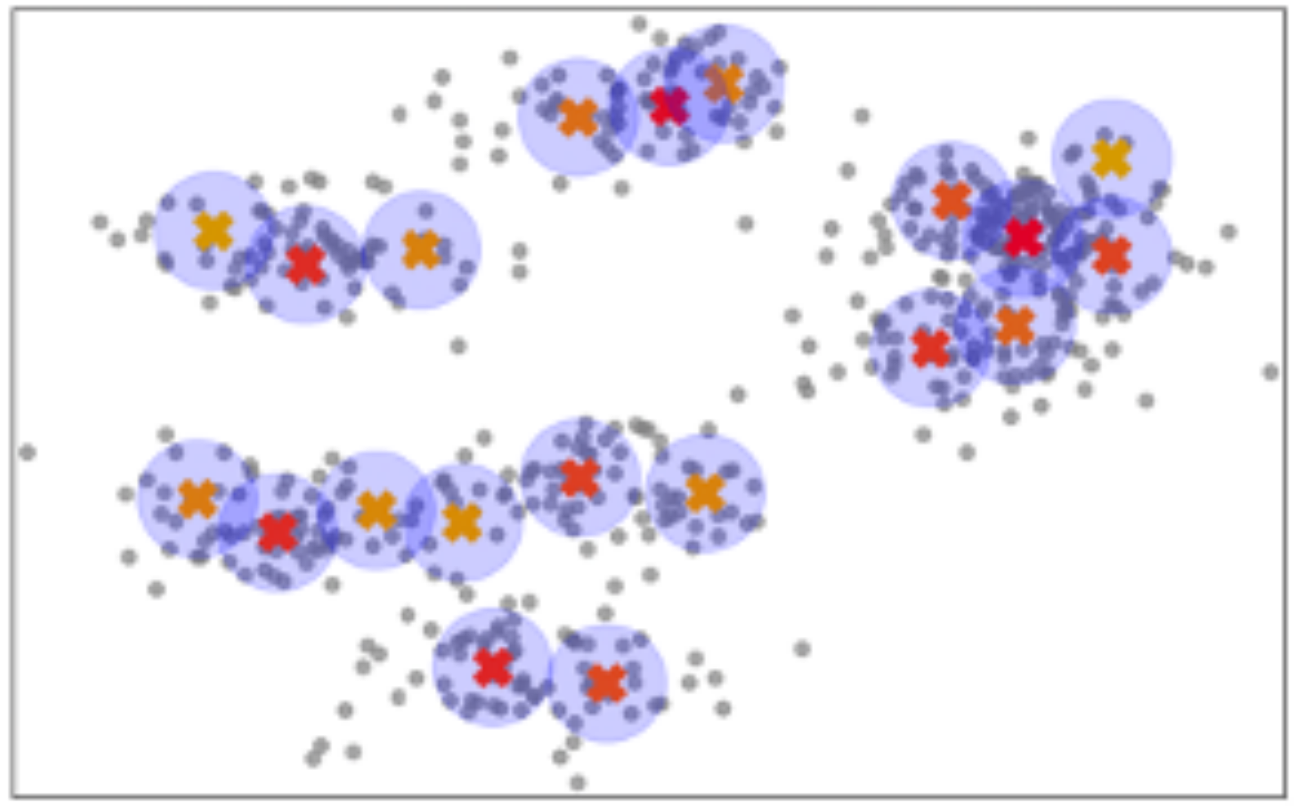
all distances in self-supervised feature space (SimCLR, DINO)

$$\leq \Pr_x\left(\mathrm{NN}_{\mathscr{L}}(x) \text{ is far from } x\right) + \Pr_x\left(\text{nearby } \mathrm{NN}_{\mathscr{L}}(x) \text{ has different label than } x\right)$$

$$\leq \left(1 - \Pr_x\left(\exists x' \in \mathscr{L} \text{ s.t. } \|x - x'\| \leq \delta\right)\right) + \Pr_x\left(\forall x' \text{ s.t. } \|x - x'\| \leq \delta, \quad f^*(x) = f(x')\right)$$

probabilistic coverage (no labels!)

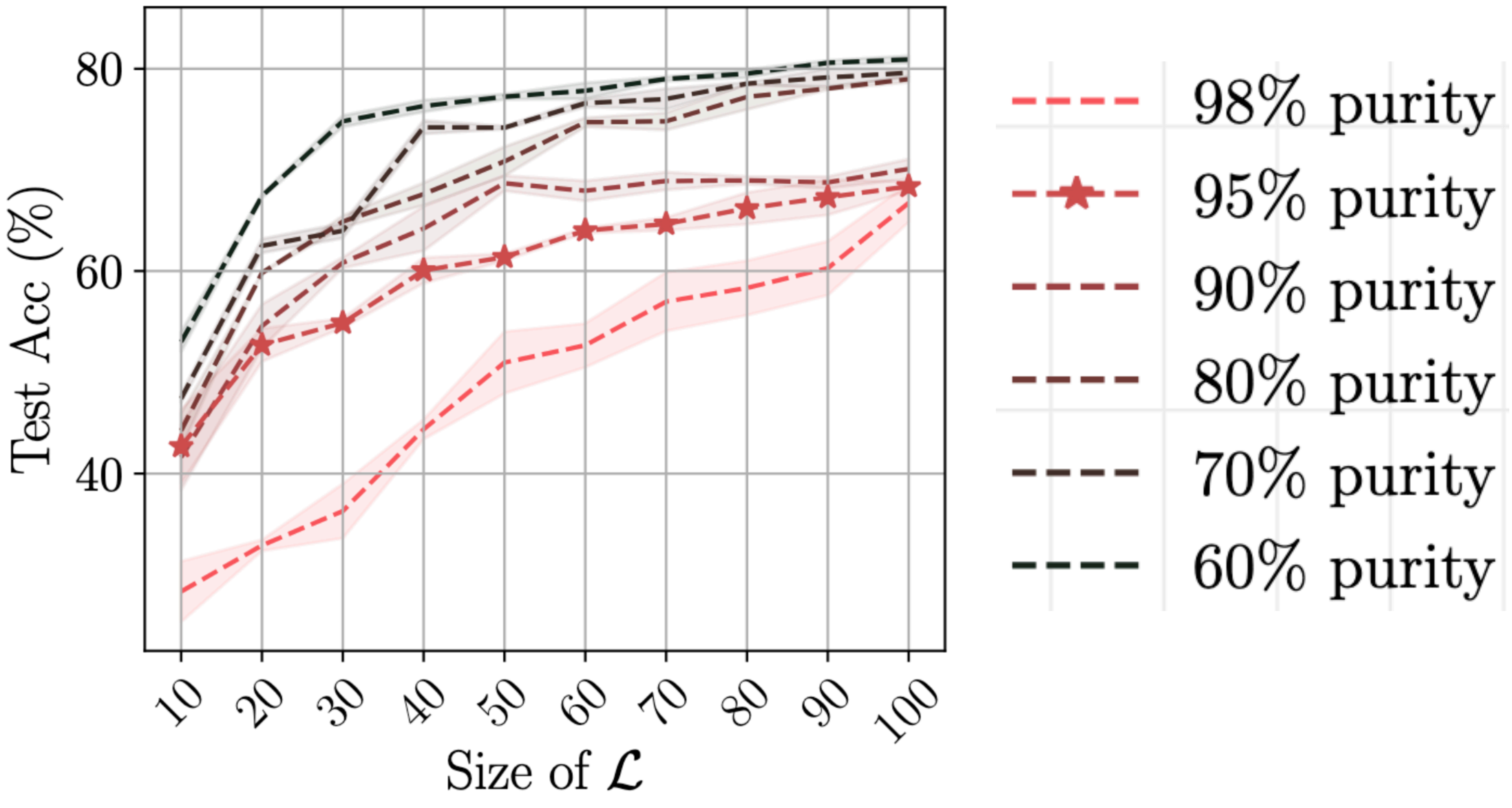
impurity (requires labels)



- Approach: choose δ small enough that impurity is small, then choose $\mathscr{L}$ to greedily maximize the coverage

The problem with ProbCover

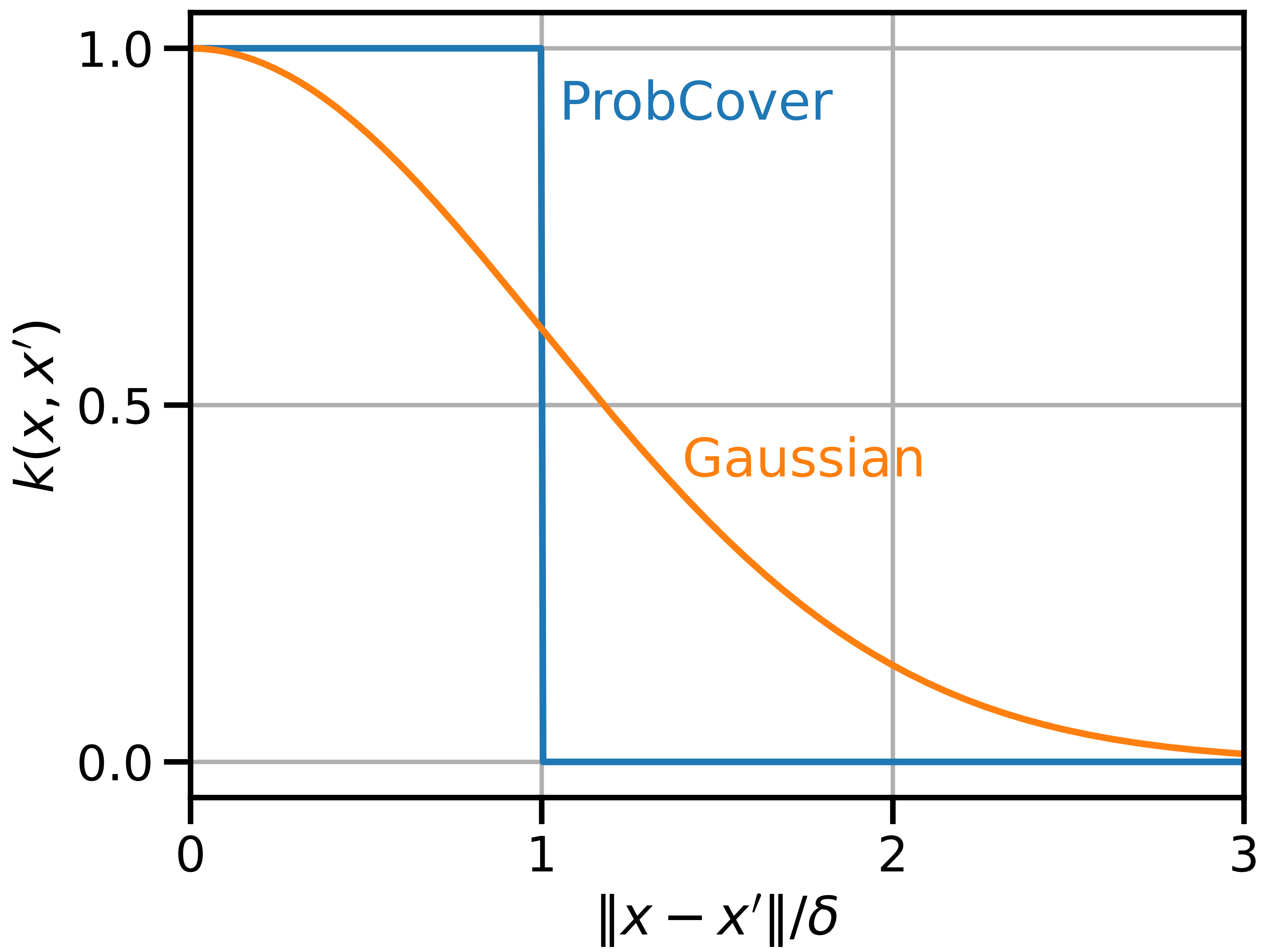
- Performance is *very* sensitive to the choice of radius δ!
- They suggest a heuristic for choosing δ to achieve a given purity level, but in our experience it's not very reliable



ProbCover

Generalized coverage

- Probabilistic coverage is a very discrete notion: a point is covered or it's not
- What about allowing "partial credit"?



$$\Pr_x\left(\hat{f}_{\mathscr{L}}(x) \text{ is wrong}\right) = \mathbb{E}_x\left[\mathbb{I}\left(f^*\left(\mathrm{NN}_{\mathscr{L}}(x)\right) \neq f^*(x)\right)\right]$$

$$= \mathbb{E}_x\left[\mathbb{I}\left(f^*\left(\mathrm{NN}_{\mathscr{L}}(x)\right) \neq f^*(x)\right)\left(1 - \max_{x' \in \mathscr{L}} k(x, x')\right)\right] + \mathbb{E}_{\tilde{x}}\left[\mathbb{I}\left(f^*\left(\mathrm{NN}_{\mathscr{L}}(x)\right) \neq f^*(x)\right)\left(\max_{x' \in \mathscr{L}} k(x, x')\right)\right]$$

$$\leq \left(1 - \mathbb{E}_x\left[\max_{x' \in \mathscr{L}} k(x, x')\right]\right) + \mathbb{E}_{\tilde{x}}\left[\max_{x' : f^*(x') \neq f^*(x)} k(x, x')\right]$$

assuming k is monotonic
in same distance as 1NN classifier

generalized coverage
(no labels!)

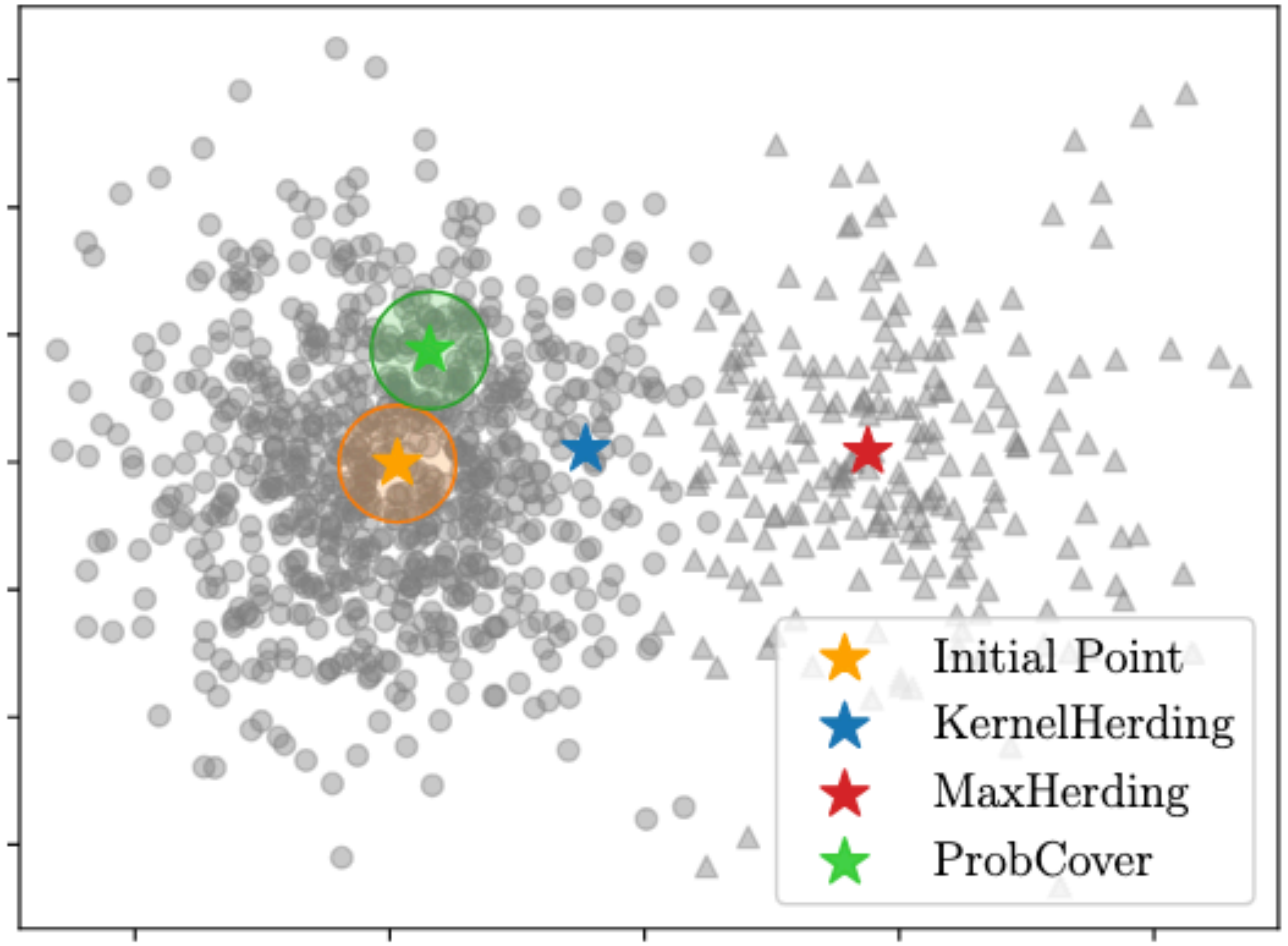
generalized impurity
(requires labels)

