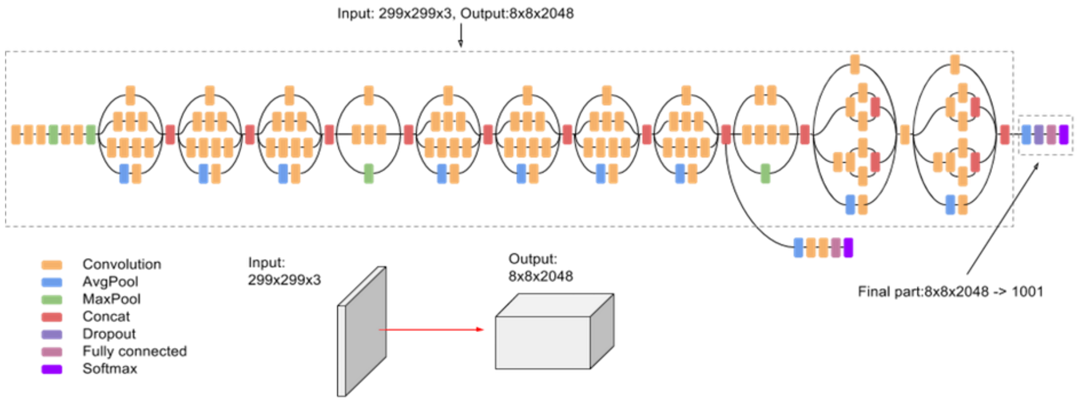
Distributing Active Learning Algorithms

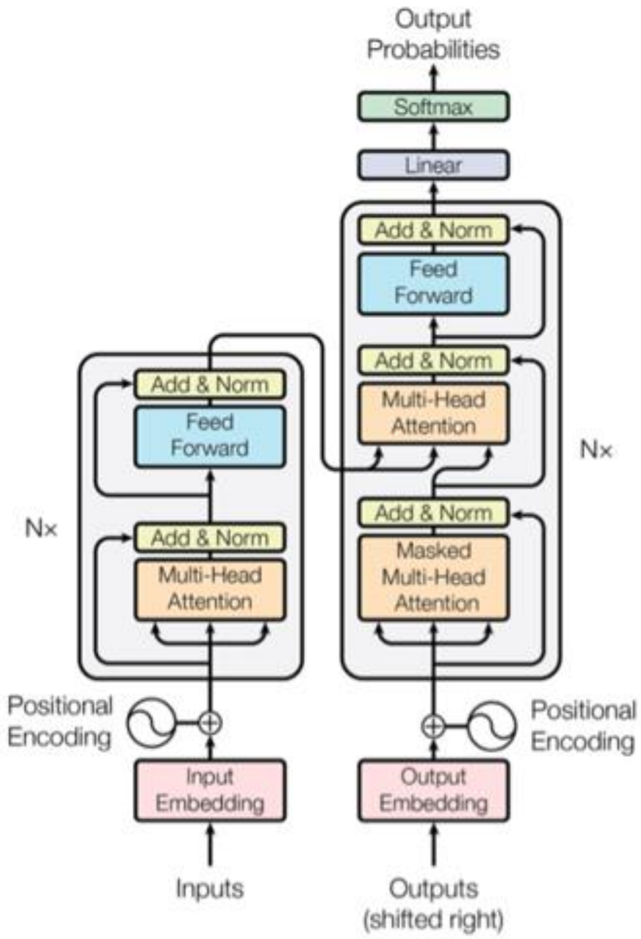
Syed Mostofa Monsur **Muhammad Abdullah Adnan**

Bangladesh University of Engineering and Technology

Active Learning

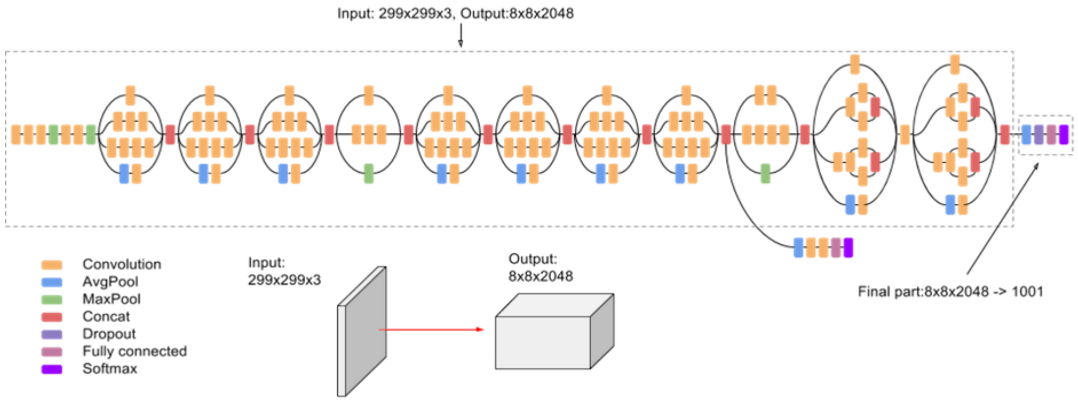
- Machine learning models require **huge amount of labeled data**

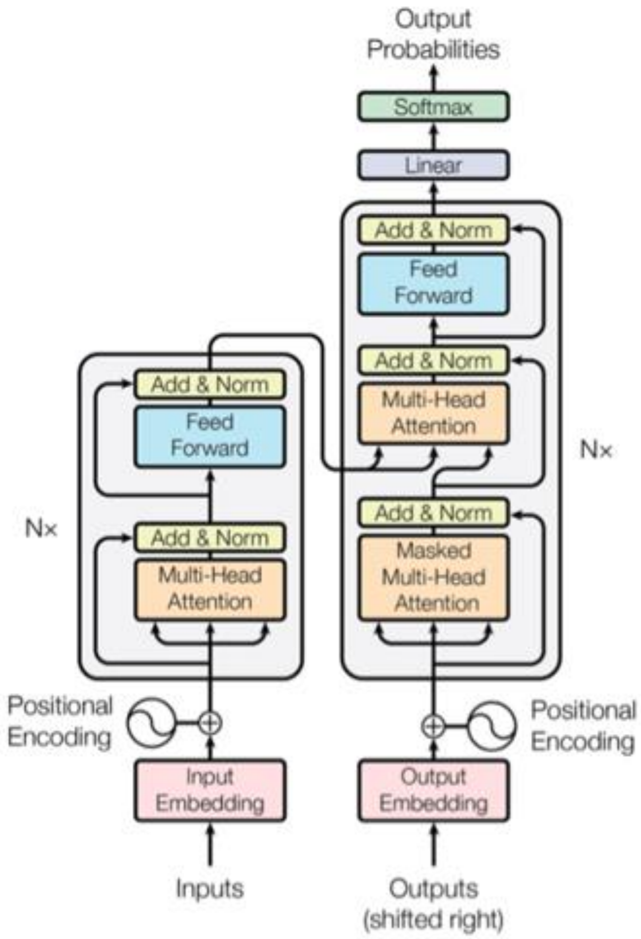




Active Learning

- Machine learning models require **huge amount of labeled data**





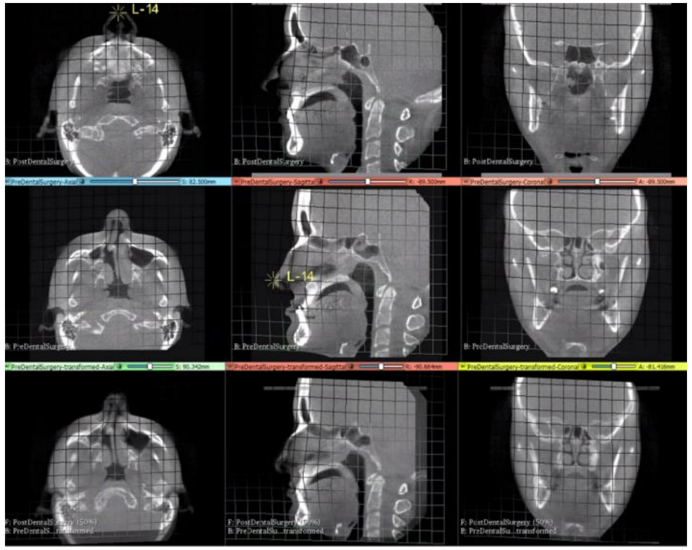
- Today a lot of data is plentiful and cheap
  - documents off the web
  - speech samples
  - images and video

Active Learning

Cannot label all the data

- Difficult to label

  Domain expert needed. **Expert physician labels medical imaging data**



Active Learning

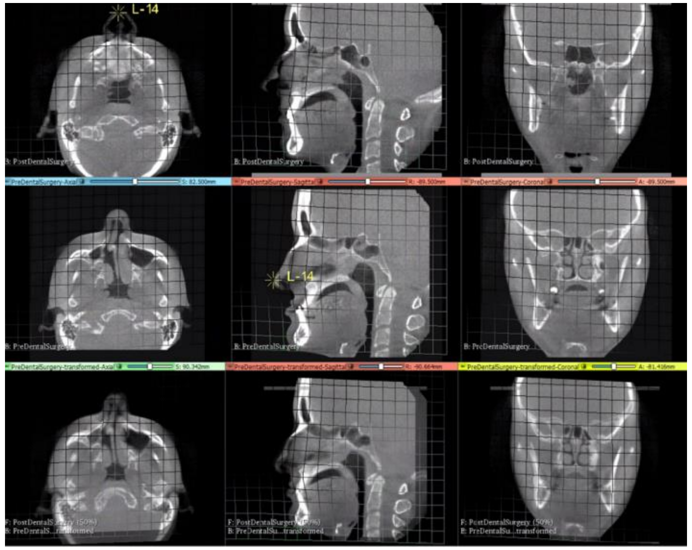
Cannot label all the data

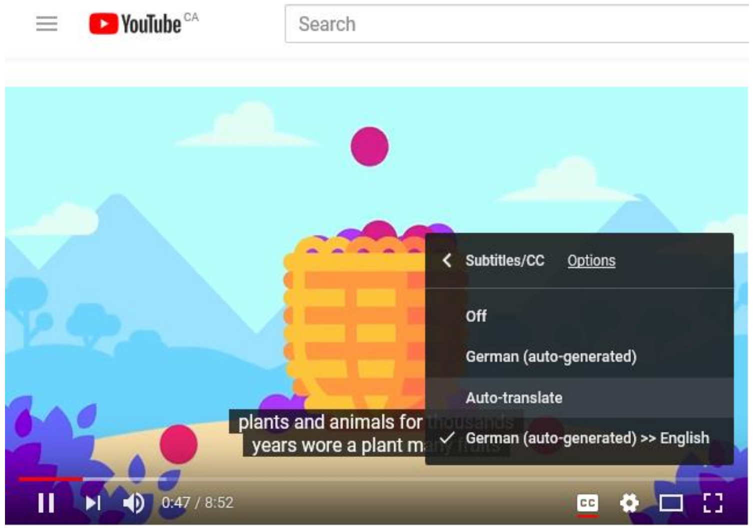
- Difficult to label

  Domain expert needed. **Expert physician labels medical imaging data**

- Time consuming

  may take hours/days. **1 minute of speech may take more than several minutes to label**

