CS 760: Machine Learning
Less-than-full Supervision

Misha Khodak

University of Wisconsin-Madison

3 December 2025

Logistics

- **Last homework due Monday before class**

- **Fill out the course survey when you get it**

- **Roadmap:**
    - **Today:** less-than-supervised learning
    - **Monday:** transfer learning
    - **Wednesday:** exam review

Outline

- **What do we do if we don’t have enough data?**
  - Motivation, approaches, taxonomy
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Active Learning**
  - Stream-based, thresholds, pool-based, margin-based
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning

Outline

- **What do we do if we don't have enough data?**
  - Motivation, approaches, taxonomy
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Active Learning**
  - Stream-based, thresholds, pool-based, margin-based
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning

What do we do if we don't have enough data?

So far our setup in supervised learning has been

- gather a set of labeled data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$
- train a model on it
- tune the model as needed

What if **collecting enough labels** to train a sufficiently expressive model **is too expensive**?

Dealing with low-data scenarios

Numerous approaches (too many to cover in detail)

- which one to take is highly application-dependent
- can construct a basic taxonomy:

less-than-full supervision
- do more with less (labeled data)
- focus of today's lecture

transfer learning
- do more with more (o.o.d. data)
- Monday's lecture

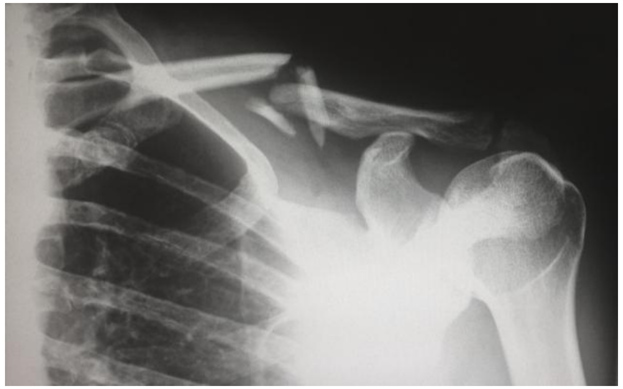
semi-supervised learning | active learning | weakly-supervised learning

multi-task learning | meta learning | foundation models

Outline

- **What do we do if we don’t have enough data?**
  - Motivation, approaches, taxonomy
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Active Learning**
  - Stream-based, thresholds, pool-based, margin-based
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning

Semi-Supervised Learning: Setup



- Our usual supervised setup:

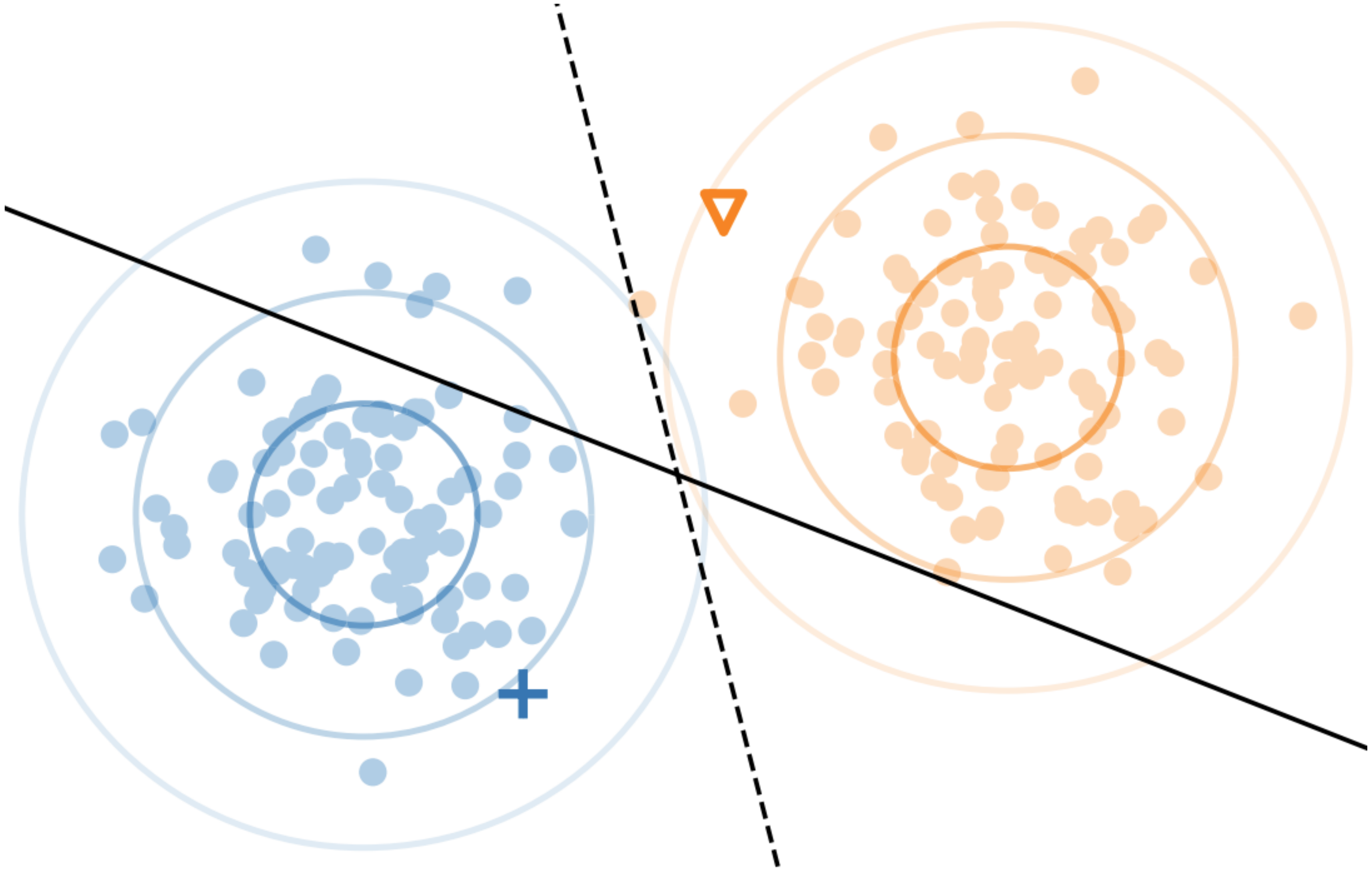
$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$$