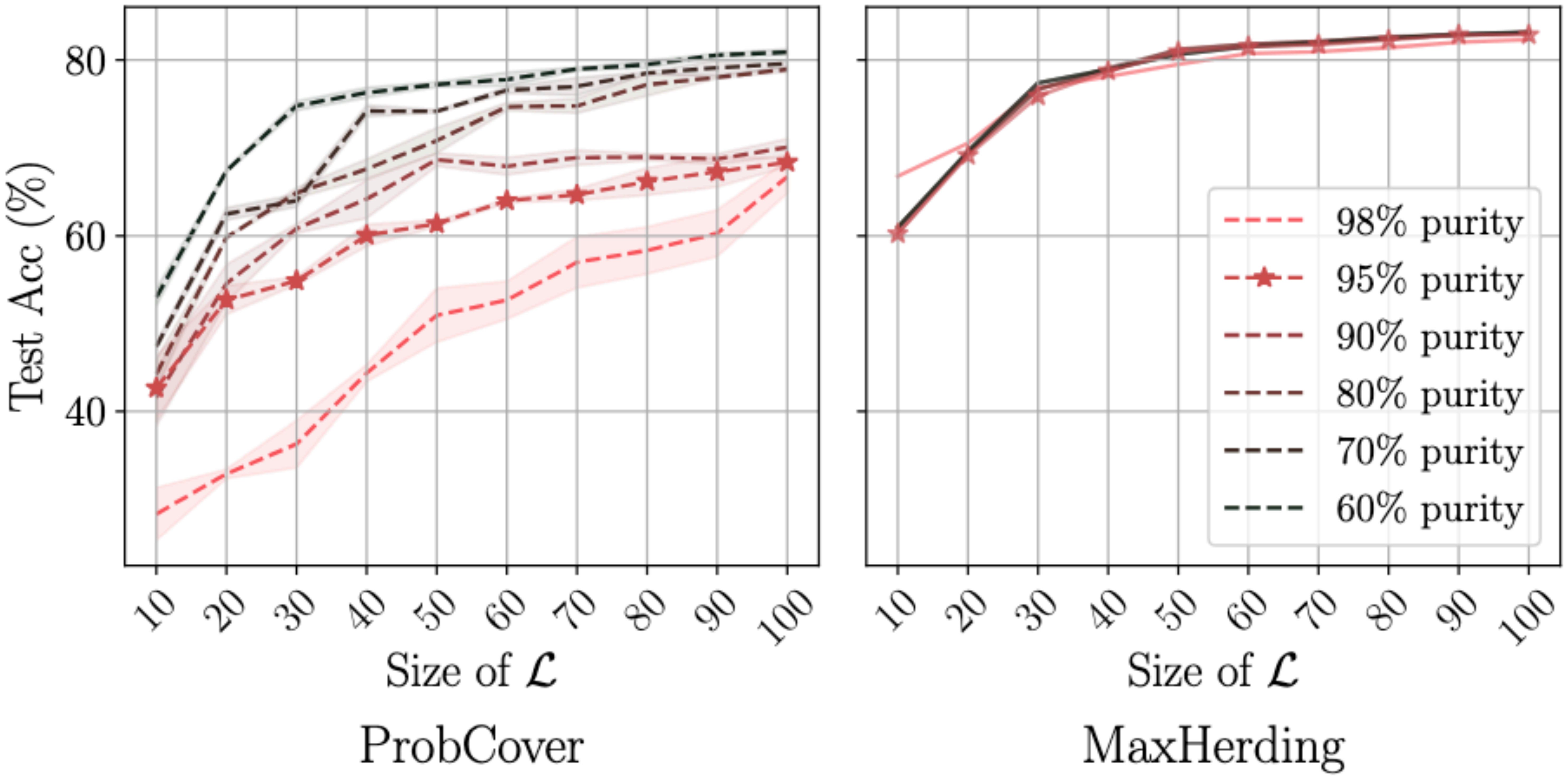
- Exactly recovers previous bound when $k(x, x') = \mathbb{I}(\|x - x'\| \leq \delta)$

MaxHerding

- Greedily maximize the generalized coverage

$$\underset{S \subseteq \mathcal{U}}{\operatorname{argmax}} \frac{1}{N} \sum_{n=1}^{N} \max_{x' \in \mathcal{L} \cup S} k(x, x')$$





- Choice of δ barely matters!

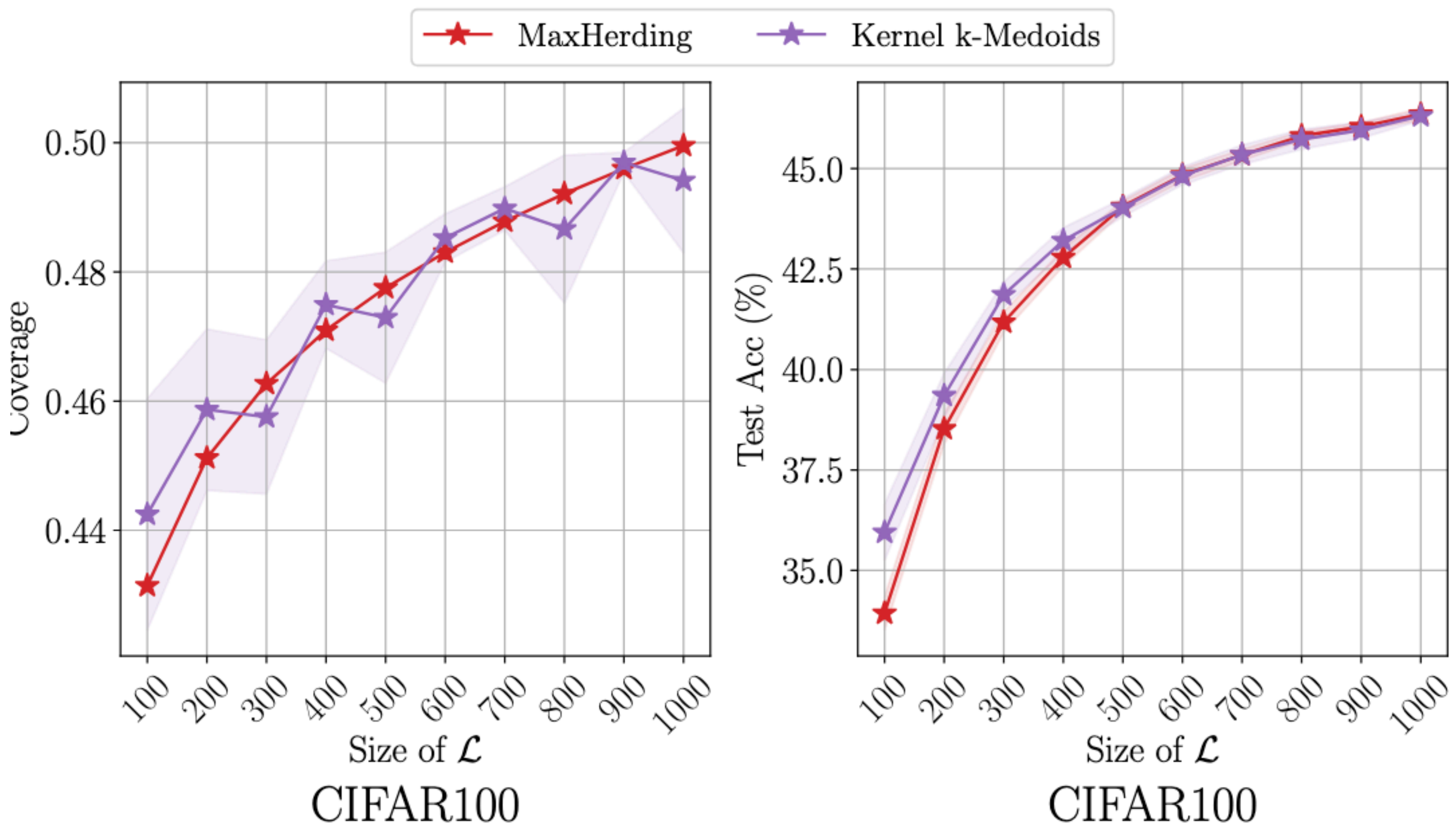
Non-greedy optimization isn't worth it

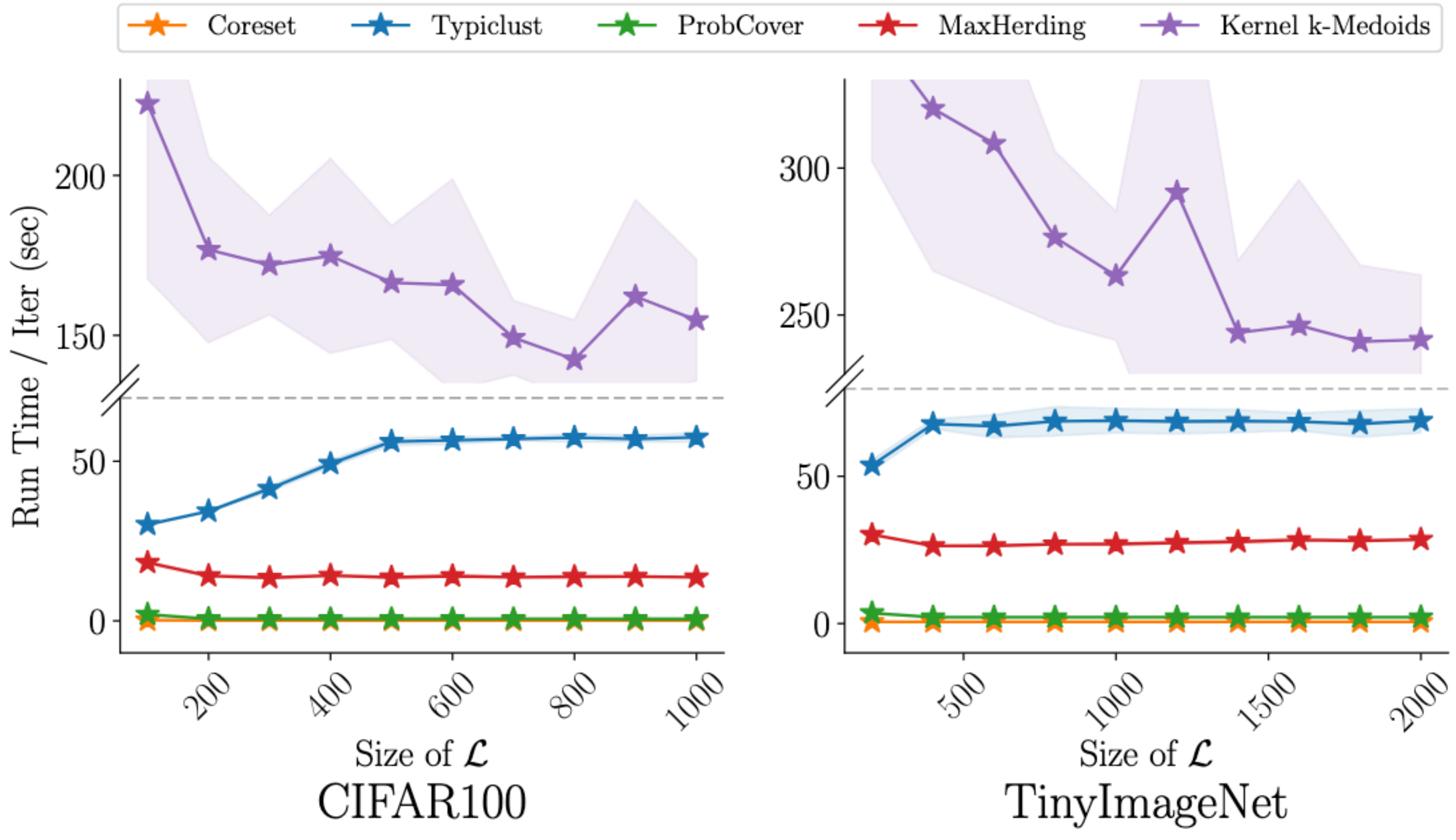
- Maximizing the coverage is exactly kernel k-medoids

$$\underset{S\subseteq\mathcal{U}}{\operatorname{argmax}} \frac{1}{N}\sum_{n=1}^{N} \max_{x'\in\mathcal{L}\cup S} k(x,x')$$

- Monotone, nonnegative, submodular:
  greedy optimization is at least 63% as good as optimal

- Non-greedy algorithm (Partitioning Around Medoids): barely better, way slower





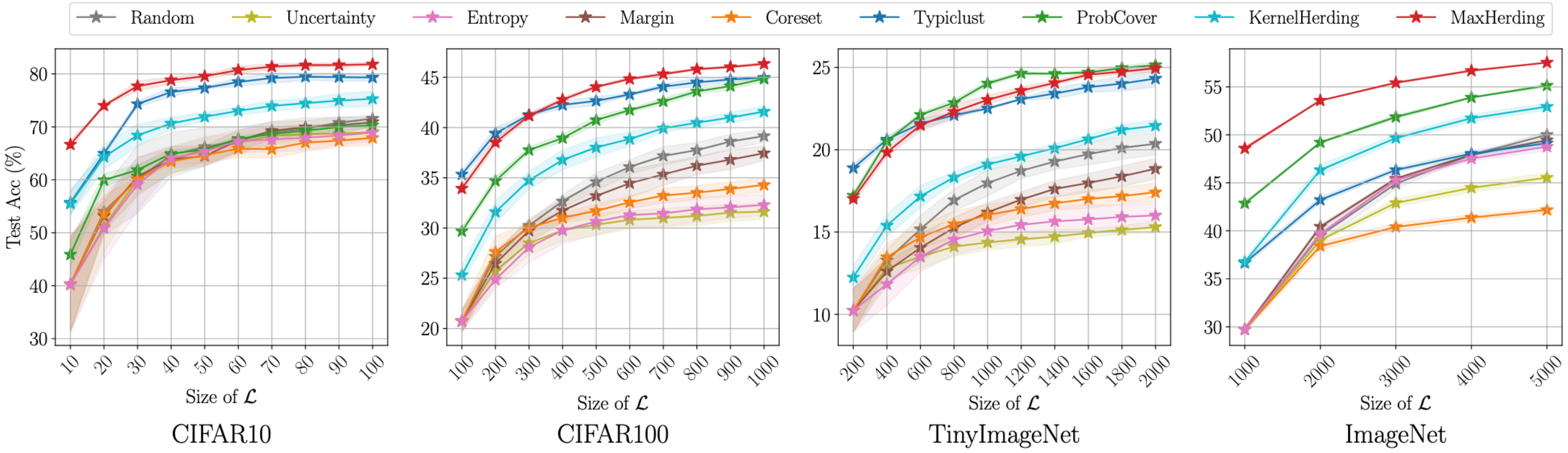


Fig. 3: Comparison on benchmark datasets using 1-NN classifier.