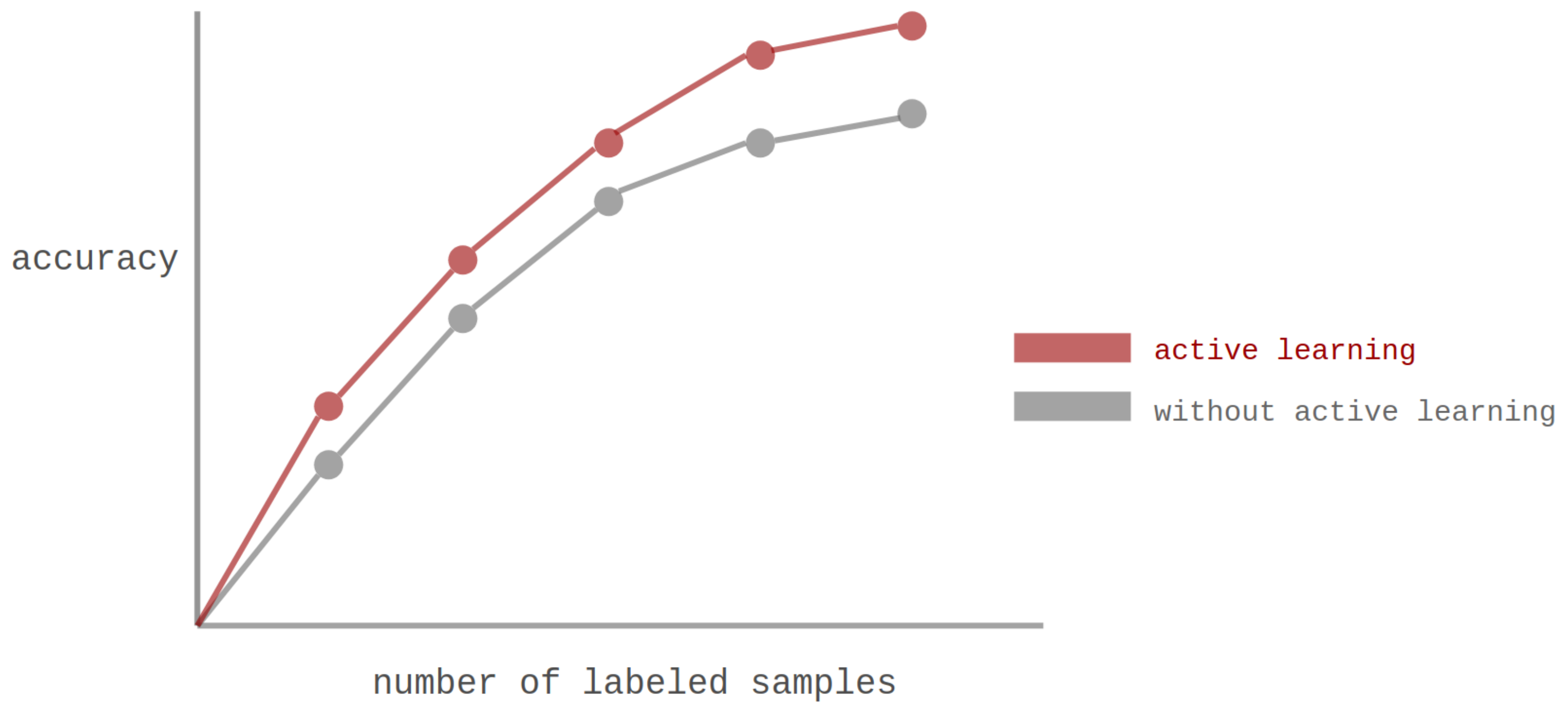




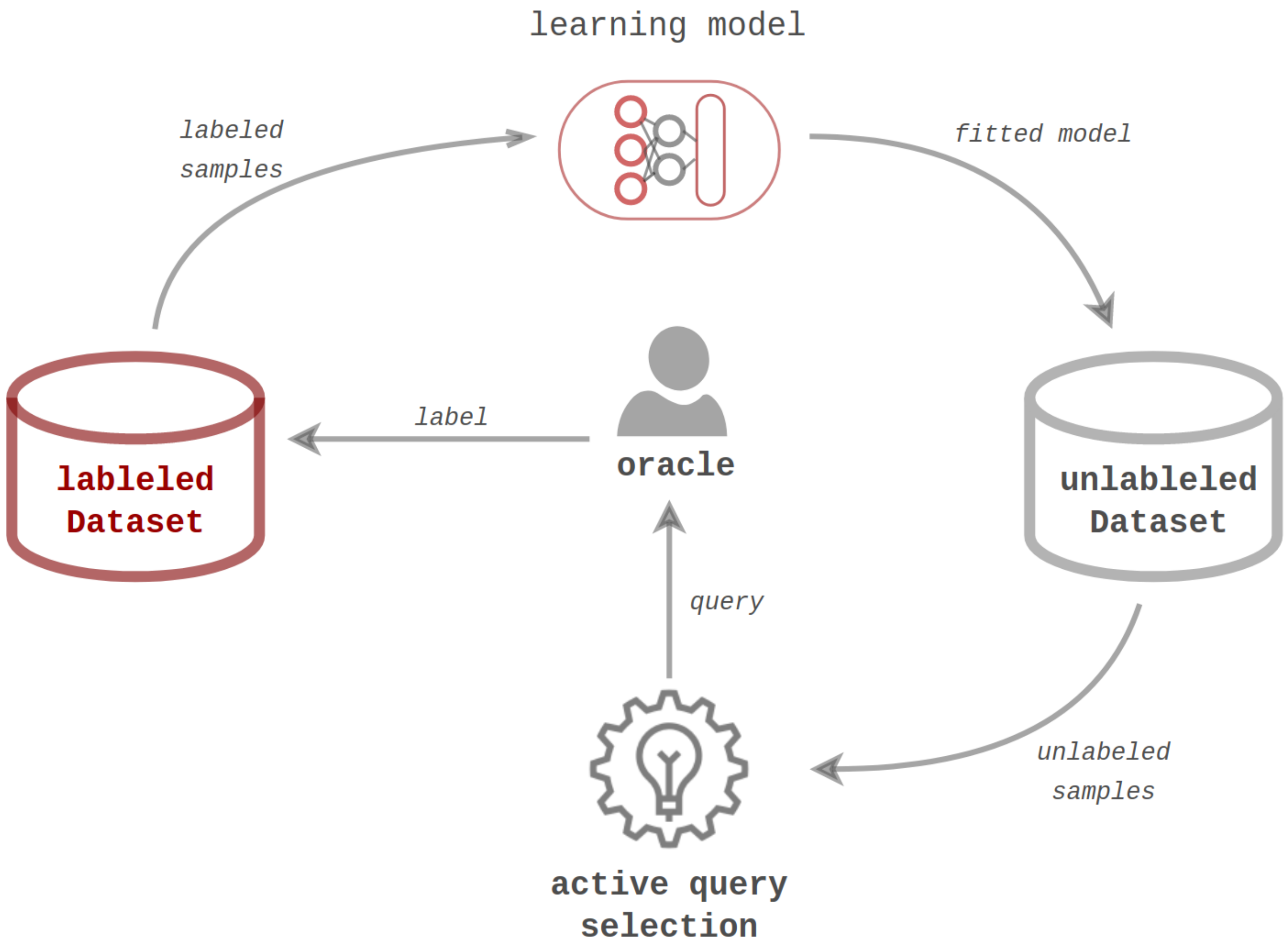
Active Learning

Goal

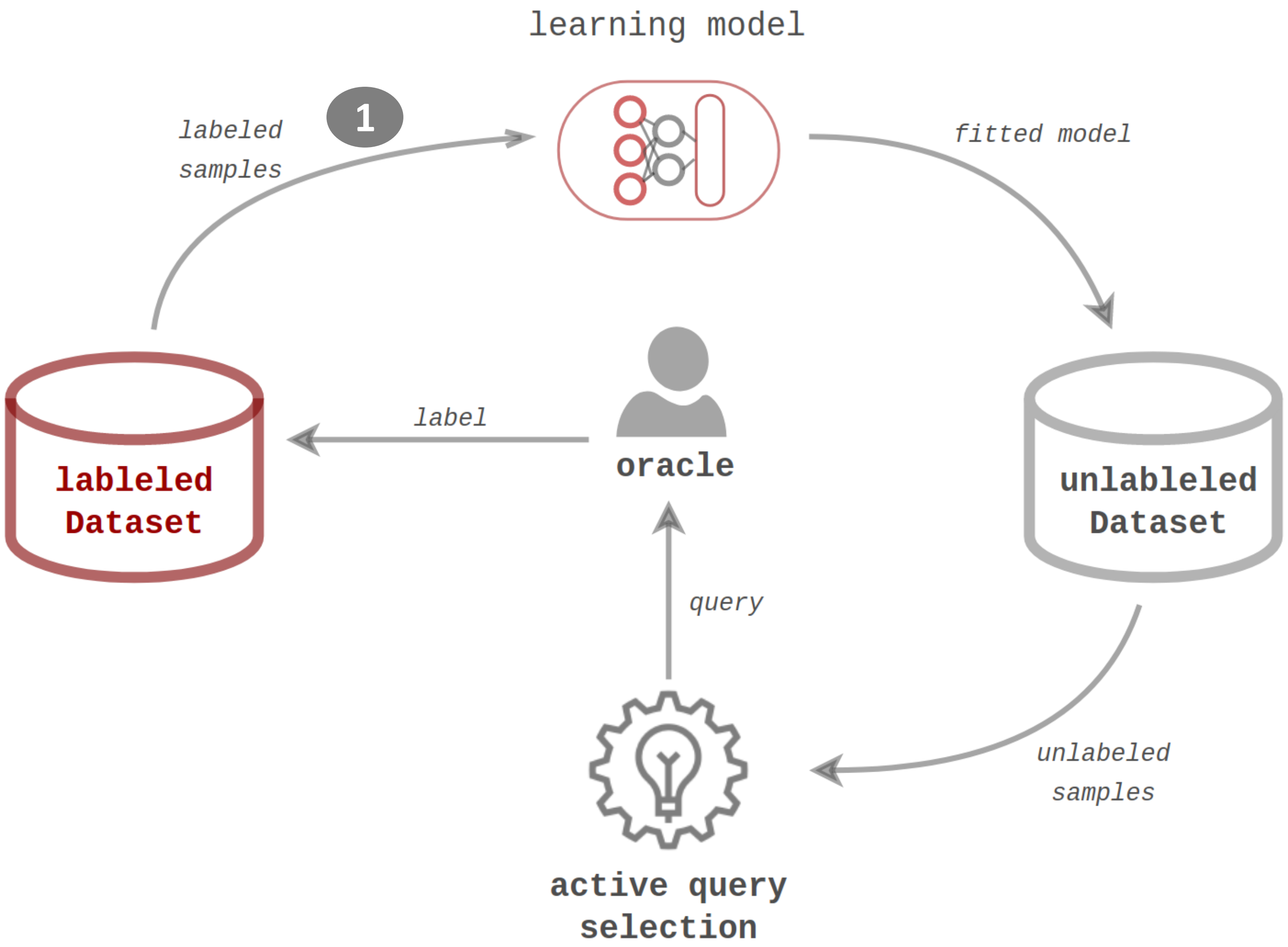
- achieve good accuracy with a relatively smaller number of labeled samples