- Downside:
  - Getting labels for all our instances might be expensive.
  - Ex: medical images: doctors need to produce labels

- Semi-supervised: some labels, **most unlabeled**

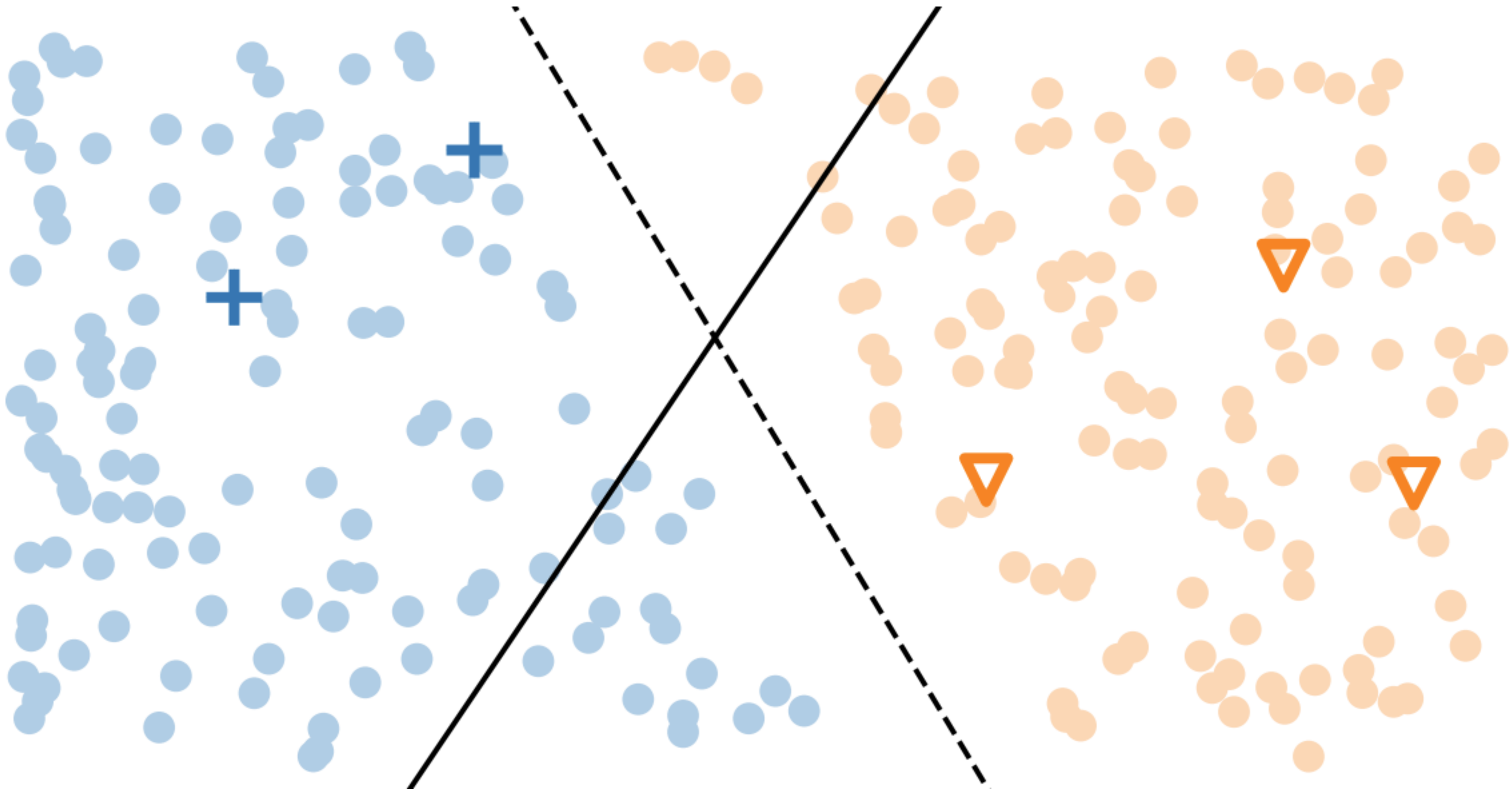
$$\underbrace{(x^{(1)}, y^{(1)}), \ldots, (x^{(n_L)}, y^{(n_L)})}_{n_L \text{ labeled points}}, \underbrace{x^{(n_L+1)}, \ldots, x^{(n_L+n_U)}}_{n_U \text{ unlabeled points}}$$

Intuition: which is the better classifier?



