CS 760: Machine Learning
Less-than-full Supervision

Fred Sala

University of Wisconsin-Madison

Nov. 11, 2021

Announcements

- **Logistics**:
  - HW 6 is out (due Tuesday night).
  - Midterm grading complete: will release today.
- Class roadmap:

Thursday, Nov. 11	Less-than-full Supervision
Tuesday, Nov. 16	Unsupervised Learning I
Thursday, Nov. 18	Unsupervised Learning II
Tuesday, Nov. 23	Learning Theory
Tuesday, Nov. 30	Reinforcement Learning I

UL + RL

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

Outline

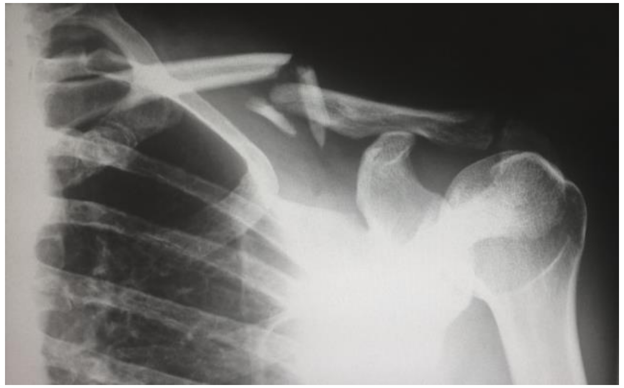
- **Semi-Supervised Learning**
  - Basic setup, label propagation, graph neural networks
- **Weak Supervision**
  - Labeling functions, accuracies & correlations, learning
- **Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR

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

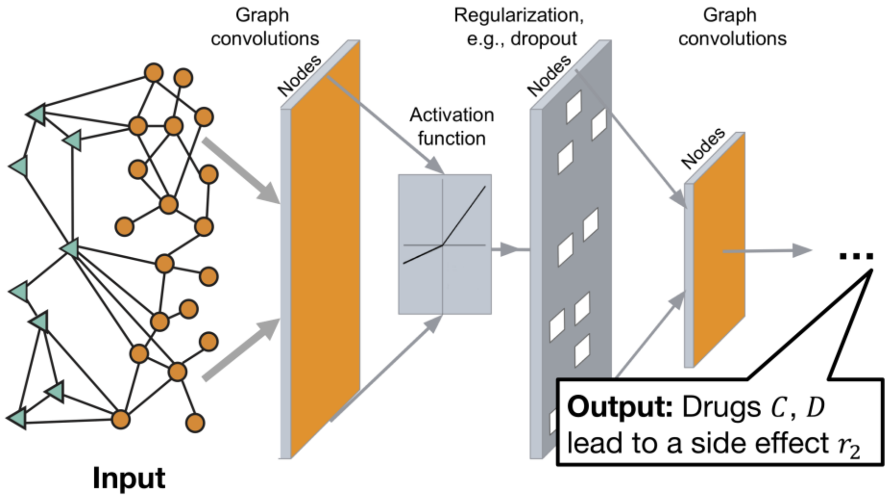
$$(x^{(1)}, y^{(1)}), \ldots, (x^{(n_L)}, y^{(n_L)}), x^{(n_L+1)}, \ldots, x^{(n_L+n_U)}$$

n_L **labeled points**

n_U **unlabeled points**

Semi-Supervised Learning: Techniques

- Huge space of approaches
  - Could cover a full class…
- We'll focus on **two** today:
- A classic technique: **label propagation**
  - **Explicit:** computes labels for the unlabeled data, then train a model
- A modern set of techniques: **graph neural networks**
  - **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

- 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: Recap

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

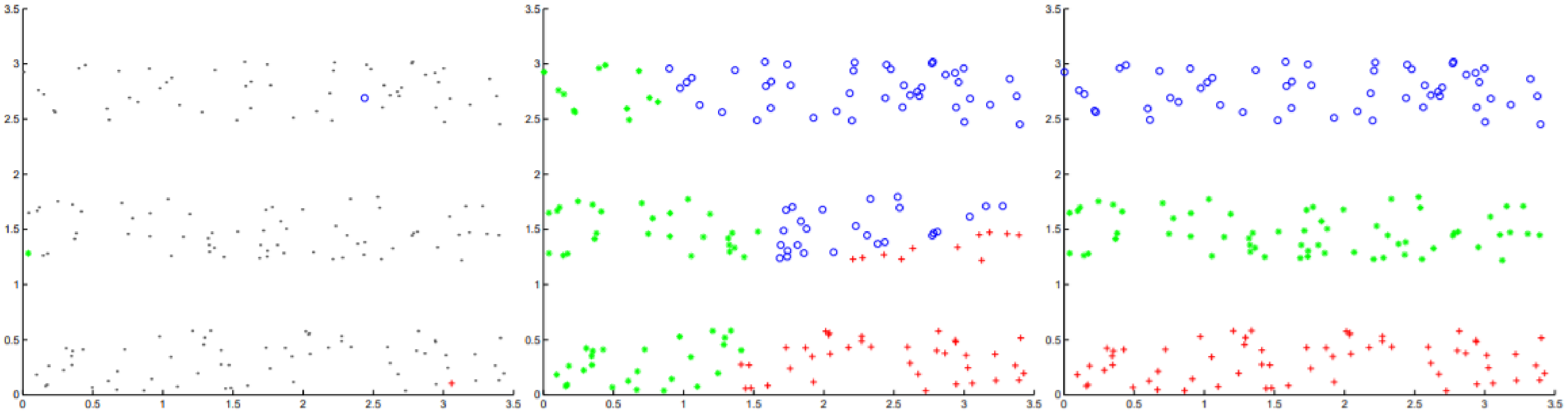
$$Y = \begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0.4 & 0.3 & 0.3 & 0 \end{bmatrix}$$

- Continue until convergence
- Initialization: for the unlabeled data, doesn't matter.

