PONTIFICIA
UNIVERSIDAD
CATÓLICA DE
VALPARAÍSO

pucv.cl

Introduction to Active Learning

Escuela de Ingeniería Informática

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Introduction to Active Learning

Beyond Supervised Learning

Sources

- Lilian Weng, OpenAI, (2021-2022)
- Löffler, Hvingelby, Goschenhofer: „Learning with Limited Labelled Data“ (2024)

Material auxiliar

- Libros (Inglés)
  - Löffler, Hvingelby, Goschenhofer. “Learning with Limited Labelled Data”. Libro “Unlocking Artificial Intelligence: From Theory to Applications” (2024)
    - https://link.springer.com/chapter/10.1007/978-3-031-64832-8_4
- Web
  - Lilian Weng (OpenAI)
    - https://lilianweng.github.io/posts/2022-02-20-active-learning/
- Papers:
  - Ash et al., Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds, ICLR 2020
  - Sener and Savarese, Active Learning for Convolutional Neural Networks: A Core-Set Approach, ICLR 2018
  - Settles, Active Learning Literature Survey, 2012 doi 10.1016/j.matlet.2010.11.072

Motivación

1. Supervised Machine Learning models require large amounts of labeled training data.

2. Labeling (annotation process) is costly, takes time, requires knowledge, involves domain experts.

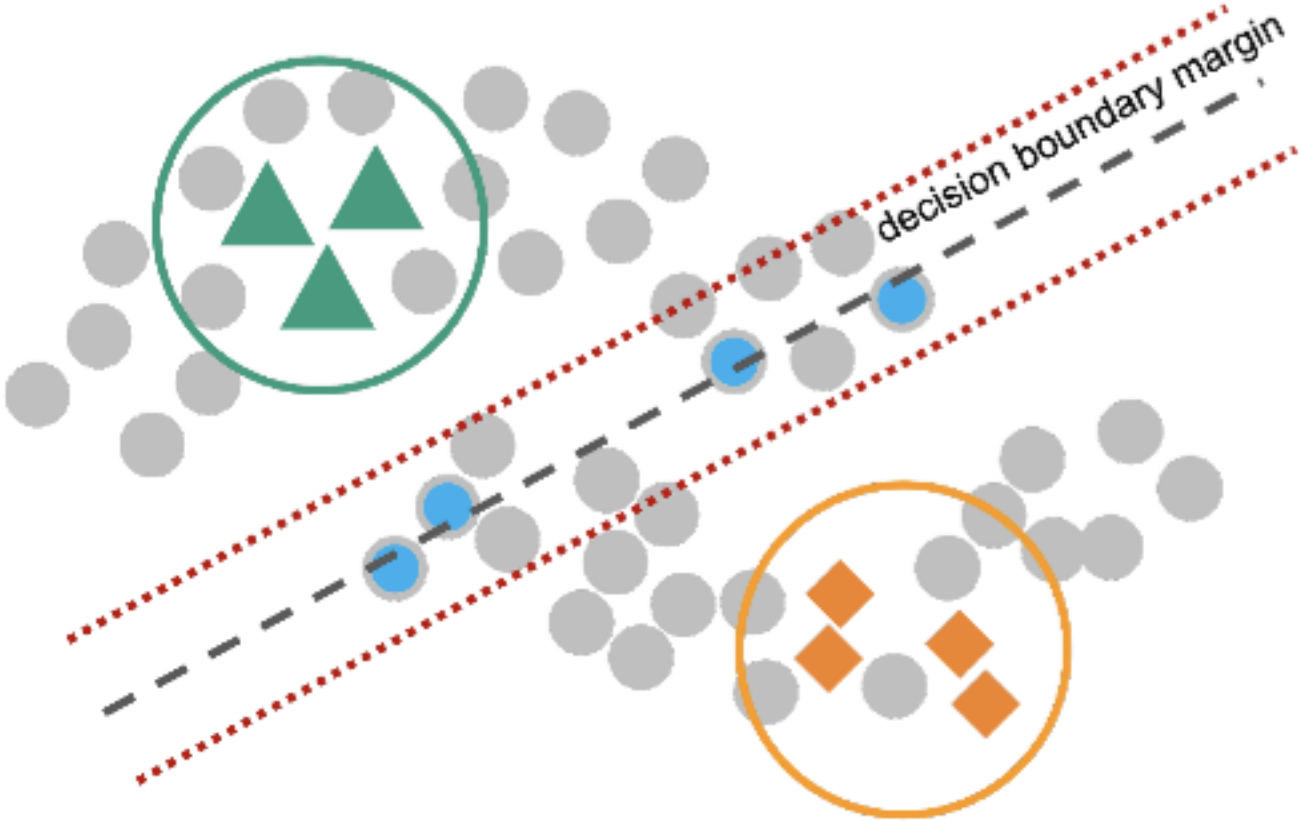
3. There is generally a larger amount of unlabeled data, that was (not yet) annotated by domain experts.

→ Machine Learning practitioners are often faced with limited labelled data scenarios

Question: How can we best leverage the information given in the unlabeled dataset next to the labeled data?

Motivación

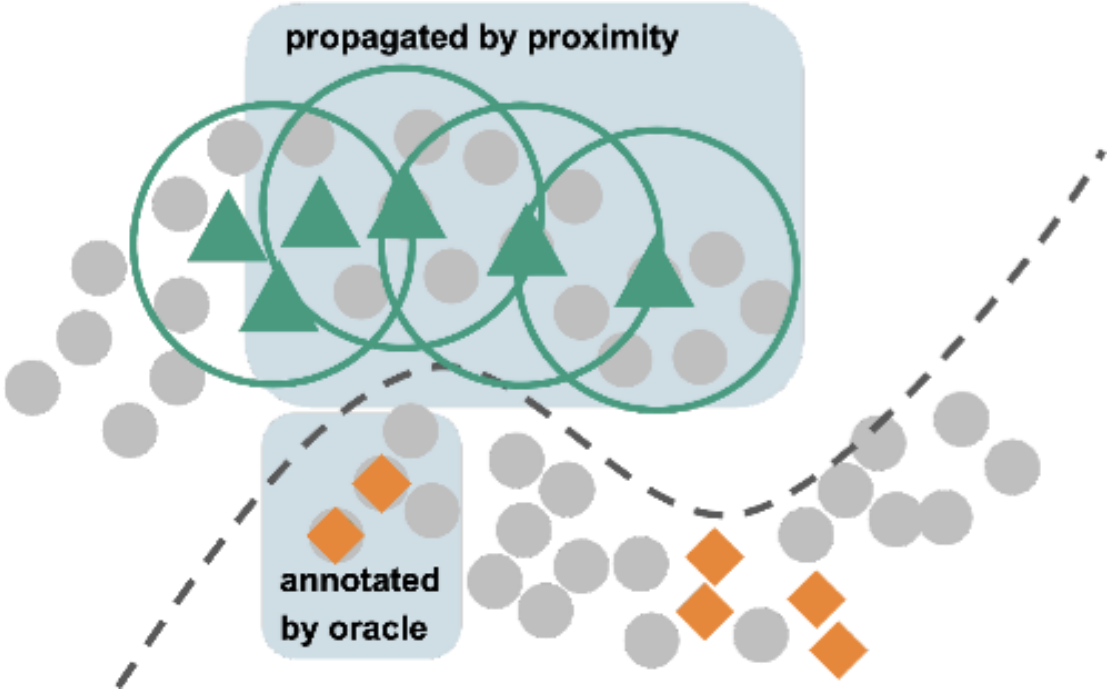
In limited labeled data scenarios, only a subset of samples is annotated (green, orange), and the majority is unlabeled.