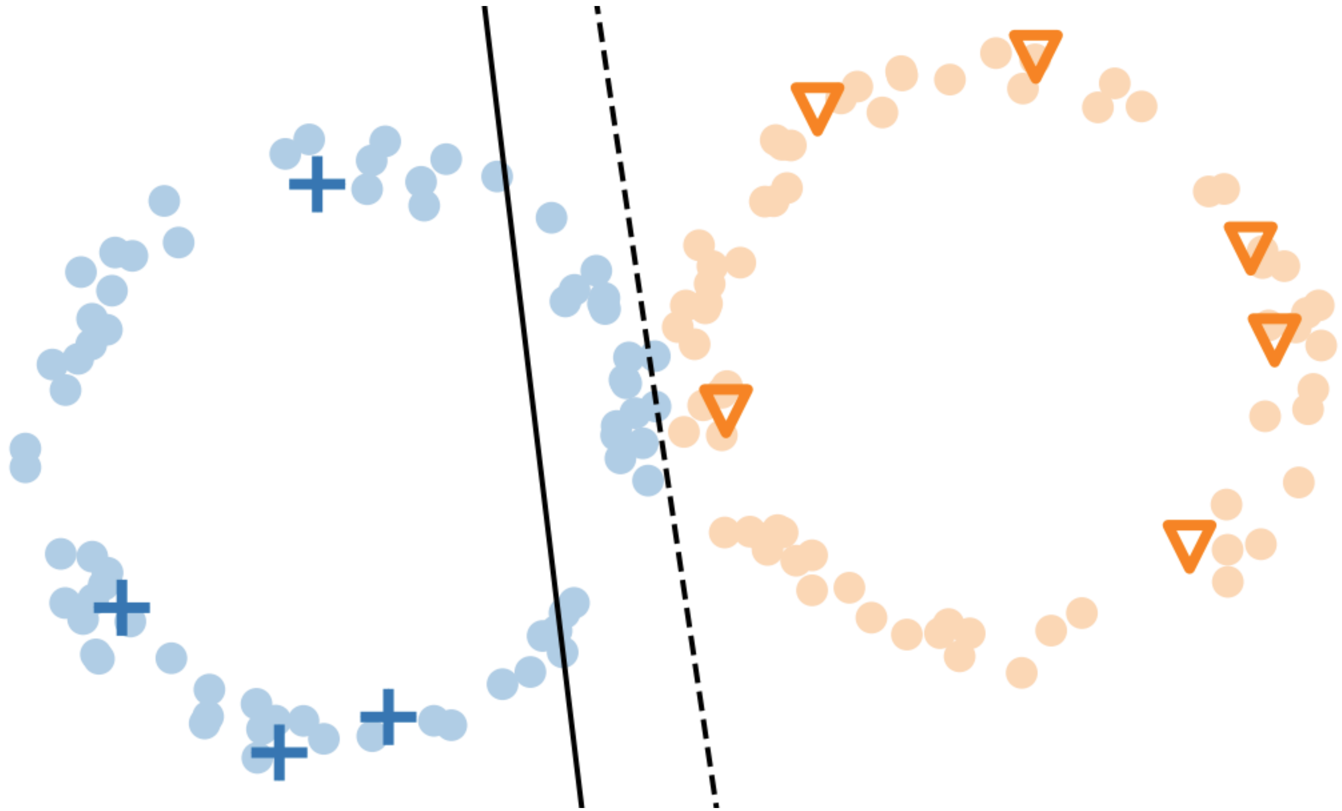
van Engelen & Hoos, 2020

Intuition: which is the better classifier?



van Engelen & Hoos, 2020

Intuition: which is the better classifier?

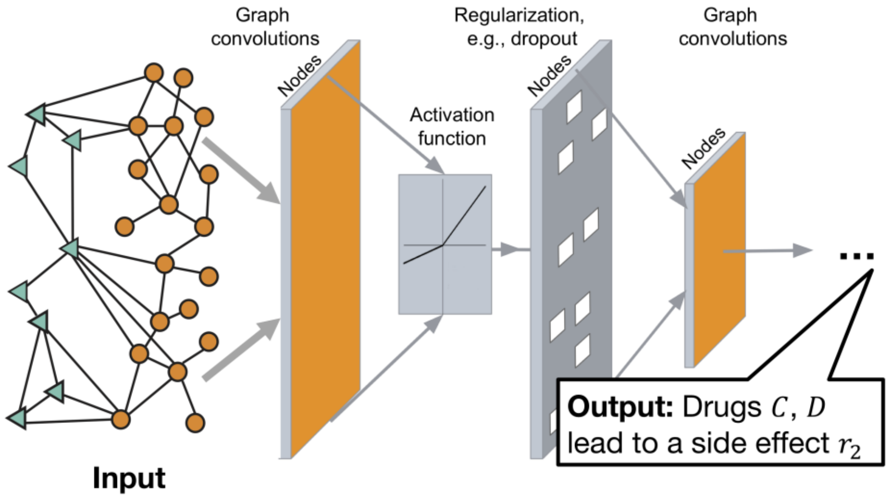


van Engelen & Hoos, 2020

Semi-Supervised Learning: Techniques

- Huge space of approaches (could cover a full class)

- We'll focus on **two** today:
  - **label propagation**
    - classic technique
    - **explicit:** compute labels for the unlabeled data, then train a model
  - **graph neural networks**
    - modern technique
    - **implicit**: use for predictions directly



Stanford Snap

Label Propagation: Setup

- Have:
$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n_L)}, y^{(n_L)}), x^{(n_L+1)}, \ldots, x^{(n_L+n_U)}$$

- **Goal**: label the n_U unlabeled points

- **Basic idea**: points that are close should have similar labels

- **Approach:** create a complete graph with edge weights

$$w_{i,j} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{\sigma^2}\right)$$

Label Propagation: Setup

- Have:

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n_L)}), x^{(n_L+1)}, \ldots, x^{(n_L+n_U)}$$

- **Approach:** create a complete graph with edge weights

$$w_{i,j} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{\sigma^2}\right)$$

- Define a transition matrix T with

$$T_{i,j} = P(j \to i) = \frac{w_{i,j}}{\sum_{k=1}^{n_L+n_U} w_{k,j}}$$

Label Propagation: Algorithm

- Set Y to be a $(n_L + n_U) \times C$ matrix with each row i the **class distribution** of point i (labeled or unlabeled)

- At each iteration,
  1. Propagate: $Y \leftarrow TY$
  2. Normalize Y (row-wise)
  3. Clamp labeled data

$$Y = \begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0.4 & 0.3 & 0.3 & 0 \end{bmatrix}$$

(force the labeled points to their known distributions, i.e. 1 for their label's class, 0 for the others)