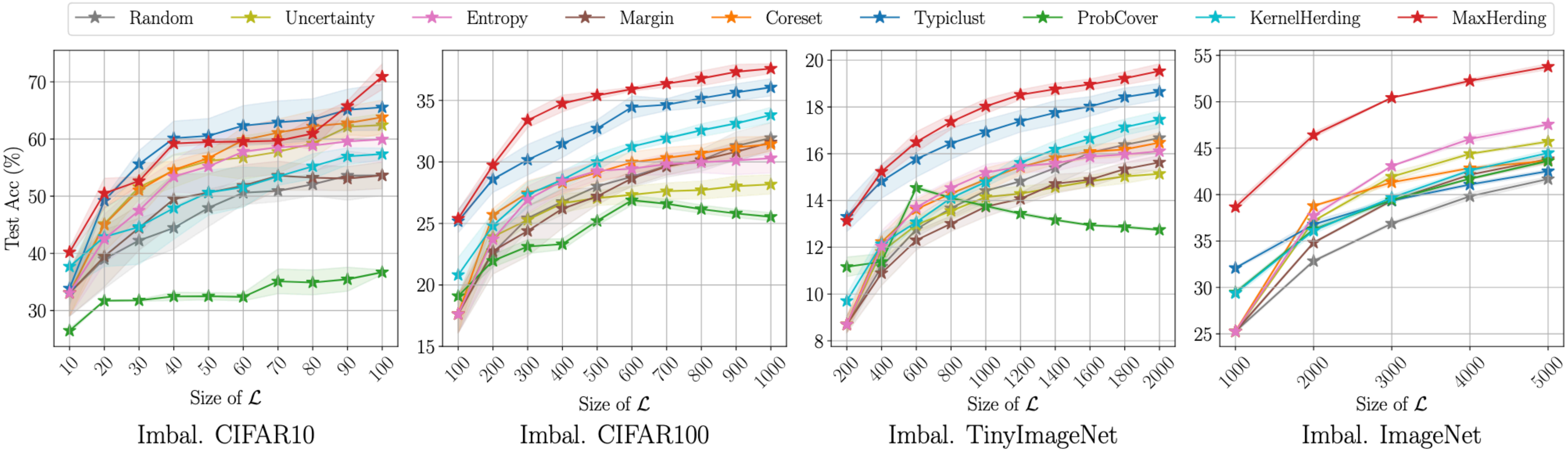
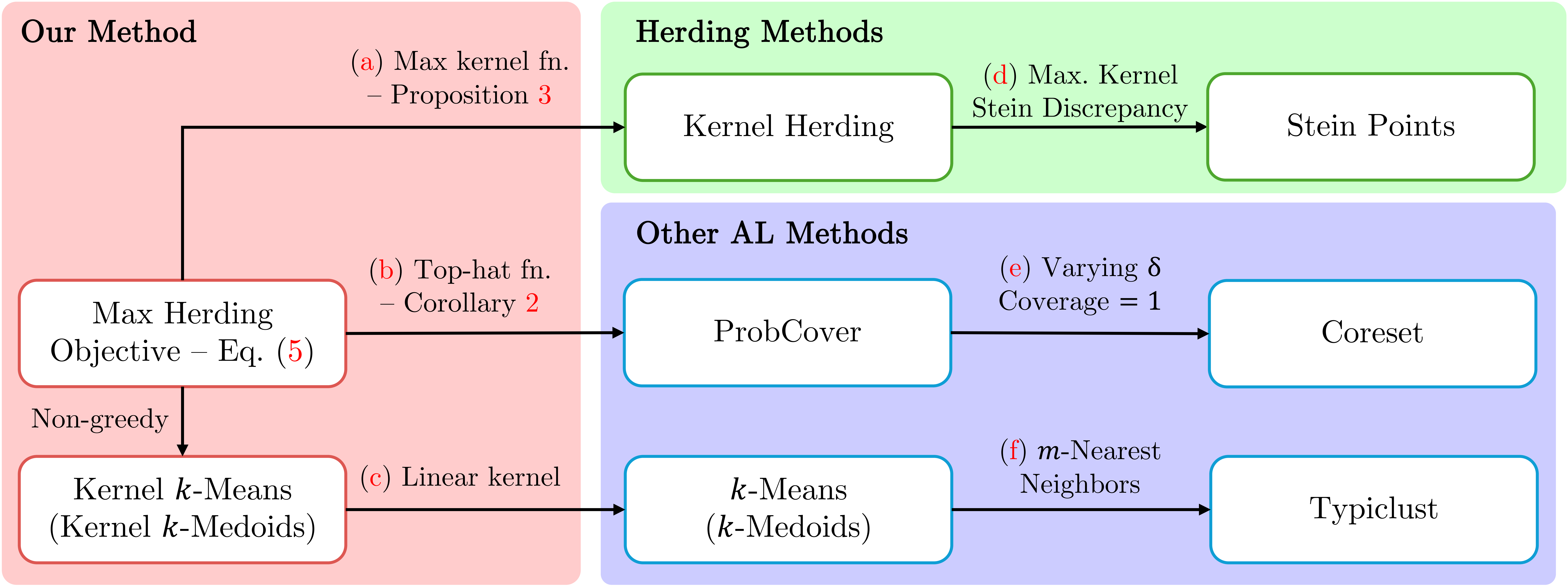
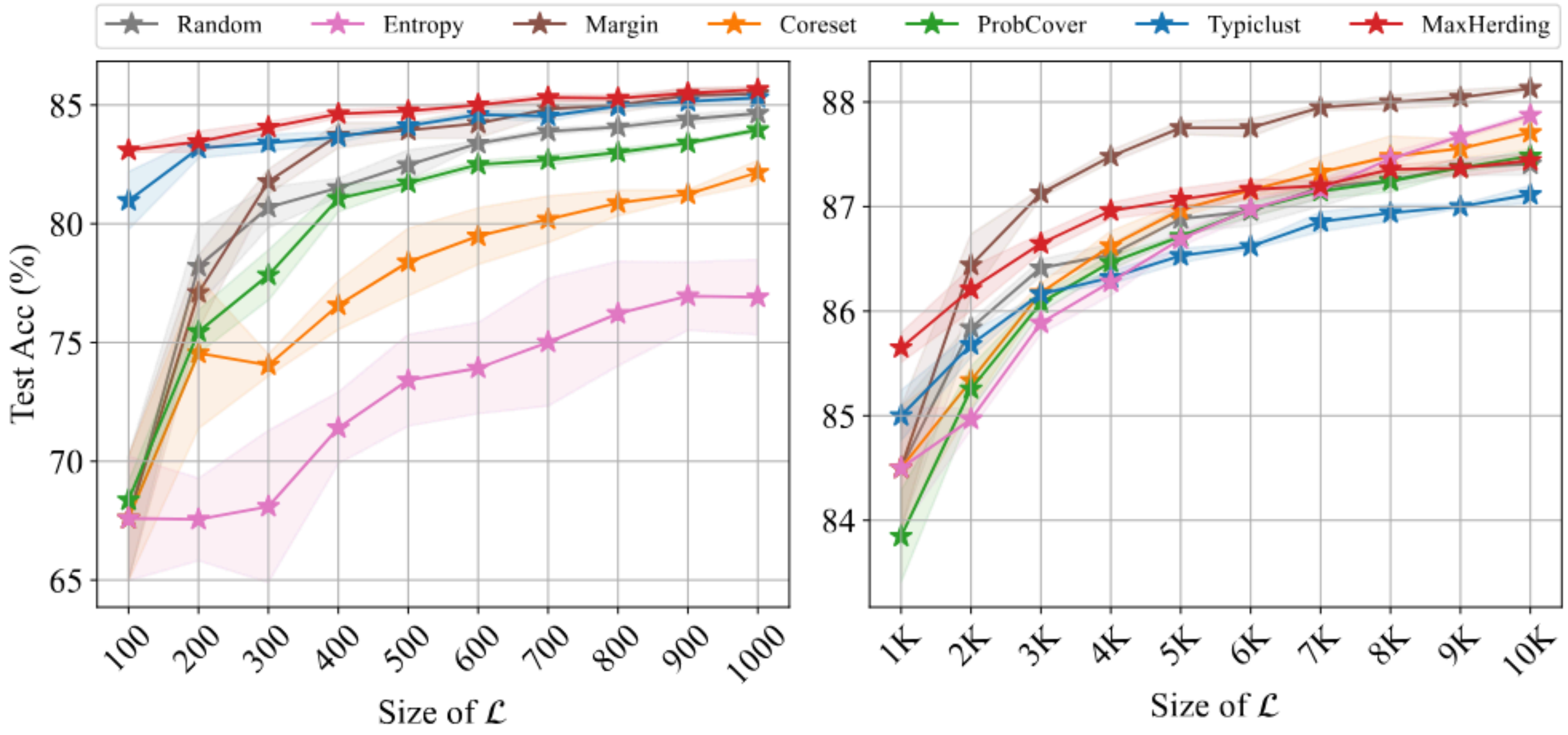


Fig. 4: Comparison on imbalanced datasets using 1-NN classifier.

Close connections to representation-based methods



…but what about later in training?



Selecting learning algorithm based on budget

- If we have a low label budget, use a representation-based method
- If we have a high label budget, use an uncertainty-based method

- …where's the line between "low" and "high"?

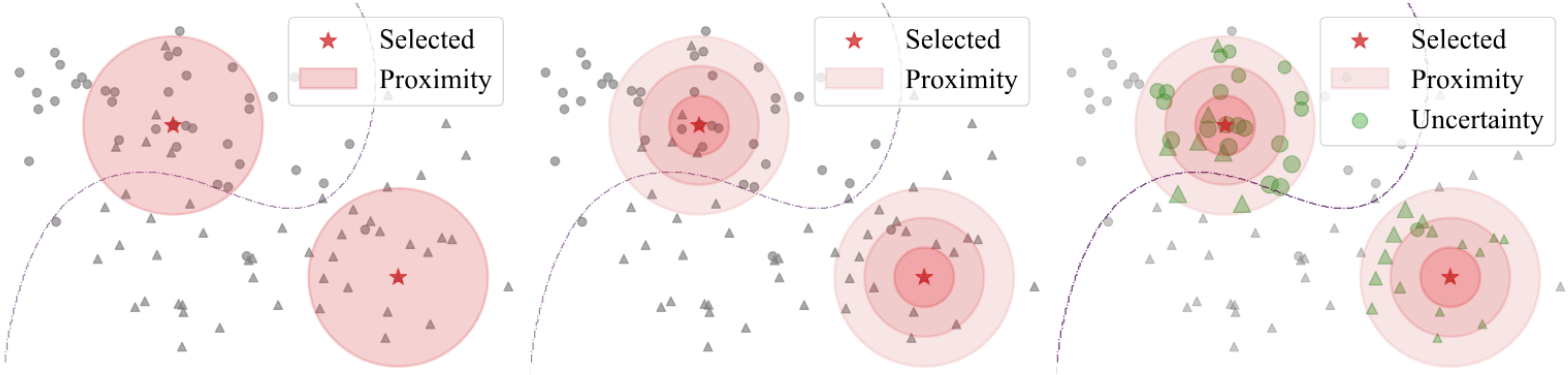
How to Select Which Active Learning Strategy is Best Suited for Your Specific Problem and Budget

Guy Hacohen[†‡], Daphna Weinshall[†]

- Problems:
  - Algorithm can't use uncertainty-based measures
  - Requires retraining many times
  - Budget regimes might not be "discrete"

Uncertainty coverage

- UCoverage: $\mathbb{E}_x\left[U(x;f)\ \max_{x' \in S} k(x,x')\right]$
  - Weight the generalized coverage by an uncertainty function $U(x;f)$



probabilistic coverage generalized coverage uncertainty coverage

Uncertainty Herding

- UCoverage: $\mathbb{E}_x\left[U(x;f)\ \max_{x'\in S} k(x,x')\right]$
  - Weight the generalized coverage by an uncertainty function $U(x;f)$
- UHerding: $\arg\max_{\tilde{x}\in\mathcal{U}} \widehat{\mathrm{UCov}}(\mathcal{L}\cup\{\tilde{x}\}) = \arg\max_{\tilde{x}\in\mathcal{U}} \frac{1}{N}\sum_{n=1}^{N} U(x_n;f) \max_{x'\in\mathcal{L}\cup\{\tilde{x}\}} k(x,x')$

(a) Margin (b) MaxHerding (c) UHerding

Uncertainty Herding

- UCoverage: $\mathbb{E}_x\left[U(x;f)\ \max_{x' \in S} k(x,x')\right]$
  - Weight the generalized coverage by an uncertainty function $U(x;f)$
- UHerding: $\arg\max_{\tilde{x} \in \mathcal{U}} \widehat{\mathrm{UCov}}(\mathcal{L} \cup \{\tilde{x}\}) = \arg\max_{\tilde{x} \in \mathcal{U}} \frac{1}{N} \sum_{n=1}^{N} U(x_n;f) \max_{x' \in \mathcal{L} \cup \{\tilde{x}\}} k(x,x')$

- Representation-based limit: MaxHerding when $U(x;f)$ is constant over x
  - Implement with temperature scaling
  - If *f* is useless but *calibrated*, then entropy/margin/etc are constant
  - As *f* improves, incorporates uncertainty more

On Calibration of Modern Neural Networks

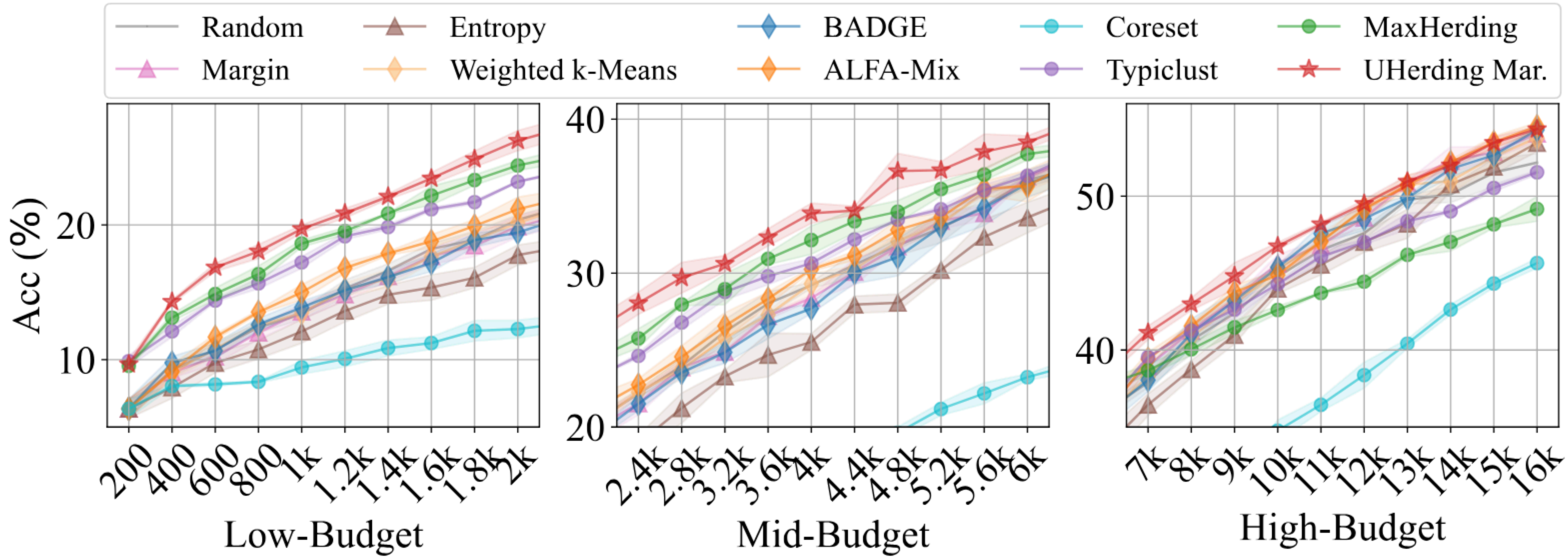
Chuan Guo[*1] Geoff Pleiss[*1] Yu Sun[*1] Kilian Q. Weinberger[1]