Dos ideas relacionadas

1. Active Learning selects most informative samples to annotate, e.g., most uncertain samples that are close to the decision boundary

2. Semi-supervised Learning assumes that nearby unlabeled samples have the same class; label propagation uses the assumtion.



Dos ideas relacionadas

1. Active Learning
    - Select unlabeled samples that would support model training the most
    - Present them to (human) oracle to annotate unlabeled samples in model-driven way

2. Semi-supervised Learning
    - Directly incorporate unlabeled data in training process

Both approaches attack the problem from opposite directions:

While active methods attempt to **explore the unknown** aspects, semi-supervised methods **exploit what the model thinks it knows** about the unlabeled data.

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Active Learning

Beyond Supervised Learning

Active Learning (AL)

Active Learning

- Select unlabeled samples that would support model training the most
- Present them to (human) oracle to annotate unlabeled samples in model-driven way

Topics covered:

- Principles: Cost, loop, scenarios
- Acquisition functions
- Deep Active Learning
- Heuristics
  - Uncertainty Sampling
  - Diversity Sampling
    - Core-Set (Sener & Savarese, 2018)
  - Balanced Criteria
    - BADGE (Ash et al. 2020)
- How to use AL in this lecture's project?

Active Learning Cost

The reduction of cost is one of the primary reasons to use Active Learning. The costs arise from different sources, e.g., the annotation task's difficulty and the associated expensive expertise of the annotators. For example, medical doctors are expensive annotators.

Internet users are cheap annotators

It is still be a good idea to intelligently select the samples that they get to annotate.

Active Learning

In **pool-based active learning**, the learner receives an unlabeled dataset U sampled according to D_X and can request labels sampled according to $D_{Y|X}$ for any $x \in U$.

- X the instance space
- Y the label space. multiclass classification, so $Y = [K]$, $[K] := \{1, 2, \ldots, K\}$.
- Dataset D from which examples are drawn
  - D_X the unlabeled data
  - $D_{Y|X}$ the labels and examples
- Classifier $f : X \rightarrow Y$ maps examples to labels

The **goal** of pool-based active learning is to find a classifier with a small expected error using as few label queries as possible.

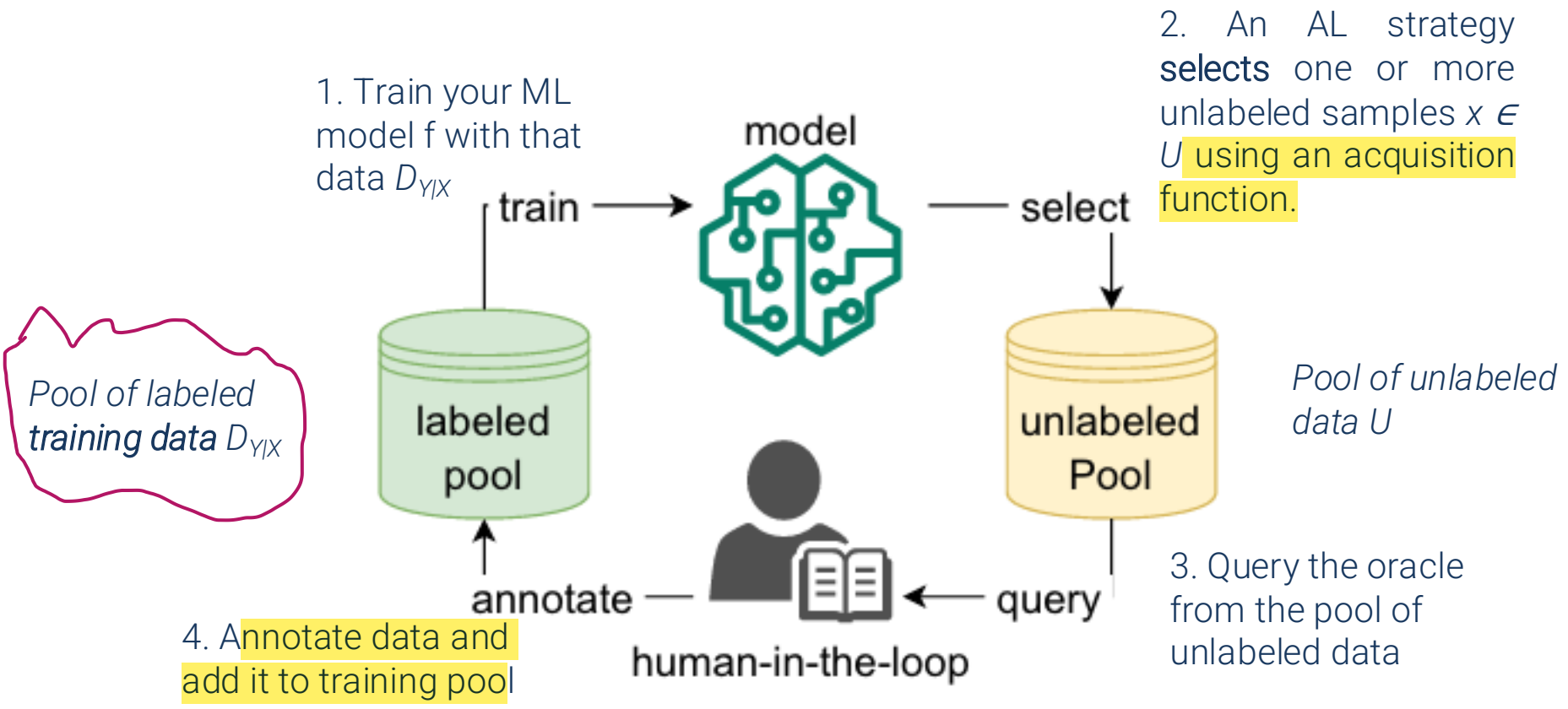
Setup: You have a large pool of unlabeled data, and you can query any instance from this pool to get its label.

Goal: Select the most informative samples from the pool.

You want the best-performing model with minimum labeled data, because labeling is expensive.

Active Learning Loop

The human-in-the-loop, that is also called the "oracle", is at the center of the AL loop.



Acquisition Functions

How does an AL strategy **select** one or more unlabeled samples?

Acquisition Functions: The method of how the AL algorithm chooses which unlabeled samples an oracle is queried with.