- Continue until convergence

Label Propagation: Recap

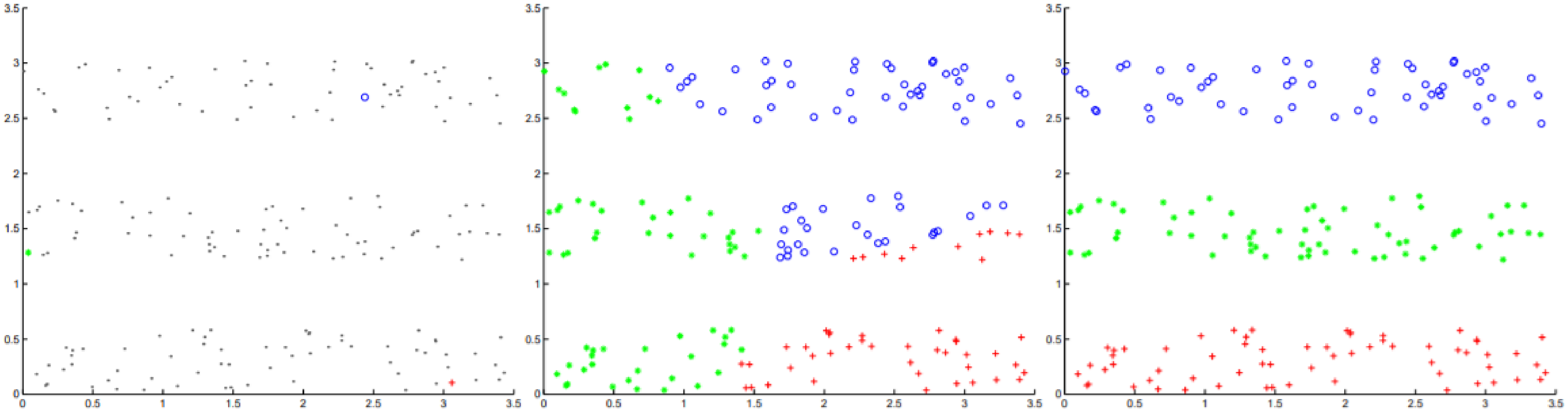
- At each iteration,
  1. Propagate: $Y \leftarrow TY$
  2. Normalize Y (row-wise)
  3. Clamp labeled data
- Continue until convergence

$$Y = \begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0.4 & 0.3 & 0.3 & 0 \end{bmatrix}$$

- Basic intuition:
  - pump signal (label distribution strength) from labeled data repeatedly into regions of low label density
  - the propagation spreads most rapidly through nearby points

Label Propagation: Results

- Let's compare this to just using kNN to label points:



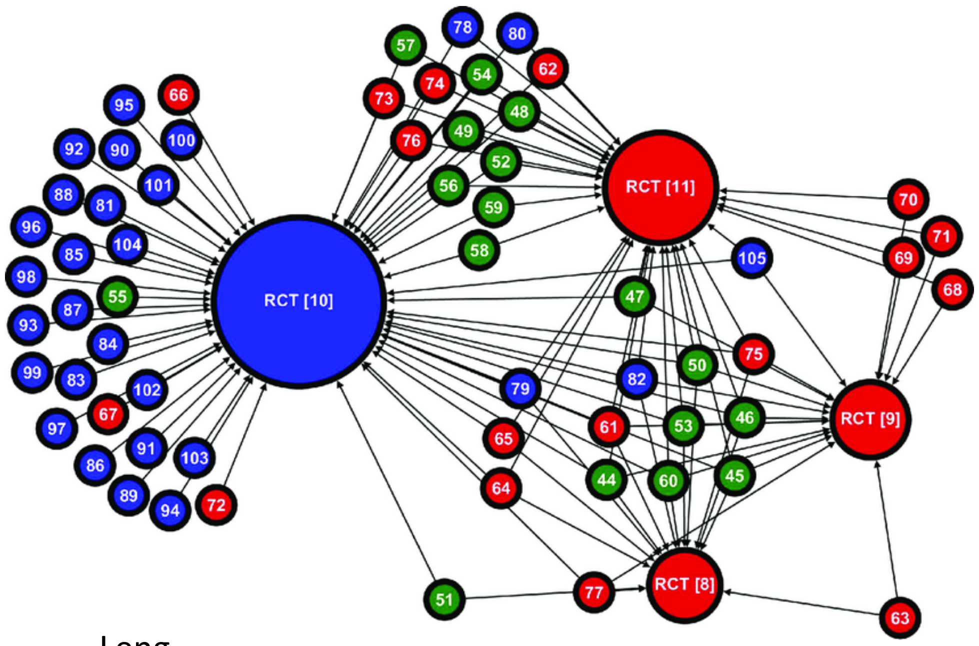
(a) The data (b) kNN (c) Label propagation

- 3 color strips:
  - one labeled point in each.
  - kNN ignores structure
  - label propagation uses structure

Graph Neural Networks: Motivations

- **Setting**: data that comes with some associated graph structure indicating similarity

- **Example:** citation networks.
  - Instances are scientific papers
  - Labels: subfield/genre
  - Graphs: if a paper cites another, there's an edge between them



Leng

Graph Neural Networks: Approach

- **Idea**: want to use the graph information in our predictions.
- **Semi-supervised aspect:** don't need all the graph's nodes to be labeled; instead, use the trained network to predict unlabeled nodes.
- One popular network: graph convolutional network (GCN)

$$f(X, A) = \text{softmax}(\underbrace{A}_{\text{adjacency matrix}}\sigma(AX\underbrace{W^{(0)}}_{\text{layer 1 weights}})\underbrace{W^{(1)}}_{\text{layer 2 weights}})$$

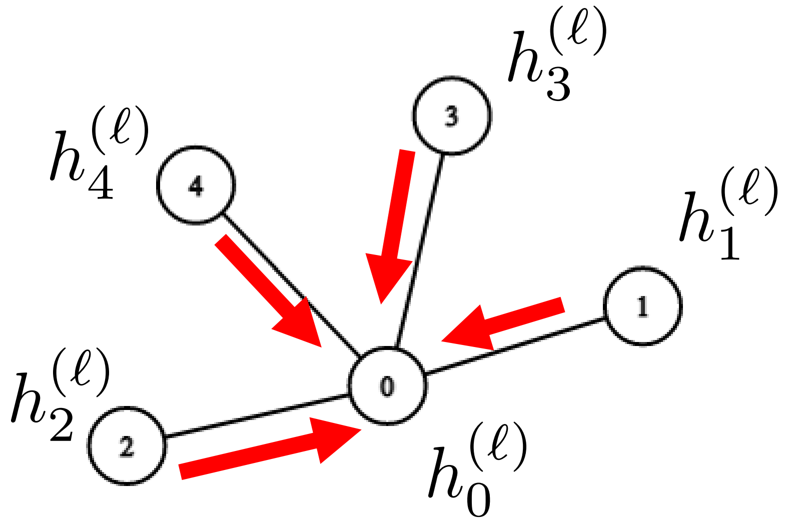
Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"