- Uncertainty-based limit: uncertainty sampling when kernel bandwidth $\to 0$
  - Use $k(x,x') = k(\|x - x'\|/\sigma)$; as $\sigma \to 0$, max UCoverage $\to \max U(x;f)$
  - Implement with $\sigma = \min_{x,x' \in \mathcal{L}: x \neq x'} \|x - x'\|$

UHerding works

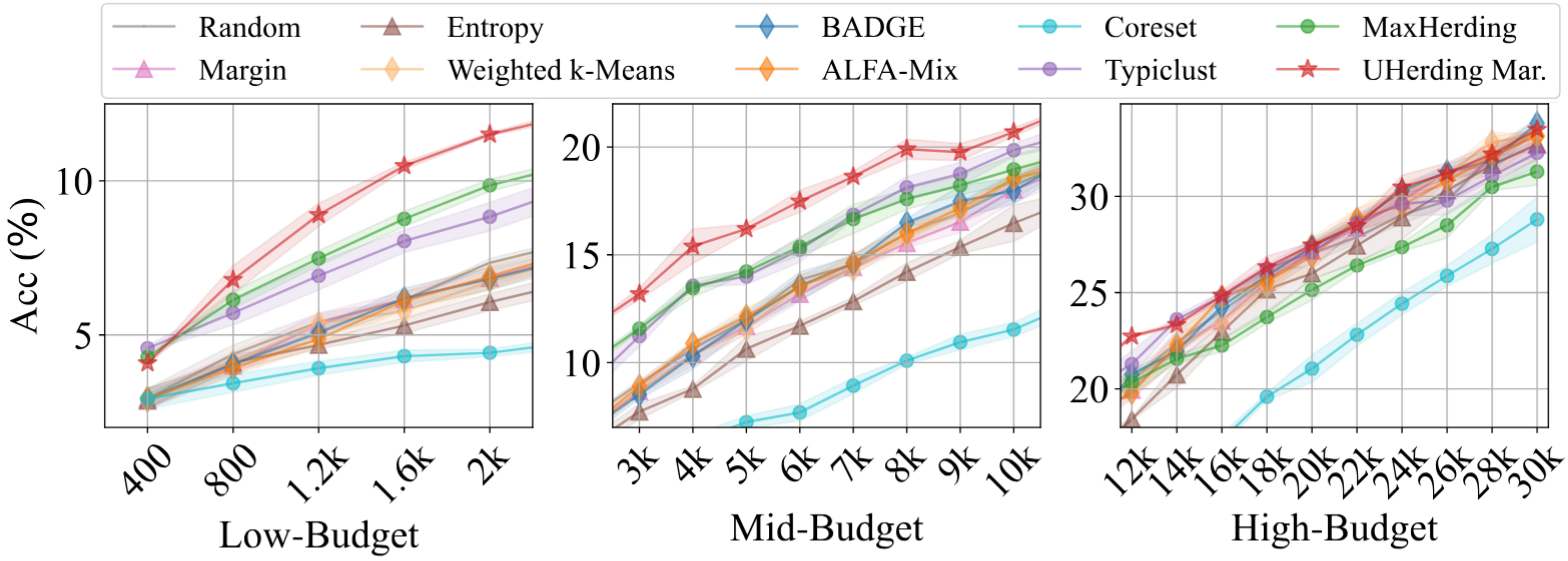
- **Theorem:** UHerding on the sample nearly maximizes UCoverage on the distribution, assuming:
  - a smooth kernel function with respect to the embeddings
  - embedding dimension isn’t too huge
  - bounded nonnegative $U(x; f)$
  - we select a small portion of the available points

UHerding works



(a) CIFAR100

UHerding works



(b) TinyImageNet

UHerding works

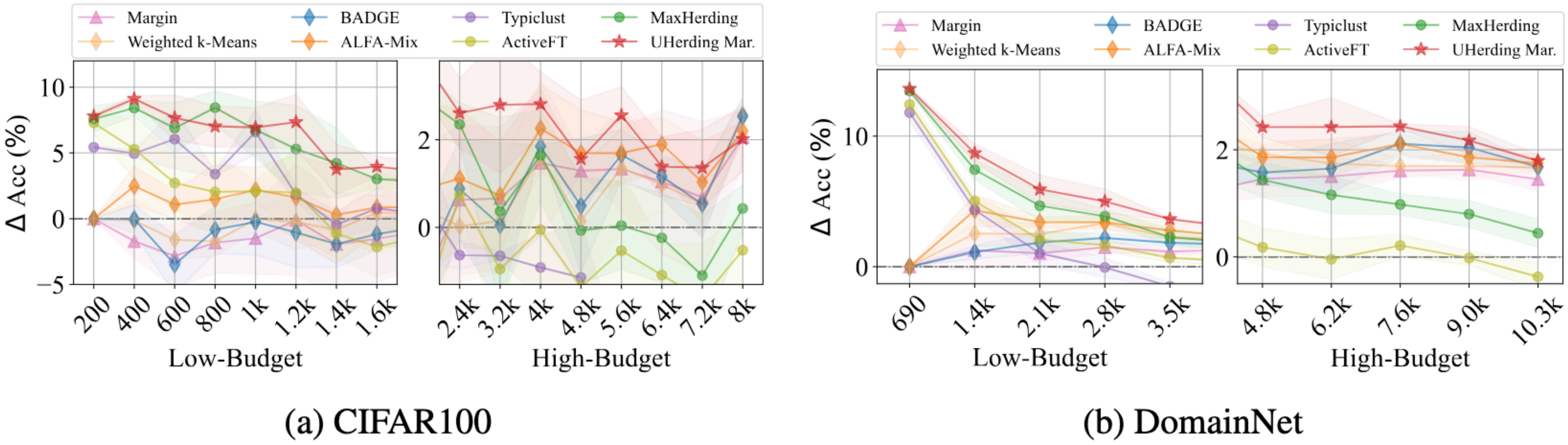
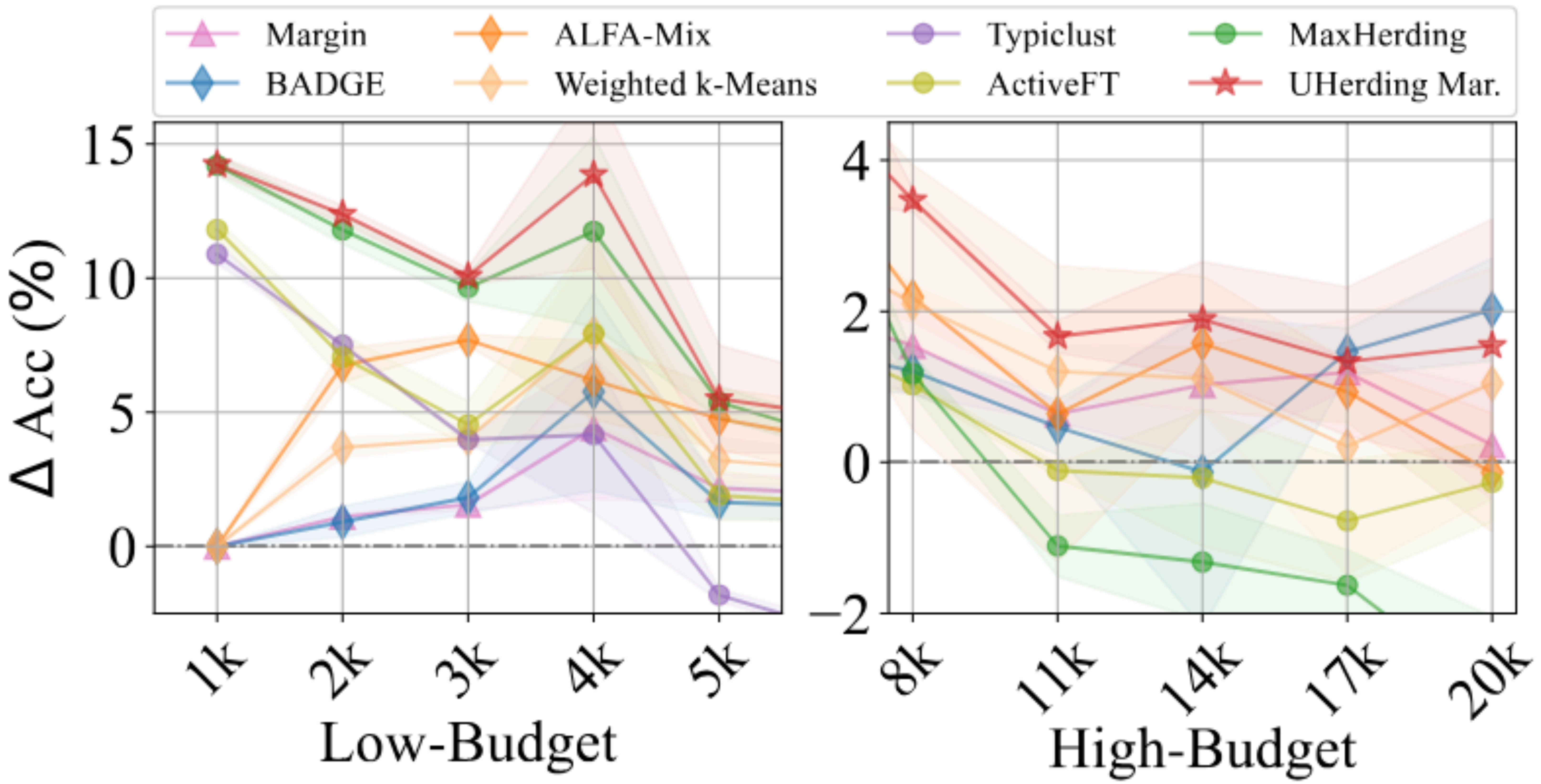


Figure 5: Comparison on CIFAR100 and DomainNet for transfer learning tasks.

UHerding works



(a) Transfer learning – ImageNet

UHerding works

Method	Low					Middle			High				
	C10	C100	Tiny.	Dom.	ImN.	C10	C100	Tiny.	C10	C100	Tiny.	Dom.	ImN.
Entropy	-1.8	-1.7	-0.6	-0.2	1.7	-1.6	-2.9	-1.7	2.2	-0.6	-0.7	0.7	**1.2**
Margin	-0.4	-0.3	-0.2	1.0	1.8	-0.1	-0.4	-0.3	**2.5**	**1.1**	0.0	1.5	0.9
BADGE	-0.5	-0.1	-0.2	1.4	2.0	0.6	-0.7	0.0	2.2	0.9	**0.4**	**1.8**	**1.0**
ALFA-M	0.1	0.9	-0.3	2.8	5.1	1.1	0.6	0.1	**2.3**	**1.3**	0.2	**1.9**	**1.0**
Weight. k	-0.5	-0.1	-0.3	2.1	3.8	0.9	0.0	-0.2	1.8	0.8	**0.3**	1.7	0.8
Coreset	-2.7	-4.5	-1.4	-3.5	-6.6	-13	-11	-5.4	-10	-9.6	-5.5	-2.7	-12
ActiveFT	–	–	–	**4.4**	**6.6**	–	–	–	–	–	–	0.0	-0.1
Typiclust	**3.7**	**3.3**	**1.6**	3.1	4.9	**4.9**	**1.8**	**2.1**	-0.8	-0.1	**0.3**	-3.2	-9.9
MaxHerd.	**5.0**	**4.1**	**2.1**	**6.2**	**10.6**	**6.2**	**2.8**	**1.9**	0.1	-2.2	-1.5	1.0	-1.2
UHerding	**5.5**	**5.5**	**3.1**	**7.4**	**11.2**	**7.8**	**4.3**	**3.7**	**3.0**	**2.1**	**0.8**	**2.3**	**2.0**

Table 1: Comparison of the mean improvement/degradation over Random selection on each budget regime and dataset. The **first**, **second**, **third** best results for each setting are marked.

Close connections to other hybrid methods

- Weighted k-means (Zhdanov 2019):
  - Swap k-means for greedy k-medoids
  - Becomes exactly UHerding with a particular U
- ALFA-Mix (Parvaneh et al. 2022):
  - Swap k-means for greedy k-medoids
  - Becomes exactly UHerding with a particular U
- BADGE (Ash et al. 2020):
  - Swap k-means++ for greedy k-medoids
  - Not exactly UHerding
    - but behaves similarly in high-temperature / low-bandwidth limits
- All of these methods are improved by our parameter adaptation scheme!

All of this was with images.
What about LLMs?

In-context learning

What Makes Good In-Context Examples for GPT-3?

Jiachang Liu[1]*, **Dinghan Shen**[2], **Yizhe Zhang**[3], **Bill Dolan**[3], **Lawrence Carin**[1], **Weizhu Chen**[2]
[1]Duke University [2]Microsoft Dynamics 365 AI [3]Microsoft Research
[1]{jiachang.liu, lcarin}@duke.edu
[2,3]{dishen, yizzhang, billdol, wzchen}@microsoft.com