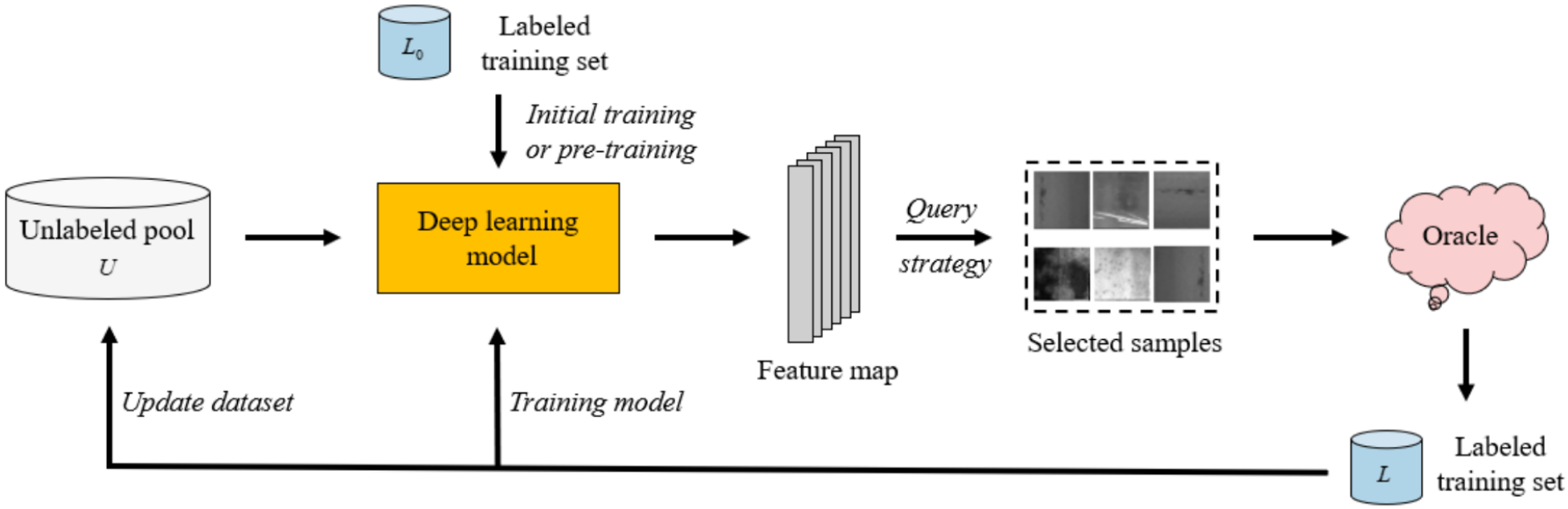
Do you an idea what samples would be the most interesting to annotate?

In Active Learning (AL), acquisition functions are strategies used by the model to decide which unlabeled instances to query (i.e., ask for labels). They are core to AL because they determine which data points are "most informative" for the learner

This refers to a cycle or loop where a deep learning model improves itself over time by actively selecting which unlabeled examples to label next.

Deep Active Learning (AL): Loop

In Deep Active Learning the acquisition function (query strategy) typically uses the extracted features (feature maps) of the Neural Network:



Sampling methods

Generally, we can group the sampling methods into three *main/basic* families.

1. Uncertainty Sampling
2. Diversity Sampling
3. Balanced Sampling that combines Uncertainty and Diversity Sampling

Uncertainty-based sampling (Softmax)

Select the least confident predictions to guide the annotators. These are $\hat{x} = argmax_x\big(1 - P(\hat{y}|x)\big)$ where $\hat{y} = argmax_y(P(y|x))$ is the most likely label $\hat{y}$ for an unlabeled sample x.

$(1 - p(3|x) = 0.4$

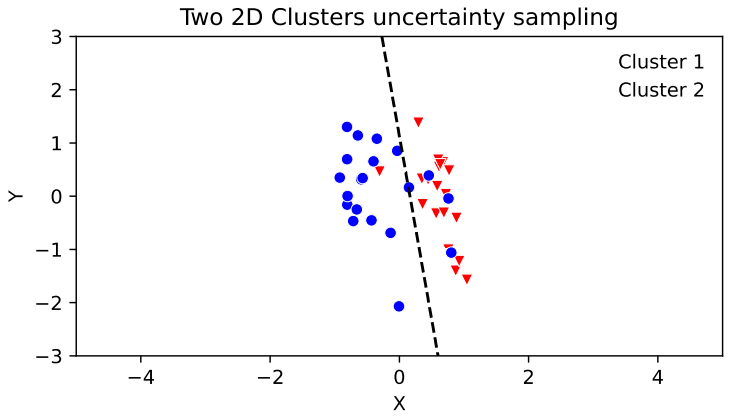
$(1 - p(3|x) = 0.6$

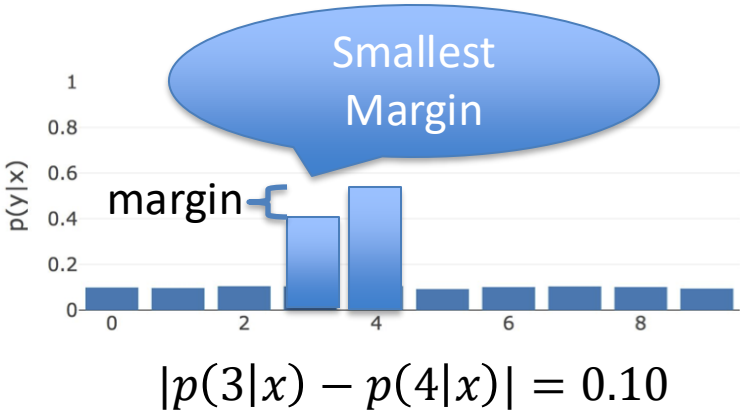
But: This tends to sample outliers, for which the model has the least information.

Uncertainty-based sampling (Softmax)

We can also sample the Smallest Margin between two predictions.

- This tends to sample along the decision boundary between two classes:





$$|p(3|x) - p(4|x)| = 0.10$$

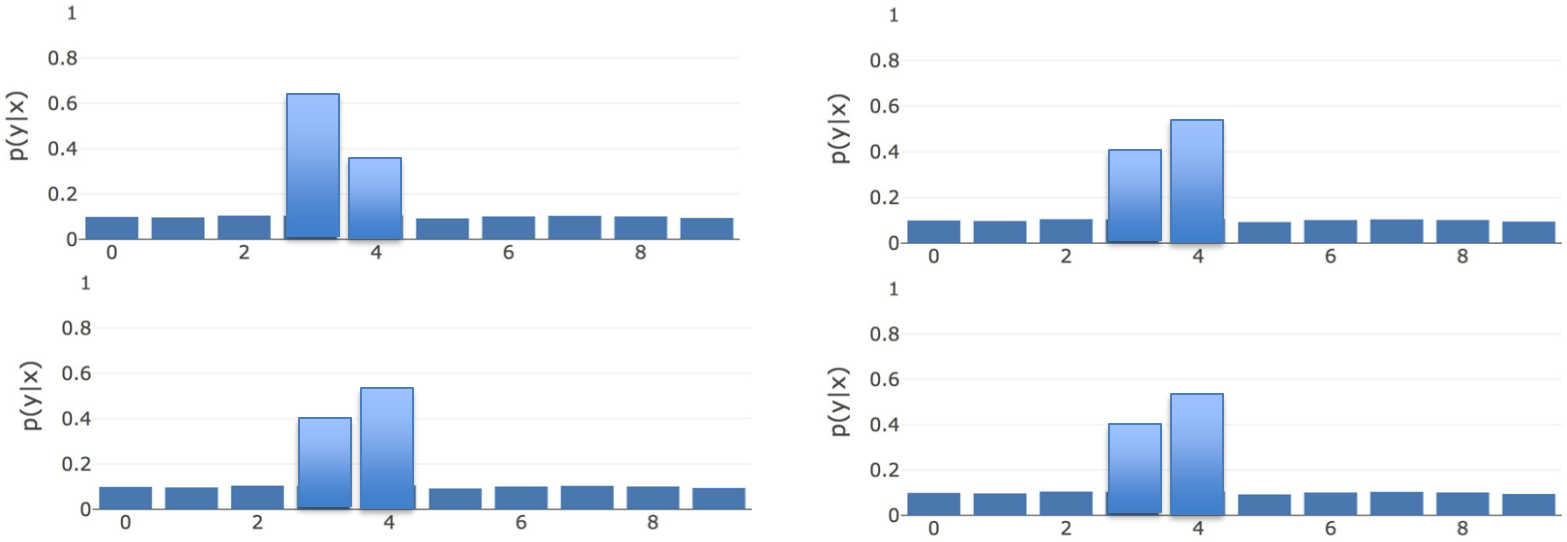
Uncertainty-based sampling (Softmax)

Besides using Softmax Probabilities one may use an ensemble of neural networks and measure their disagreement.