Graph Convolutional Networks

- One popular network: graph convolutional network (GCN)

$$f(X, A) = \mathrm{softmax}(A\sigma(AXW^{(0)})W^{(1)})$$

- Just like a feedforward network, but also mix together nodes by multiplying by adjacency matrix
- Can also normalize, use Laplacian, many variations



Graph Convolutional Networks

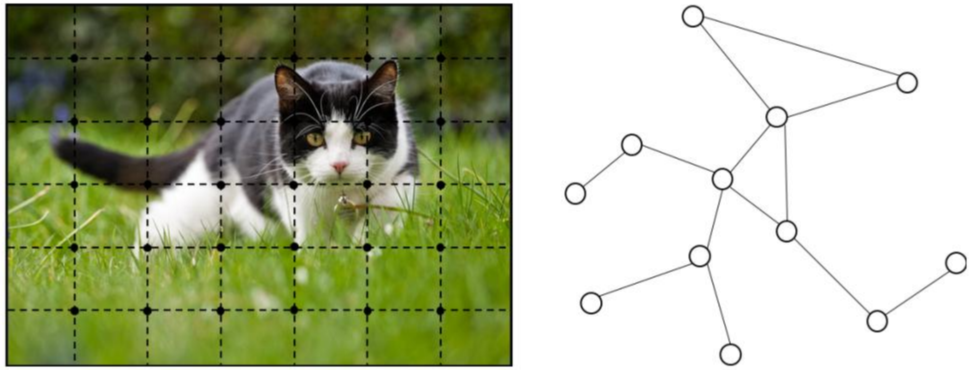
- One popular network: graph convolutional network (GCN)

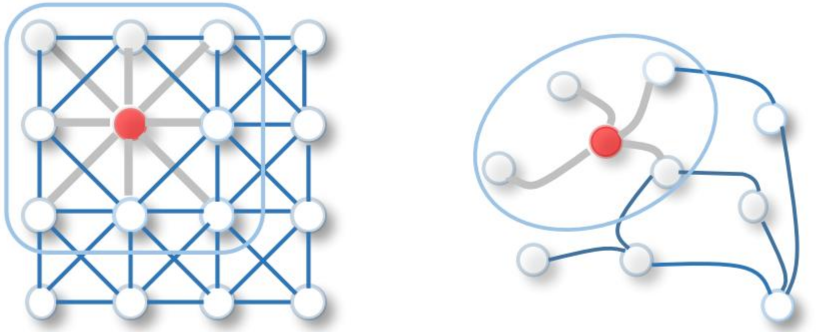
$$f(X, A) = \mathrm{softmax}(A\sigma(AXW^{(0)})W^{(1)})$$

Note the resemblance to CNNs:

- Pixels: arranged as a very regular graph
- Want: more general configurations (less regular)

Wu et al, A Comprehensive Survey on Graph Neural Networks





Zhou et al, Graph Neural Networks: A Review of Methods and Applications

Break & Quiz

Break & Quiz

True or False

1. Label propagation will produce similar outcomes to k nearest neighbors when label density is high
2. Label propagation is guaranteed to recover the true labels for its unlabeled points.

A. True and True
B. True and False
C. False and True
D. False and False

Break & Quiz

True or False

1. Label propagation will produce similar outcomes to k nearest neighbors when label density is high
2. Label propagation is guaranteed to recover the true labels for its unlabeled points.

A. True and True
B. **True and False**
C. False and True
D. False and False

If label density is high, there will be nearby points (i.e. small distances) that are labeled so LabelProp will have similar behavior to kNN.

LabelProp works only if underlying distance assumption holds

Outline

- **What do we do if we don’t have enough data?**
  - Motivation, approaches, taxonomy
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Active Learning**
  - Pooling vs. streaming, learning thresholds, using margins
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning

