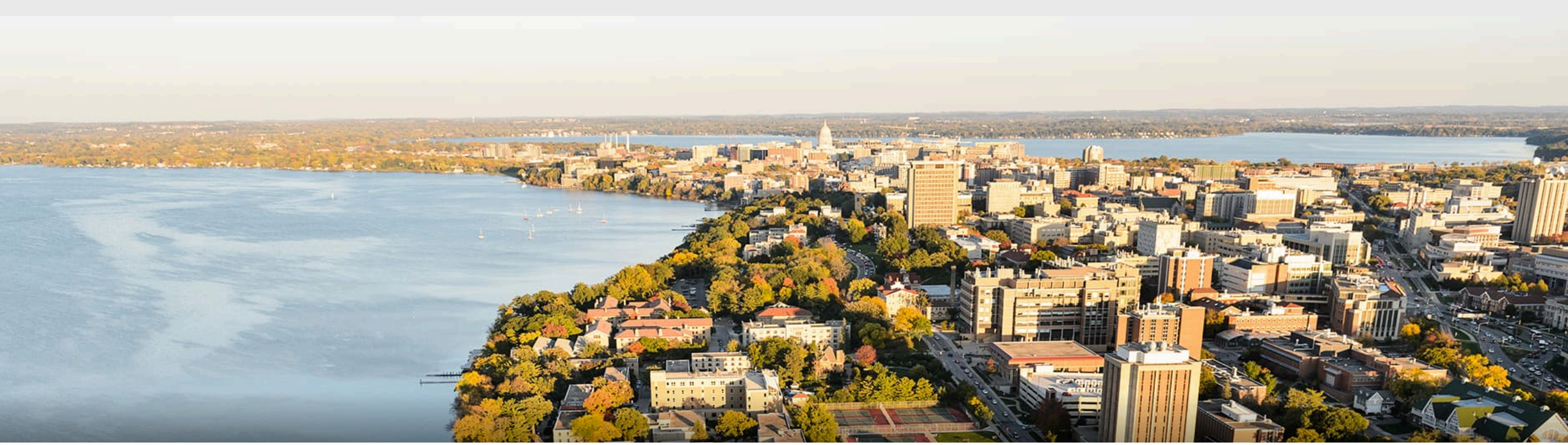


CS 760: Machine Learning
Less-than-full Supervision

Kirthi Kandasamy

University of Wisconsin-Madison

April 3, 2023

Outline

- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning
- **Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR
- **Next lecture: Kernels & SVMs**

Outline

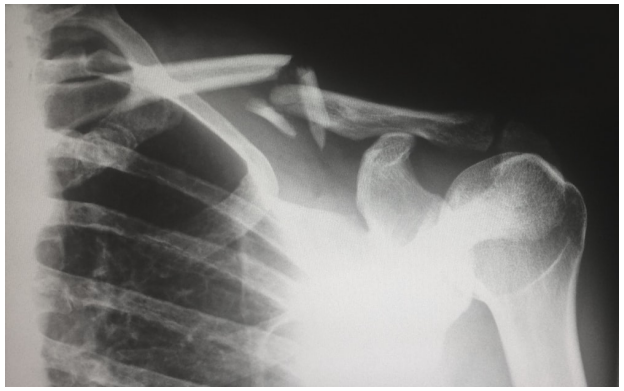
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning
- **Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR

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

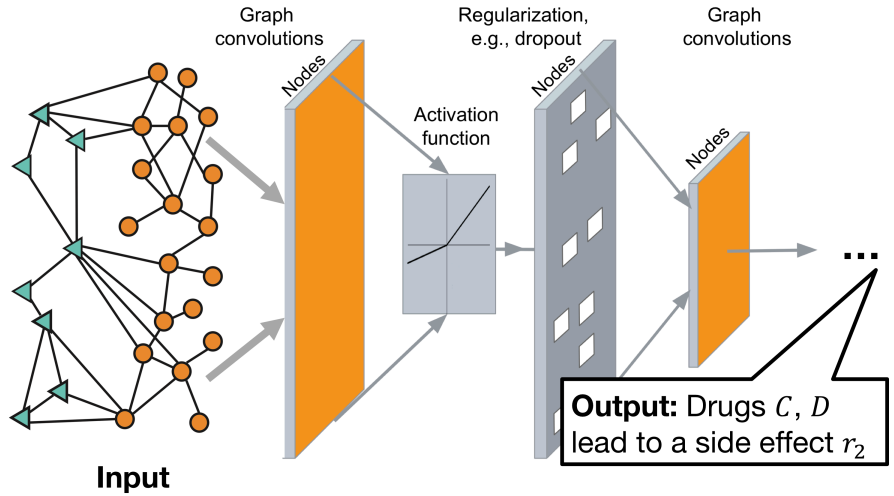
$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n_L)}, y^{(n_L)}), x^{(n_L+1)}, \ldots, x^{(n_L+n_U)}$$

n_L **labeled points**

n_U **unlabeled points**

Semi-Supervised Learning: Techniques

- Large space of approaches
  - Could cover a full class…
- We'll focus on a classic technique: **label propagation**
  - **Explicit:** computes labels for the unlabeled data, then train a model
- A modern set of techniques: **graph neural networks**
  - will not cover in this class