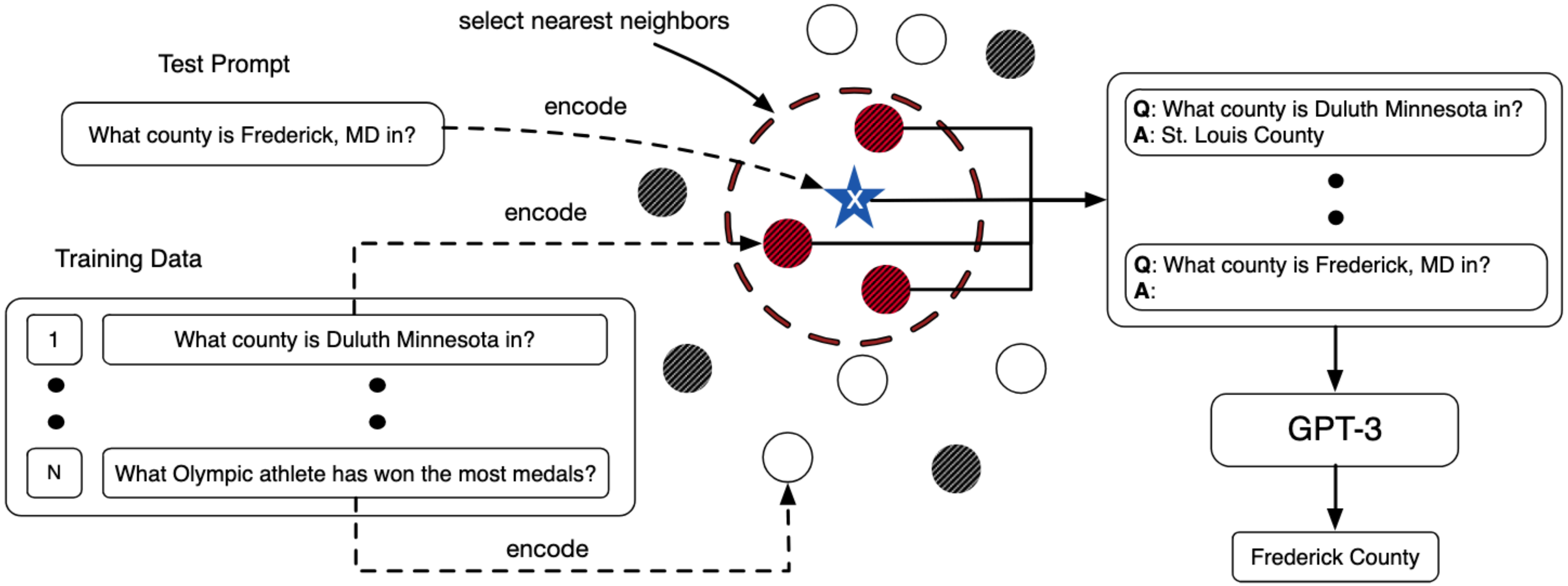
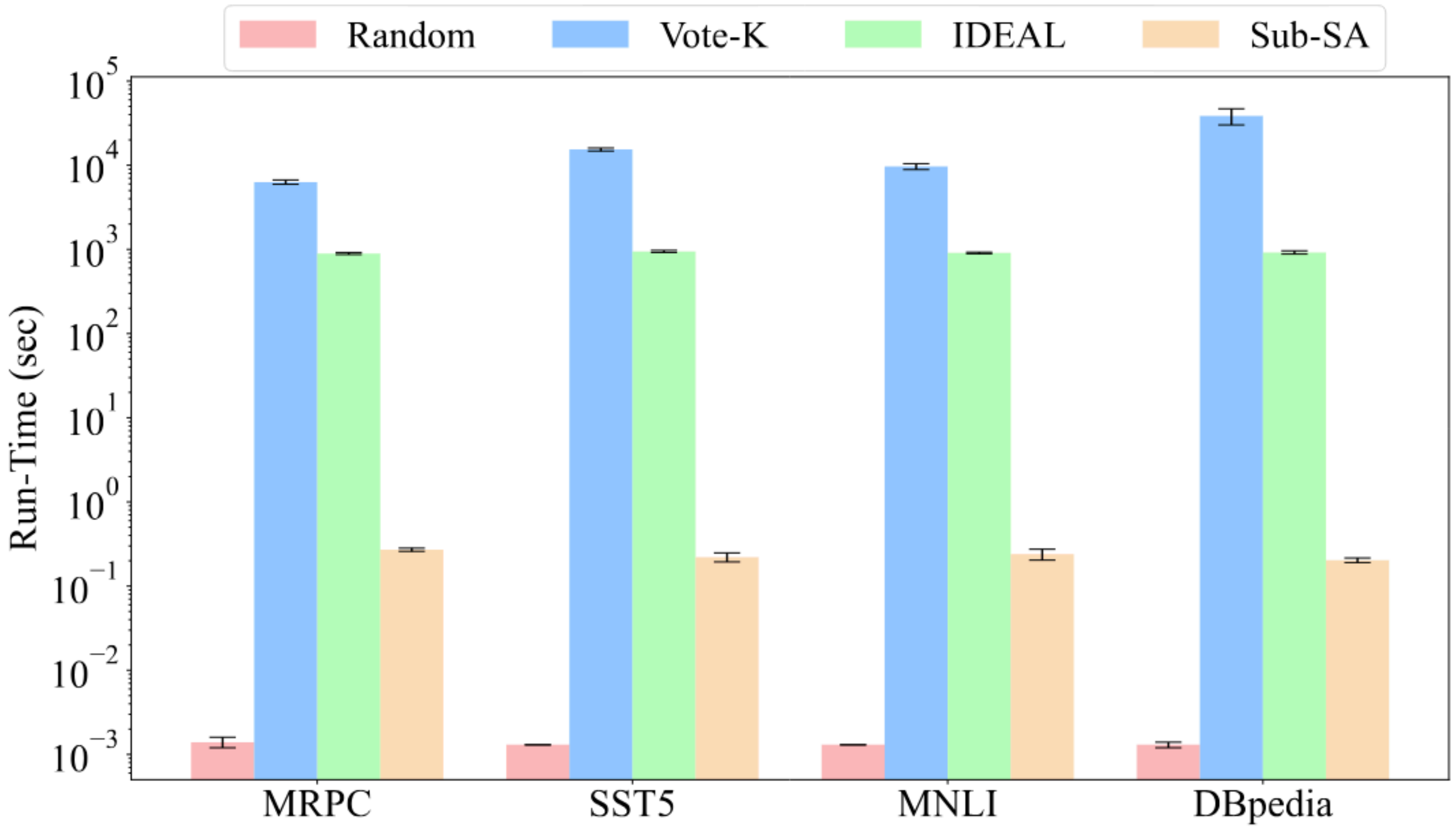


Figure 2: In-context example selection for GPT-3. White dots: unused training samples; grey dots: randomly sampled training samples; red dots: training samples selected by the k-nearest neighbors algorithm in the embedding space of a sentence encoder.

Active selection for in-context learning

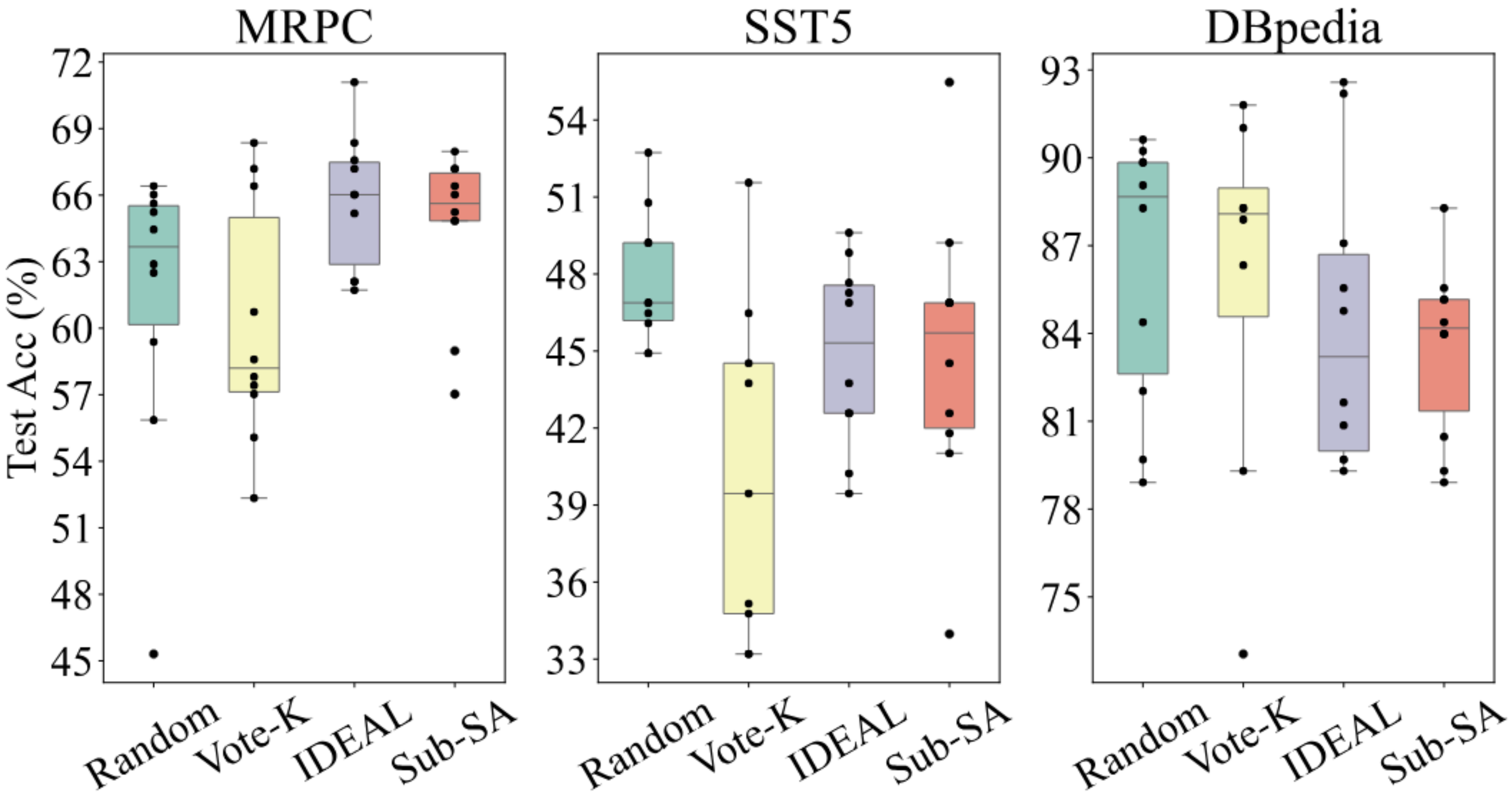
- Collect labels to maximize coverage in this space,
  so new queries have good nearby in-context examples

- Several existing papers; algorithms have gotten much faster over time



(a) Comparison in runtime for $k = 18$

Does active selection for in-context learning work?



(b) Comparison in test accuracy for $k = 18$

Does active selection for in-context learning work?

k-Shot	Method	Classification				
		MRPC	SST5	MNLI	DBPedia	RTE
$k = 100$	Random	64.5 ± 4.4	47.9 ± 2.3	39.6 ± 3.0	91.2 ± 2.3	55.7 ± 3.0
	Vote-k	62.6 ± 3.2	46.2 ± 3.4	38.8 ± 3.0	**86.6 ± 2.8**	**57.5 ± 0.4**
	IDEAL	65.1 ± 2.3	47.0 ± 3.3	38.6 ± 1.7	92.1 ± 1.9	**58.8 ± 2.3**
	Sub-SA	65.3 ± 2.3	48.4 ± 3.8	**42.3 ± 4.9**	91.9 ± 1.8	57.3 ± 1.2
$k = 18$	Random	61.4 ± 6.2	47.8 ± 2.5	38.6 ± 4.1	86.3 ± 4.4	56.6 ± 2.3
	Vote-k	60.1 ± 5.2	**40.2 ± 6.3**	37.4 ± 3.3	85.7 ± 6.0	57.5 ± 1.5
	IDEAL	65.7 ± 2.9	**44.9 ± 3.4**	38.9 ± 3.1	84.3 ± 4.8	55.5 ± 2.7
	Sub-SA	64.6 ± 3.5	**44.9 ± 5.4**	39.8 ± 4.8	83.5 ± 2.9	**61.6 ± 0.9**
$k = 5$	Random	55.7 ± 5.9	42.0 ± 4.6	37.9 ± 3.7	72.9 ± 6.3	54.9 ± 5.8
	Vote-k	**47.5 ± 6.5**	41.4 ± 5.8	37.3 ± 2.5	**87.9 ± 3.9**	53.9 ± 0.7
	IDEAL	61.3 ± 7.5	39.5 ± 6.4	36.7 ± 3.6	72.2 ± 11.1	52.3 ± 3.7
	Sub-SA	60.3 ± 4.1	36.6 ± 9.6	39.2 ± 6.8	73.9 ± 7.0	53.6 ± 1.1

Table 1: Comparison in performance of state-of-the-art methods for classification tasks.

k-Shot	Method	Multi-Choice	Dialogue
		Hellaswag	MWoZ
$k = 100$	Random	65.6 ± 2.4	40.2 ± 4.0
	Vote-k	65.1 ± 2.4	**47.7 ± 2.2**
	IDEAL	65.3 ± 2.6	42.9 ± 4.3
	Sub-SA	65.6 ± 2.6	38.8 ± 4.0
$k = 18$	Random	65.2 ± 3.0	32.0 ± 4.2
	Vote-k	65.2 ± 3.2	**42.3 ± 4.3**
	IDEAL	64.6 ± 2.9	34.7 ± 6.1
	Sub-SA	64.1 ± 2.4	33.6 ± 6.5

Table 2: Comparison on multi-choice and dialogue.

k-Shot	Method	Generation	
		GeoQ	Xsum
$k = 100$	Random	57.6 ± 3.2	19.8 ± 0.7
	Vote-k	58.2 ± 1.8	20.0 ± 0.5
	IDEAL	58.4 ± 1.6	19.3 ± 0.3
	Sub-SA	59.4 ± 1.7	19.6 ± 0.7
$k = 18$	Random	44.3 ± 2.6	19.1 ± 1.1
	Vote-k	**49.7 ± 1.7**	19.7 ± 0.6
	IDEAL	47.7 ± 5.6	19.6 ± 0.6
	Sub-SA	**52.4 ± 2.3**	19.3 ± 0.8

Table 3: Comparison on generation tasks.

Blue: statistically better than random
Red: statistically worse than random

Does active selection for in-context learning work?

Model	Size	Method	Classification				
			MRPC	SST5	MNLI	DBPedia	RTE
MiniLM	23M	Random	61.6 ± 6.7	47.5 ± 3.7	38.4 ± 3.3	86.6 ± 4.3	55.7 ± 2.4
		Vote-k	60.1 ± 6.1	44.9 ± 4.7	38.6 ± 4.6	86.8 ± 4.0	**52.8 ± 0.8**
		IDEAL	65.5 ± 3.8	44.9 ± 3.8	37.2 ± 1.9	83.3 ± 5.3	56.1 ± 2.3
		Sub-SA	62.1 ± 6.0	43.9 ± 4.9	40.5 ± 3.8	**81.0 ± 3.1**	**52.4 ± 1.5**
GPT-J	6B	Random	60.9 ± 6.4	48.0 ± 3.0	36.8 ± 5.0	85.8 ± 4.7	56.3 ± 2.4
		Vote-k	63.2 ± 3.3	**43.8 ± 4.7**	37.9 ± 2.5	82.9 ± 2.7	58.2 ± 1.5
		IDEAL	**66.5 ± 2.1**	47.7 ± 4.7	38.7 ± 2.2	84.5 ± 4.3	57.5 ± 2.5
		Sub-SA	**66.7 ± 2.3**	**45.7 ± 3.6**	37.3 ± 4.5	**65.5 ± 12.0**	**59.2 ± 1.0**

Table 4: Comparison of selection methods with different embedding models.

Model	Method	Classification				
		MRPC	SST5	MNLI	DBPedia	RTE
Pythia-1B	Random	50.3 ± 9.4	32.7 ± 2.6	35.2 ± 2.6	22.0 ± 1.1	49.3 ± 2.3
	Vote-k	46.0 ± 2.7	31.2 ± 3.7	34.8 ± 2.3	23.0 ± 4.0	50.5 ± 0.9
	IDEAL	59.3 ± 4.2	29.6 ± 2.6	35.5 ± 3.4	24.6 ± 6.1	49.6 ± 3.4
	Sub-SA	55.2 ± 1.8	31.6 ± 2.9	35.2 ± 2.2	23.4 ± 4.3	**54.7 ± 0.7**
GPT-Neo-2.7B	Random	62.8 ± 6.0	40.2 ± 1.9	33.1 ± 3.6	77.1 ± 2.1	53.4 ± 2.6
	Vote-k	61.0 ± 3.9	39.7 ± 4.3	34.1 ± 2.4	80.1 ± 2.0	56.4 ± 1.2
	IDEAL	65.3 ± 1.2	39.5 ± 2.5	33.9 ± 3.3	69.4 ± 7.1	51.7 ± 4.0
	Sub-SA	67.1 ± 1.2	42.1 ± 4.6	36.7 ± 4.6	67.3 ± 7.7	**58.4 ± 0.5**

Table C.1: Performance comparison of inference-based LLMs in classification tasks.

Does active selection for in-context learning work?