Variation Ratio (Vr) is the proportion of predictions from our ensemble, that are not the modal class prediction

Example: N=4 networks predict [3,4,4,4], here m=1 are not modal class

$$\mathrm{Vr}(x) = 1 - \frac{m}{N} = 1 - \frac{1}{N} = 0.25$$

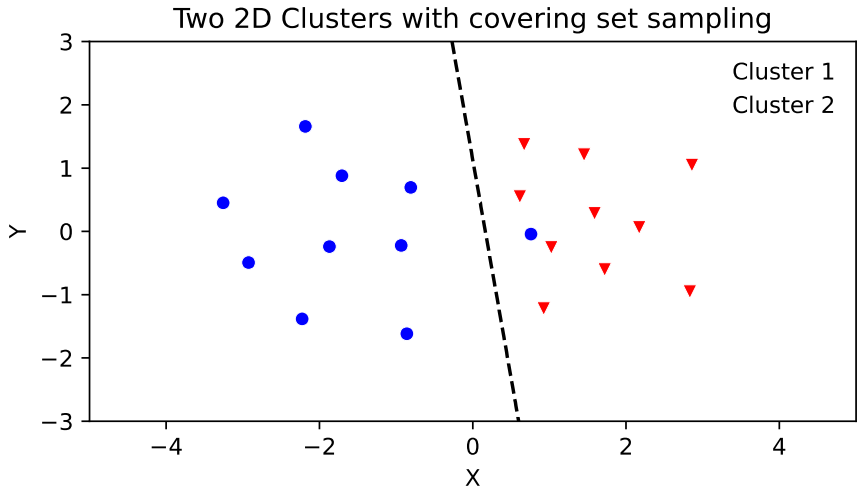


Diversity-based sampling: clustering of data

Idea: sample a representative set of data from all parts of the data distribution.

This promises to be useful for acquiring more than 1 sample at a time
→ good for annotating batches
→ Redundant information, slow learning

Example: find covering sets for data clusters, so that the clusters' members are equidistant among each other:



Core-Set (Sener & Savarese, 2018)

Limitations: Active Learning Heuristics are not effective when applied to CNNs in batch settings.

- A single data point will not have statistically significant impact on the model (only local optimization problem).
- Infeasible to train as many models as there are data points
  - large scale problems + large scale Neural Networks

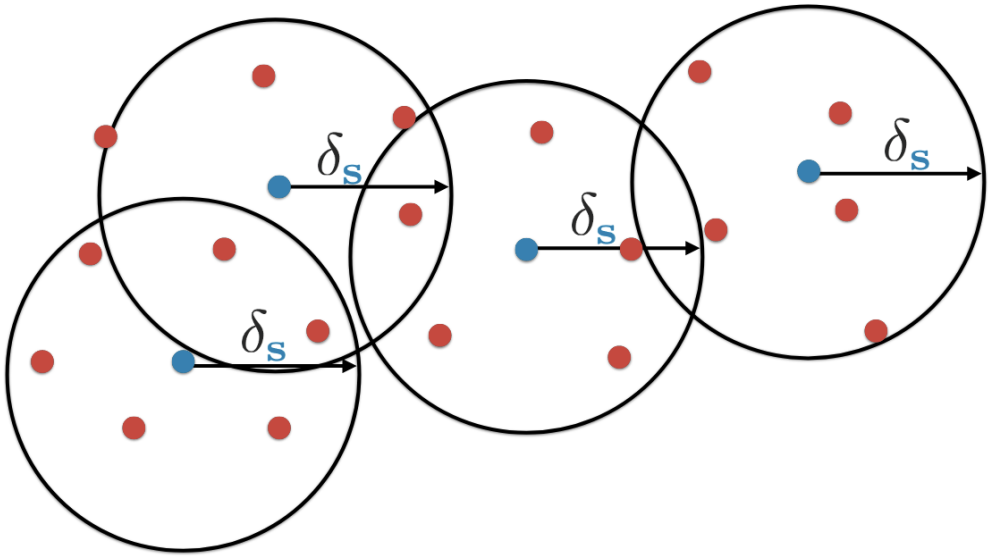
Proposal: Formulate problem as *core-set selection*

What does this mean?

> *«[...] choosing set of points such that a model learned over the selected subset is competitive for the remaining data points.»*

Visualization

Select points s and remainder of the dataset $[n] \backslash s$, then Sener & Savarese show that if s is the δ_s cover of the dataset, the active learning problem depends on a good selection of s that can cover the entire set



Theoretical error bound of loss

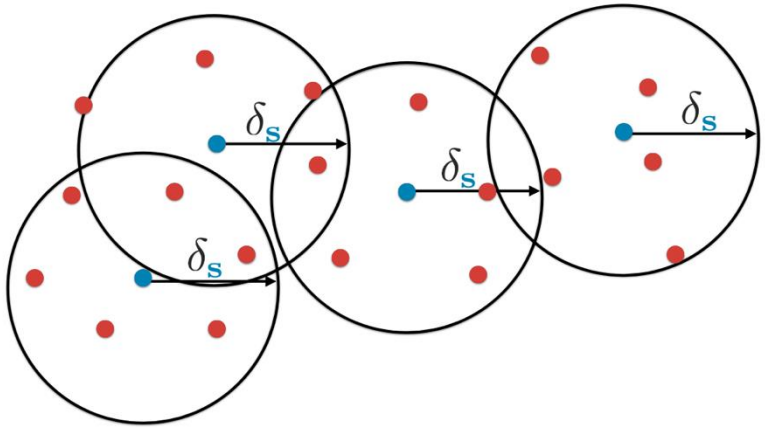
Intuitively, bound error of core-set selection with radius size

