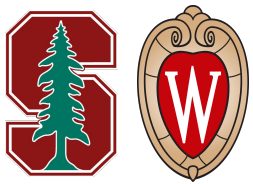
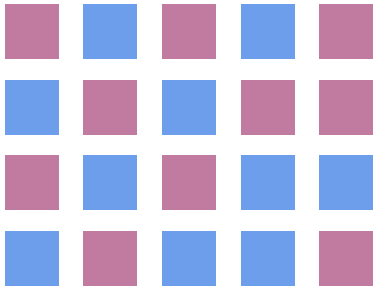
Liger: Fusing weak supervision with foundation model embeddings

Mayee F. Chen*, Daniel Y. Fu*, Dyah Adila, Michael Zhang, Fred Sala, Kayvon Fatahalian, Christopher Ré

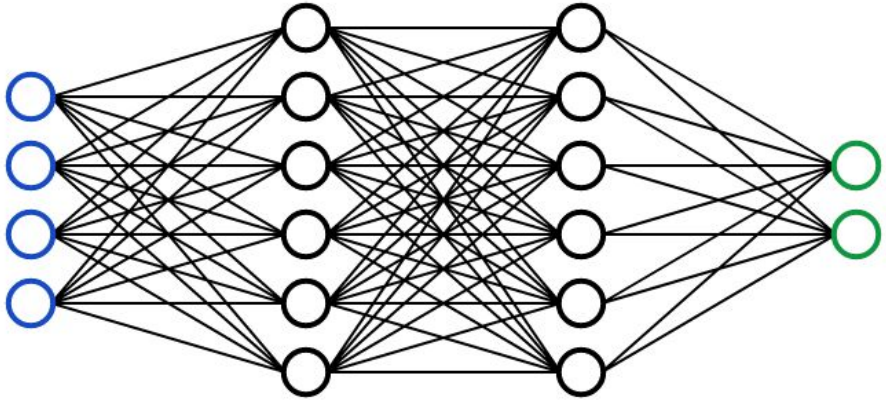
Snorkel AI MLW, April 8th, 2022



Motivation: traditional supervised deep models

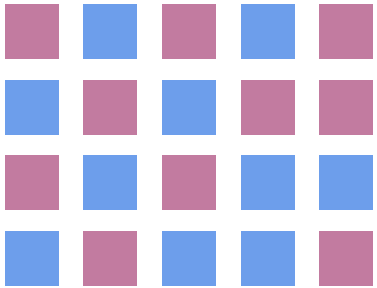


Labeled data ✎

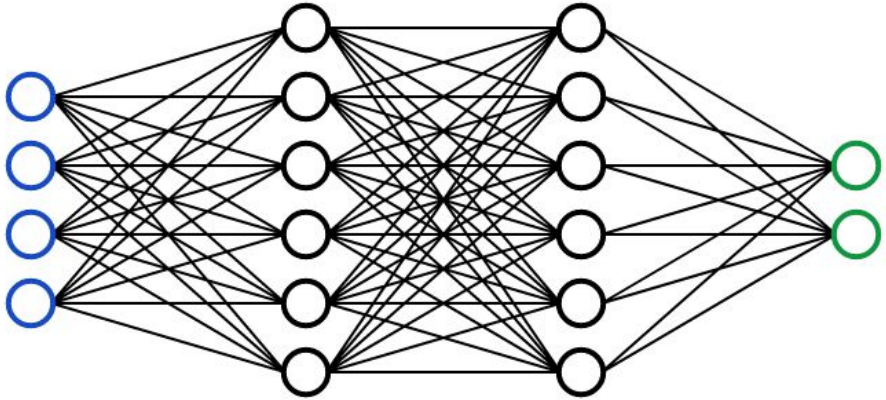