Active Learning

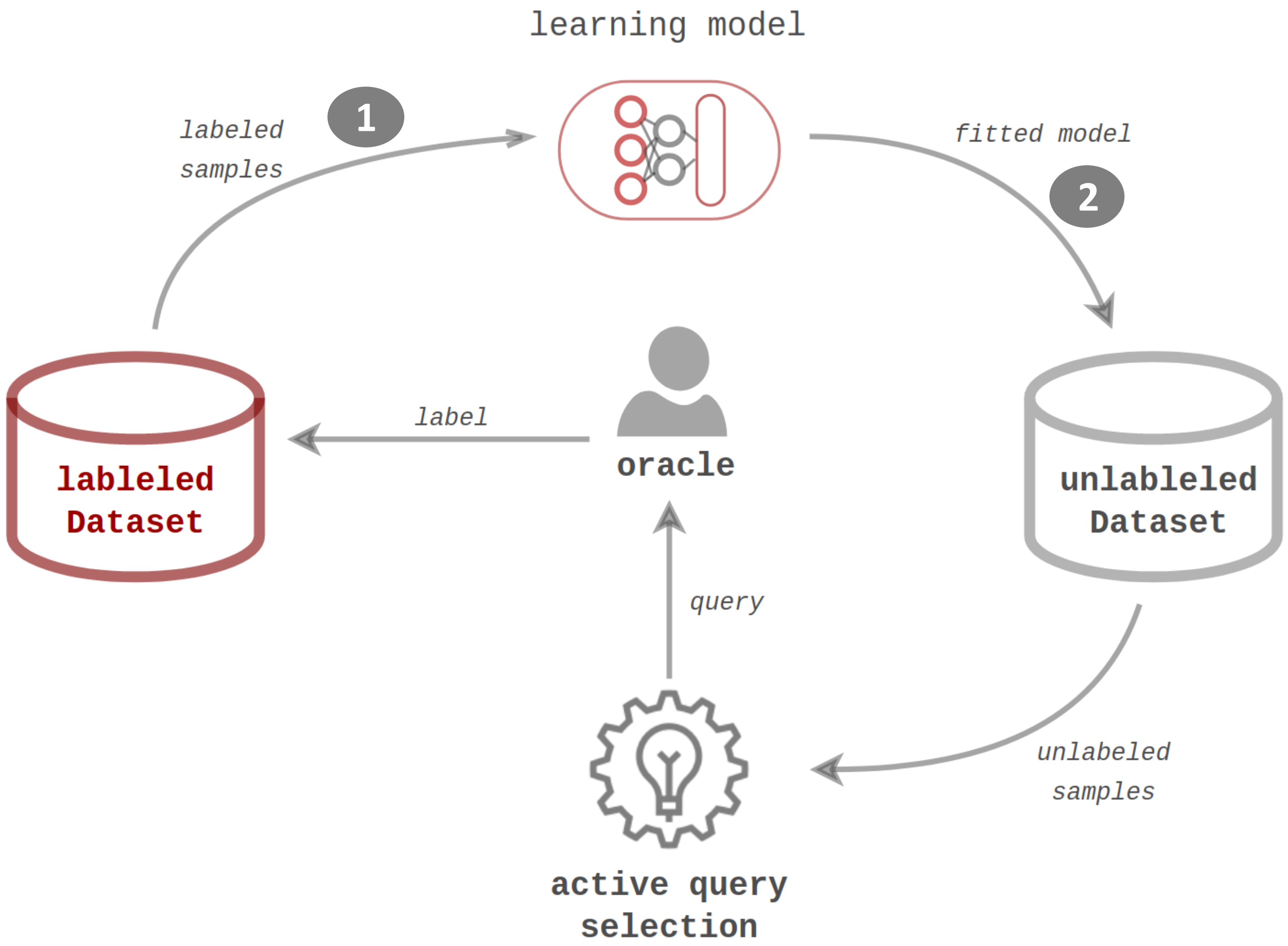


Active Learning



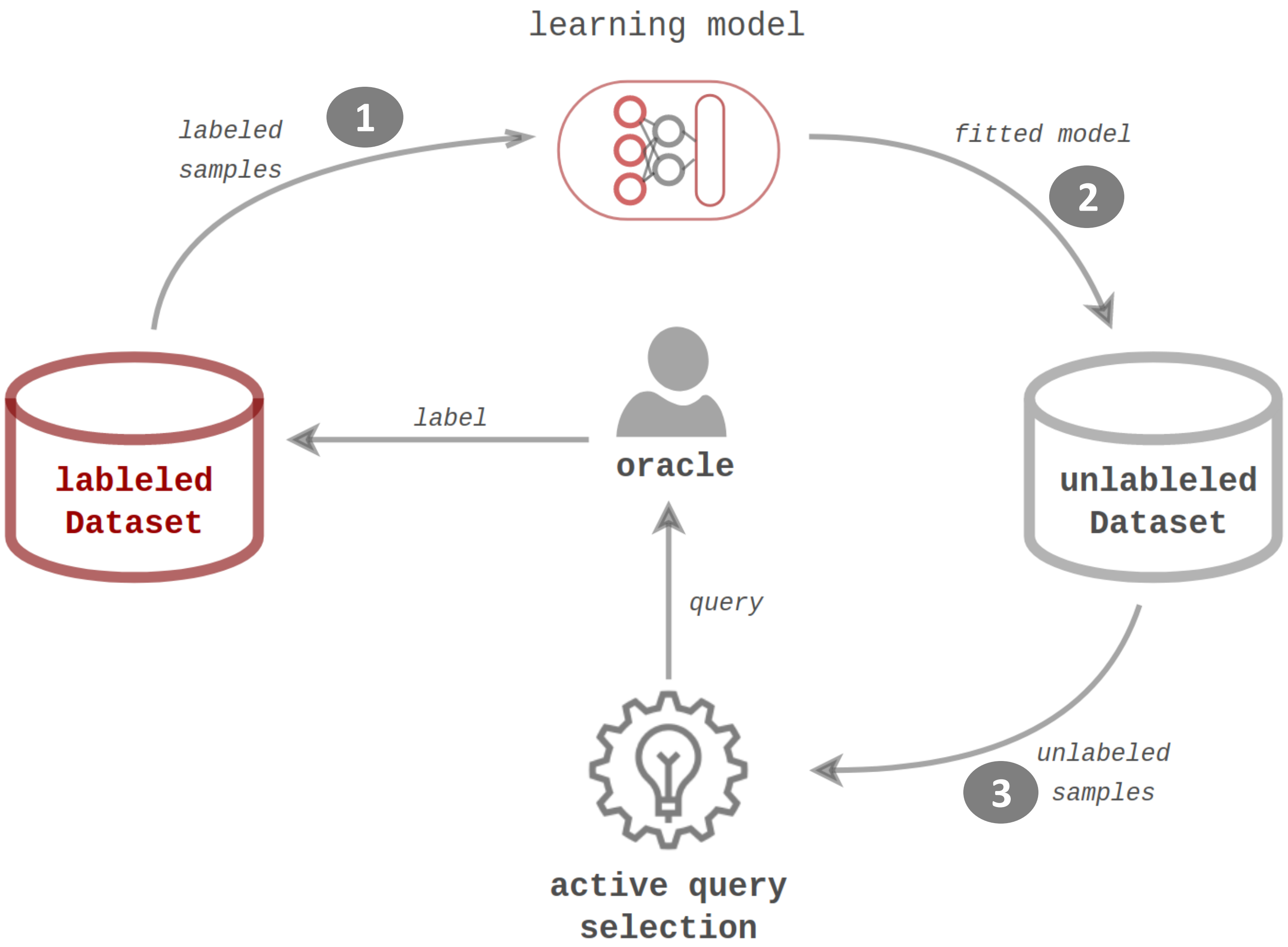
1. Train model

Active Learning



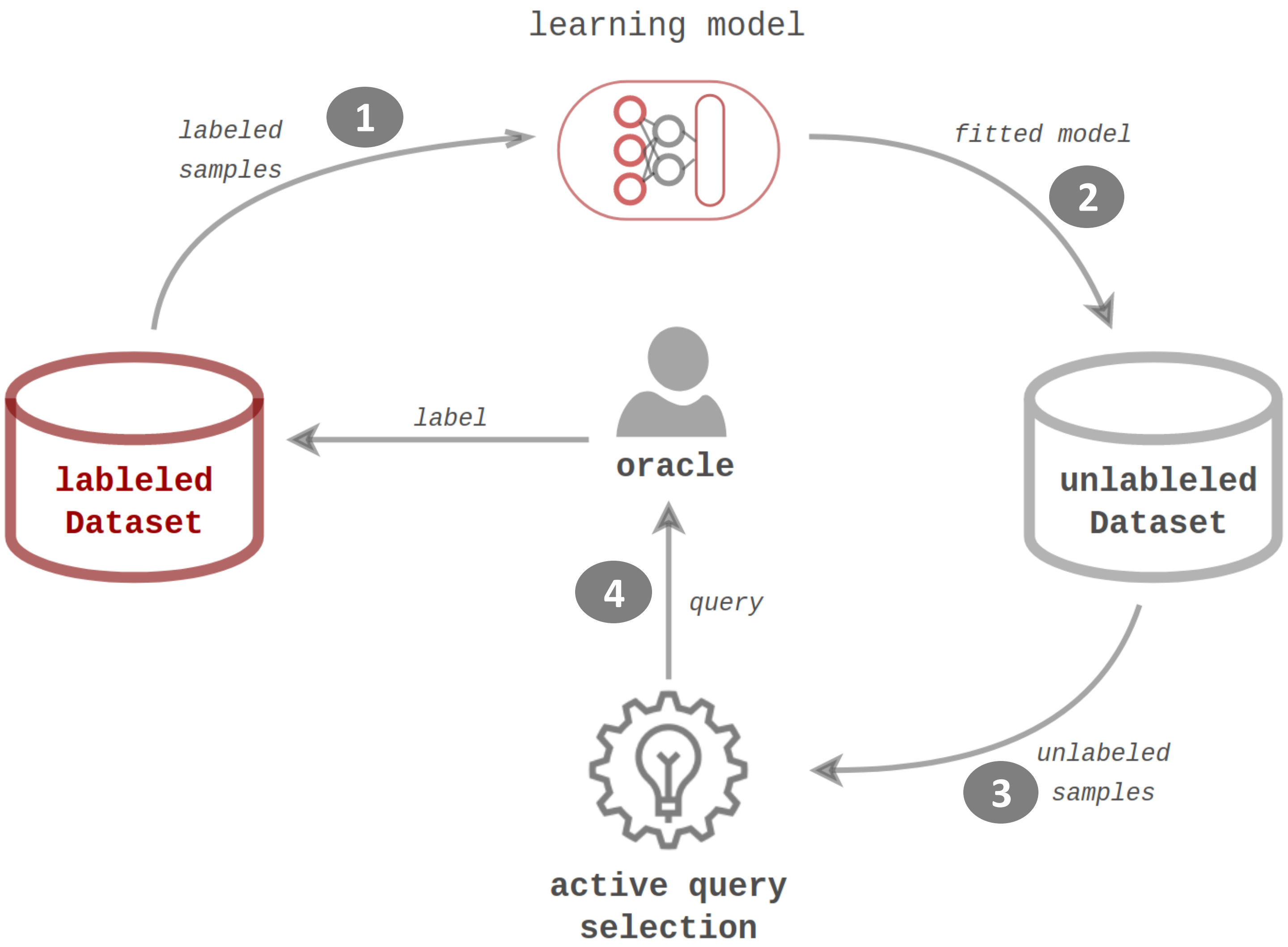
1. Train model
2. Make Predictions on unlabeled data