1. *Loss over the whole dataset – loss over the selected subset* s

$$\left| \frac{1}{n} \sum_{i \in [n]} l(\mathbf{x}_i, y_i, A_{\mathbf{S}}) - \frac{1}{|\mathbf{S}|} \sum_{j \in \mathbf{S}} l(\mathbf{x}_j, y_j; A_{\mathbf{S}}) \right| \leq \mathcal{O}(\delta_{\mathbf{S}}) + \mathcal{O}\left(\sqrt{\frac{1}{n}}\right)$$

is bounded by

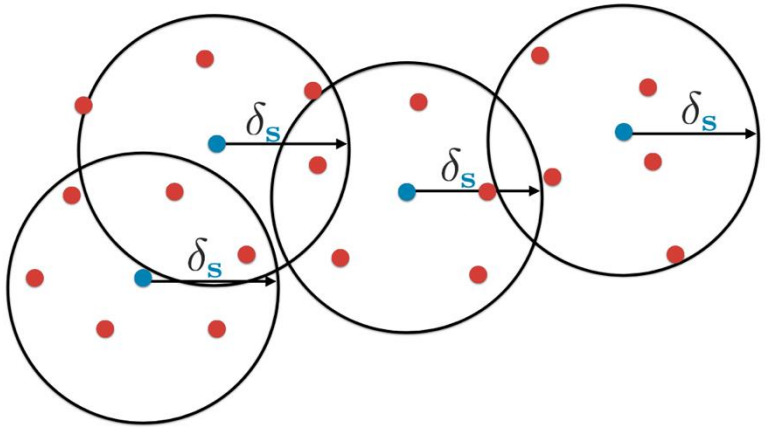
2. *Covering radius + term that goes to zero with larger 'n'*



Implemention as k-Center-Problem

Problem: combinatorial optimization of center selection and radius size is NP-hard.

Choose centers s such that the maximal distance of any other point to its nearest center point is minimal. This maximal distance is called the coverage radius δ_s.



Heuristic Algorithm (1)
Greedy initialization

Intead of an optimal set s and radius δ_s we use a greedy heuristic to come to these initial s and δ_s.

Algorithm 1 k-Center-Greedy

Input: data $\mathbf{x}_i$, existing pool $\mathbf{s}^0$ and a budget b
Initialize $\mathbf{s} = \mathbf{s}^0$
repeat
  $u = \arg\max_{i \in [n] \setminus \mathbf{s}} \min_{j \in \mathbf{s}} \Delta(\mathbf{x}_i, \mathbf{x}_j)$
  $\mathbf{s} = \mathbf{s} \cup \{u\}$
until $|\mathbf{s}| = b + |\mathbf{s}^0|$
return $\mathbf{s} \setminus \mathbf{s}^0$

Heuristic Algorithm (2)
Robust k-Center

Using initial s and δ_s and optimize them with Mixed Integer Programming (MIP) and Binary Search.

1. Search better δ_s radius iteratively
    1. Binary Search between $\delta_s/2$ and δ_s (middle point is new upper/lower δ_s)
2. Check if conditions are still fulfilled (MIP)
    1. Coverage
    2. Budget
    3. Number of uncovered samples
3. Convergence when lower and upper radius are close to each other or the same value

In step 2, relax coverage condition by allowing a small number of points to no be covered by radius.

Using Core-Set for active learning

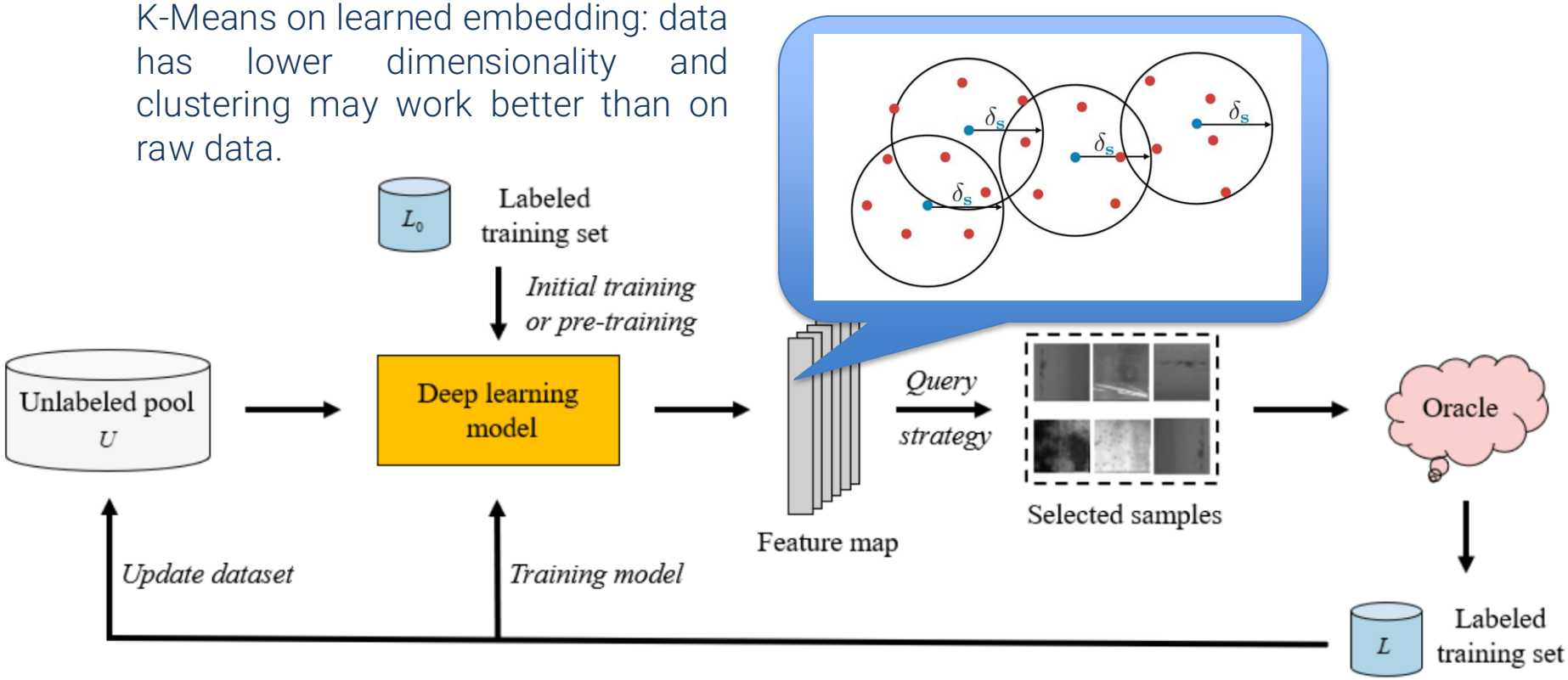
The selected points s are those data sampes that the annotator will label.

Every iteration of the active learning loop:

1. Train model fully on labeled pool
2. Find new s and δ_s
3. Annotate new data

Core-Set AL clusters in embedding space

K-Means on learned embedding: data has lower dimensionality and clustering may work better than on raw data.