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

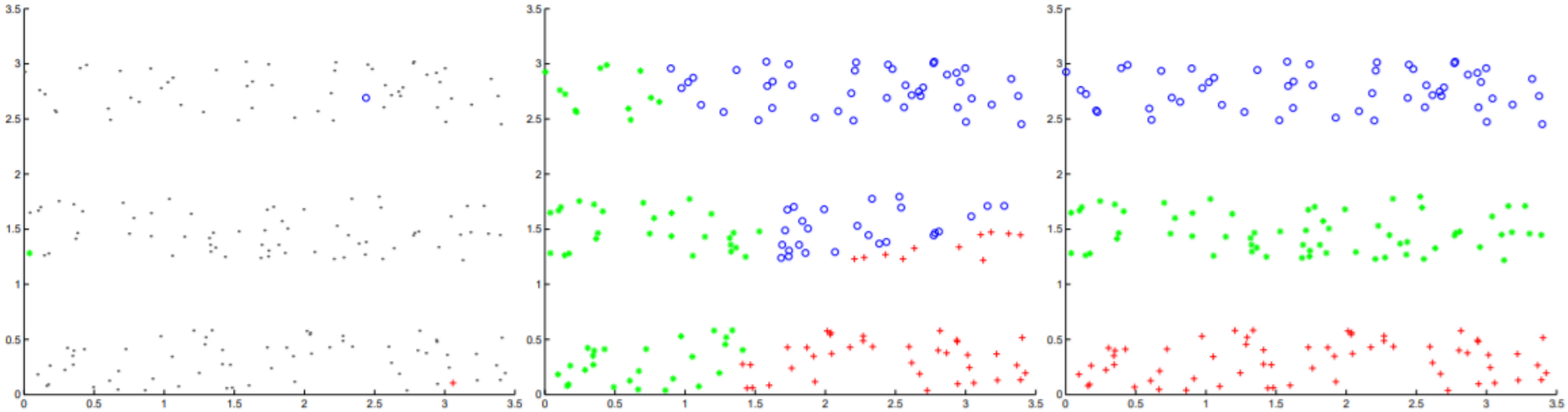
- The algorithm is simple. Set Y to be a *(nL+nU)*x*C* matrix with each row the distribution of point I (labeled or unlabeled)
- At each iteration,
  1. Propagate: Y ← TY
  2. Normalize (row-wise) Y
  3. Clamp labeled data

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

- Continue until convergence
  - **Clamping**: force the labeled points to their known distributions (ie, 1 for their label's class, 0 for the others)

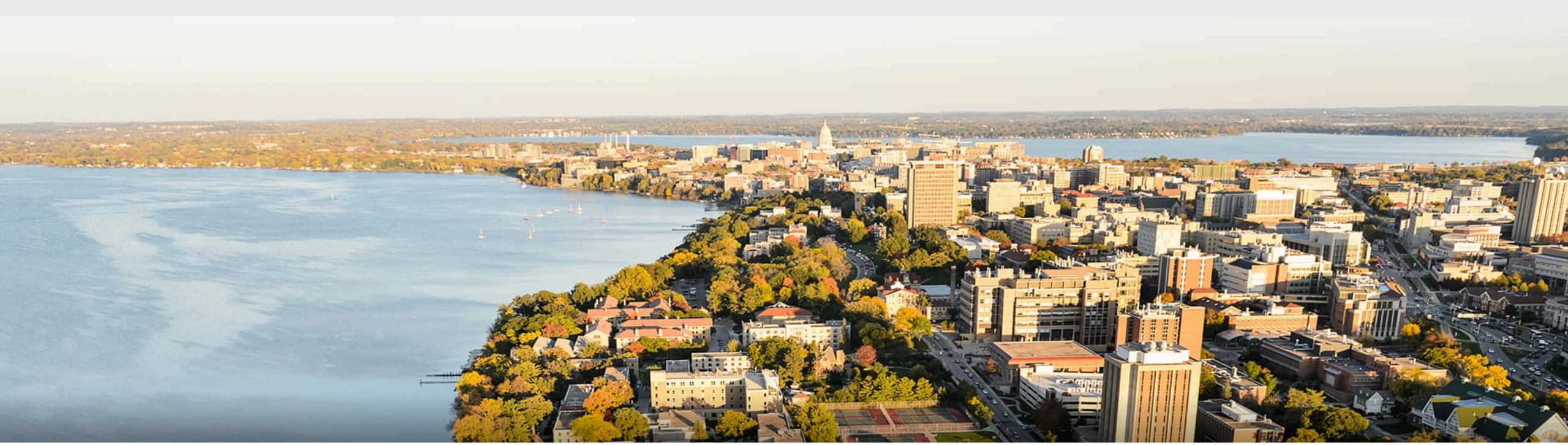
Label Propagation: Results

- 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 it.



Break & Quiz

Break & Quiz

Q 1-1: 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

Q 1-1: 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 (ie, small distances) that are labeled so LabelProp will have similar behavior to kNN.

LabelProp works only if underlying distance assumption holds.