Active Learning



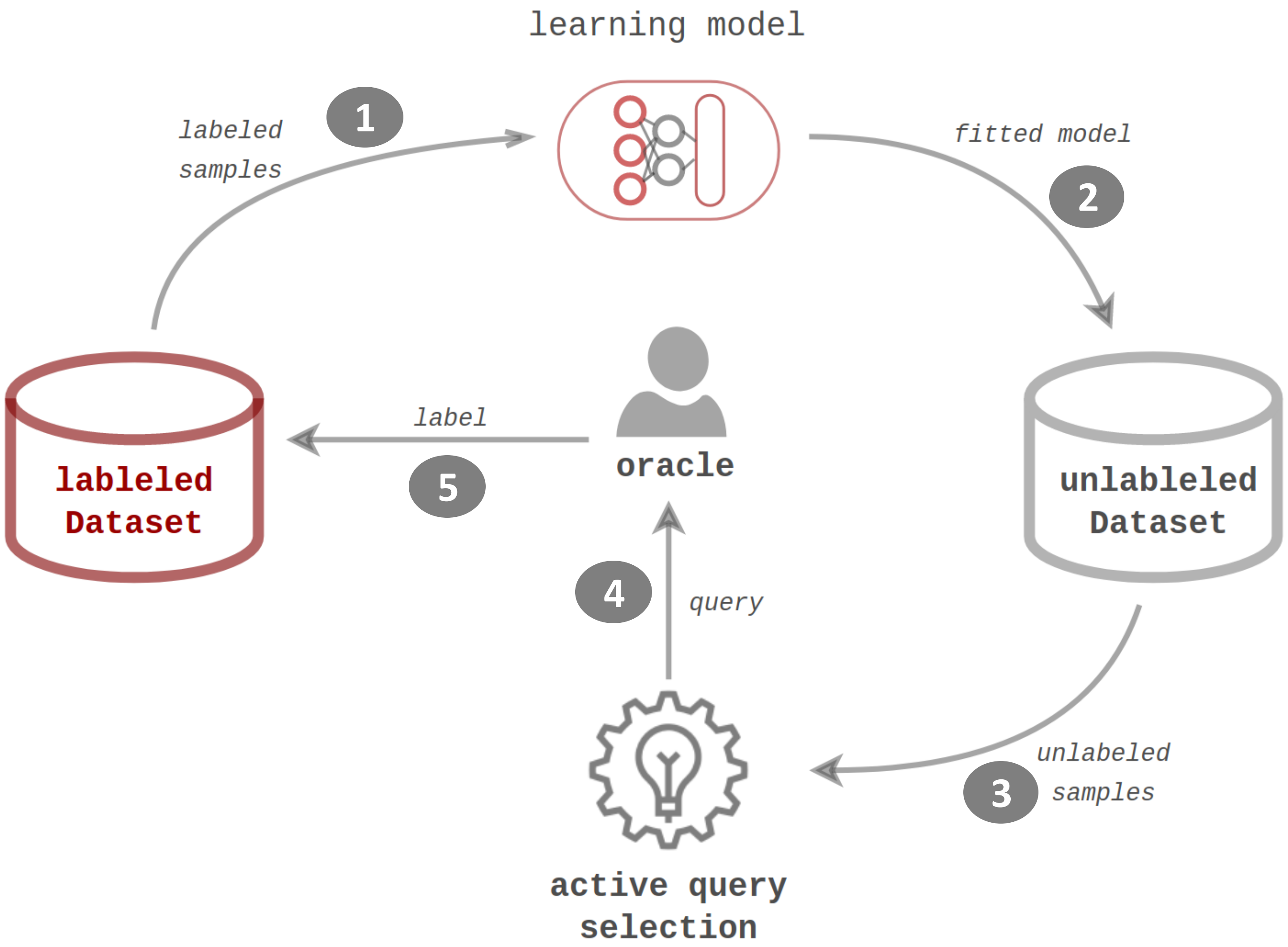
1. Train model
2. Make Predictions on unlabeled data
3. Gather information from the predictions and feed them to query selection procedure

Active Learning



1. Train model
2. Make Predictions on unlabeled data
3. Gather information from the predictions and feed them to query selection procedure
4. Selection procedure makes queries to oracle to get label

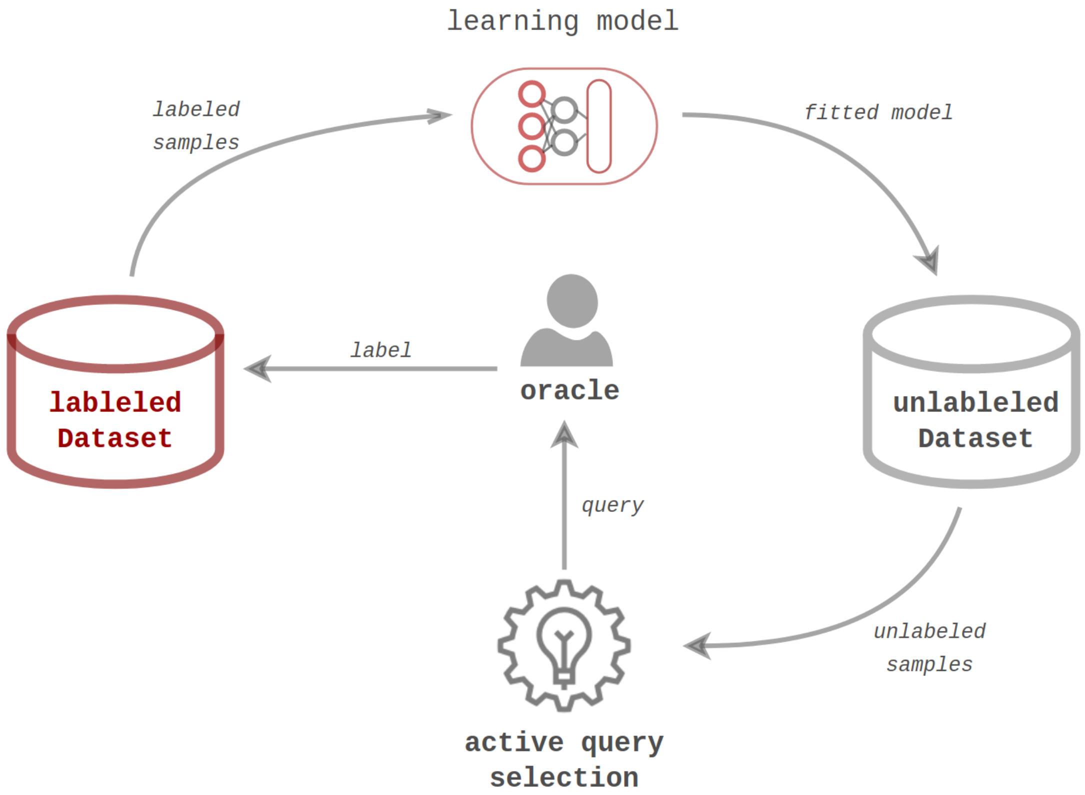
Active Learning



1. Train model
2. Make Predictions on unlabeled data
3. Gather information from the predictions and feed them to query selection procedure
4. Selection procedure makes queries to oracle to get label
5. Add labeled point to dataset

Active Learning

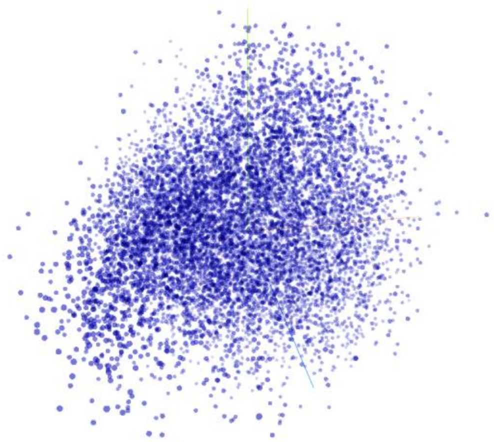
- ML algorithms that can **query**
- Selection procedure decide **which instances we should label first**
- Tries to derive an **optimal labeling sequence**
- **Oracle** provides labels
- Models achieve high accuracy with a **small number of labeled samples**



Distributed Active Learning

High Dimensional Data

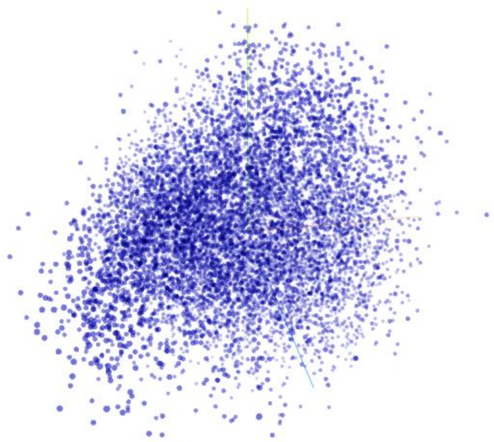
- Complexity due to volume
- Need distributed storage