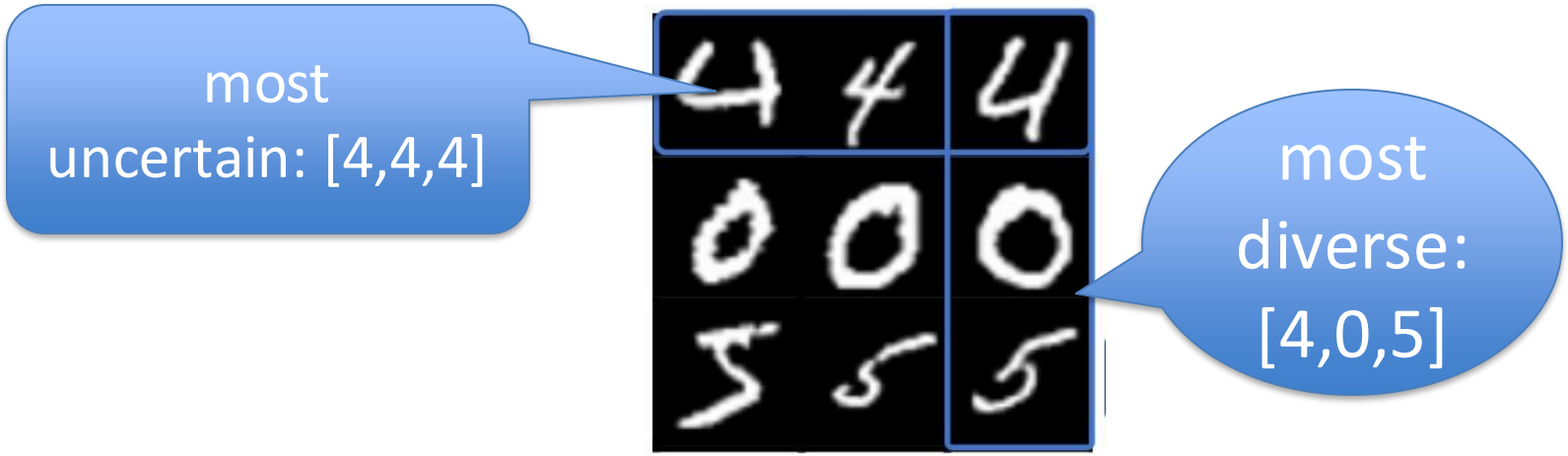


Balancing of uncertainty and diversity

Pure Uncertainty and Diversity Sampling each has disadvantages.

Idea: **combine both concepts** as follows

1. Find a large amount of uncertain samples
2. From this subset, select the most diverse samples



BADGE (Ash et al. 2020)

Idea: select a batch of samples to annotate that is both diverse and contains uncertain samples.

Batch Active learning by Diverse Gradient Embeddings (BADGE)

1. Represent points in a hallucinated *gradient space*
2. Sample groups of points that are *disparate* and have a *high magnitude*

BADGE (Ash et al. 2020)

Idea: select a batch of samples to annotate that is both diverse and contains uncertain samples.

Batch Active learning by Diverse Gradient Embeddings (BADGE)

1. Represent points in a hallucinated *gradient space*

   For each unlabeled sample, calculate a vector in relation to the current Deep Neural Network.

   vector ← loss of sample wrt. pseudo-label (most likely class). Hence "hallucinated" gradient and not the true, real gradient.

BADGE (Ash et al. 2020)

Idea: select a batch of samples to annotate that is both diverse and contains uncertain samples.

Batch Active learning by Diverse Gradient Embeddings (BADGE)

2. Sample groups of points that are *disparate* and have a *high magnitude*

Hallucinated gradient conveys uncertainty (large loss magnitude → high uncertainty). A certain model would have small loss.

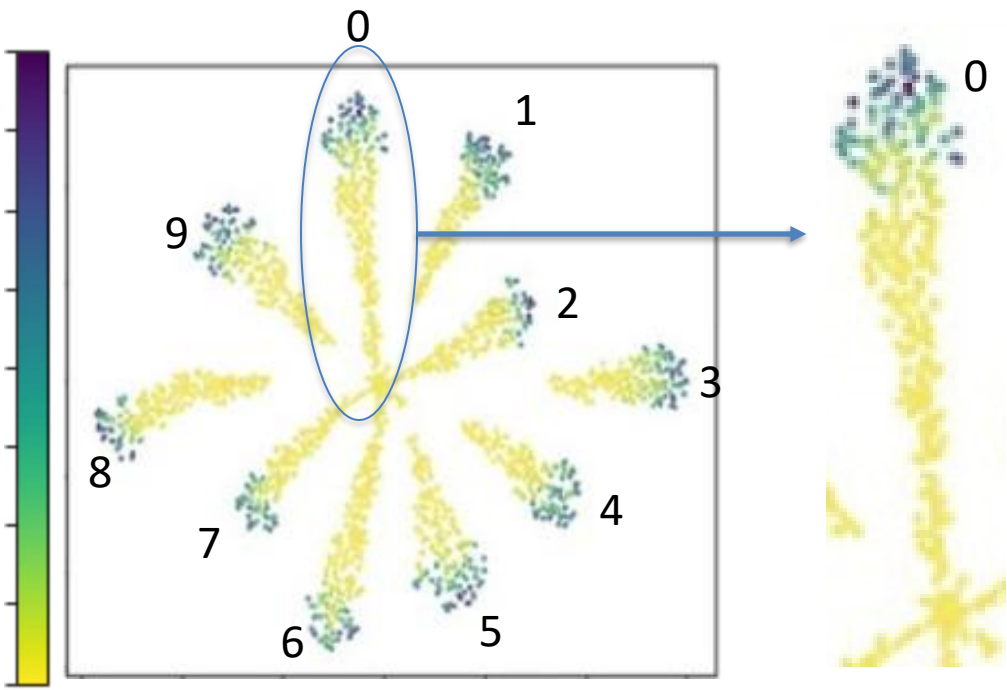
Diverse gradient embedding vectors indicate diversity in sample space

BADGE (Ash et al. 2020)

BADGE visualized: hallucinated Gradient Embedding for MNIST dataset (10 classes) using a CNN. Bright yellow shows certain predictions, yellow shows high magnitude and

High magnitude (uncertain)

Low magnitude (certain)



These are gradient vectors with similar value and high loss magnitude

These are samples with Loss near zero, already learned by CNN

Image from Ash et al., 2020

BADGE (Ash et al. 2020)

Algorithm 1 BADGE: Batch Active learning by Diverse Gradient Embeddings

Require: Neural network $f(x;\theta)$, unlabeled pool of examples U, initial number of examples M, number of iterations T, number of examples in a batch B.

Labeled dataset $S \leftarrow M$ examples drawn uniformly at random from U together with queried labels.
Train an initial model θ_1 on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.
for $t = 1, 2, \ldots, T$: **do**
    For all examples x in $U \setminus S$:
        1. Compute its hypothetical label $\hat{y}(x) = h_{\theta_t}(x)$.
        2. Compute gradient embedding $g_x = \frac{\partial}{\partial \theta_{\text{out}}} \ell_{\text{CE}}(f(x;\theta), \hat{y}(x))|_{\theta=\theta_t}$, where θ_{out} refers to parameters of the final (output) layer.
    Compute S_t, a random subset of $U \setminus S$, using the k-MEANS++ seeding algorithm on $\{g_x : x \in U \setminus S\}$ and query for their labels.
    $S \leftarrow S \cup S_t$.
    Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.
end for
return Final model θ_{T+1}.

BADGE (Ash et al. 2

- Initialize algorithm: draw initial set *M* of *x from U*.
- Set budget: *T* iterations with *B* samples per each Batch
- Train first model, label first samples S

Algorithm 1 BADGE: Batch Active learni by Diverse Gradient Embeddings

Require: Neural network $f(x;\theta)$, unlabeled pool of examples U, initial number of examples M, number of iterations T, number of examples in a batch B.