Outline

- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning
- **Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR

Weak Supervision: Motivation

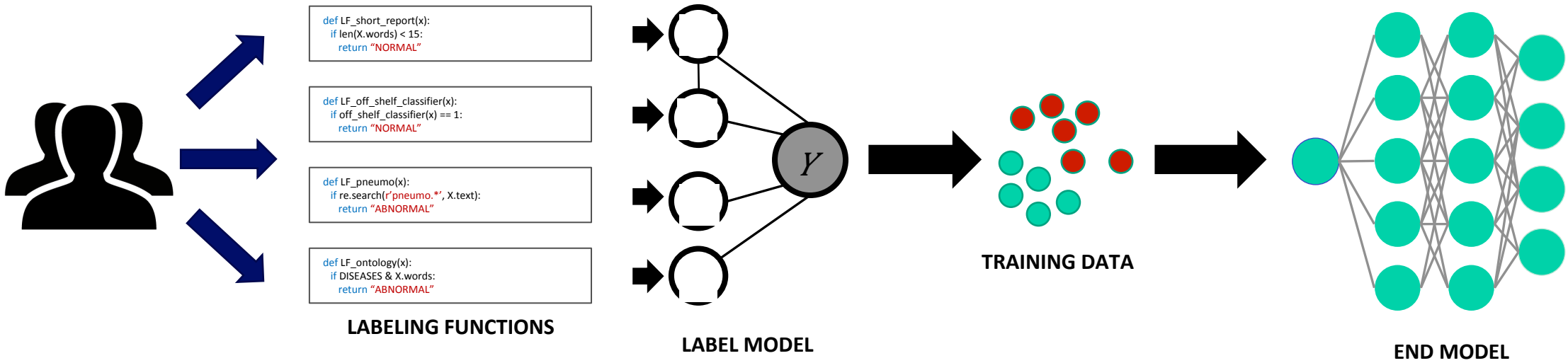
- As before, labels are very expensive to get.
- Sometimes we can get **cheaper sources** to label points
  - Noisy…
  - But can acquire several of them

- 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

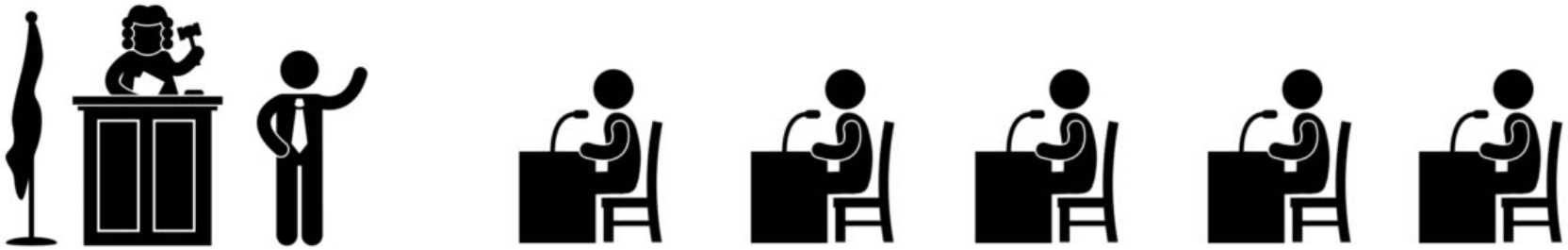
2. **Model** and **combine** these labels

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

Weak Supervision: Intuition & Majority Vote

- Pretend we're in court:

Witnesses

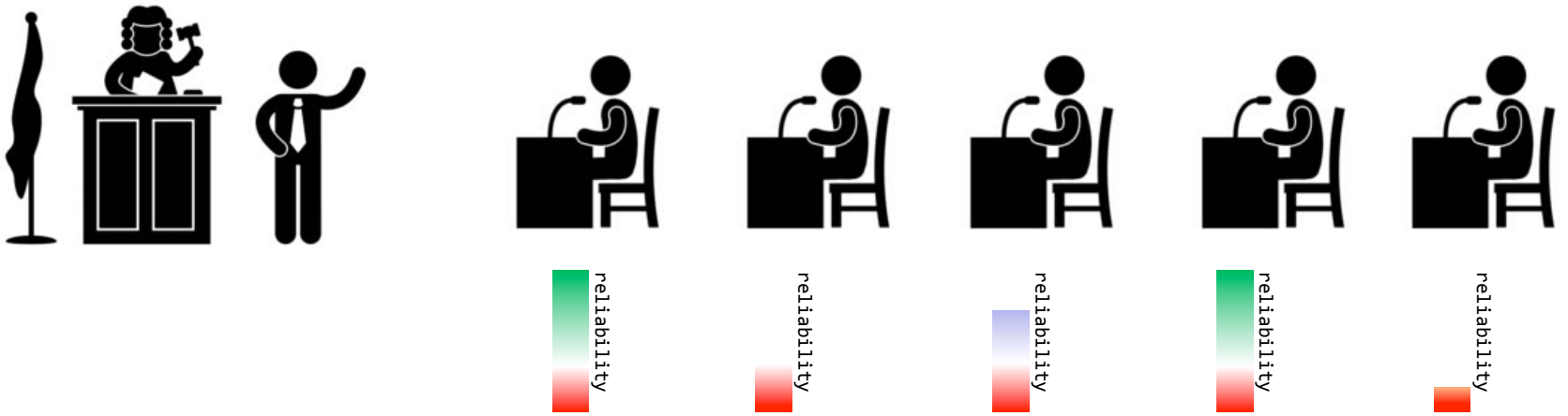


Votes

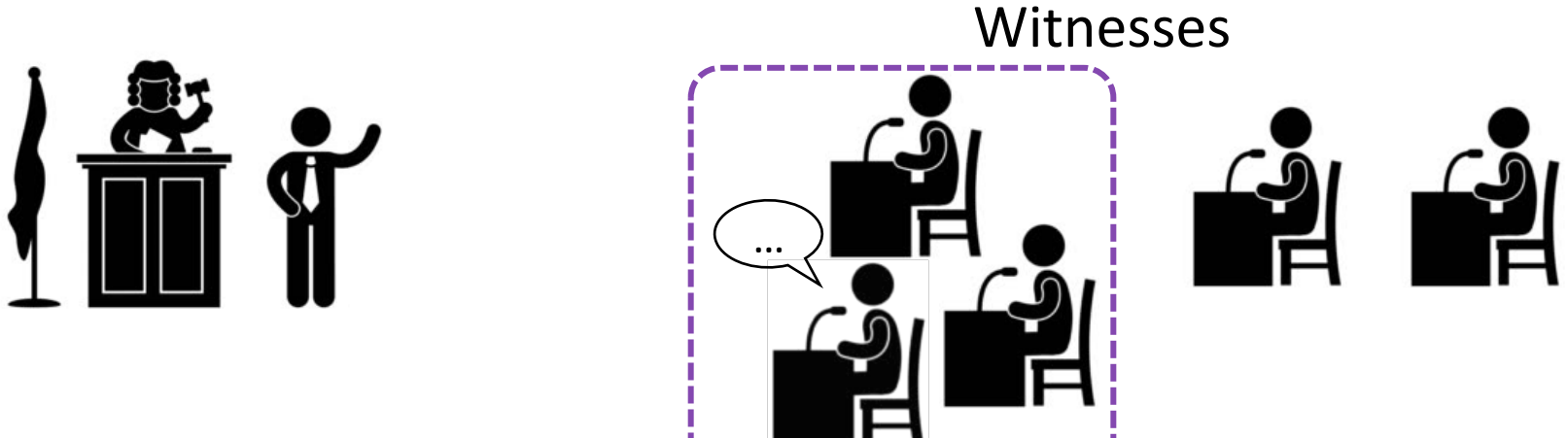
Naïve approach: **majority vote**

Weak Supervision:

- Can we do better?
  - Some witnesses more reliable, others are voting in a bloc



1. Incorporate **accuracies**



2. Incorporate **correlations**

Weak Supervision: Label Model

- Suppose we have labeling functions $\lambda_1, \lambda_2, \ldots, \lambda_m$ and the true (unobserved) label is .
- **Goal:** we want to compute the **conditional probability**

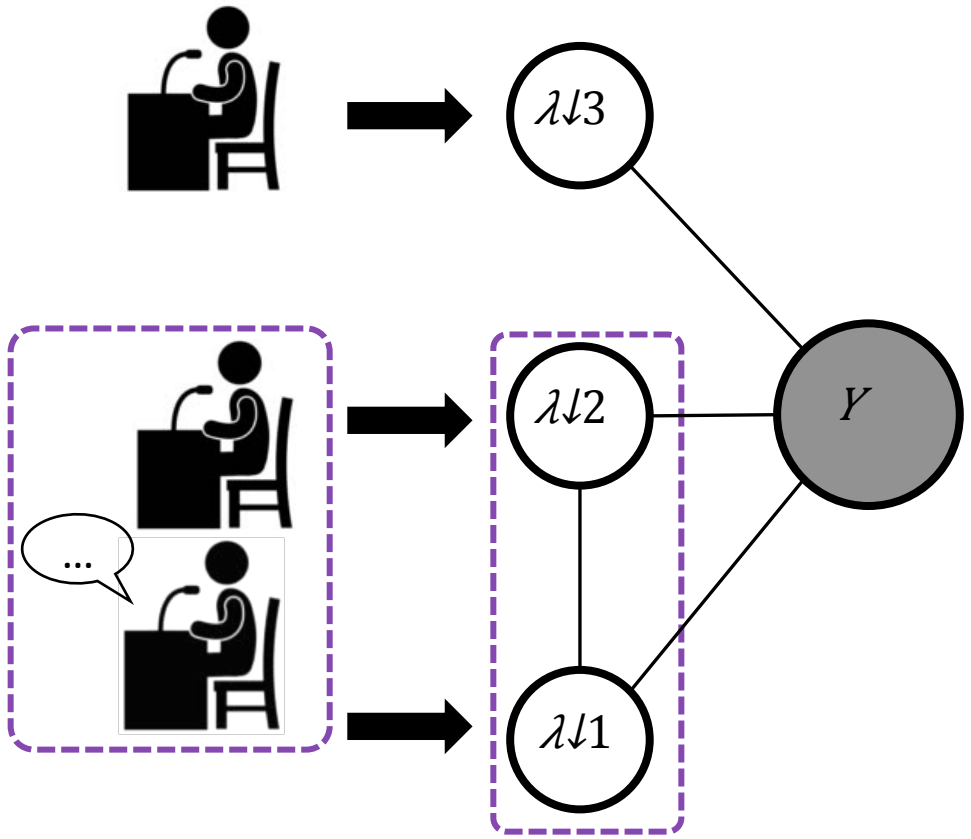
$$\mathbb{P}(Y|\lambda_1, \lambda_2, \ldots, \lambda_m)$$

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

- Q: Encode this information into an undirected graphical model: the **Label Model**

Weak Supervision: Label Model Structure

- Basic idea: $p(\lambda_1, \ldots, \lambda_m, y) = \frac{1}{Z} \exp\left(\theta_1 \lambda_1 y + \theta_2 \lambda_2 y + \ldots + \theta_m \lambda_m y\right)$