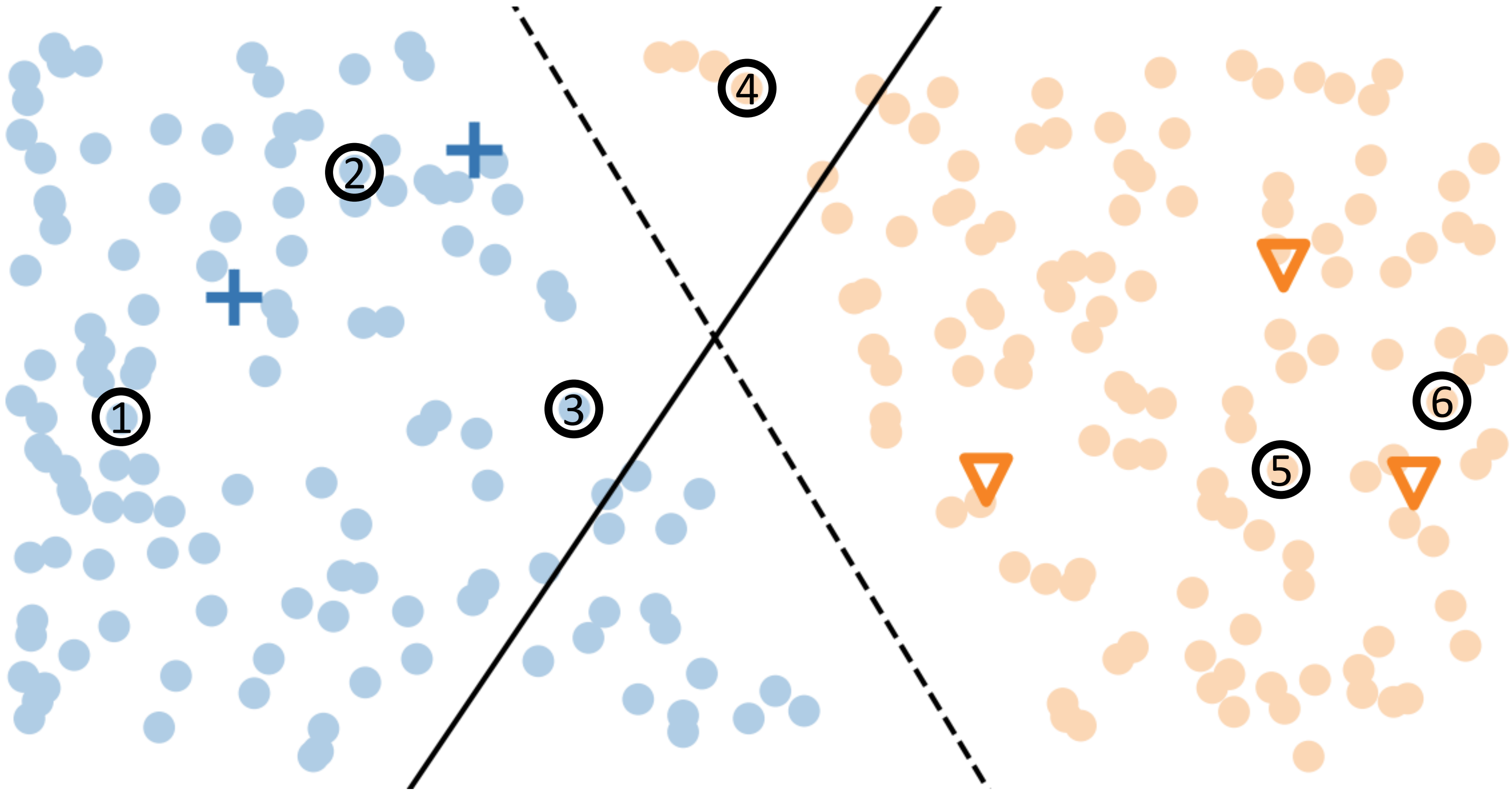
Active Learning: Setup

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$$

- So far we have collected data **passively:**
  1. sample unlabeled data points i.i.d. from a distribution
  2. label a random subset of them

- This reflects one way of obtaining data in practice, but we can also collect data **actively:**
  - the unlabeled points still come i.i.d. from some distribution
  - the learning algorithm decides whether or not to label them

Intuition: which point's label is most useful?



van Engelen & Hoos, 2020

Different active learning settings

Pool-based active learning: you are given a set of unlabeled i.i.d. points $x_1, \ldots, x_n \sim D$ and can pick which ones to label

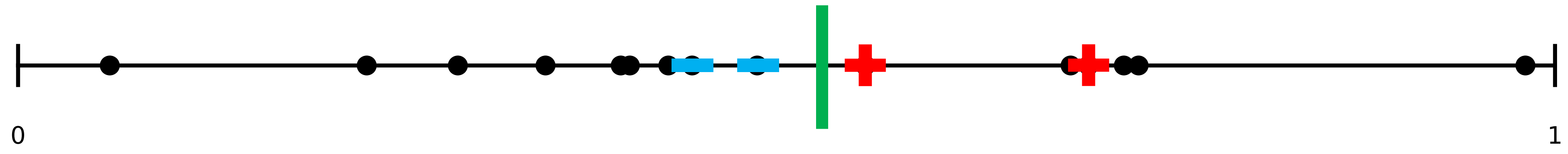
Stream-based active learning: you see the i.i.d. points $x_1, \ldots$ sequentially and must **irrevocably** decide whether to label x_i before seeing x_{i+1}

Other settings exist

The one you're in depends on the application

Canonical example: Learning a threshold

Suppose we want to classify $x \in \mathrm{Uniform}[0,1]$ that we know can be labeled by a threshold function $h_\theta(x) = 1\{x \geq \theta\}$



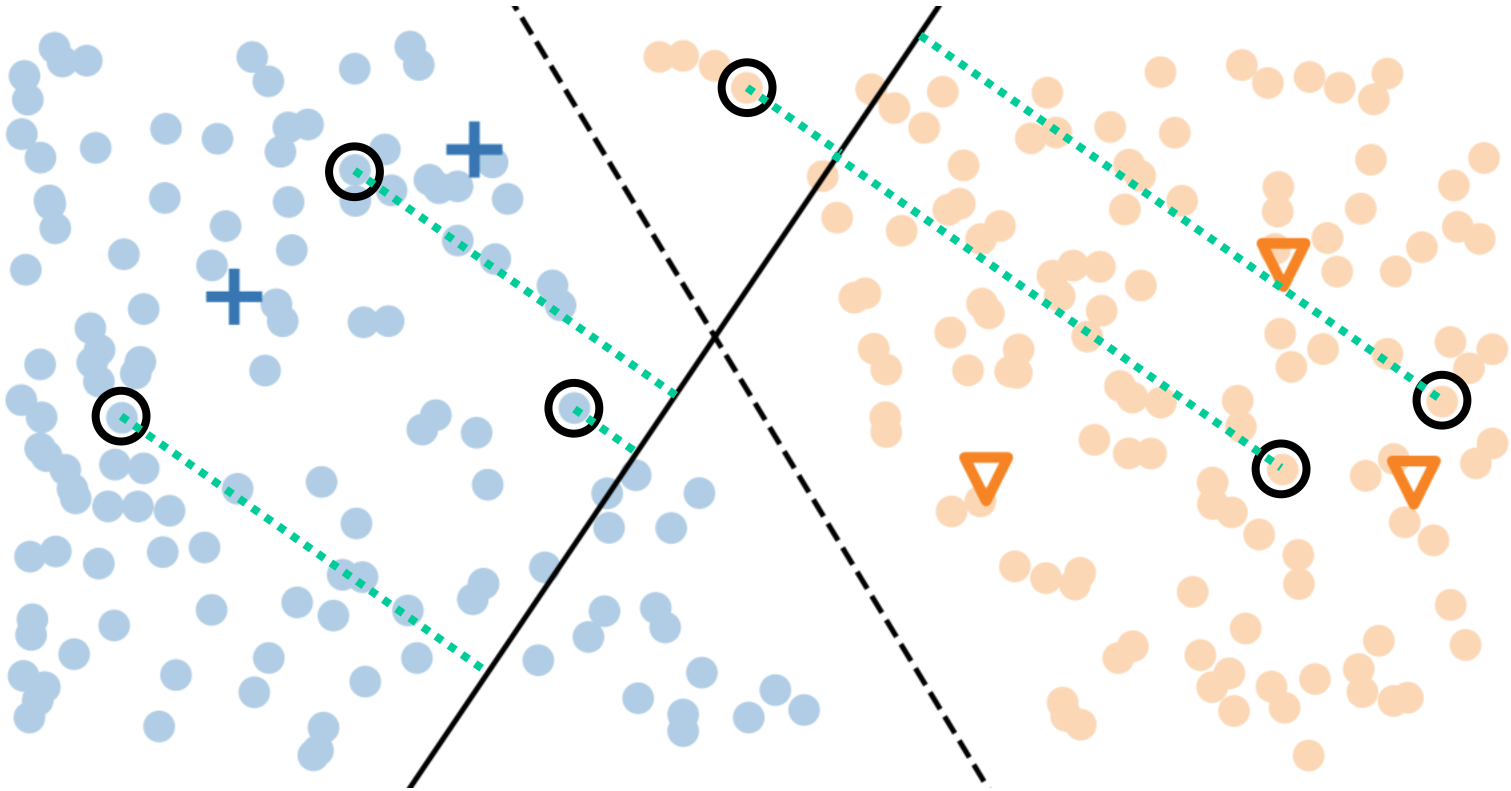
- if we label the points i.i.d. we need $\Omega({}^{1}/_{\varepsilon})$ samples to learn a threshold $\widehat{\theta}$ with error ε