1: Labeled dataset $S \leftarrow M$ examples drawn uniformly at random from U together with queried labels.
2: Train an initial model θ_1 on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.
3: **for** $t = 1, 2, \ldots, T$: **do**
4: For all examples x in $U \setminus S$:
1. Compute its hypothetical label $\hat{y}(x) = h_{\theta_t}(x)$.
2. Compute gradient embedding $g_x = \frac{\partial}{\partial \theta_{\text{out}}} \ell_{\text{CE}}(f(x;\theta), \hat{y}(x))|_{\theta=\theta_t}$, where θ_{out} refers to parameters of the final (output) layer.
5: Compute S_t, a random subset of $U \setminus S$, using the k-MEANS++ seeding algorithm on $\{g_x : x \in U \setminus S\}$ and query for their labels.
6: $S \leftarrow S \cup S_t$.
7: Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.
8: **end for**
9: **return** Final model θ_{T+1}.

BADGE (Ash et al. 2020)

- Predict pseudo-label (hypothetical) using the model
- Calculate gradient embedding vector using the loss function cross-entropy
- Vector is sampled from final layer before SoftMax activation

Algorithm 1 BADGE: Batch Ac

Require: Neural network $f(x; \theta$ mples M, number of iterations T, number of exam

1: Labeled dataset $S \leftarrow M$ examp with queried labels.
2: Train an initial model θ_1 on S by mini $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.
3: **for** $t = 1, 2, \ldots, T$: **do**
4: For all examples x in $U \setminus S$:
   1. Compute its hypothetical label $\hat{y}(x) = h_{\theta_t}(x)$.
   2. Compute gradient embedding $g_x = \frac{\partial}{\partial \theta_{\text{out}}} \ell_{\text{CE}}(f(x;\theta), \hat{y}(x))|_{\theta=\theta_t}$, where θ_{out} refers to parameters of the final (output) layer.
5: Compute S_t, a random subset of $U \setminus S$, using the k-MEANS++ seeding algorithm on $\{g_x : x \in U \setminus S\}$ and query for their labels.
6: $S \leftarrow S \cup S_t$.
7: Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.
8: **end for**
9: **return** Final model θ_{T+1}.

BADGE (Ash et al. 2020)

- Sample batch of samples S_t similar to Core-Set Sampling (k-means++ or optimal)
  - high magnitude and diverse vectors
- Query labels for samples S_t and add to training dataset

Algorithm 1 BADGE: Batch Ac

Require: Neural network $f(x; \theta$ mples M, number of iterations T, number of exam

1: Labeled dataset $S \leftarrow M$ exa th queried labels.
2: Train an initial model θ_1 on
3: **for** $t = 1, 2, \ldots, T$: **do**
4: For all examples x in $U \setminus$
1. Compute its hypoth
2. Compute gradient embeddi $\frac{\partial}{\partial \theta_{\text{out}}} \ell_{\text{CE}}(f(x; \theta), \hat{y}(x))|_{\theta=\theta_t}$, where θ_{out} refers to parameters of the final (output) la
5: Compute S_t, a random subset of $U \setminus S$, using the k-MEANS++ seeding algorithm on $\{g_x : x \in U \setminus S\}$ and query for their labels.
6: $S \leftarrow S \cup S_t$.
7: Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x; \theta), y)]$.
8: **end for**
9: **return** Final model θ_{T+1}.

BADGE (Ash et al. 2020)

Algorithm 1 BADGE: Batch Active learning by Diverse Gradient Embeddings

Require: Neural network $f(x; \theta)$, unlabeled pool of examples U, initial number of examples M, number of iterations T, number of exam

1: Labeled dataset $S \leftarrow M$ exa … th queried labels.
2: Train an initial model θ_1 on …
3: **for** $t = 1, 2, \ldots, T$: **do**
4: For all examples x in $U \setminus$ …
1. Compute its hypo …
2. Compute gradient … θ_{out} refers to parameters of the final (…
5: Compute S_t, a random subset of $U \setminus$ … k-MEANS++ seeding algorithm on $\{g_x : x \in U \setminus S\}$ and query for their labels.
6: $S \leftarrow S \cup S_t$.
7: Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x; \theta), y)]$.
8: **end for**
9: **return** Final model θ_{T+1}.

- Train the model on the additional data
- Continue for *T* iterations (until budget is depleted)
- Finally, return the trained model

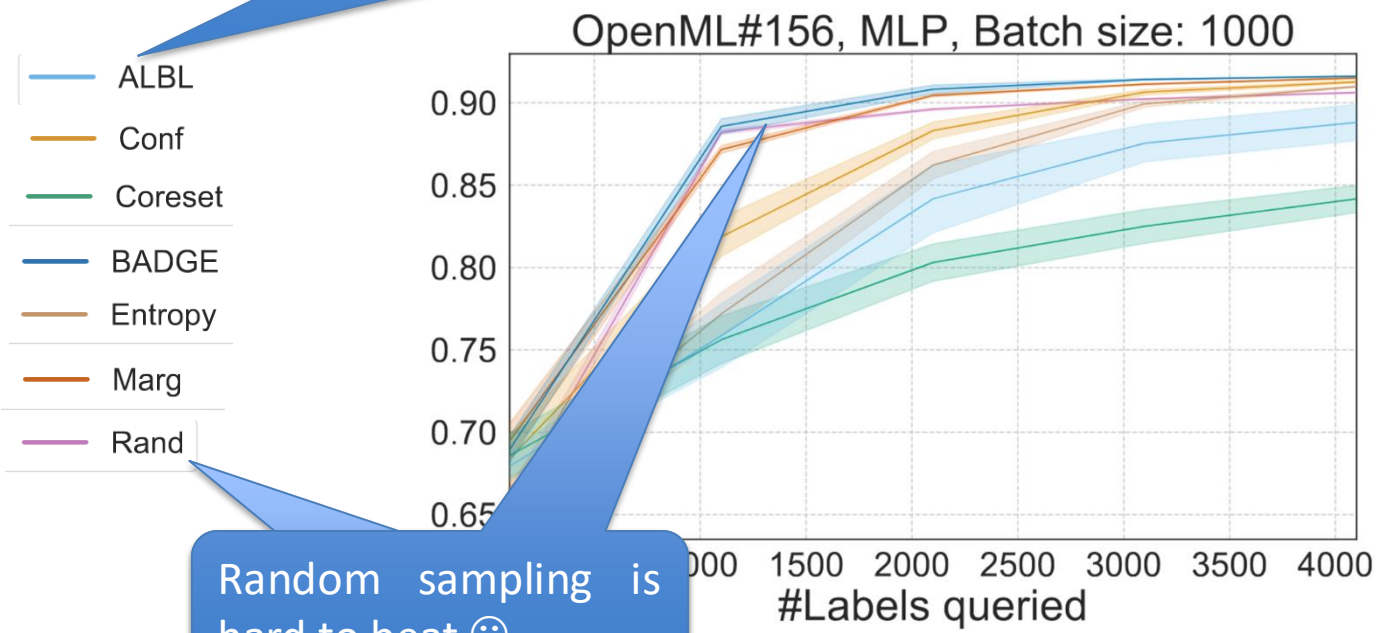
Evaluating Active Learning

Goal: … number of anno…

- W… …rge difference b…

(Dataset: Ope… …odel MLP, 1000 samples per Batch acquisition)