Outline

- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning
- **Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR

Self Supervision: Basic Idea

- Suppose we have no labeled data, nor weak sources
- What can we do with unlabeled data?
  - Generative modeling, etc.
  - Could also obtain **representations** (ie new features) for **downstream use.**
- Need to create tasks from unlabeled data: “Pretext tasks”
  - Ex: predict stuff you already know

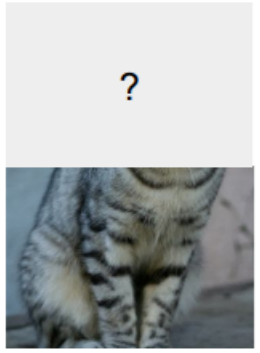
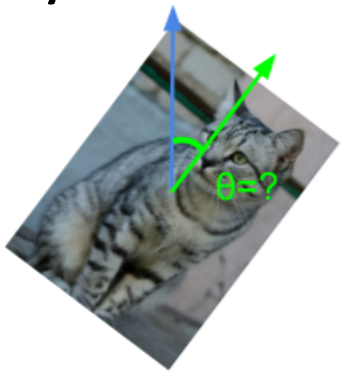


image completion



rotation prediction

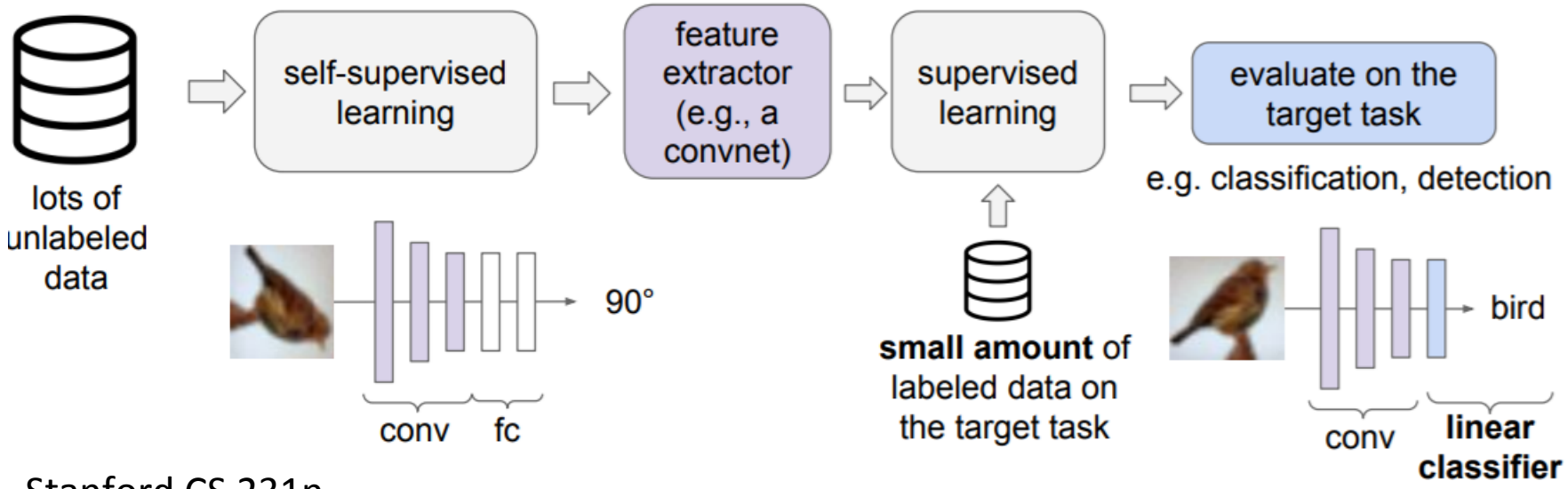
“jigsaw puzzle”

colorization

Representations using pretext tasks

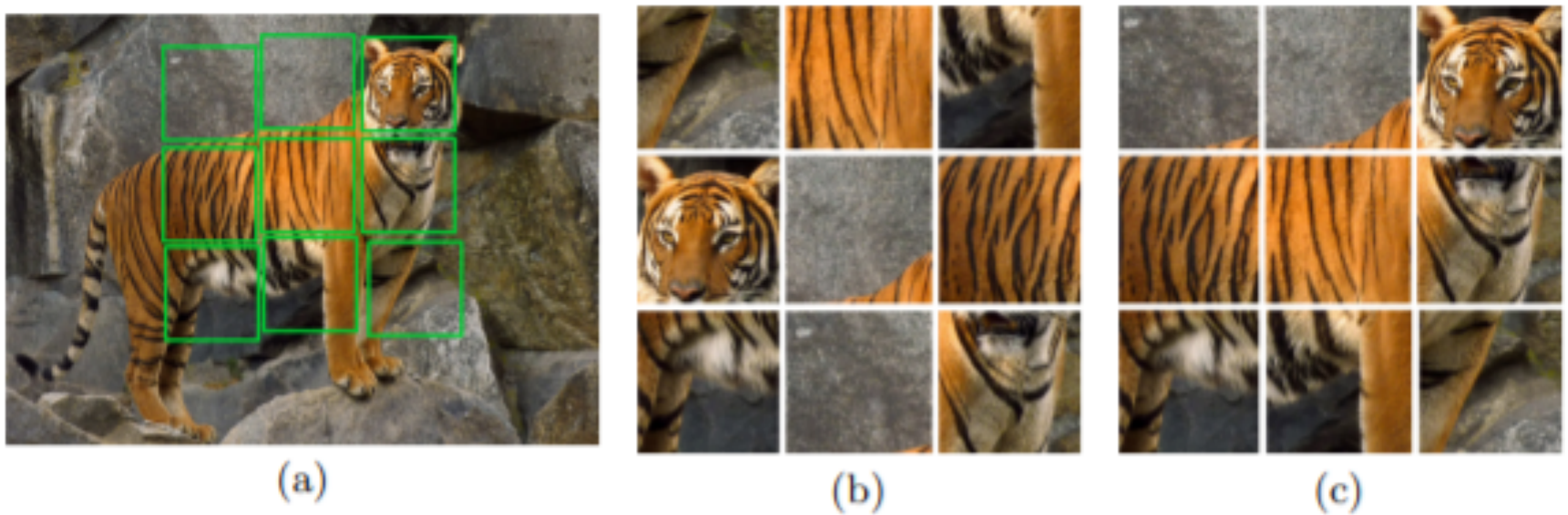
Intuition: a network that does well on pretext tasks has also learned some representation that is useful for the actual task.