Distributed Active Learning

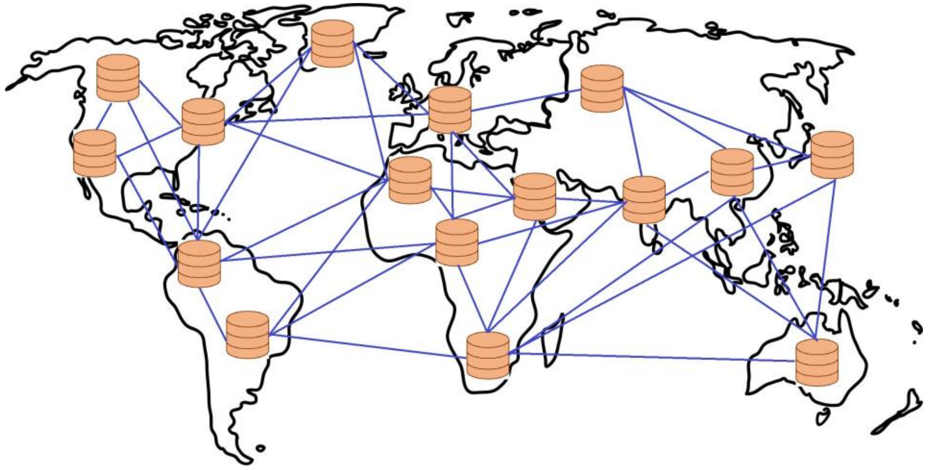
High Dimensional Data

- Complexity due to volume
- Need distributed storage

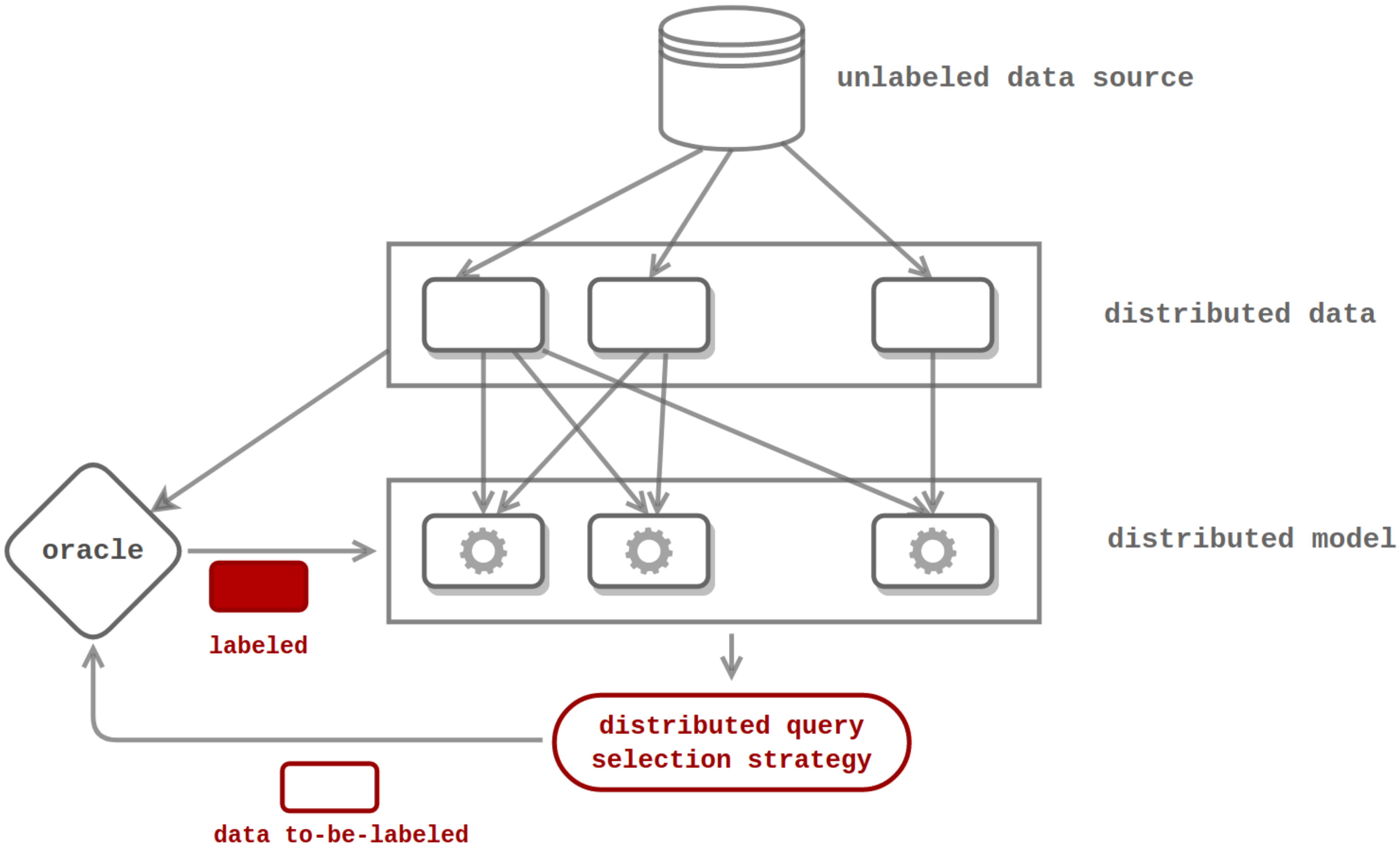


Distributed ML on the rise

- Need for scalable ML algorithms for **cloud**
- Data is distributed across the world
- Distributed frameworks: **hadoop, spark**



Distributed Active Learning



Related Works

Classic AL Strategies

- Uncertainty Sampling [Lewis et. Al]
- Query-by-Committee [Seung et. Al]

Extensive research on AL has resulted in various strategies

- Deep Reinforcement AL [Fang et. Al]
- AL with GAN (Generative Adversarial Nets) [Zhu and Bento]
- Deep Active Learning [Kronrod et. Al]
- Learning Active Learning from Data [Konyushkova et. al]

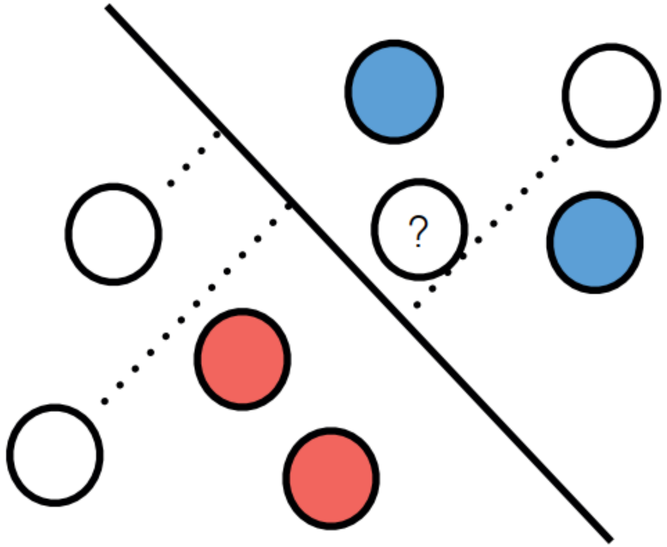
Current Researches do not study Active Learning from a Distributed Context

Related Works

Uncertainty Sampling [Lewis et. Al]

- Query the **most uncertain** sample first
- Samples **near decision boundary**
- *H* is measurement of uncertainty

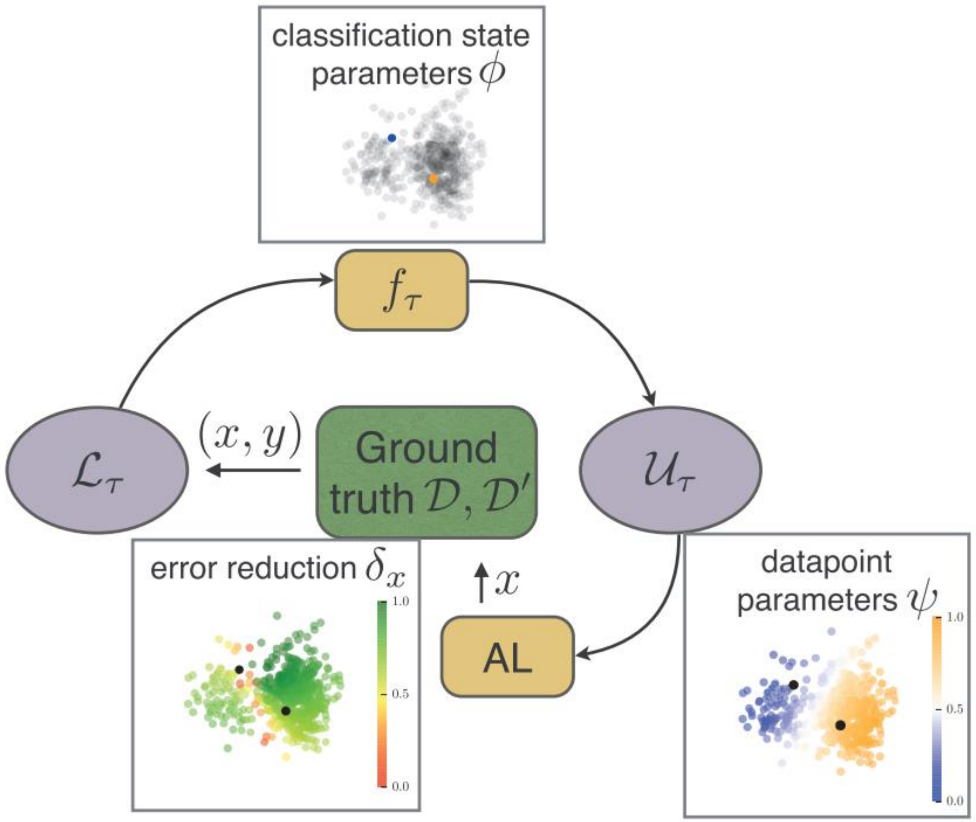
$$x^\star = \arg\max_{x \in \mathcal{U}_t} H(\hat{y} = y \mid x)$$



Related Works

- **Learning Active Learning from Data [Konyushkova et. al ’17]**

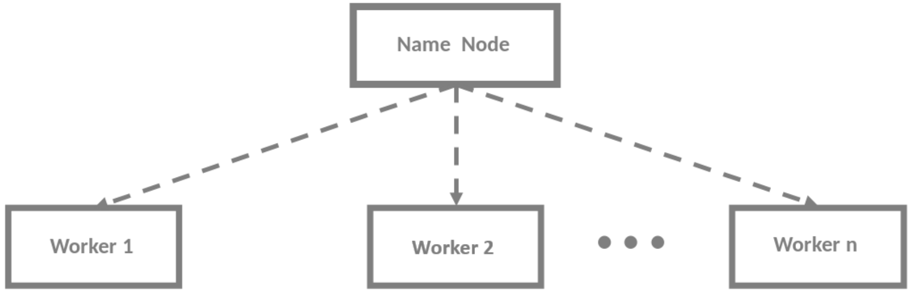
- Query selection procedure is a **Regressor**
- Uses *properties of classifiers* and *data* to *predict the potential error reduction*.
- Label sample which **reduces the generalization most**



HDFS

Hadoop Distributed File System (*Hadoop*)

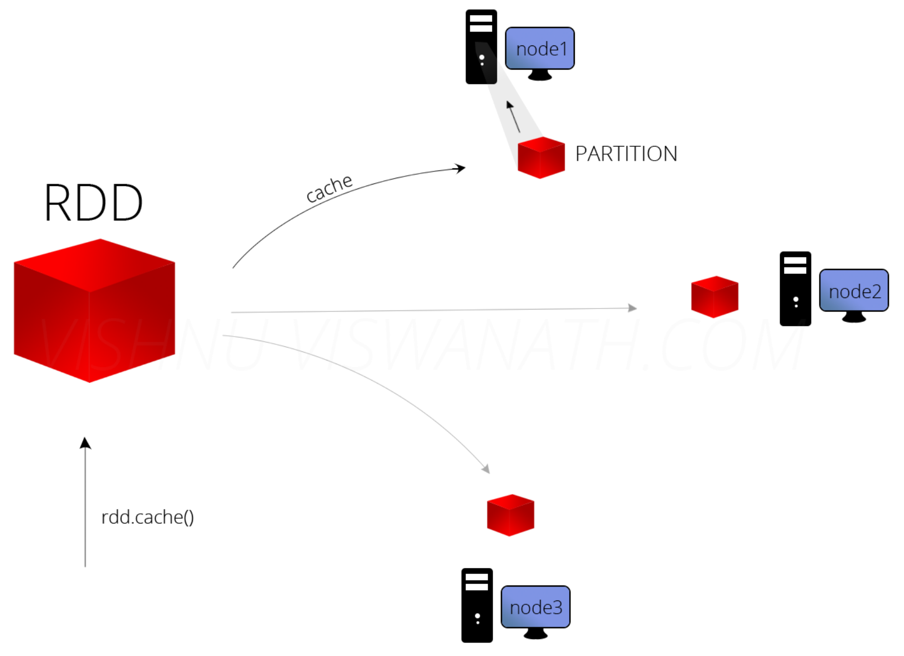
- Stores data across machines
- Blocks of data
- **Workers** store
- **Namenode** manages/schedules workers



Spark RDD

Resilient Distributed Datasets (*Spark*)

- Stored on a cluster
- Each machine has **partitions** (chunks)
- Fault tolerant
- No specific **ordering** of data (no indexing)



Distributed Active Learning

2 Distributed Active Learning Algorithms

- **Distributed Uncertainty Sampling**
- **Distributed Density Weighting**
  - Uses : algorithm ***construct-proximity-matrix***