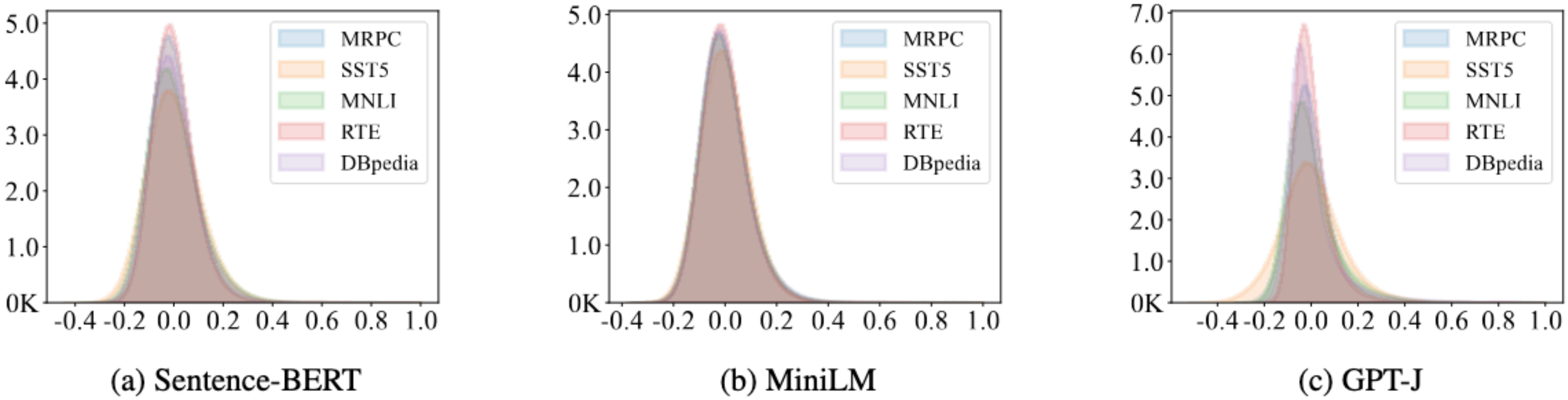


Figure 2: Density of cosine similarity between embeddings of training examples by dataset and embedding models.

Changing gears slightly:
why do we need to be so careful with DPO?

Learning dynamics

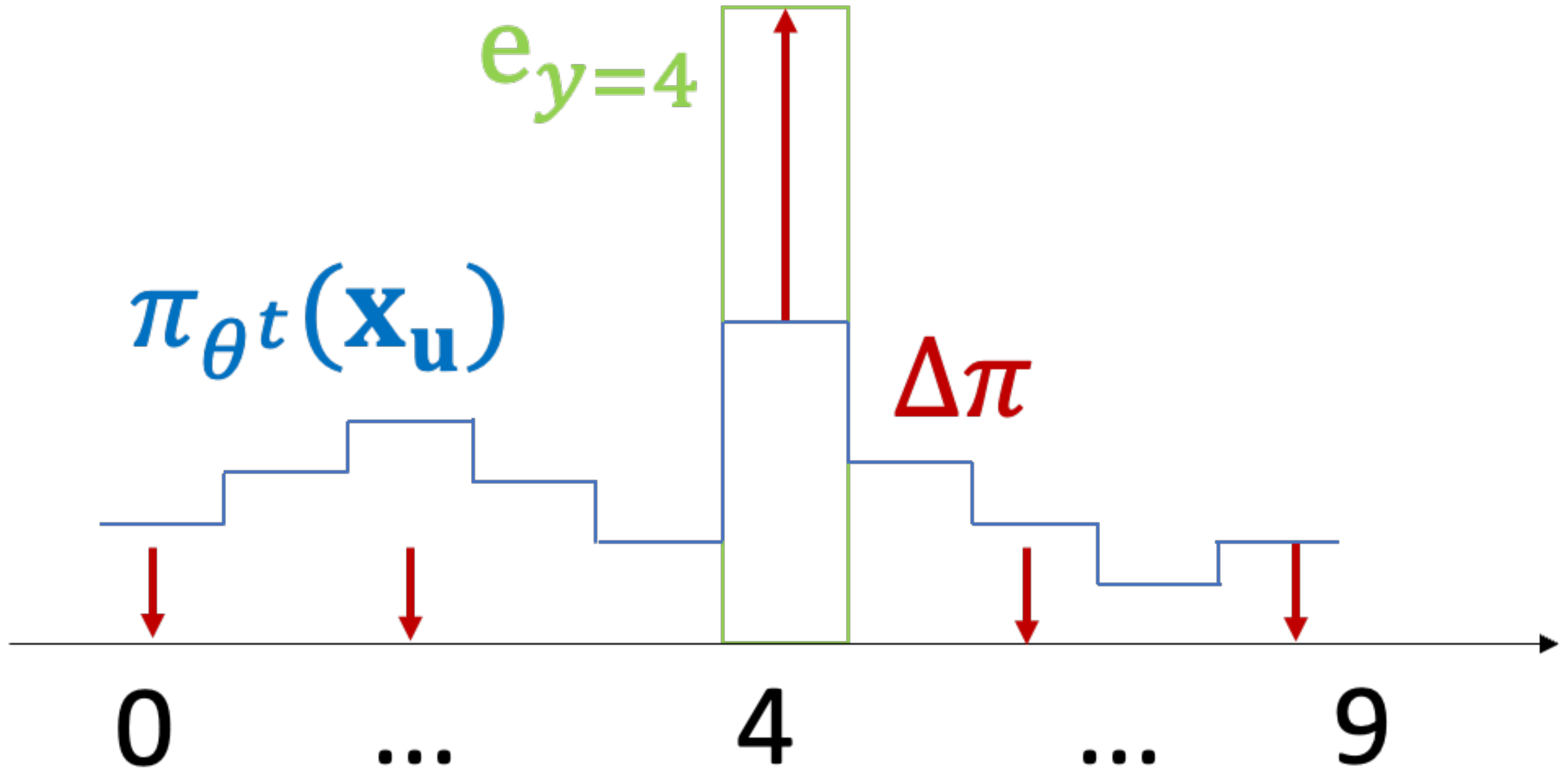
After an update on $\boldsymbol{x_u}$, how does the model's prediction on $\boldsymbol{x_o}$ change?

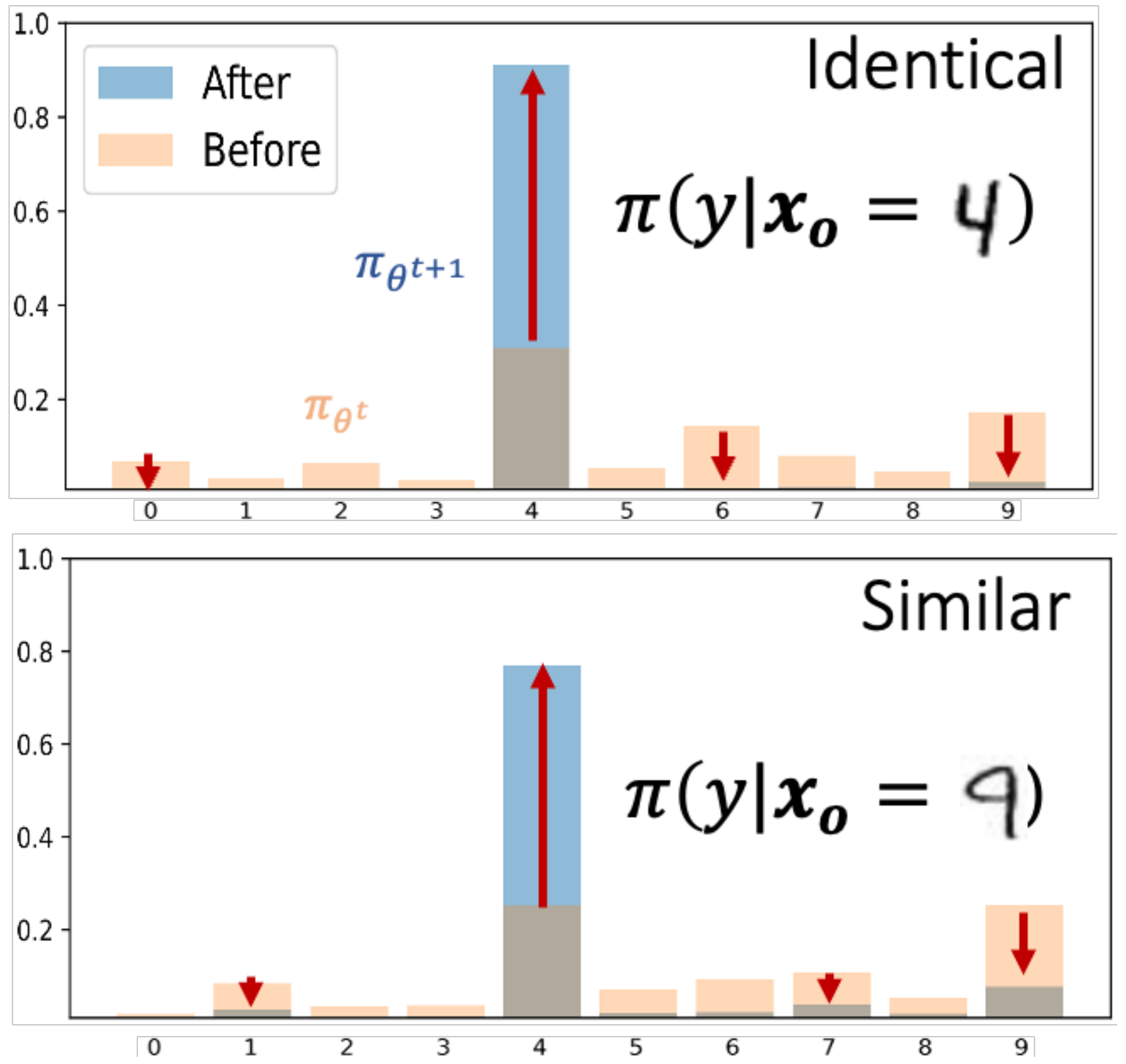
$$\Delta \log \pi^t(\boldsymbol{x_o}) = -\eta \mathcal{A}^t(\boldsymbol{x_o}) \mathcal{K}^t(\boldsymbol{x_o}, \boldsymbol{x_u}) \mathcal{G}^t(\boldsymbol{x_u}, \boldsymbol{y_u}) + \mathcal{O}(\eta^2)$$

$$I - \mathbf{1}(\pi^t)^\top = \begin{bmatrix} 1-\pi_1 & -\pi_1 & \cdots & -\pi_1 \\ -\pi_2 & 1-\pi_2 & \cdots & -\pi_2 \\ \cdots & \cdots & \ddots & \cdots \\ -\pi_V & -\pi_V & \cdots & 1-\pi_V \end{bmatrix}$$

eNTK: $\nabla_\theta \mathrm{z_o} (\nabla_\theta \mathrm{z_u})^{\mathrm{T}}$

$\pi_\theta(\mathbf{y}|\boldsymbol{x_u}) - \mathbf{y_u}$





Learning dynamics in LLMs

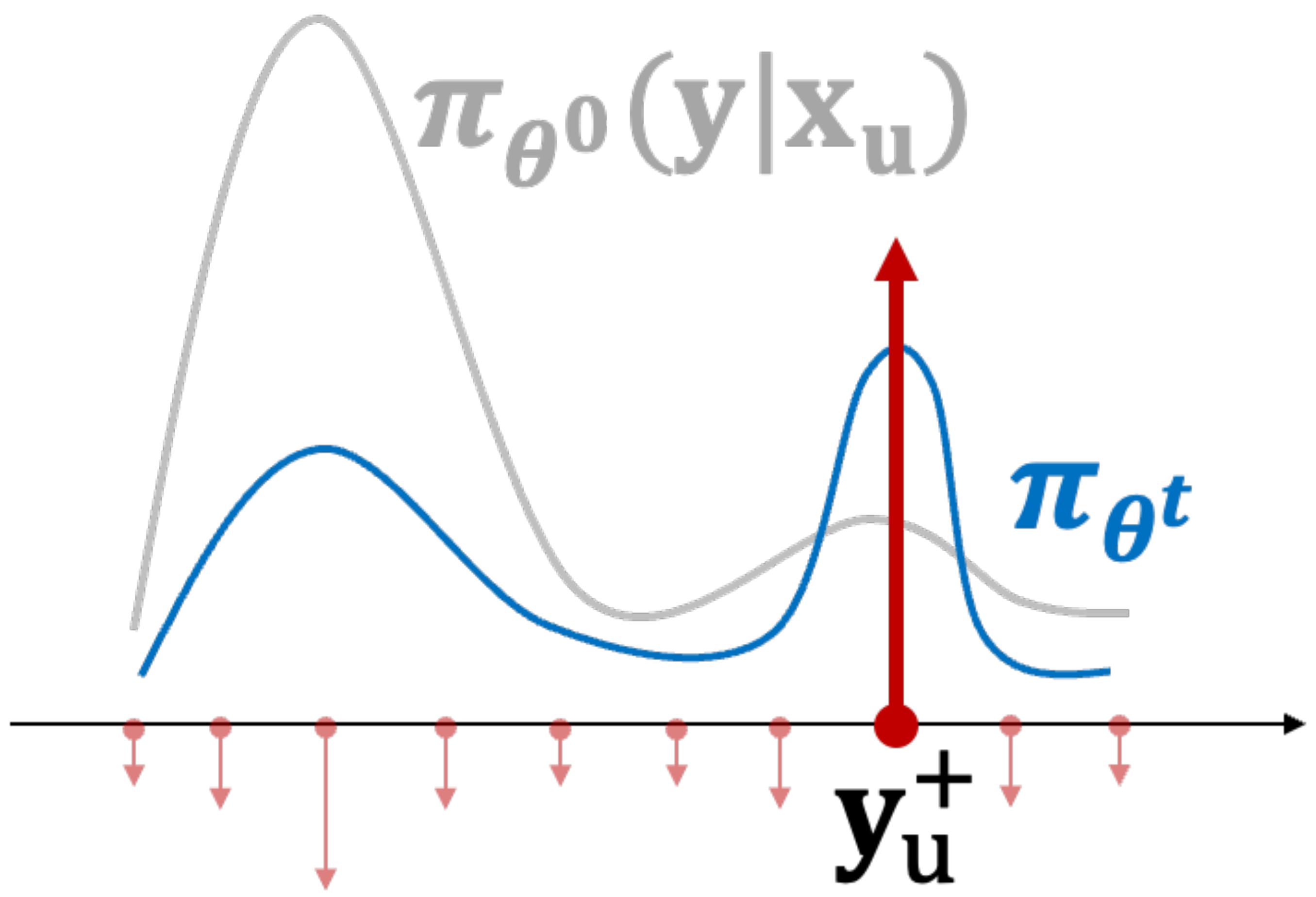
For a prompt $\boldsymbol{x_u}$, how does learning the response $\boldsymbol{y_u^+}$ influence the model's belief about another $\boldsymbol{y_u'}$?