- Use the learned network as a feature extractor
- Then, train using a small amount of data



Self Supervision: Pretext Tasks

- Lots of options for pretext tasks
  - Predict rotations
  - Coloring
  - Fill in missing portions of the image
  - Solve puzzles:

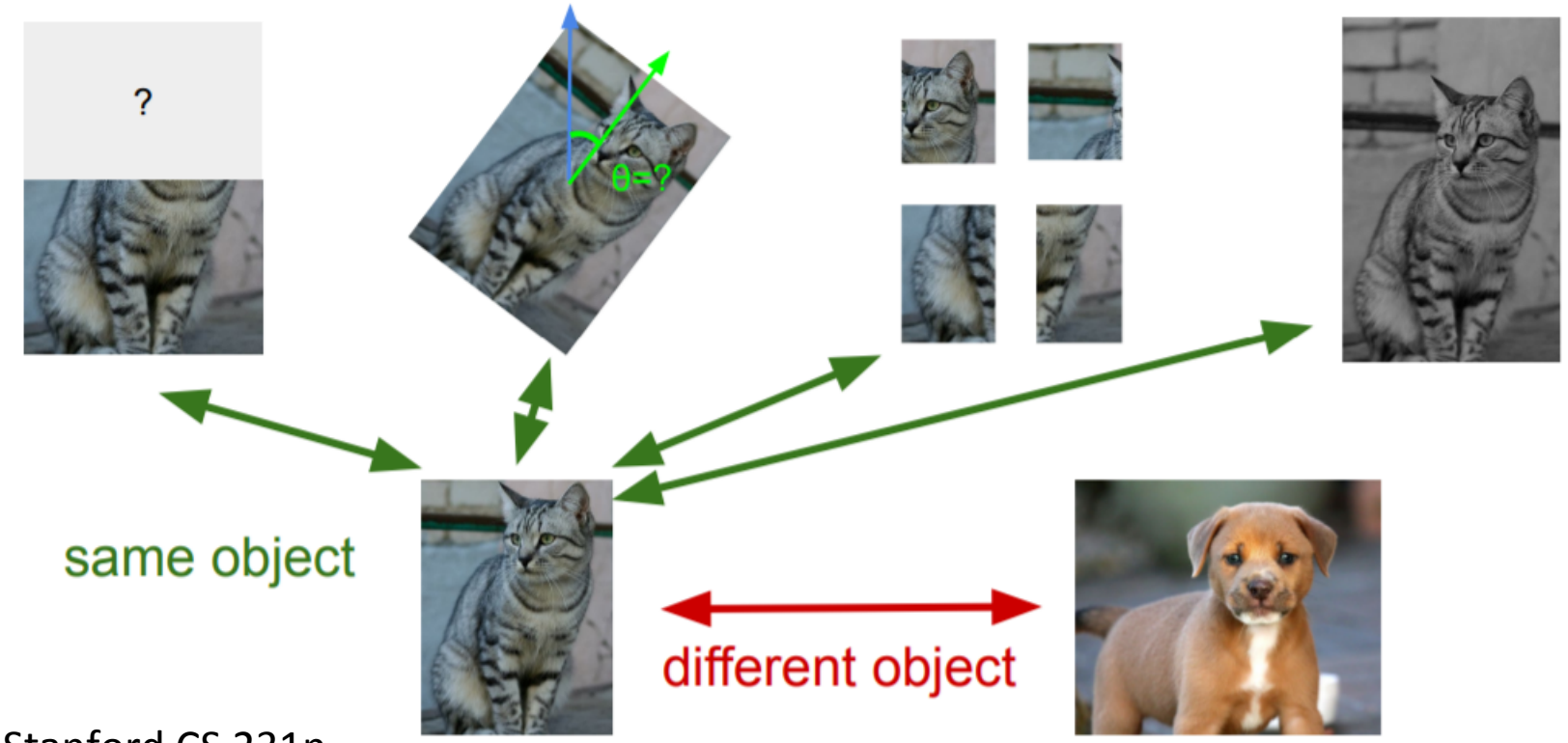


(a) (b) (c)

Noroozi and Favaro

Contrastive Learning: Basics

- Want to learn representations so that:
  - Transformed versions of single sample are similar
  - Different samples are different