Distributed Uncertainty Sampling

Algorithm 1

Distributed Uncertainty Sampling

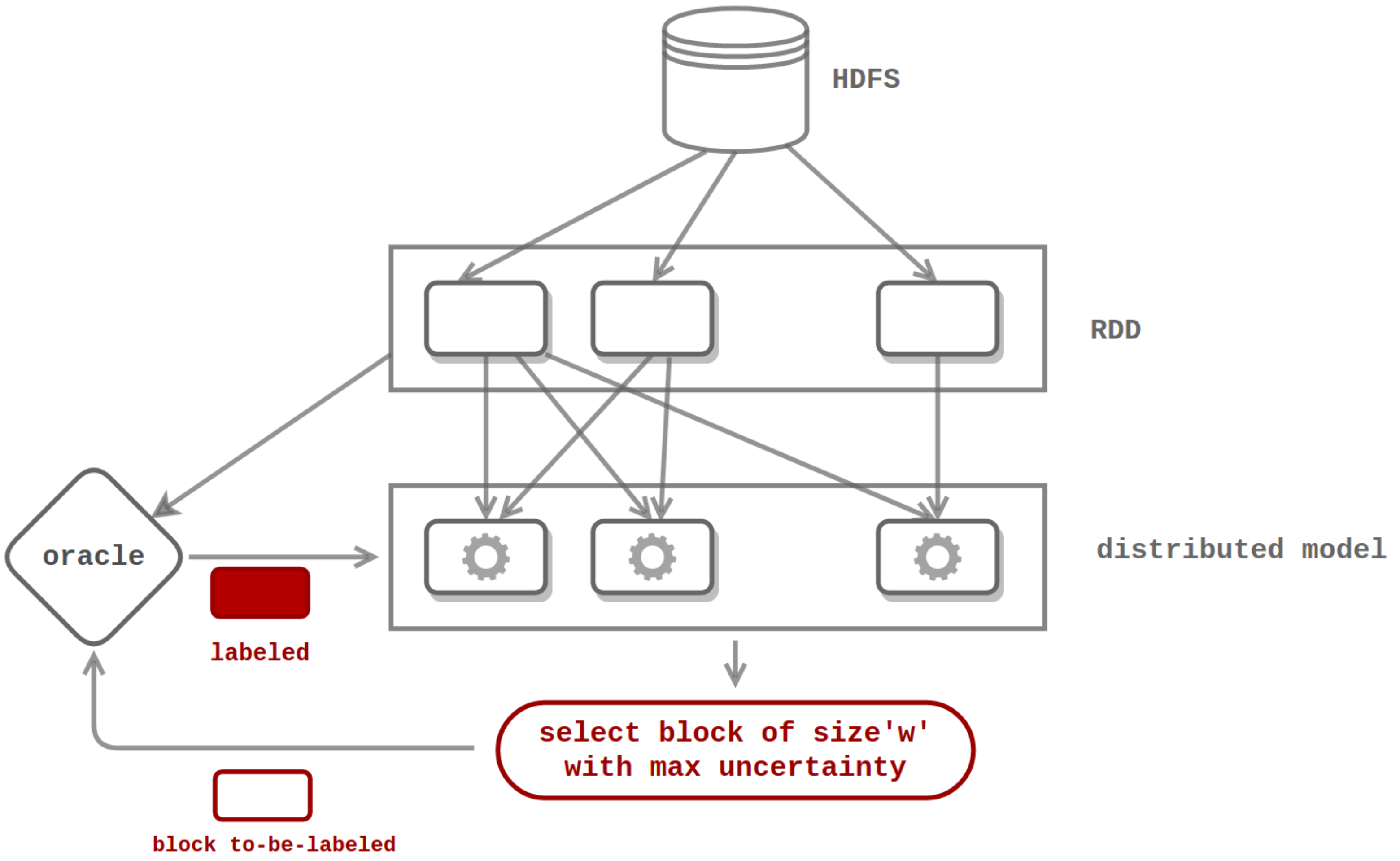
- Distributed extension of Uncertainty sampling
- Select block of data which incurs the most uncertainty and send for labeling

$$\mathcal{X}^* = \underset{\{x_m^*, \ldots x_n^*\} \subset \mathcal{U}_t}{\operatorname{argmax}} \sum_{\{x_m^*, \ldots x_n^*\}} H(\hat{y} = y \mid x)$$

Distributed Uncertainty Sampling

Algorithm 1

Distributed Uncertainty Sampling

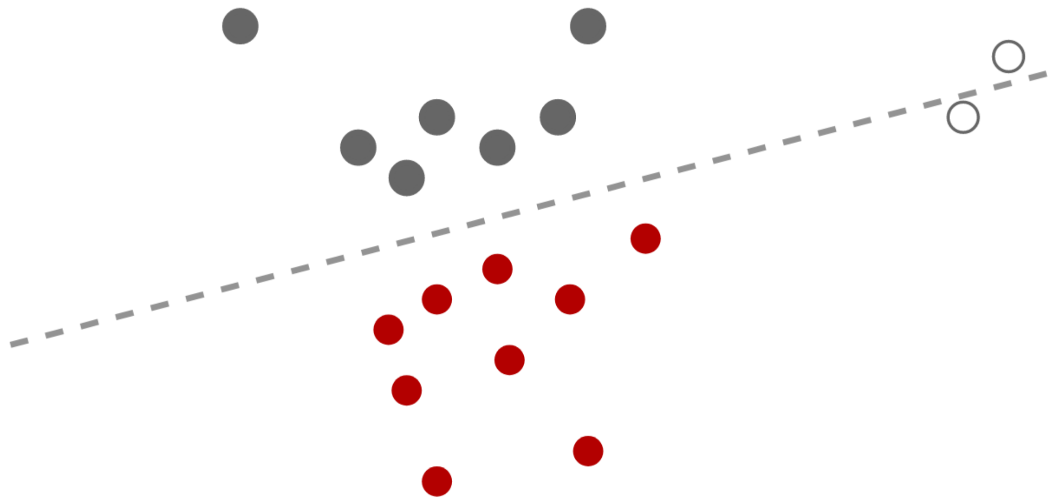


Distributed Density Weighting

Algorithm 2

Distributed Density Weighting

- **Density Weighting**
  - Uncertainty sampling lacks robustness to **outliers**
  - Query strategy needs to incorporate **representativeness** information
  - Density heuristic captures representativeness of a block of data

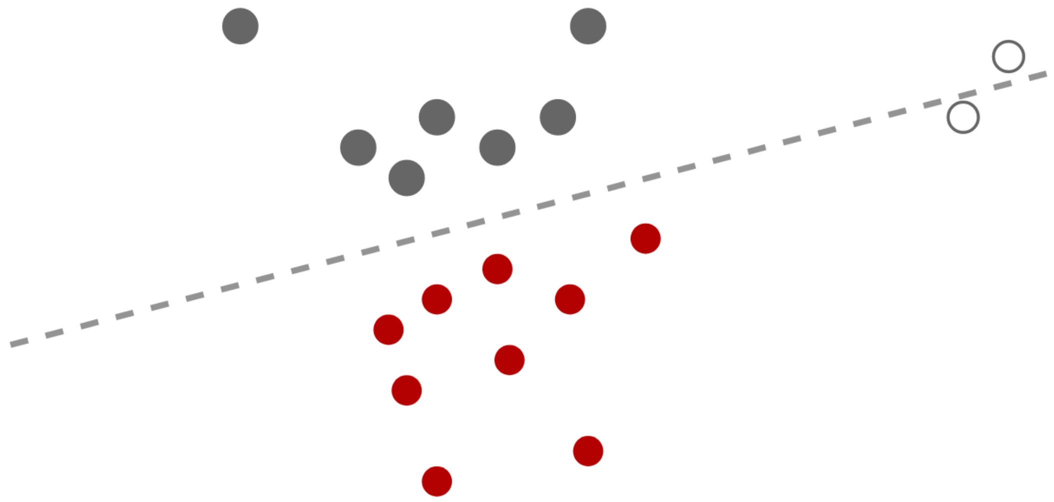


Distributed Density Weighting

Algorithm 2

Distributed Density Weighting

- **Density Weighting**
  - Uncertainty sampling lacks robustness to **outliers**
  - Query strategy needs to incorporate **representativeness** information
  - Density heuristic captures representativeness of a block of data



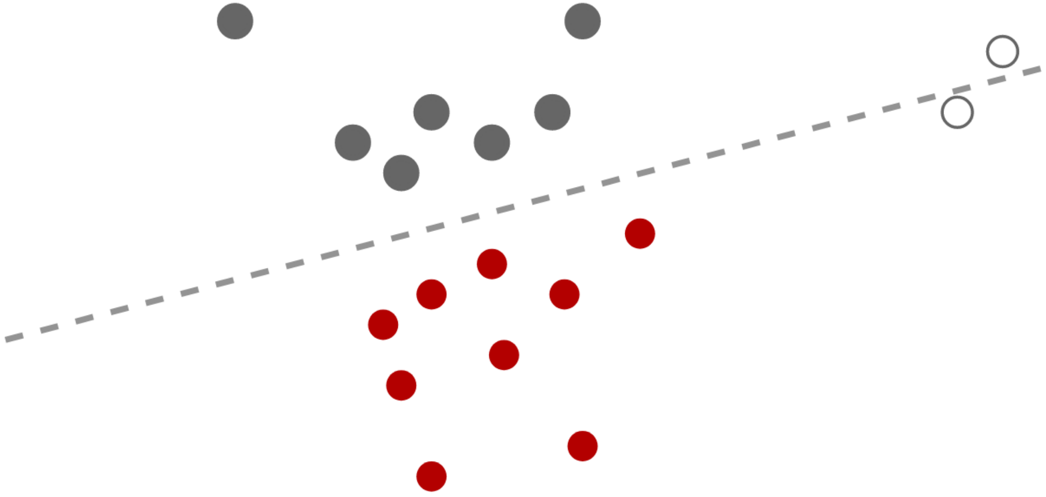
$$\mathcal{X}^*_{ID} = \underset{\{x^*_m,\ldots x^*_n\}\subset\mathcal{U}_t}{\operatorname{argmax}} \sum_{\{x^*_m,\ldots x^*_n\}} \phi_A(x) \times \left(\frac{1}{U}\sum_{x'\in\mathcal{U}} S(x,x')\right)^{\beta}$$

Distributed Density Weighting

Algorithm 2

Distributed Density Weighting

- **Density Weighting**
  - Uncertainty sampling lacks robustness to **outliers**
  - Query strategy needs to incorporate **representativeness** information
  - Density heuristic captures representativeness of a block of data