- if we label using binary search we only need $O(\log \frac{1}{\varepsilon})$!

Margin-based active learning

Intuition: harder-to-classify points are more informative

Idea: train a linear classifier on a few i.i.d. points, then actively pick points based on distance from the classifier



van Engelen & Hoos, 2020

Active learning summary

Theoretically:

- **Goal:** prove much (e.g. exponentially) smaller sample complexity relative to passive learning (e.g. $O(\log\frac{1}{\varepsilon})$ vs. $\Omega({}^{1}/_{\varepsilon})$)
- **Reality:** hard to show outside linear models / nice distributions

Empirically:

- Lots of heuristics for selecting points, often based on their estimated difficulty (as in margin-based active learning)
- Numerous applications where labels are expensive

Outline

- **What do we do if we don’t have enough data?**
  - Motivation, approaches, taxonomy
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Active Learning**
  - Stream-based, thresholds, pool-based, margin-based
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning

Weak Supervision: Motivation

- As before, labels are very expensive to get.

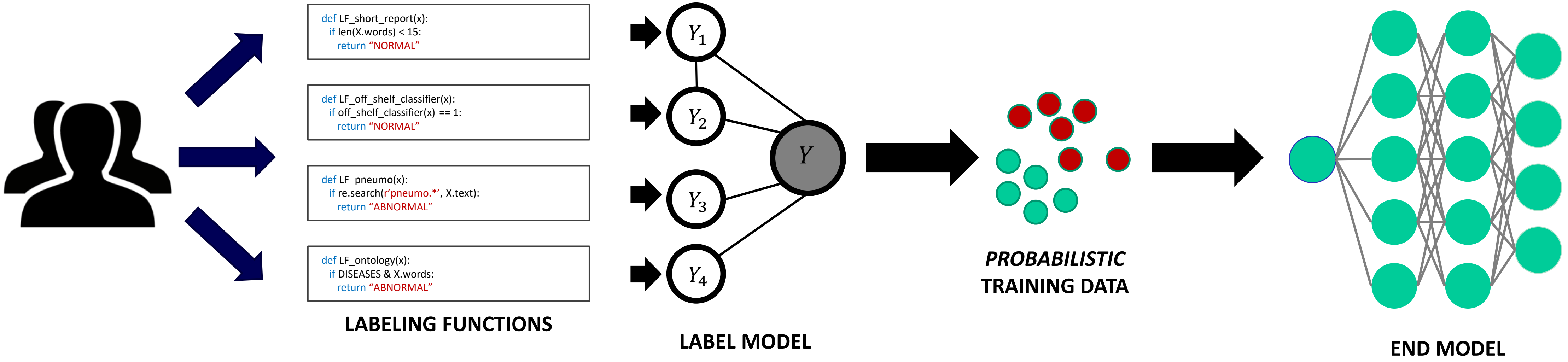
- Sometimes we can get **cheaper sources** to label points
  - Noisy…
  - But can acquire several of them

- Some examples of sources:
  - Heuristics (expressed via small programs)
  - Pre-trained models
  - Lookups in knowledge bases
  - Crowdsourced workers

```
@labeling_function()
def check_out(x):
    return SPAM if "check out" in x.text.lower() else ABSTAIN
```

Weak Supervision: Pipeline

Three components



1. Users write ***labeling functions*** to create noisy labels

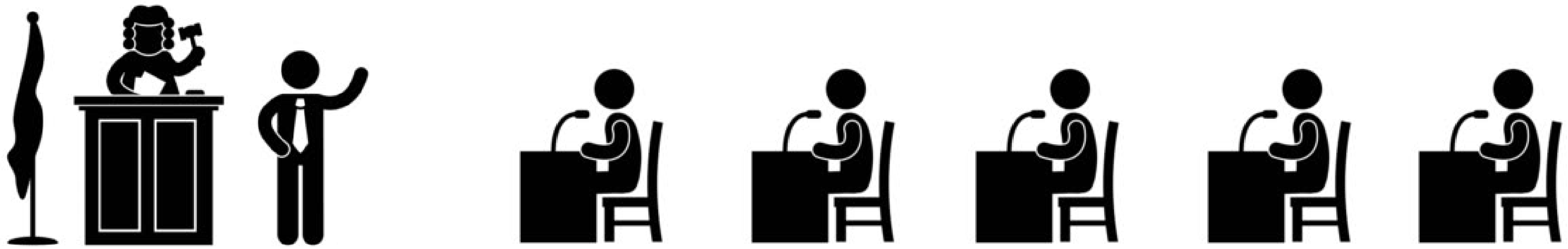
2. We **model** and **combine** these labels