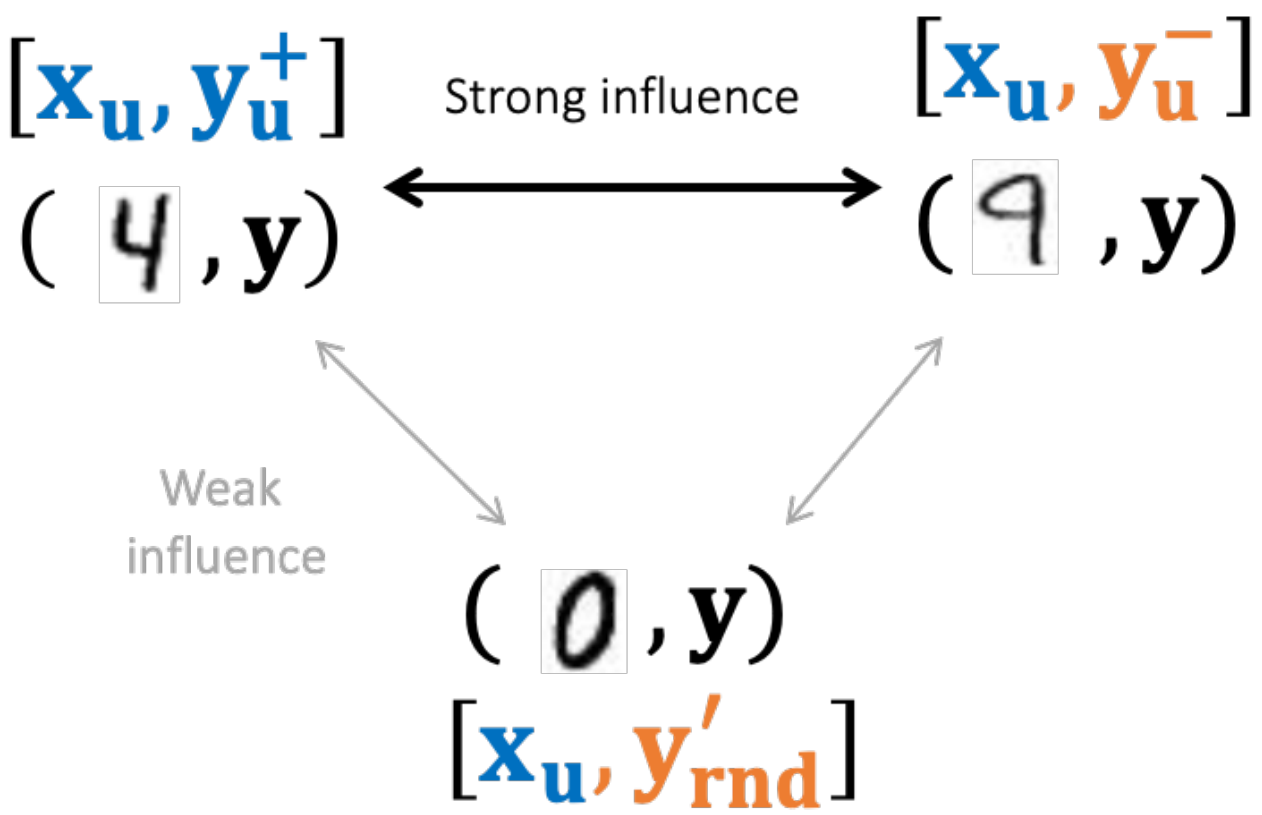
$$\mathcal{L}_{\text{SFT}} \triangleq -\log \mathbf{z} = -\log \pi_\theta(\mathbf{y}|\mathbf{x}) = -\sum \log \pi_\theta(y_l|\mathbf{x}, \mathbf{y}_{<\mathbf{l}})$$

$$\chi = [\mathbf{x}; \mathbf{y}]; \qquad \mathbf{z} = h_\theta(\chi); \qquad \pi_\theta(\mathbf{y}|\chi) = \text{Softmax}\,(\mathbf{z})$$

$$\underbrace{[\Delta \log \pi^t(y|\chi_o)]_m}_{V\times M} = -\sum_{l=1}^{L} \eta \underbrace{[\mathcal{A}^t(\chi_o)]_m}_{V\times V\times M} \underbrace{[\mathcal{K}^t(\chi_o, \chi_u)]_l}_{V\times V\times L} \underbrace{[\mathcal{G}(\chi_u)]_l}_{V\times L} + \mathcal{O}(\eta^2)$$

Learning dynamics in LLMs

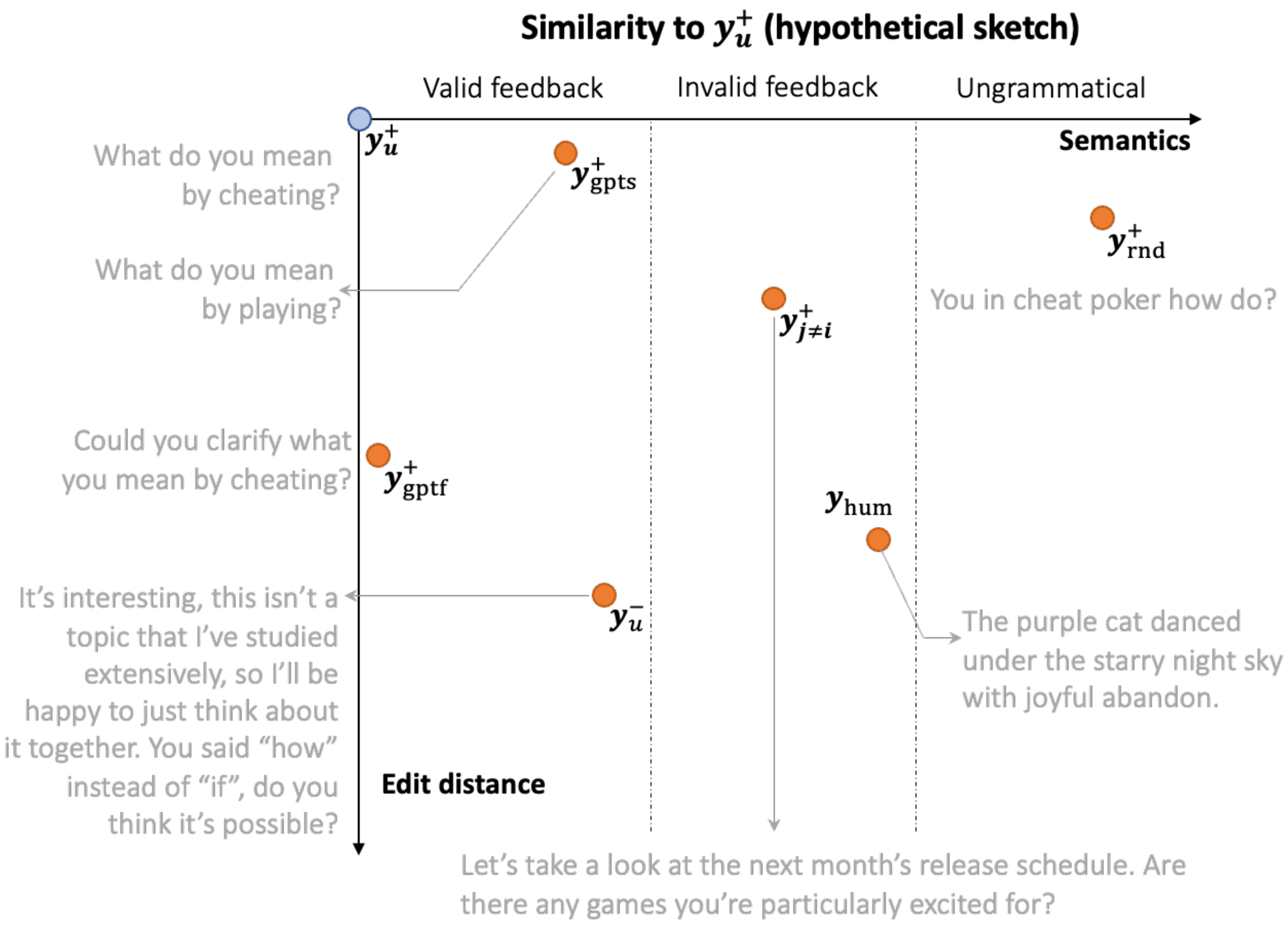




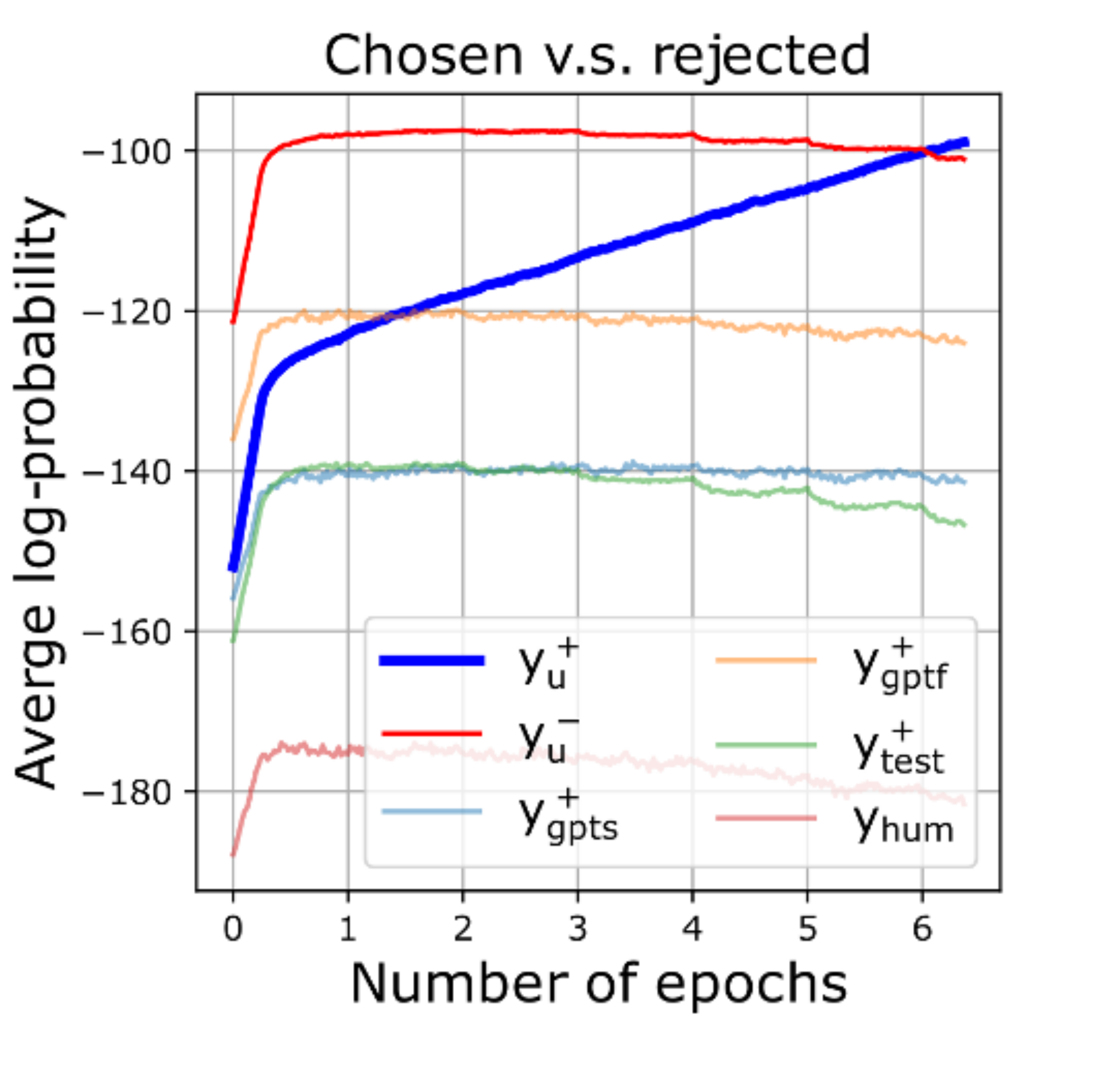
Prompt: x_u

How do you cheat in poker?

1. **Chosen response y_u^+**
   - 1.1 GPT rephrase chosen, preserving semantics y_{gpts}^+
   - 1.2 GPT rephrase chosen, preserving format y_{gptf}^+
2. **Rejected response y_u^-**
   - 2.1 GPT rephrase rejected, preserving semantics y_{gpts}^-
   - 2.2 GPT rephrase rejected, preserving format y_{gptf}^-
3. **Irrelavent from train set $y_{j\neq i}^+$**
4. **Random sentence by GPT y_{hum}**
5. **Random permuted chosen y_{rnd}^+**



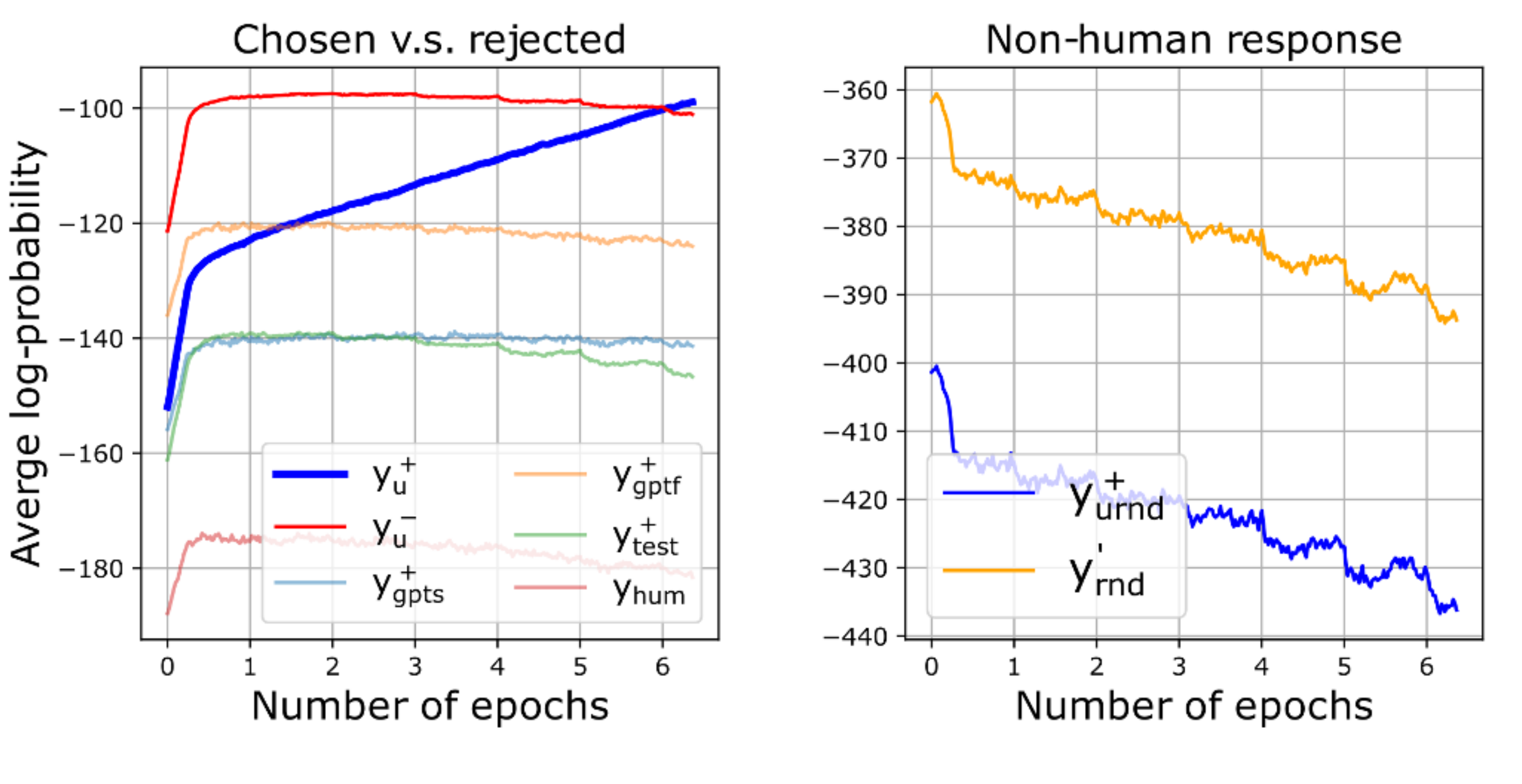
Learning dynamics in supervised finetuning