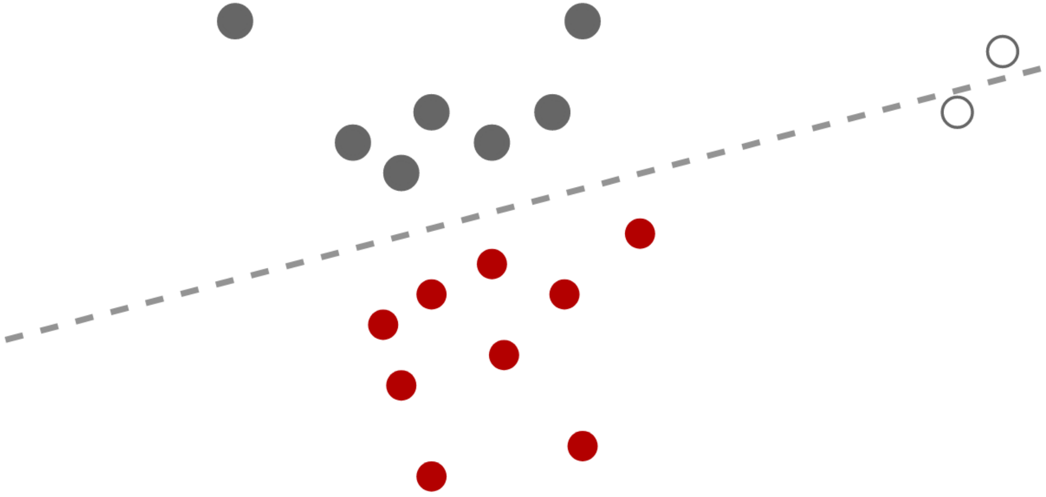


$$\mathcal{X}^*_{ID} = \operatorname*{argmax}_{\{x^*_m,\dots x^*_n\}\subset\mathcal{U}_t} \underbrace{\sum_{\{x^*_m,\dots x^*_n\}} \phi_A(x) \times}_{\text{base uncertainty metric}} \left(\frac{1}{U}\sum_{x'\in\mathcal{U}} S(x,x')\right)^{\beta}$$

Distributed Density Weighting

Algorithm 2
Distributed Density Weighting

- **Density Weighting**
  - Uncertainty sampling lacks robustness to **outliers**
  - Query strategy needs to incorporate **representativeness** information
  - Density heuristic captures representativeness of a block of data



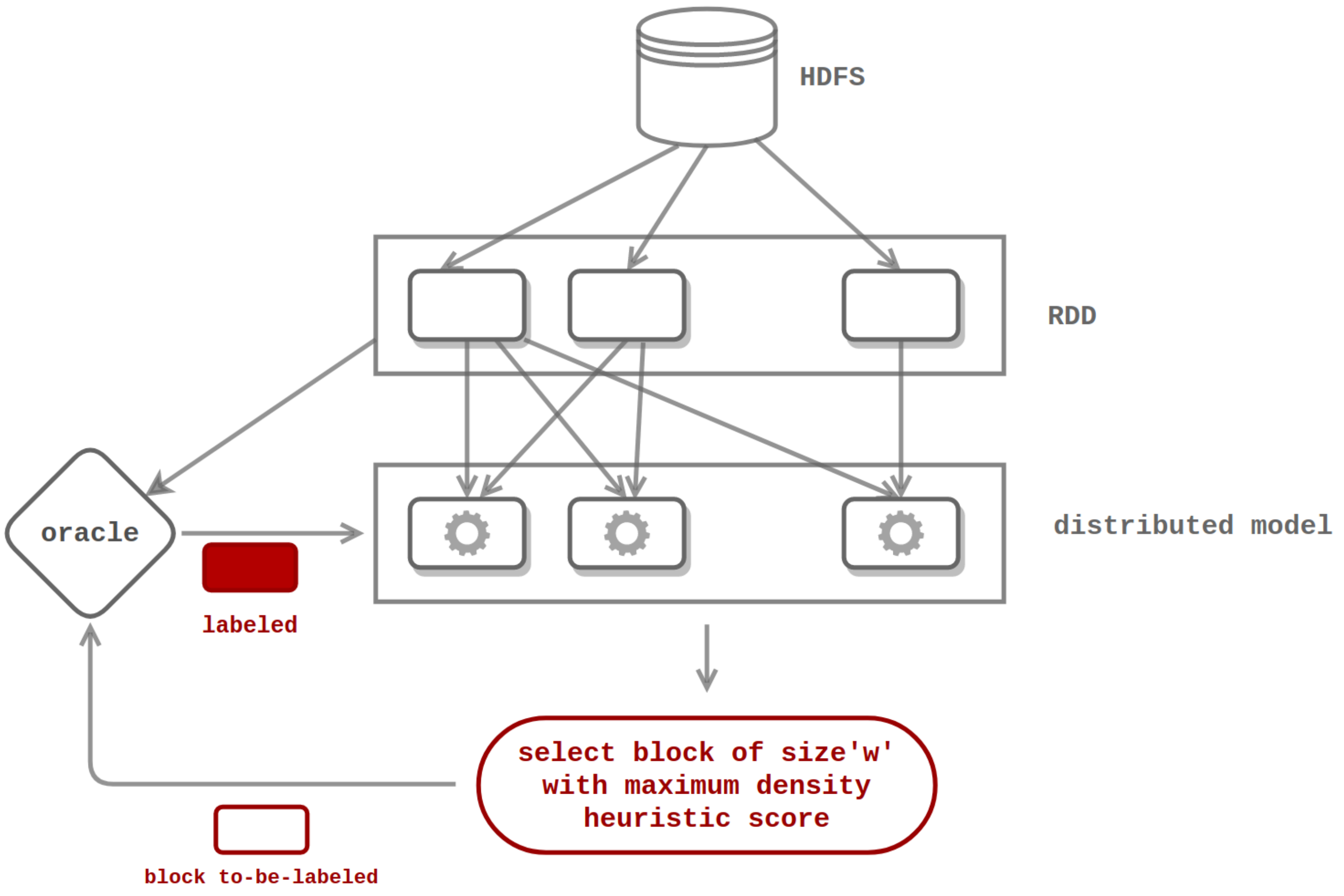
$$\mathcal{X}^*_{ID} = \underset{\{x^*_m,\ldots x^*_n\}\subset\mathcal{U}_t}{\operatorname{argmax}} \sum_{\{x^*_m,\ldots x^*_n\}} \phi_A(x) \times \left(\frac{1}{U}\sum_{x'\in\mathcal{U}} S(x,x')\right)^{\beta}$$

Similarity between two points

Distributed Density Weighting

Algorithm 2

Distributed Density Weighting



Construct Proximity Matrix

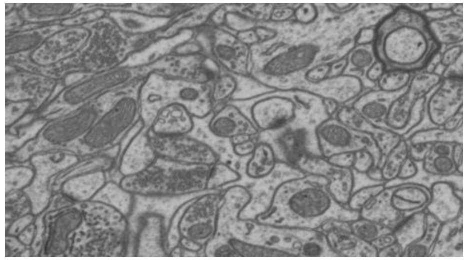
Algorithm 3 Construct-Proximity-Matrix

- Need to calculate proximity of datapoints for **Distributing Density Weighting**
- Implemented using a series of transformations of Spark's **CoordinateMatrix** and **BlockMatrix** libraries

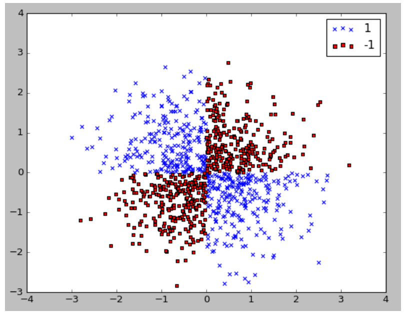
Experimental Setup

- **6 Node Cluster**
  - 6*16 GB Memory
  - 6*8 cores
  - 160G Disk Space
  - Apache-Hadoop, Apache-Spark, Ubuntu
- **Datasets (Real and Synthetic)**
  - **Google Quickdraw** (28x28 bitmap images; subset 1G
  - **Striatum** (10k x 100 3D Electron Microscopy stack of rat neural tissue)
  - **XOR** (0.5G checker board synthetic data set)