Deep neural network trained from scratch

Motivation: traditional supervised deep models



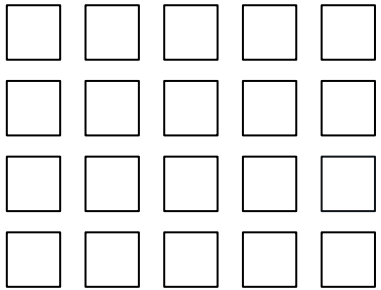
Labeled data ✎



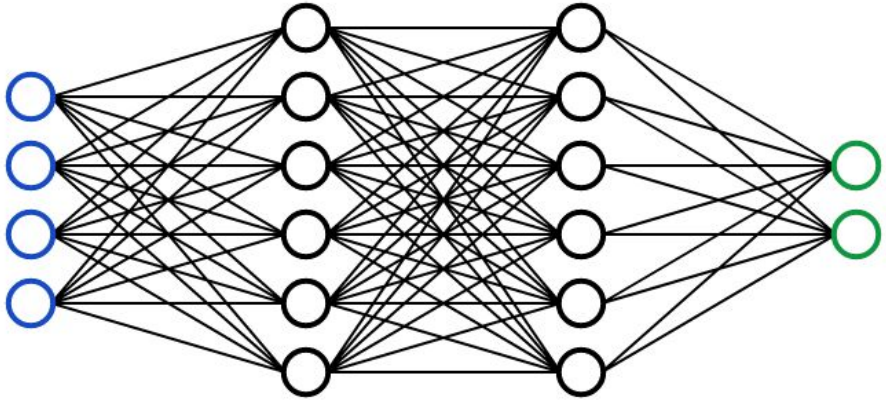
Deep neural network trained from scratch

Both: expensive and time-consuming!

Motivation: traditional supervised deep models

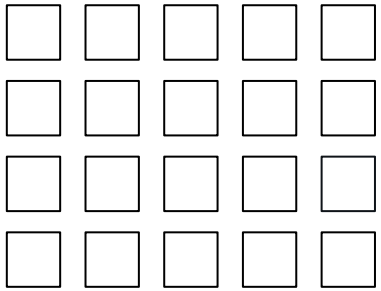


Unlabeled data



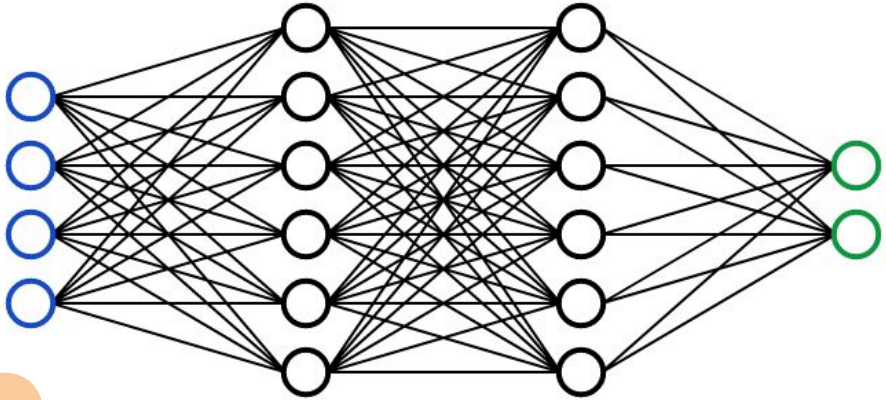
Deep neural network trained from scratch