- Basic intuition: pump signal from labeled data repeatedly into regions of low label density
- One more thing: can learn σ via heuristics

Label Propagation: Results

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



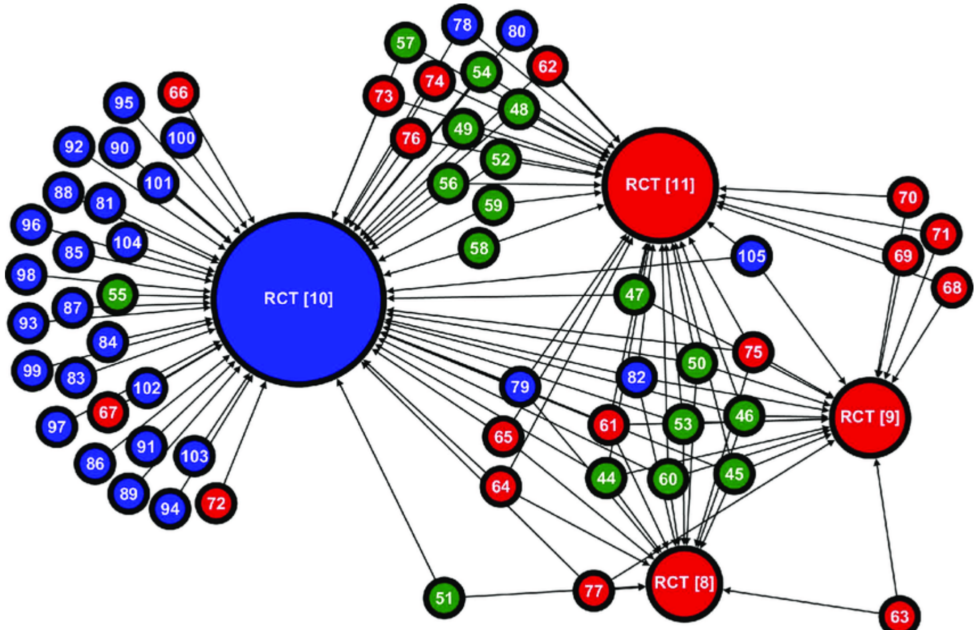
(a) The data (b) kNN (c) Label propagation

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

Graph Neural Networks: Motivations

- **Idea**: data comes with some associated graph structure that indicates similarity
  - Not necessarily built from features.

- 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---use the trained network to predict unlabeled nodes.
- One popular network: graph convolutional network (GCN)

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

Adjacency Matrix

Layer 1 Weights

Layer 2 Weights

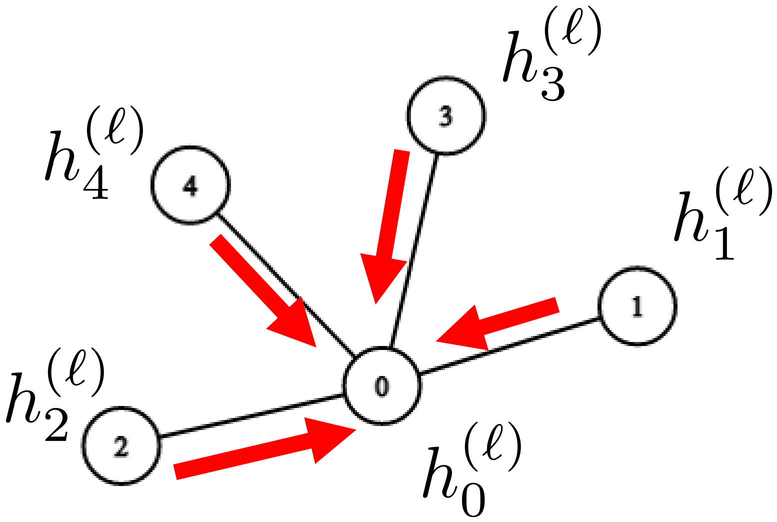
Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"

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

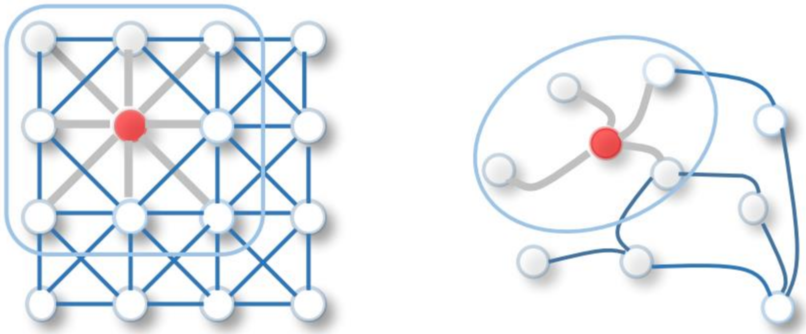
- One popular network: graph convolutional network (GCN)

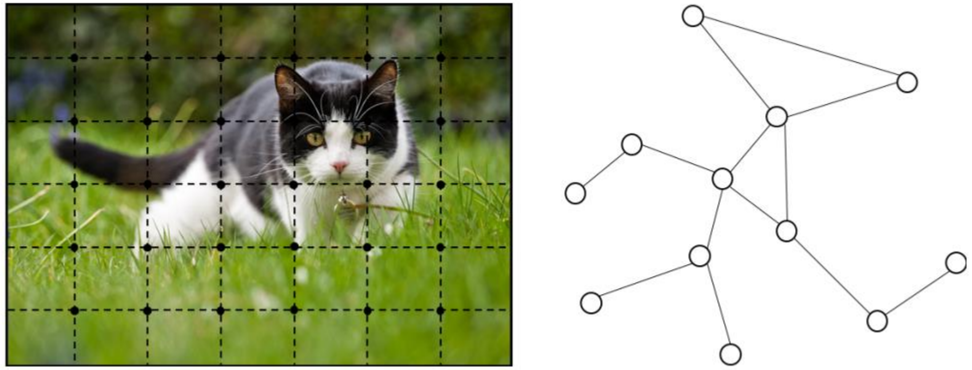
$$f(X, A) = \mathrm{softmax}(A\sigma(AXW^{(0)})W^{(1)})$$