3. The generated labels are used to **train a downstream model**

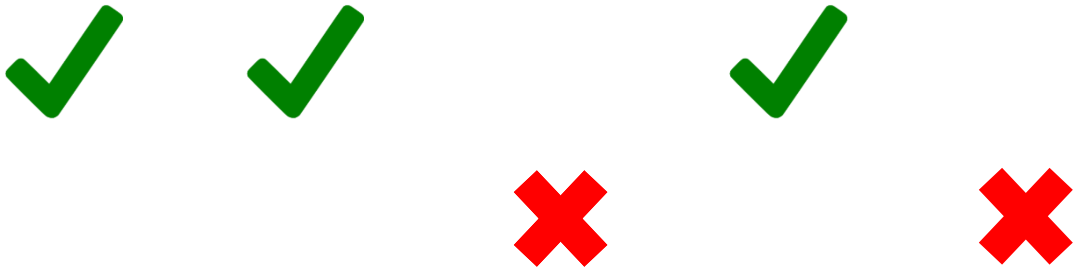
Weak Supervision: Intuition & Majority Vote

Pretend we're in court:

Witnesses



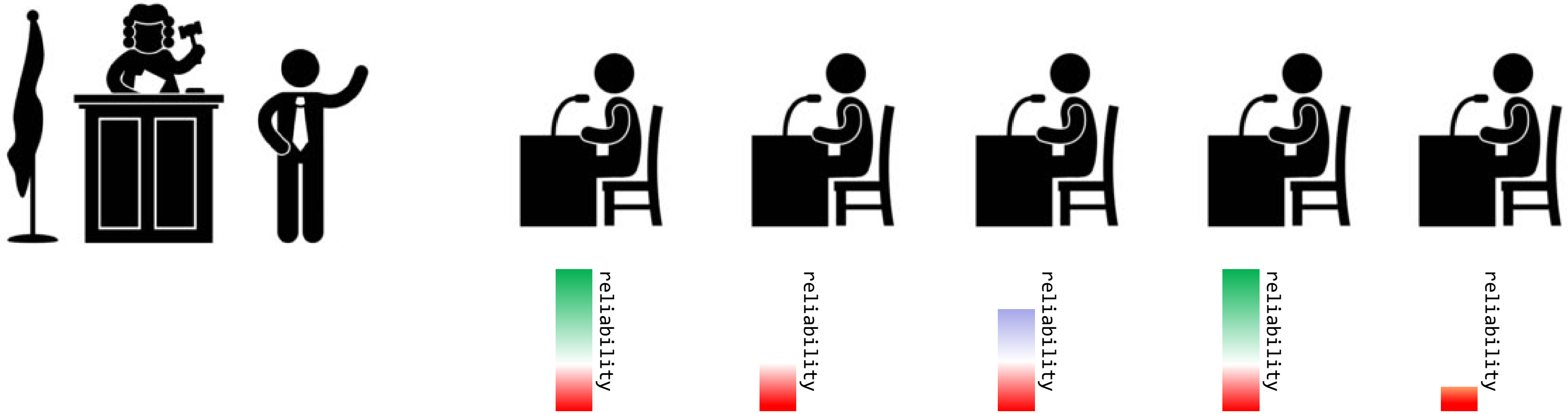
Votes



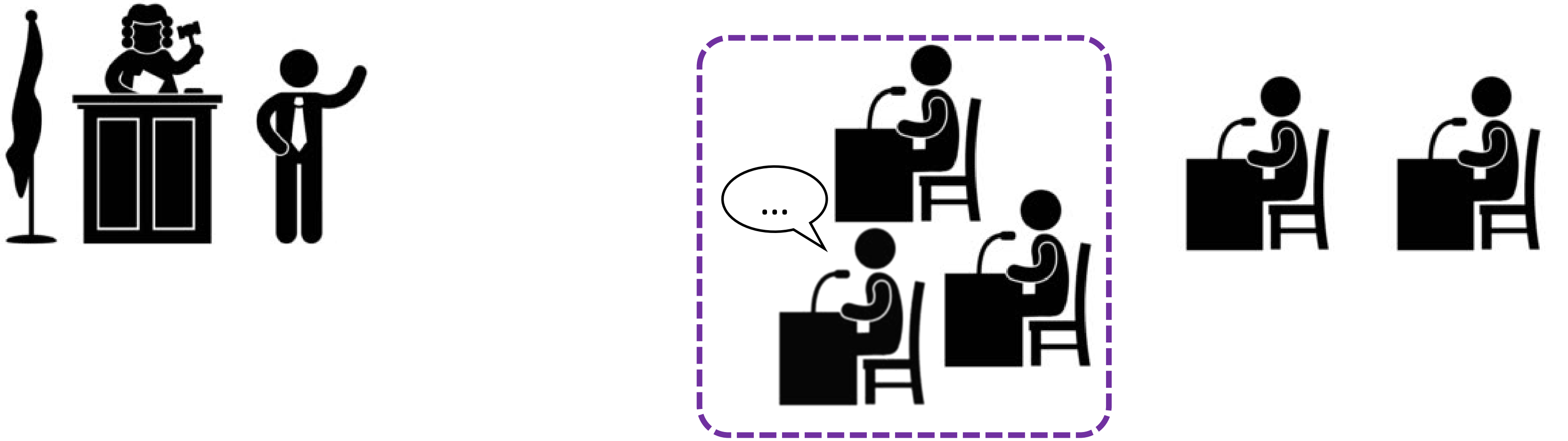
Naïve approach: **majority vote**

Weak Supervision:

Can we do better? Some witnesses can be less reliable, and/or some of them may vote in a block.



Incorporate **accuracies**



Incorporate **correlations**

Weak Supervision: Label Model

Suppose we have labeling functions $\lambda_1, \lambda_2, \ldots, \lambda_m$ and the true (unobserved) label is Y.

- **Goal:** we want to compute the **conditional probability**

$$P(Y|\lambda_1, \lambda_2, \ldots, \lambda_m)$$

- **Why**: want to know given a set of votes from the *m* labeling functions, how likely is Y to be 0? To be 1?

- **Approach:**
  - model the accuracies and correlations of different labeling functions using a **probabilistic (graphical) model**
  - infer the model parameters using **unsupervised learning**
  - Use the resulting model to produce **(soft) labels for supervised training**

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Fred Sala