Motivation: traditional supervised deep models



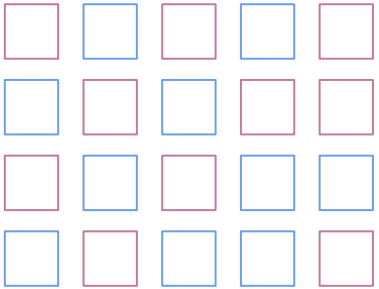
Unlabeled data +

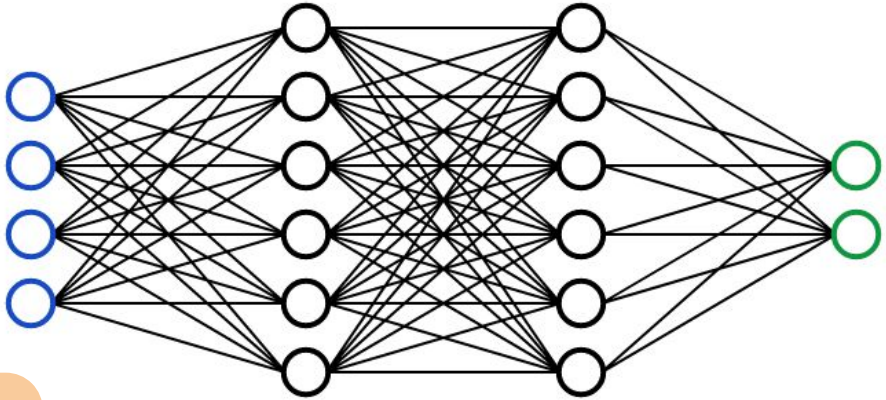
Noisier sources of signal: crowd workers, heuristics, external KBs, etc.



Deep neural network trained from scratch

Motivation: traditional supervised deep models



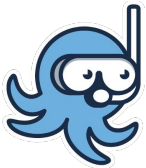


Unlabeled data +

Noisier sources of signal: crowd workers, heuristics, external KBs, etc.

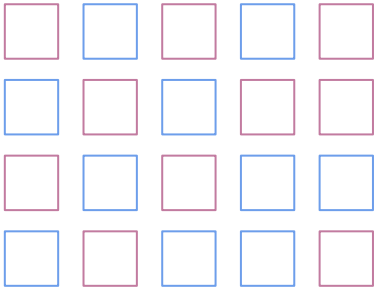
Deep neural network trained from scratch

=



Weak Supervision!

Motivation: traditional supervised deep models

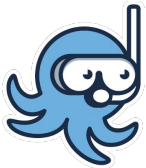


Unlabeled data +

Noisier sources of signal: crowd workers, heuristics, external KBs, etc.

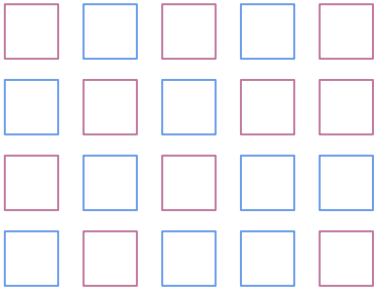
Large, pre-trained models (BERT, GPT-3, DALL-E 2, CLIP)

=



Weak Supervision!

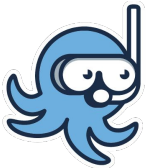
Motivation: traditional supervised deep models



Unlabeled data +

Noisier sources of signal: crowd workers, heuristics, external KBs, etc.

=



Weak Supervision!

Large, pre-trained models (BERT, GPT-3, DALL-E 2, CLIP)

=

Foundation models (FMs)*

*Bommasani et al. "On the opportunities and risks of foundation models." 2021.

Q: How do we combine Weak Supervision and Foundation Models?

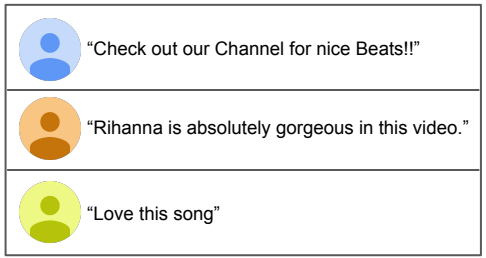
Outline

1. Background
    a. The weak supervision pipeline
    b. Foundation models
2. Technical review of Weak Supervision
3. Liger 🦁🐯
4. Why it works: embedding “smoothness”
5. Results
6. Summary and Future Directions