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

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

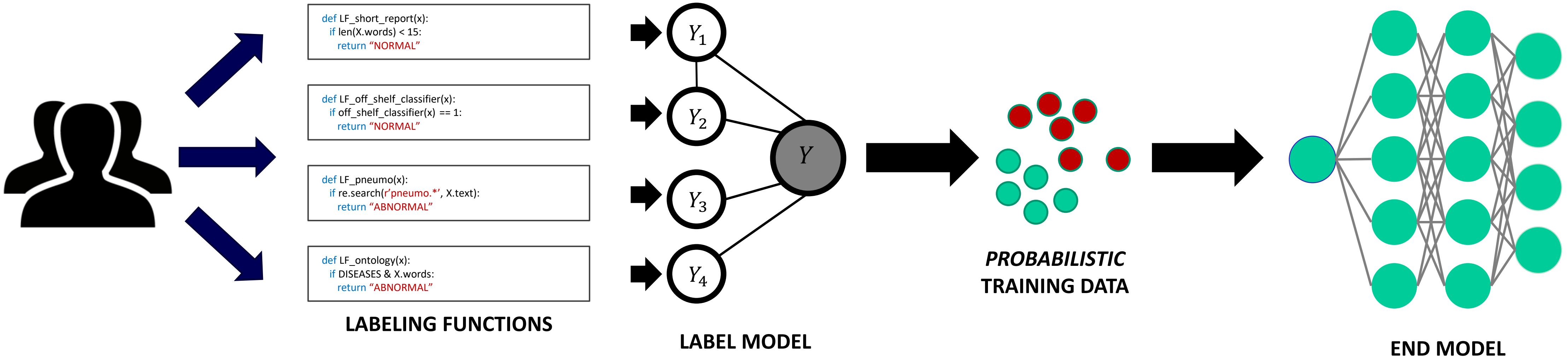
- As before, labels are very expensive to get.
- Sometimes we can get **cheaper sources** to label points
  - Noisy…
  - But can acquire several of them

- 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

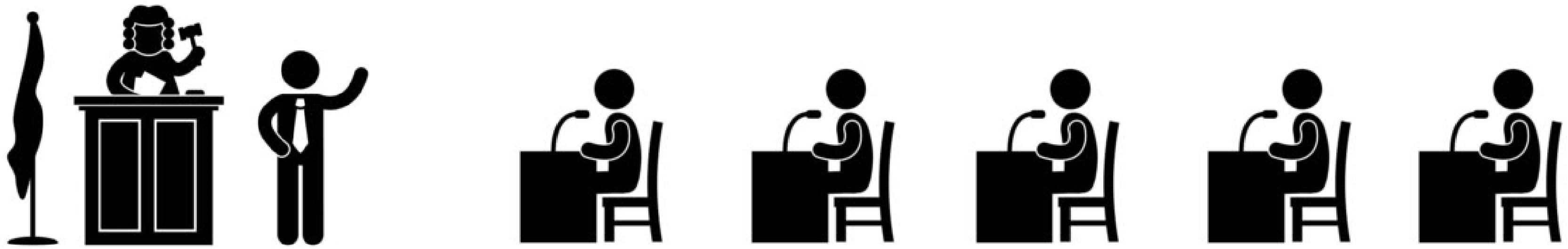
2. We **model** and **combine** these labels

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

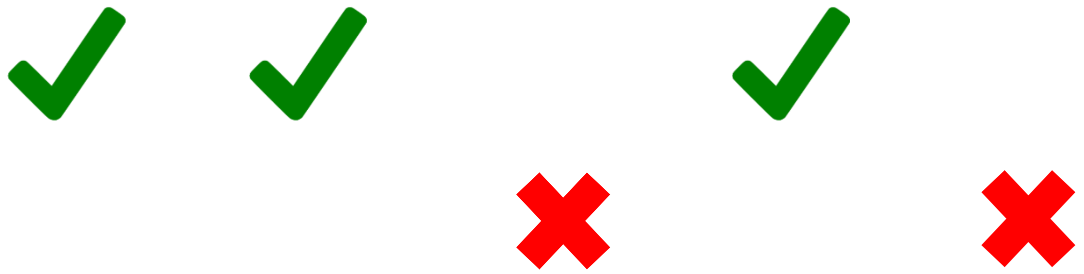
Weak Supervision: Intuition & Majority Vote

- Pretend we're in court:

Witnesses



Votes

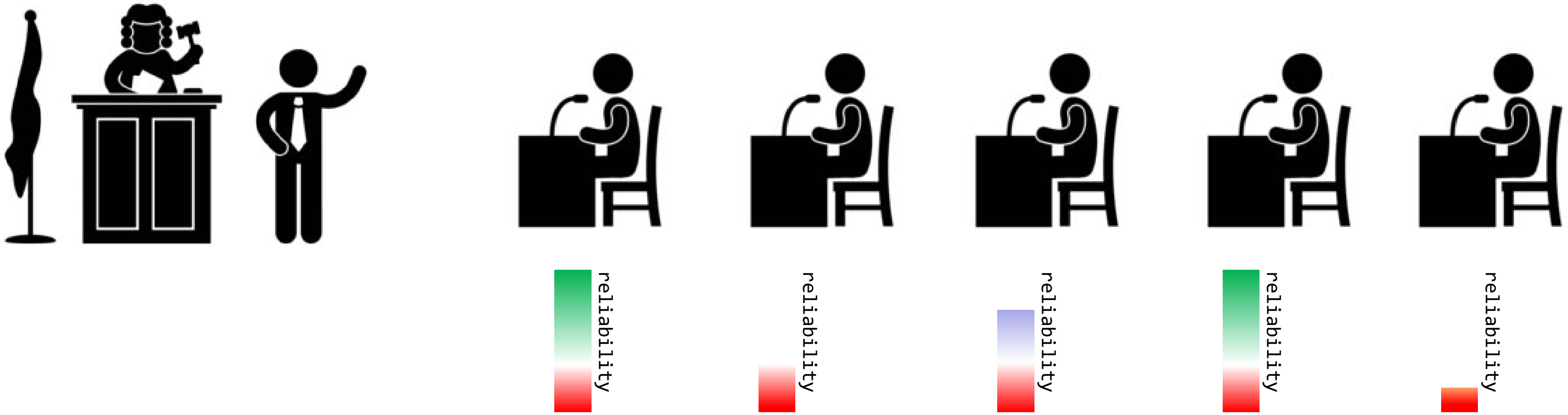


Naïve approach: **majority vote**

Weak Supervision:

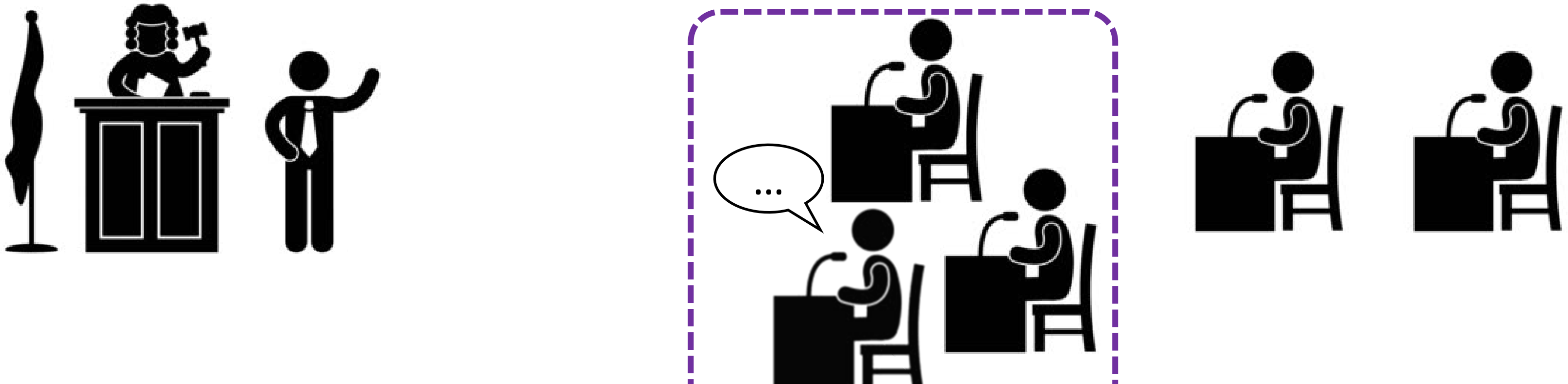
- Can we do better?
  - Some witnesses more reliable, others are voting in a bloc

Witnesses



1. Incorporate **accuracies**

Witnesses



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

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

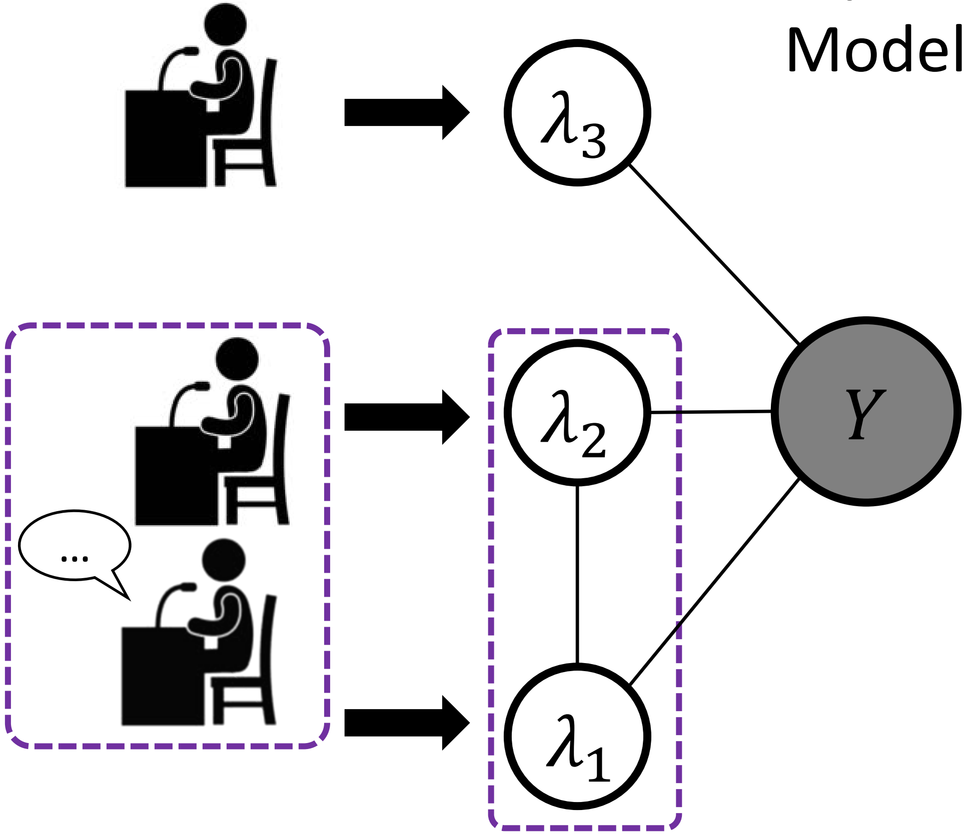
  - **Read**: given a set of votes from the *m* labeling functions, how likely is Y to be 0? To be 1? Etc…

- Q: What do we need to compute this? A: Encode this information into an undirected graphical model: the **Label Model**

Weak Supervision: Label Model Structure

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

Graphical Model



Parameters

1. **Means** $\mathbf{E}[\lambda_i], \mathbf{E}[Y]$ (easy-ish)