- Uncetainty: CONFidence, Entropy, MARGin
- Diversity: Coreset
- Mix: BADGE
- ALBL (AL by Learning)



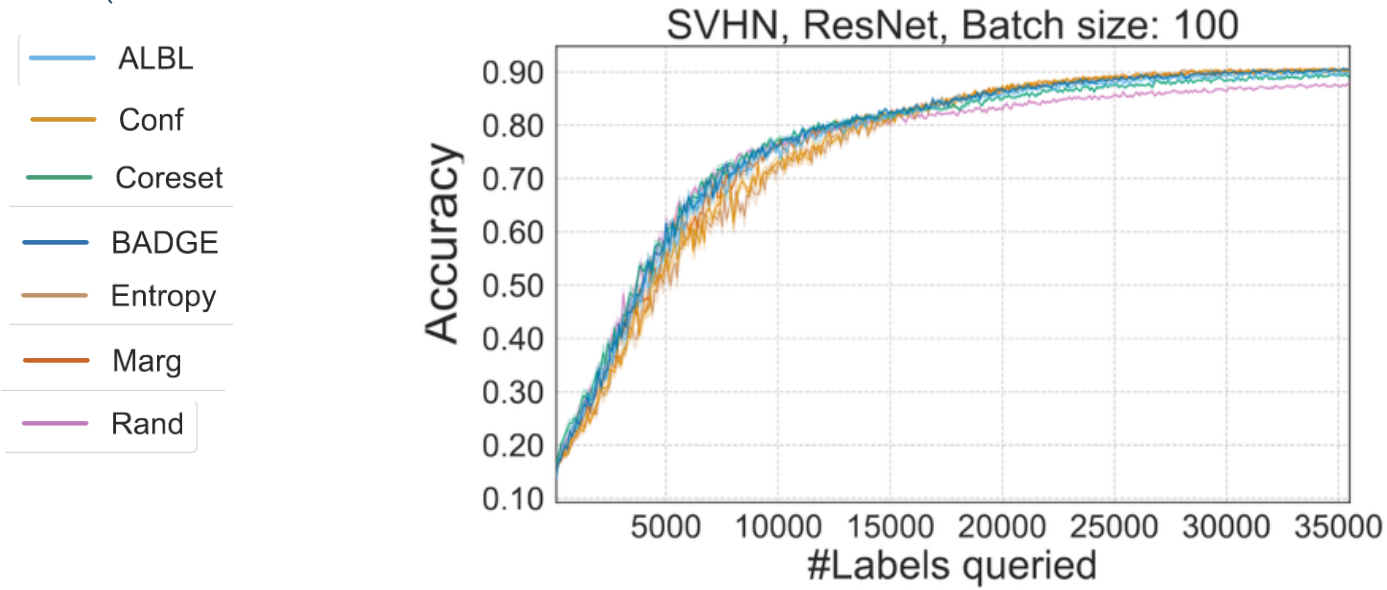
Random sampling is hard to beat ☺

Evaluating Active Learning

Issues with active learning:

1. Larger models or datasets are often very hard to optimize and training gets very noisy.
2. Smaller batch sizes make selection less dependent on good batch selection and the results are barely different

(Dataset: SVHN, Model ResNet, 100 samples per Batch acquisition)

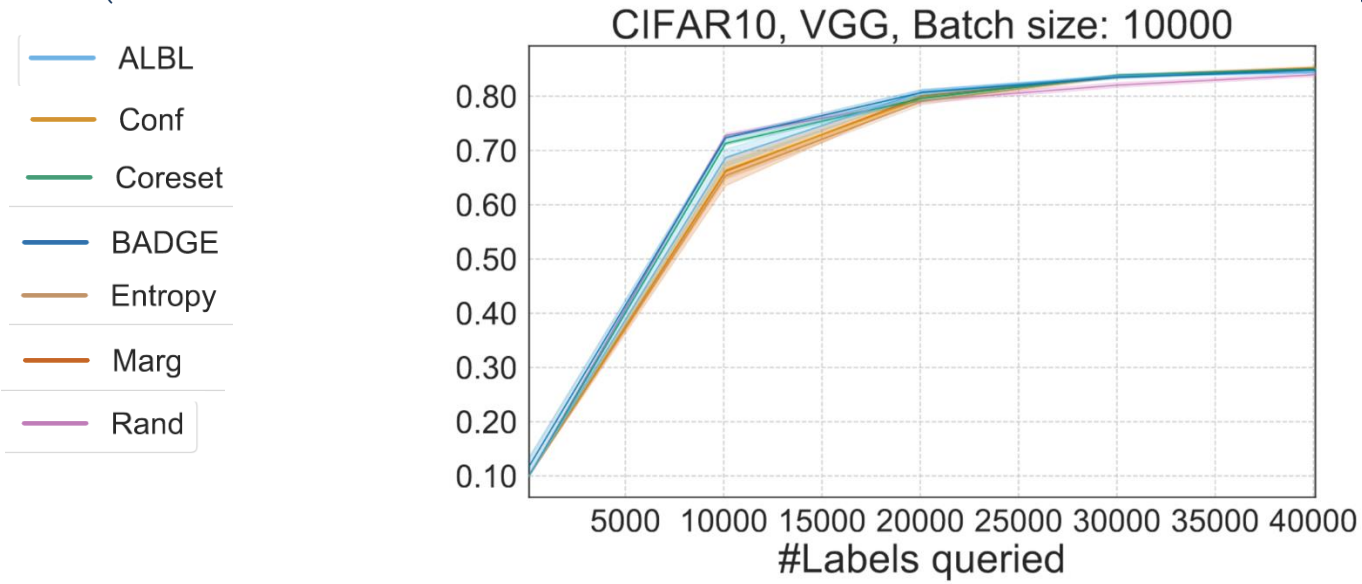


Evaluating Active Learning

Issues with active learning:

1. *Very large batch sizes (10.000) diverge to random sampling, (or random sampling becomes more diverse)*

(Dataset: CIFAR10, Model VGG, 10.000 samples per Batch acquisition)



Active Learning in practice

How can you use Active Learning in this lecture's project? Given a model to classify the Tool Tracking data, implement Active Learning as follows:

1. Prepare the dataset
    1. Randomly select small subset of labeled data (train/val/test)
    2. Use remaining data *as if it is unlabeled*
2. Implement Active Learning Strategies
    1. modAL library implements, e.g., uncertainty sampling.
    2. Deep Active Learning in https://github.com/JordanAsh/badge
3. Implement AL Loop
    1. Train on labelled data
    2. Use AL to pick unlabeled data to annotate
        1. Annotate (using original labels from original dataset)
    3. Retrain model, evaluate it on annotated validation data
4. Stop when out of budget
5. Evaluate on Test dataset

Experiment with hyper parameters: *batch size*, *AL strategies*, different *Neural Networks* (MLP, LSTM, RNN, TCN, Transformers,..), etc.

Resumen

Topics covered:

- Principles: Cost, loop, scenarios
- Acquisition functions
- Deep Active Learning
- Heuristics
  - Uncertainty Sampling
  - Diversity Sampling
    - Core-Set (Sener & Savarese, 2018)
  - Balanced Criteria
    - BADGE (Ash et al. 2020)
- How to use AL in this lecture's project?