Background

The Weak Supervision (WS) Pipeline

$x \in \mathcal{X}$

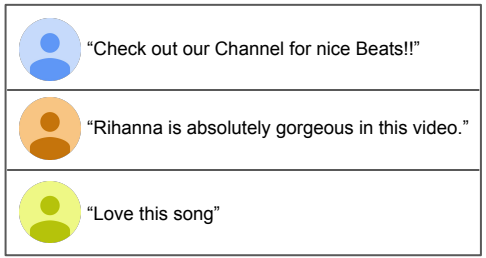


Unlabeled dataset

Unknown labels $y \in \{-1, 1\}$

The Weak Supervision (WS) Pipeline

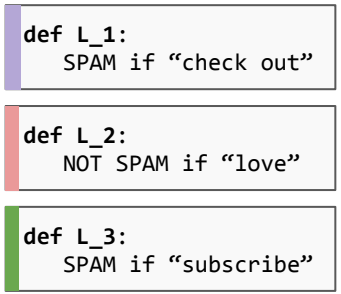
$x \in \mathcal{X}$



Unlabeled dataset

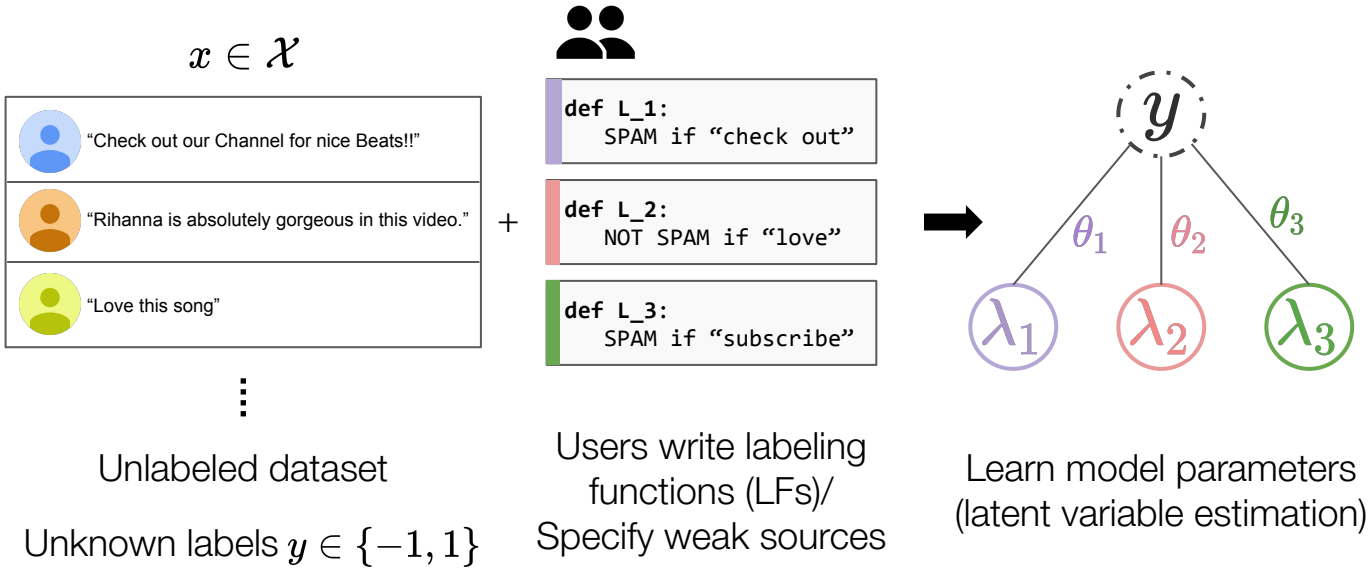
Unknown labels $y \in \{-1, 1\}$

+



Users write labeling functions (LFs)/ Specify weak sources

The Weak Supervision (WS) Pipeline



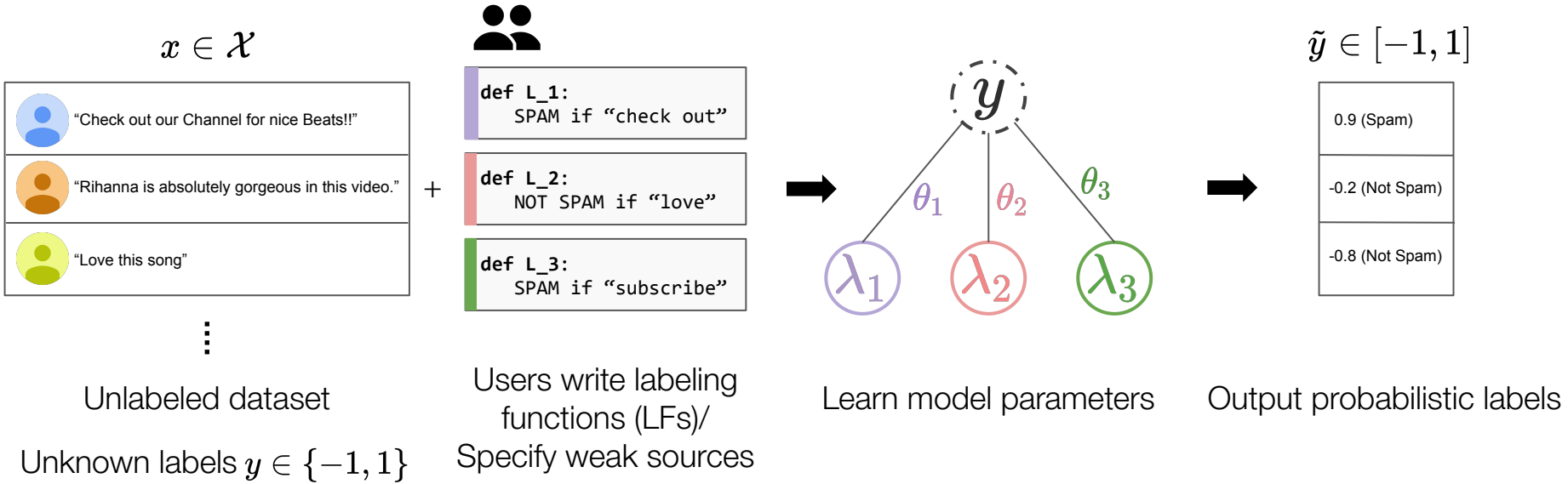
Unlabeled dataset

Unknown labels $y \in \{-1, 1\}$

Users write labeling functions (LFs)/ Specify weak sources

Learn model parameters (latent variable estimation)