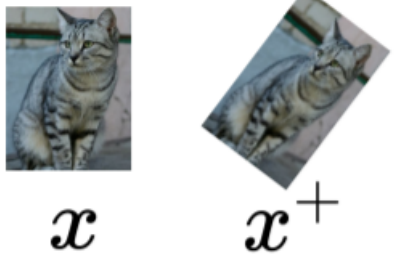


Contrastive Learning: Motivation

- Contrastive learning goal:
  - Keep together related representations, push unrelated apart.
  - The InfoNCE loss function:

Van den Oord et al., 2018

$$L = -E_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{k-1} \exp(s(f(x), f(x_j^-))} \right]$$

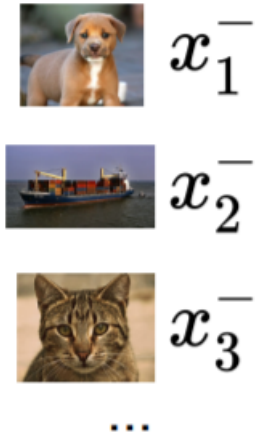


x x^+

Positive sample: keep close

Negative sample keep far

x

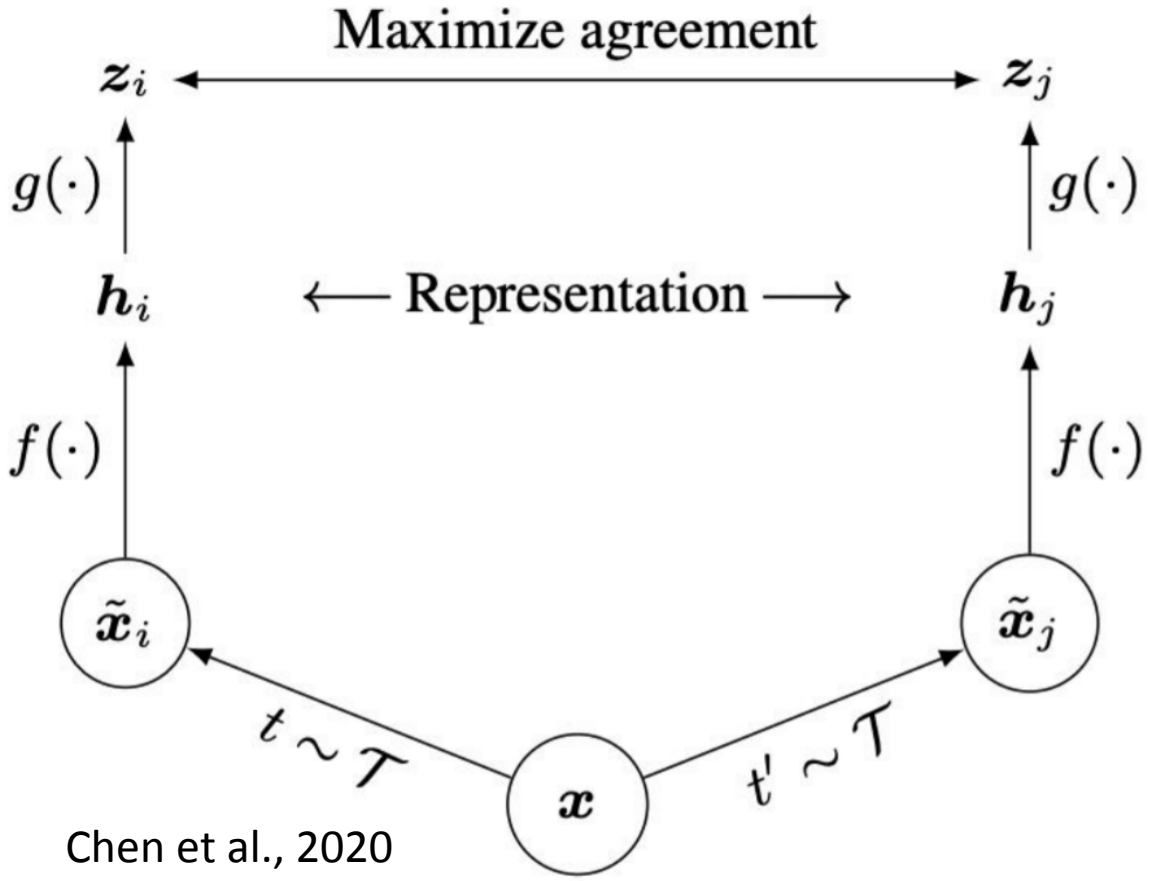


x_1^- x_2^- x_3^- ...