Desired response becomes more likely

Other decent responses stay about the same

Learning dynamics in supervised finetuning

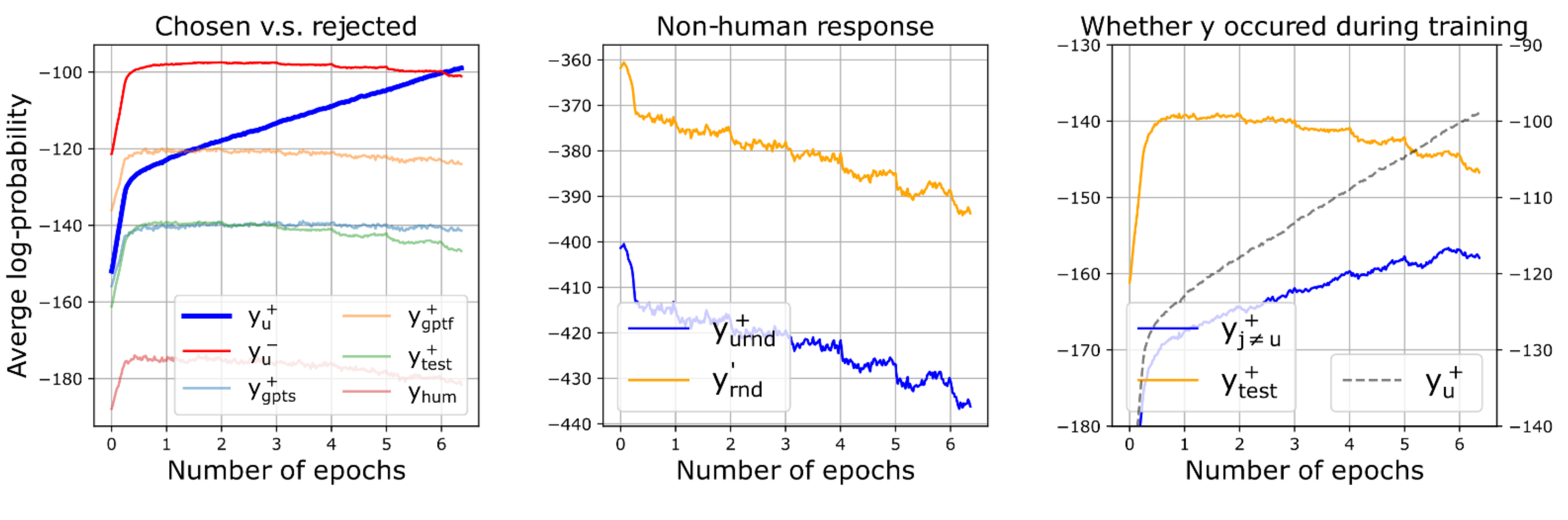


Desired response
becomes more likely

Ungrammatical responses
become less likely

Other decent responses
stay about the same

Learning dynamics in supervised finetuning



Desired response becomes more likely

Ungrammatical responses become less likely

Other decent responses stay about the same

Irrelevant responses in the training dataset become more likely!

Learning dynamics in supervised finetuning

Desired response
becomes more likely

Ungrammatical responses
become less likely

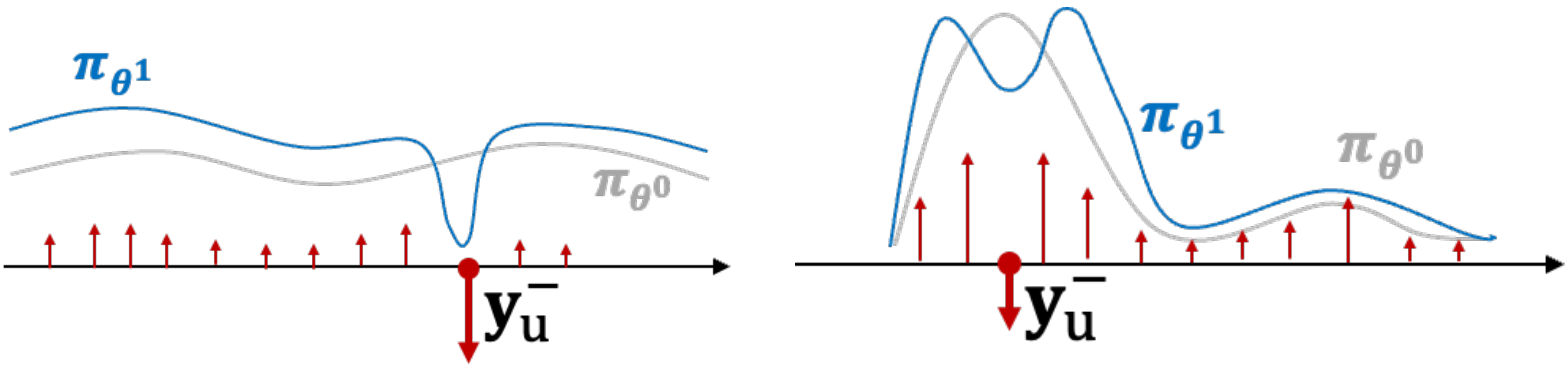
Other decent responses
stay about the same

Irrelevant responses
in the training dataset
become more likely!

Direct preference optimization (DPO)

$$-\sum_{l=1}^{L} \eta [\mathcal{A}^t(\chi_o)]_m [\mathcal{K}^t(\chi_o, \chi_u^+)\mathcal{G}^t_{\mathrm{DPO+}} - \mathcal{K}^t(\chi_o, \chi_u^-)\mathcal{G}^t_{\mathrm{DPO-}}]_l$$

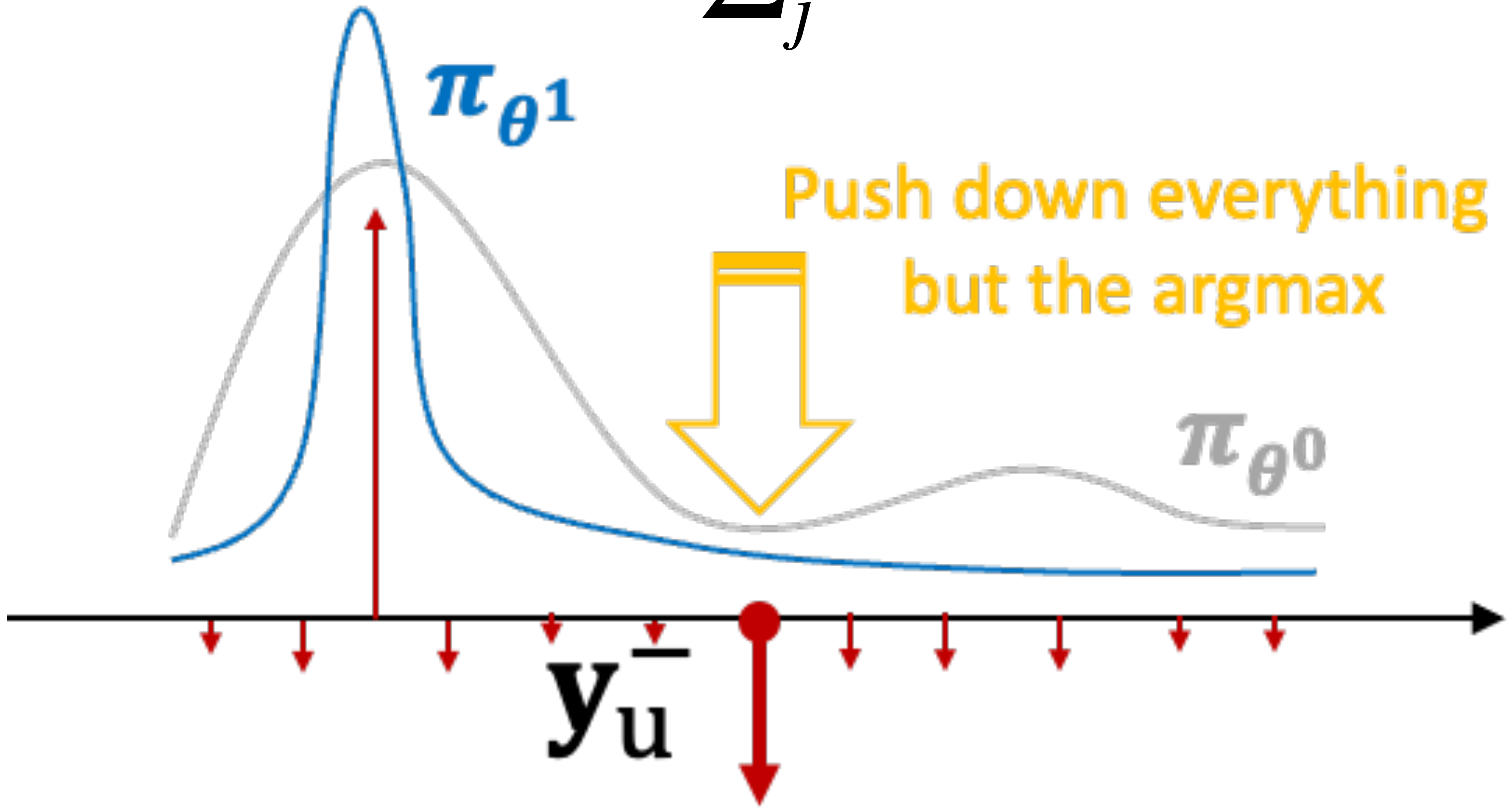
- Negative gradient helps the model not say y_u^-

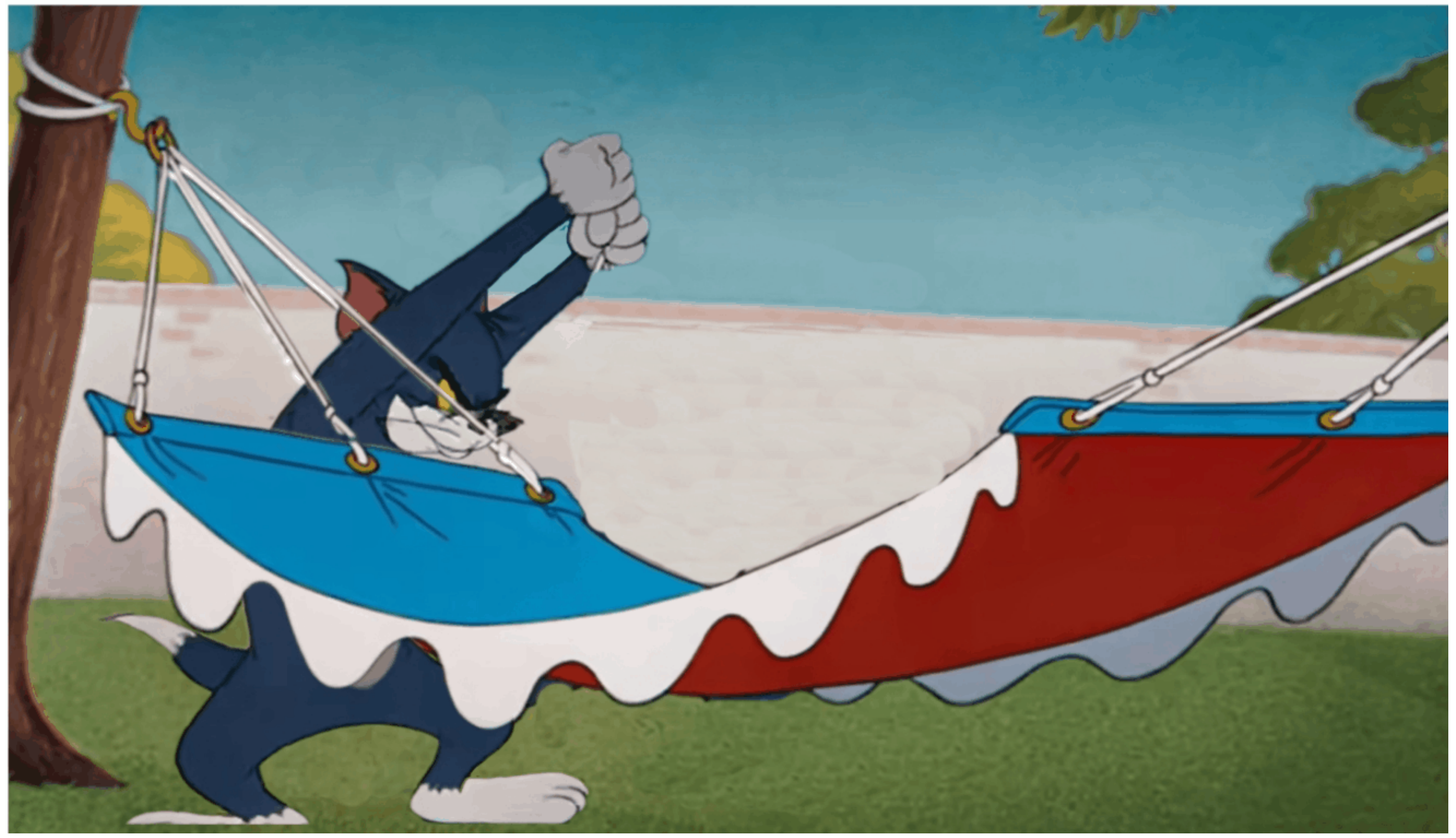


Direct preference optimization (DPO)

$$-\sum_{l=1}^{L}\eta[\mathcal{A}^t(\chi_o)]_m[\mathcal{K}^t(\chi_o,\chi_u^+)\mathcal{G}^t_{\mathrm{DPO+}}\boxed{-\mathcal{K}^t(\chi_o,\chi_u^-)\mathcal{G}^t_{\mathrm{DPO-}}}]_l$$

- Negative gradient helps the model not say y_u^-
- …but if y_u^- was already very unlikely, weird things happen!
- To decrease $\frac{e^{l_i}}{\sum_j e^{l_j}}$, can decrease numerator or increase denominator





Learning dynamics in DPO

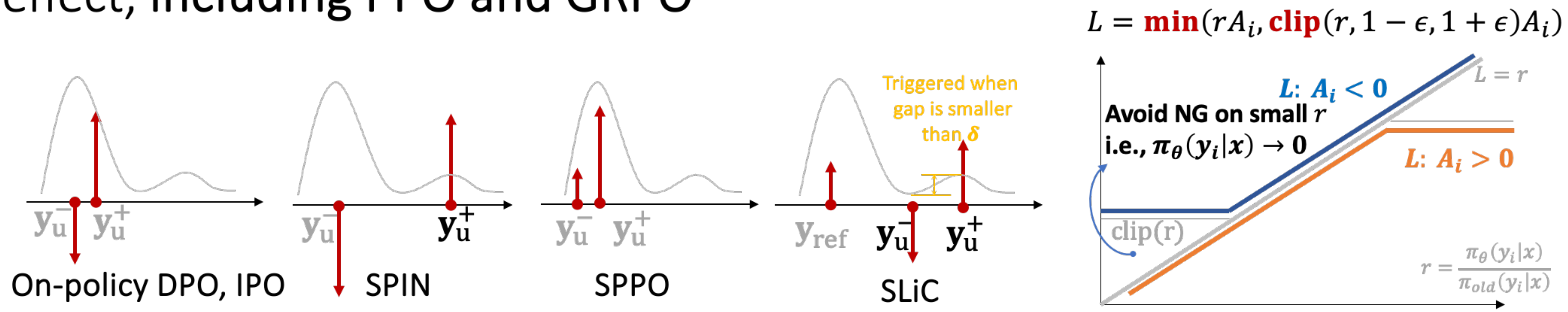
Desired response becomes *less* likely!

Basically *everything* becomes less likely

Greedy decoding becomes much more likely

Learning dynamics in DPO

- Fun fact: many effective methods (unintentionally) mitigate squeezing effect, including PPO and GRPO



Thanks!

- Active learning can help
  - For low label budgets, need representation-based methods
    - Smooth notions of representation help!
  - For high label budgets, need uncertainty-based methods
  - Uncertainty herding can smoothly adapt

- But only when “coverage” is a reasonable notion
  - i.e. not for selecting points for in-context learning

- Learning dynamics can help explain preference finetuning
  - Surprisingly simple negative gradient / squeezing effect explains DPO weirdness

- Overall lesson: thinking about theory can be useful :)