The Weak Supervision (WS) Pipeline



Unlabeled dataset

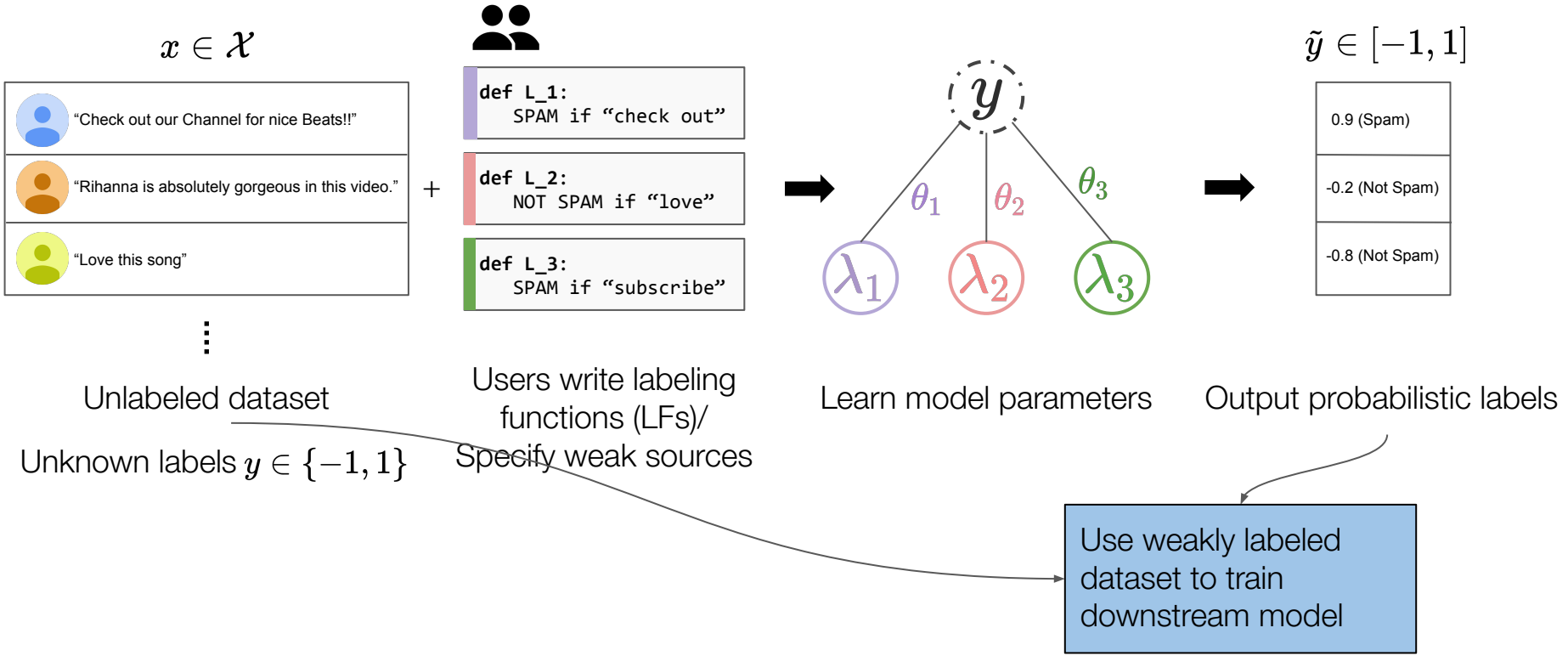
Unknown labels $y \in \{-1, 1\}$

Users write labeling functions (LFs)/ Specify weak sources

Learn model parameters

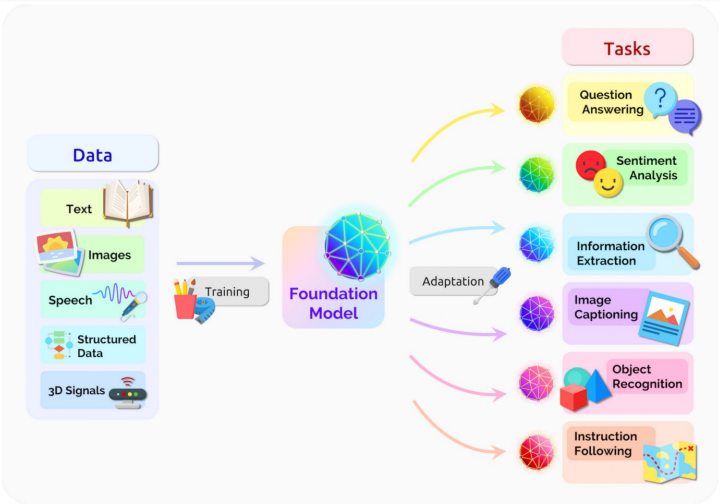
Output probabilistic labels

The Weak Supervision (WS) Pipeline



Foundation Models (FMs)

- Off-the-shelf access, fine-tune the FM on your target task
- Limited interfaces:
  - Cannot access model weights/costly to fine-tune FM