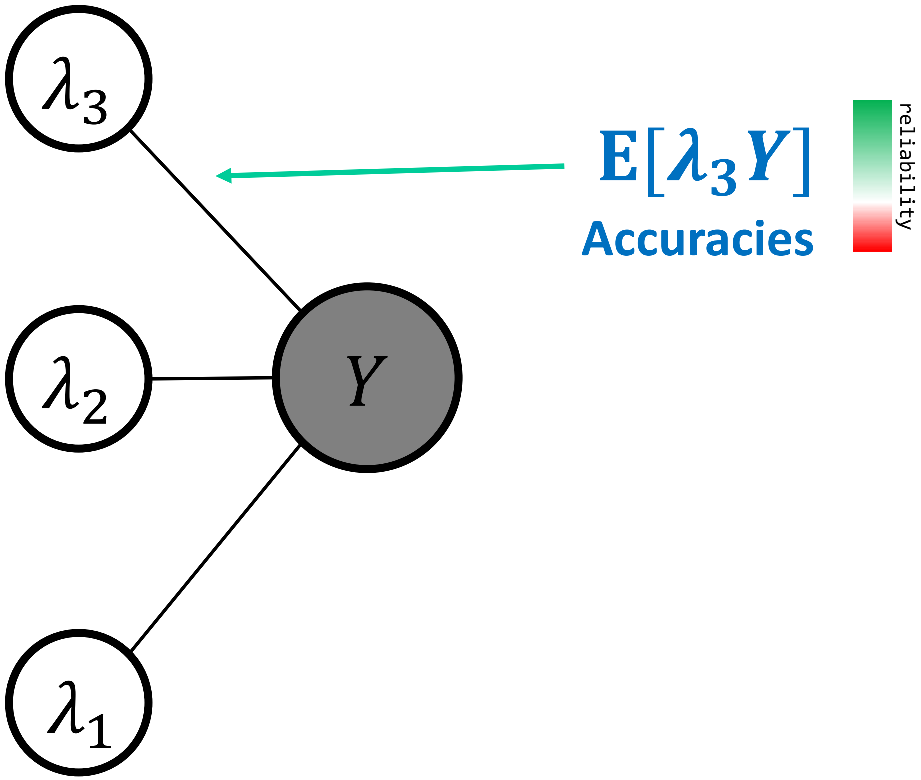
2. **Accuracies** 3. **Correlations**

$$\underbrace{\mathbf{E}[\lambda_i Y] \qquad \mathbf{E}[\lambda_i \lambda_j]}_{\textbf{Covariances!}}$$

Problem: how do we learn parameters, **without observing Y?**

Weak Supervision: Label Model Learning

- Harder than our usual fully supervised graphical model learning… but a simple **independence property** helps
  - Don't see the accuracy parameters, but we know (an estimate of) their product for each pair of labeling functions….



Neat independence property:

$$\mathbf{E}[\lambda_1 Y \lambda_2 Y] = \mathbf{E}[\lambda_1 Y]\, \mathbf{E}[\lambda_2 Y]$$

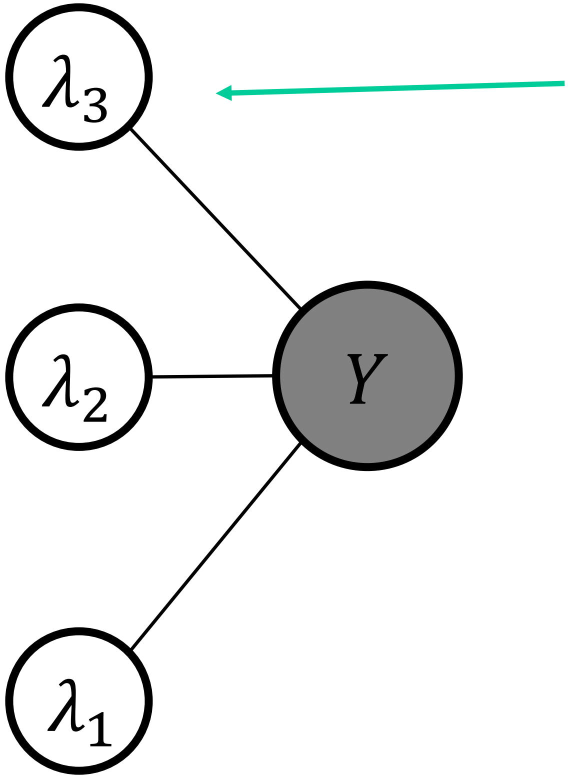
or

$$\mathbf{E}[\lambda_1 \lambda_2] = \mathbf{E}[\lambda_1 Y]\, \mathbf{E}[\lambda_2 Y]$$

Average rate of agreement/disagreement. We can estimate this from samples (we se the labeling functions).

Weak Supervision: Label Model Learning

- Harder than our usual fully supervised graphical model learning... but a simple **independence property** helps



$\mathbf{E}[\boldsymbol{\lambda_3 Y}]$

Accuracies

Let's write three equations:

$$\mathbf{E}[\boldsymbol{\lambda_1\lambda_2}] = \mathbf{E}[\boldsymbol{\lambda_1 Y}]\ \mathbf{E}[\boldsymbol{\lambda_2 Y}]$$
$$\mathbf{E}[\boldsymbol{\lambda_1\lambda_3}] = \mathbf{E}[\boldsymbol{\lambda_1 Y}]\ \mathbf{E}[\boldsymbol{\lambda_3 Y}]$$
$$\mathbf{E}[\boldsymbol{\lambda_2\lambda_3}] = \mathbf{E}[\boldsymbol{\lambda_2 Y}]\ \mathbf{E}[\boldsymbol{\lambda_3 Y}]$$

Three equations, three variables. Let's solve:

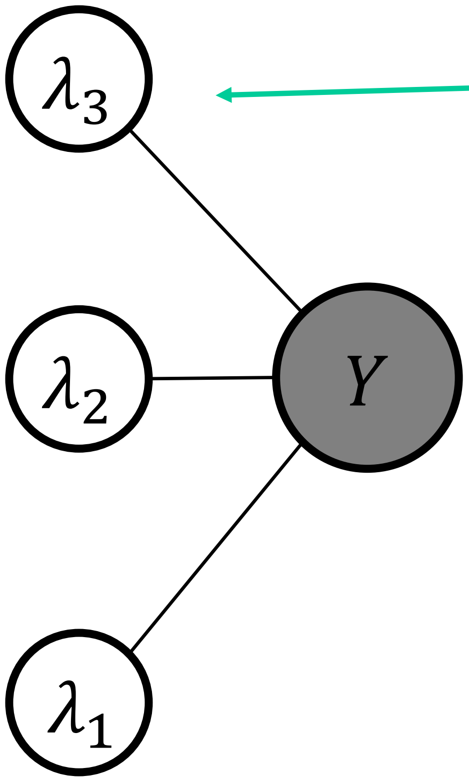
$$\mathbf{E}[\boldsymbol{\lambda_1 Y}] = \sqrt{\mathbf{E}[\boldsymbol{\lambda_1\lambda_2}]\mathbf{E}[\boldsymbol{\lambda_1\lambda_3}]/\mathbf{E}[\boldsymbol{\lambda_2\lambda_3}]}$$

(Multiply first two equations, divide by third)

So: we obtain the **accuracies**! **Correlations** are even easier.

Weak Supervision: Label Model Learning

- Harder than our usual fully supervised graphical model learning… but a simple **independence property** helps



$$\mathbf{E}[\lambda_1 Y] = \sqrt{\mathbf{E}[\lambda_1\lambda_2]\mathbf{E}[\lambda_1\lambda_3]/\mathbf{E}[\lambda_2\lambda_3]}$$

Overall approach:

1. Divide the labeling functions into sets of three (“triplets”)

2. Estimate the agreement/disagreement rate

3. Solve the system for each triplet.

Weak Supervision: Using the Parameters

- The learned accuracies & correlations can be used to compute the conditional probability

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

- Leads to a bunch of **probabilistic** (soft) labels
  - Ex: Y_1 = [0.2 0.8]
  - Can use for training with cross-entropy loss (or others)

Break & Quiz

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

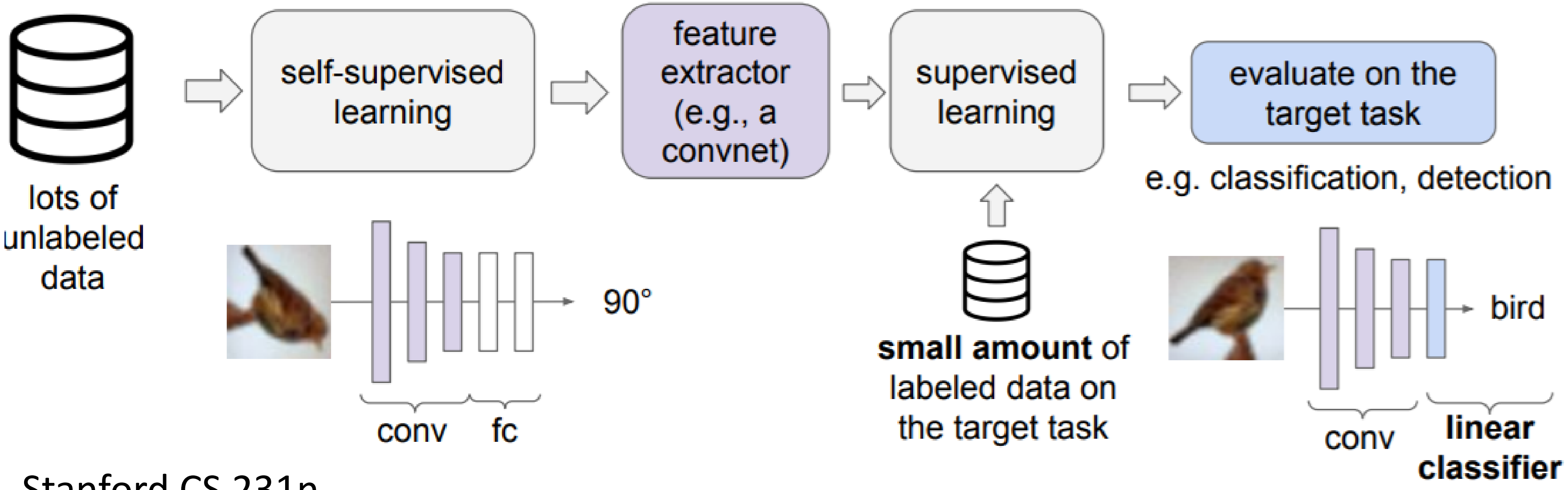
- Suppose we have no labeled data, nor weak sources
- What can we do with unlabeled data?
  - Generative modeling, etc.
  - Could also obtain **representations** (ie new features) for **downstream use.**
- Need to create tasks from unlabeled data: "Pretext tasks"
  - Ex: predict stuff you already know