quick draw dataset



striatum dataset



XOR dataset

Performance Evaluation

Accuracy Gain Per Iteration

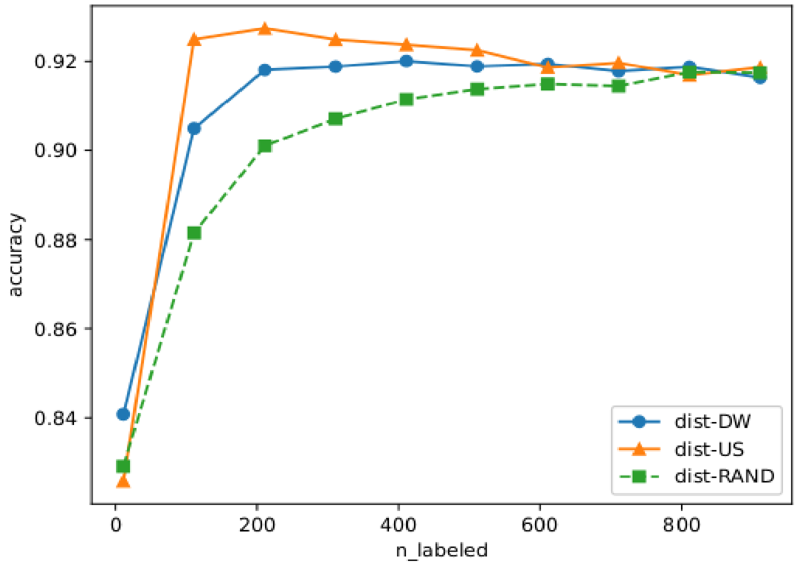


Figure 6.1: Accuracy gain per iteration for Striatum dataset

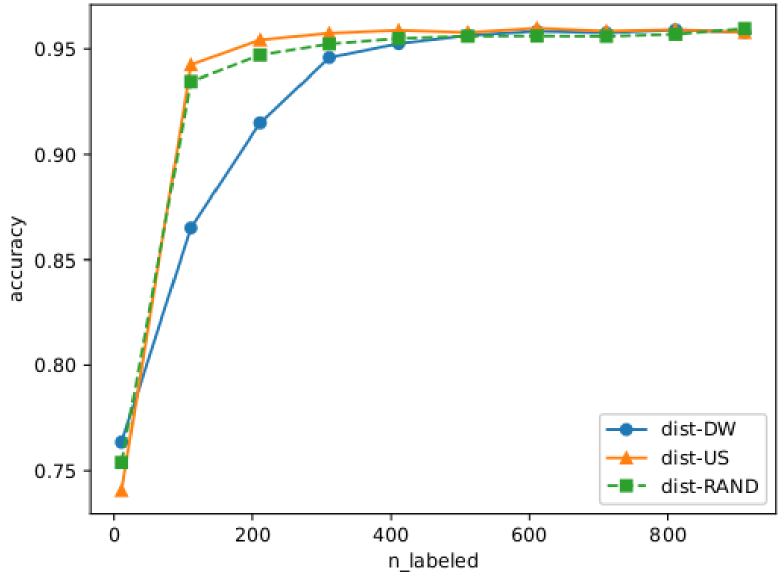


Figure 6.2: Accuracy gain per iteration for Quick draw dataset

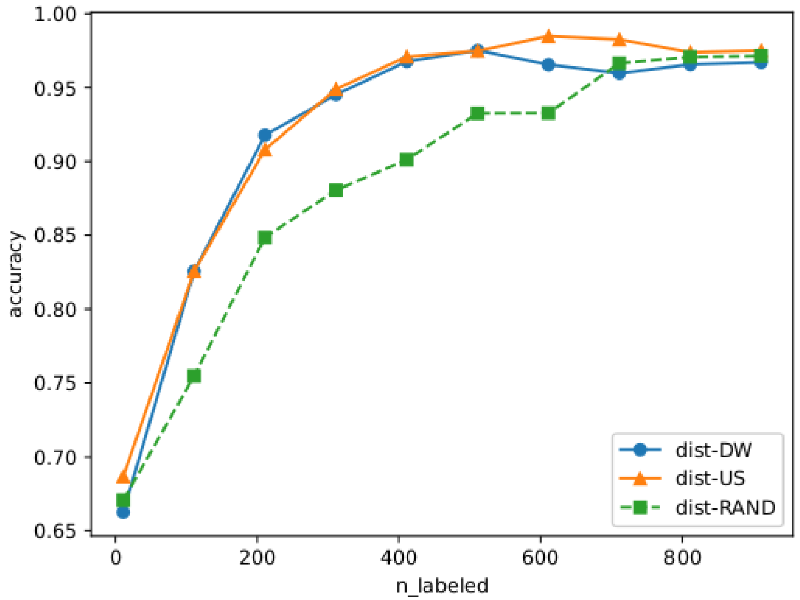


Figure 6.3: Accuracy gain per iteration for XOR dataset

Performance Evaluation

Accuracy vs Running time

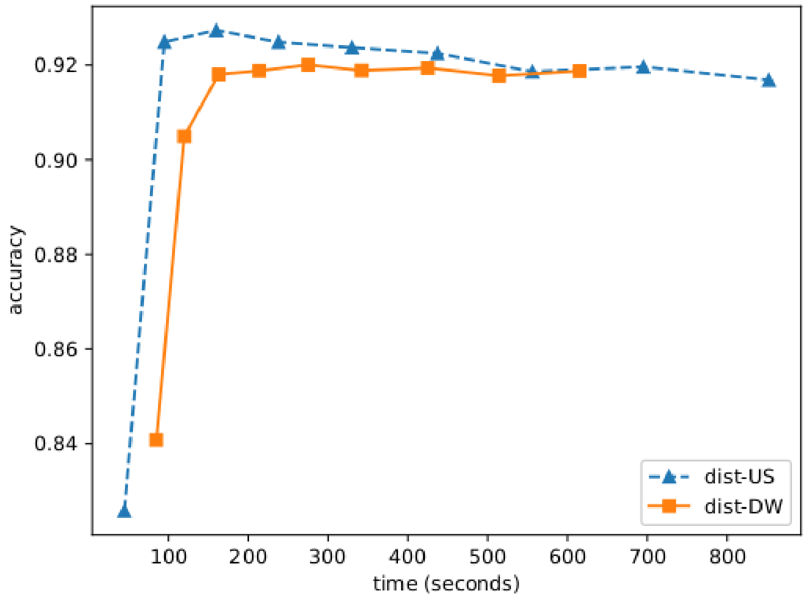


Figure 6.5: Accuracy vs running time for Striatum Data

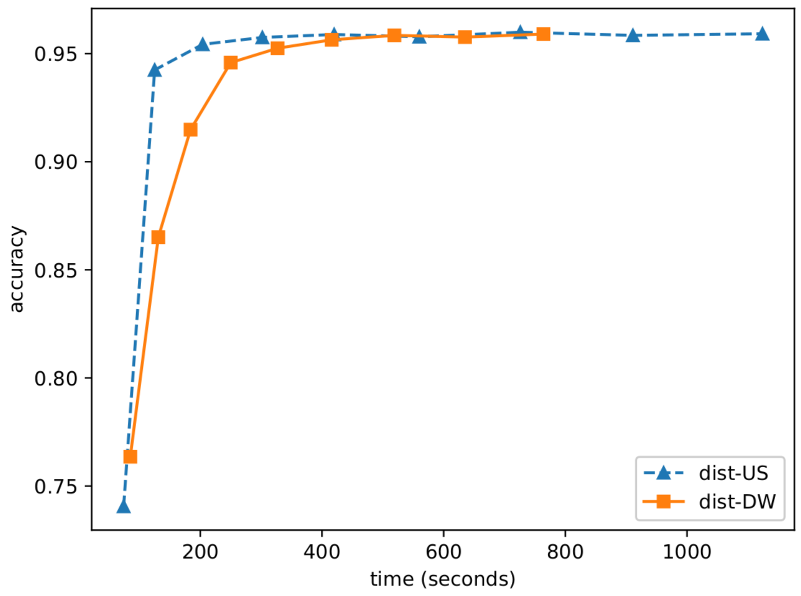


Fig. 6. Accuracy vs running time for Quick Draw Data

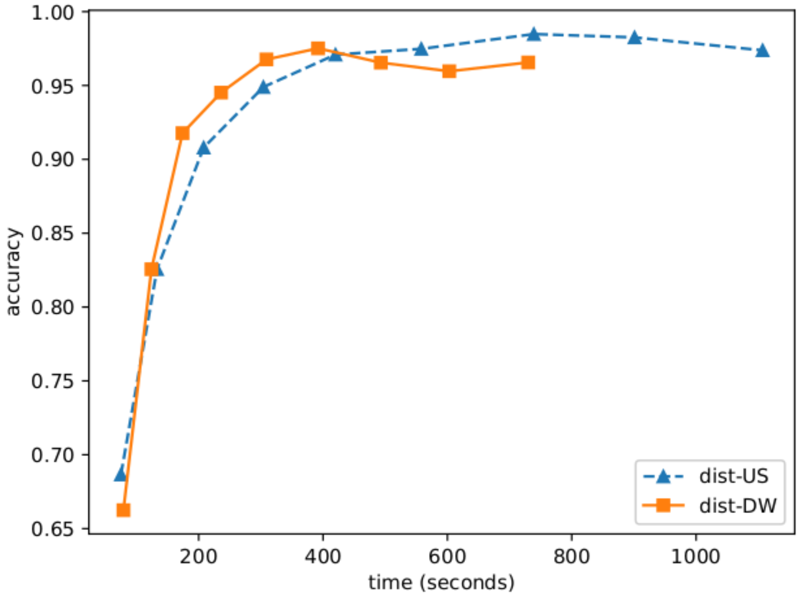


Figure 6.7: Accuracy vs running time for XOR Data

Performance Evaluation

Running Time

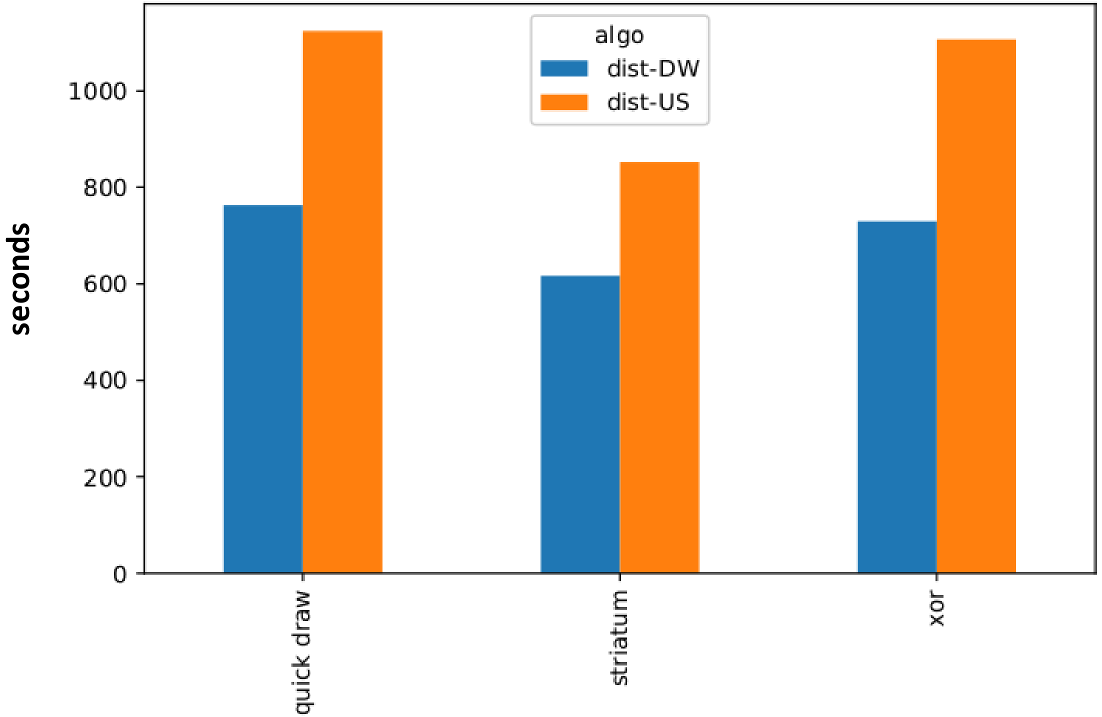


Figure 6.4: Running time for various datasets

Shuffle I/O

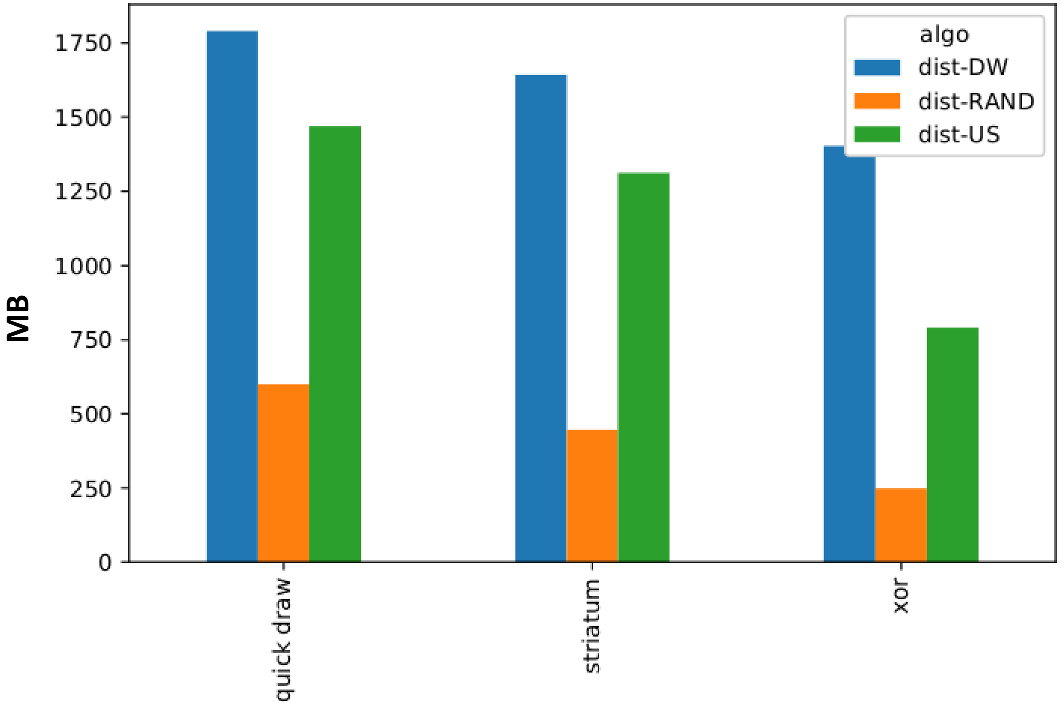


Figure 6.8: Amount of bytes read in shuffle for algorithms

Performance Evaluation

ColumnSimilarity() Vs Create-Proximity-Matrix Algorithm

	Time in seconds	Time in seconds
Size	Columnsimilarity	Create-Proximity
1000	22	12
2000	62	13
3000	148	14
4000	297	15
5000	out of memory	17

Striatum

	Time in seconds	Time in seconds
Size	Columnsimilarity	CreateProximity
1000	15	16
2000	40	17
3000	96	18
4000	188	22
5000	out of memory	19

Quick Draw Dataset

Conclusion

- In this work, we have implemented two distributed active learning algorithms
- The algorithms are tested with both real and synthetic datasets.
- In future we want to explore other areas of active learning in scale.

Thank You