Contrastive Learning: Frameworks

- Many approaches (very active area of research)
  - A popular approach: SimCLR. Score function is cosine similarity,

    - Generate positive samples:
    Choose random augmentations



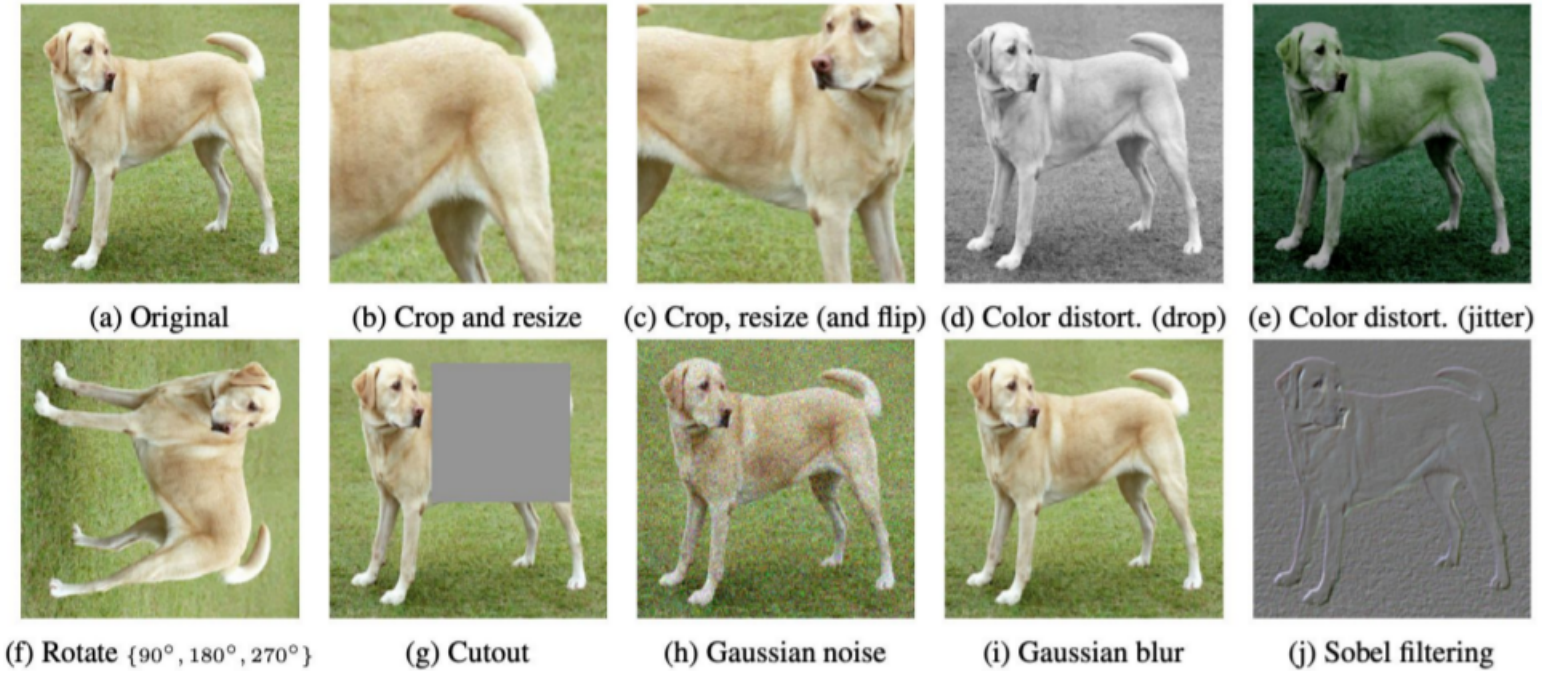
Chen et al., 2020

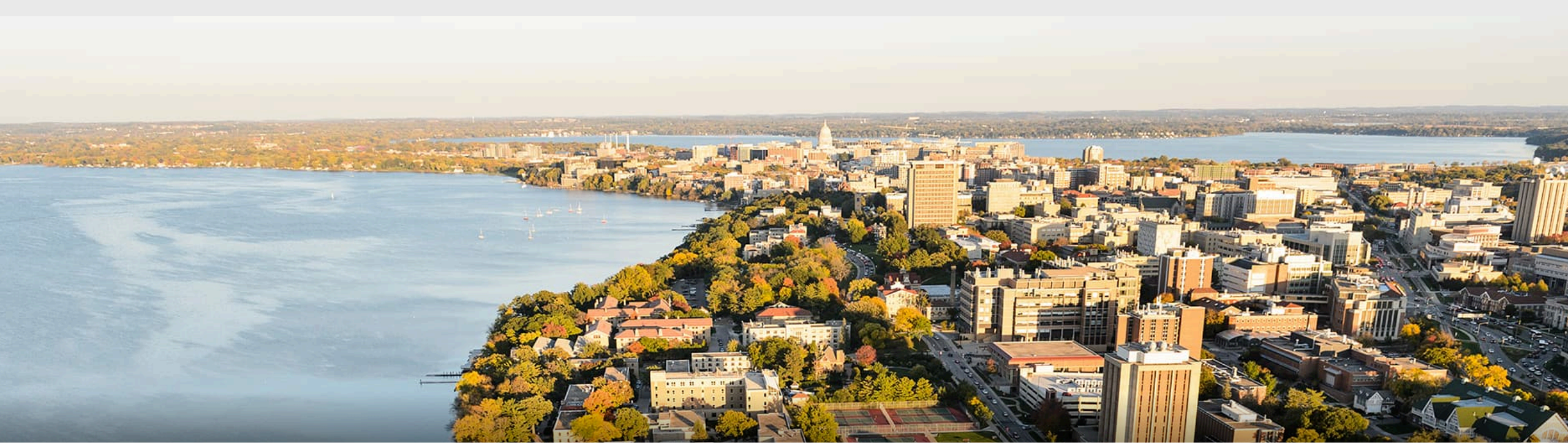
Contrastive Learning: Frameworks

- Many approaches (very active area of research)
  - A popular approach: SimCLR. Score function is cosine similarity,

  - Generate positive samples:

    Choose random augmentations





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