image completion | rotation prediction | "jigsaw puzzle" | colorization

Stanford CS 231n

Self Supervision: Using the Representations

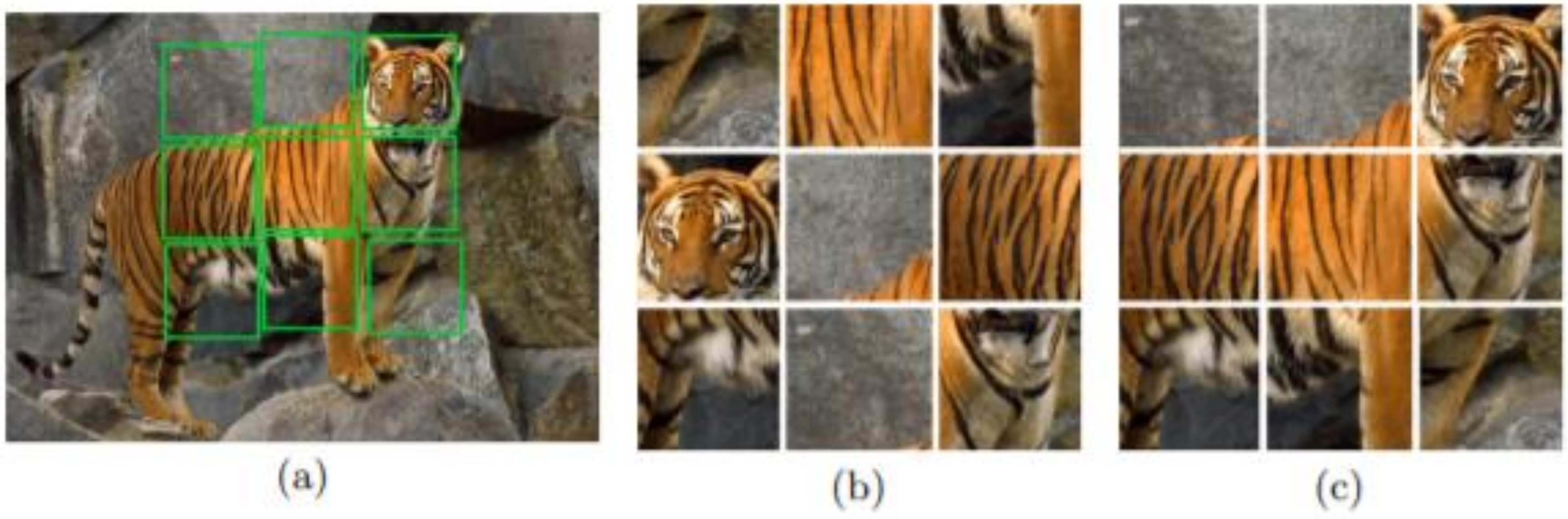
- Don’t care specifically about our performance on pretext task
- Use the learned network as a feature extractor
- Once we have labels for a particular task, train
  - A small amount of data



Stanford CS 231n

Self Supervision: Pretext Tasks

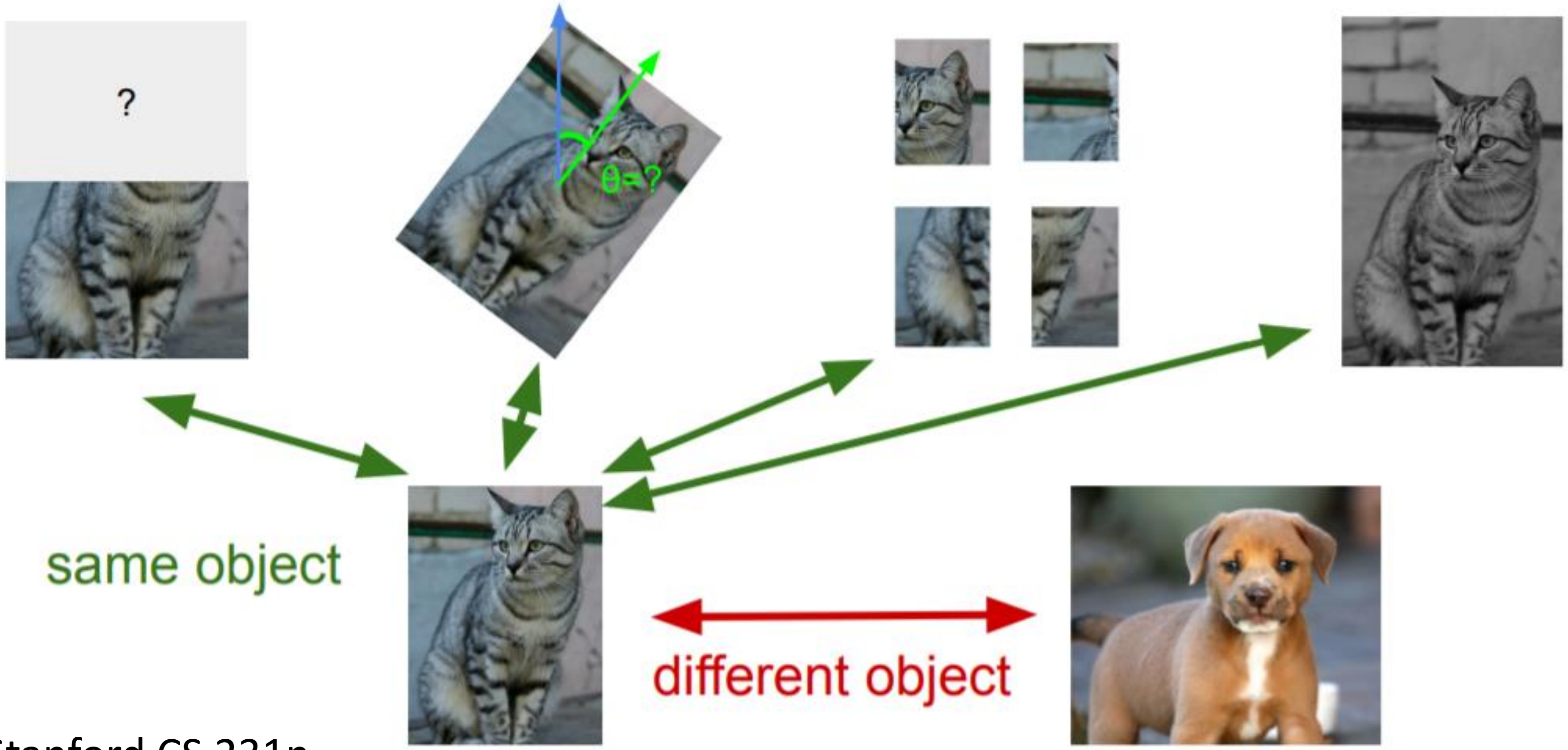
- Lots of options for pretext tasks
  - Predict rotations
  - Coloring
  - Fill in missing portions of the image
  - Solve puzzles:



Noroozi and Favaro

Contrastive Learning: Basics

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



Stanford CS 231n

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^-

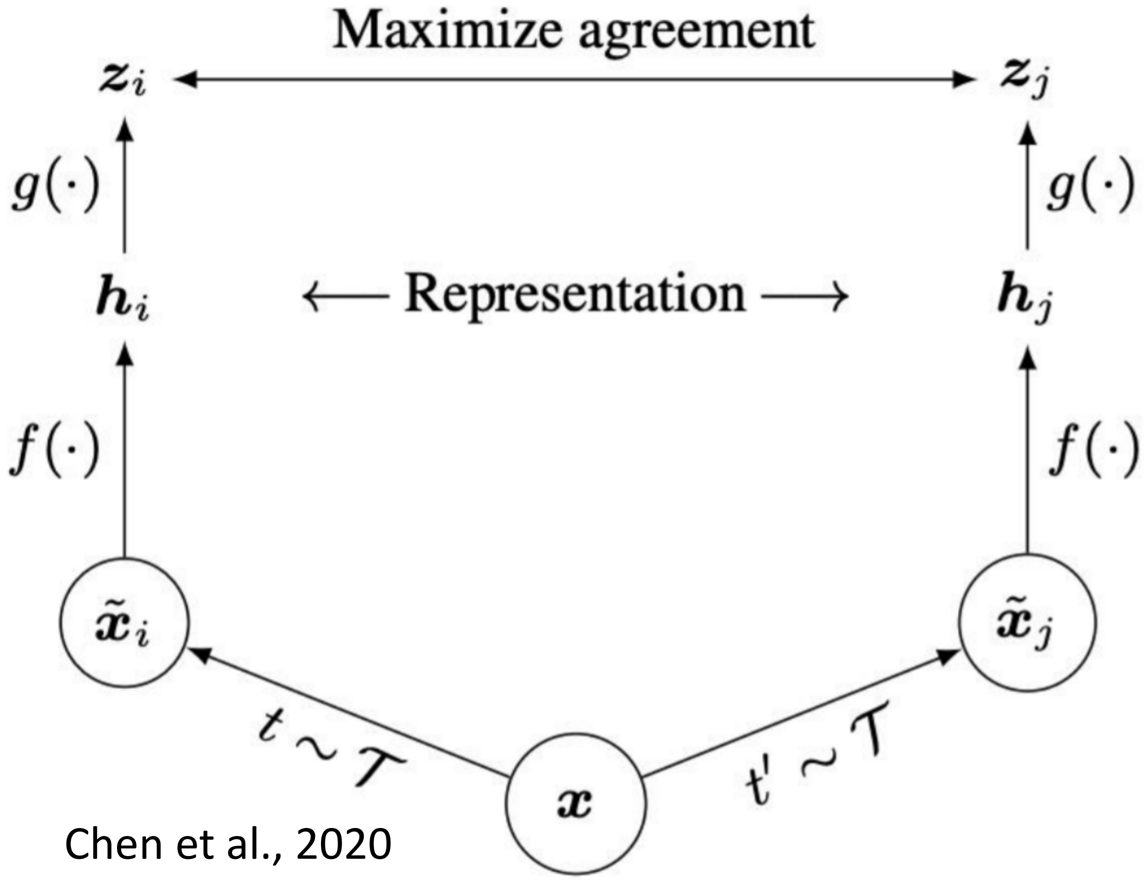
x_3^-

...

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



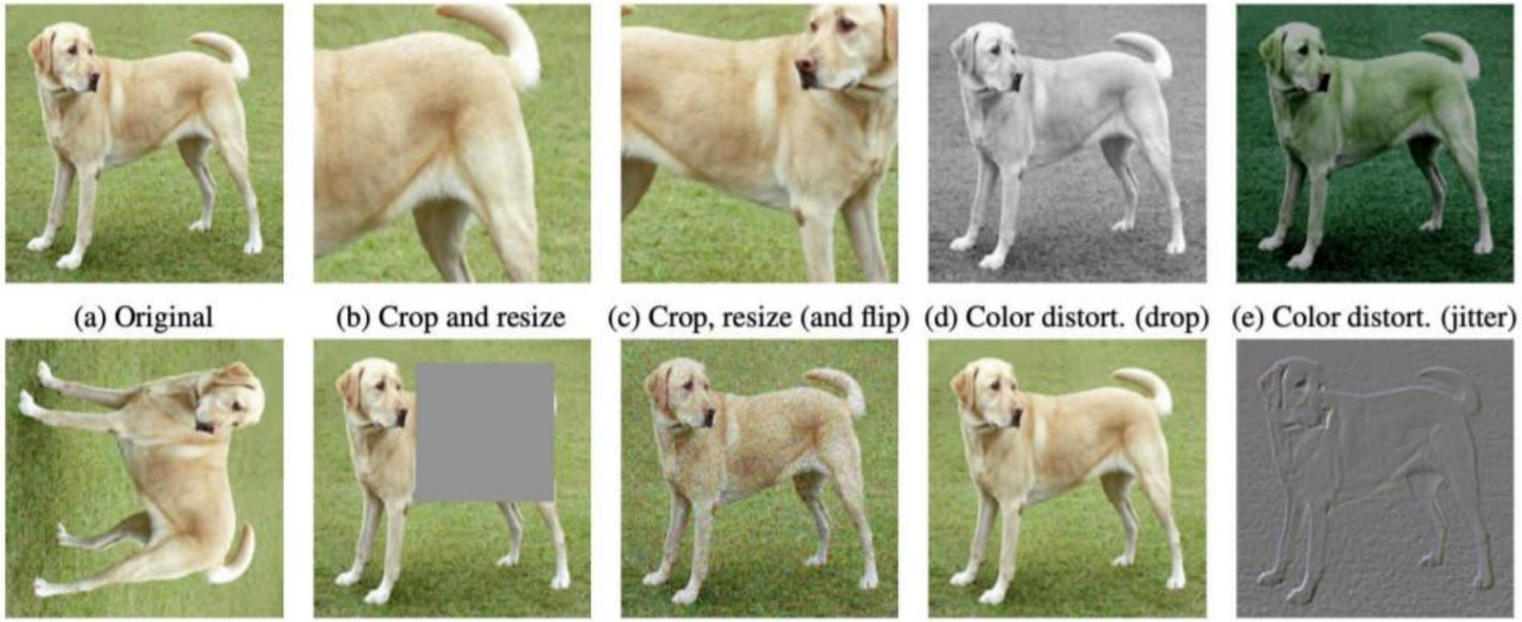
Chen et al., 2020

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



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

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