  - Access FM embeddings of data as *f(x)*, with fixed mapping *f*



Simple approaches?

Use weak supervision to make a dataset $(x, \tilde{y})$

Then what?

Simple approaches?

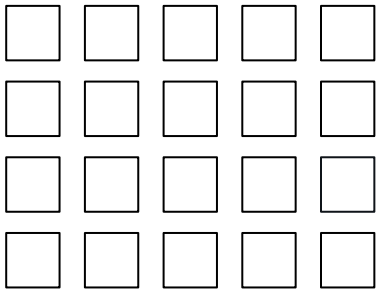
Use weak supervision to make a dataset $(x, \tilde{y})$

Then what?

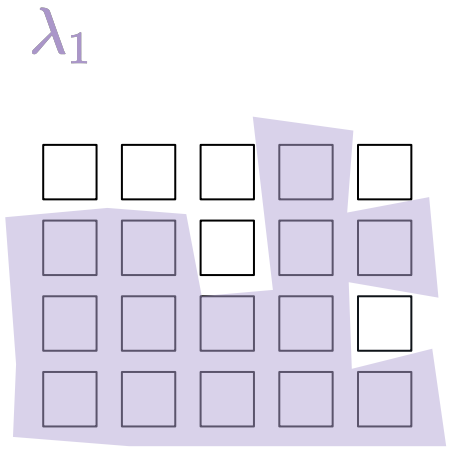
- Fine-tuning? ✗
- KNN with *f(x)*
- Linear probes, adapters (simple MLPs on *f(x)*)

Can we do better than sequential application of weak supervision and FM embeddings?

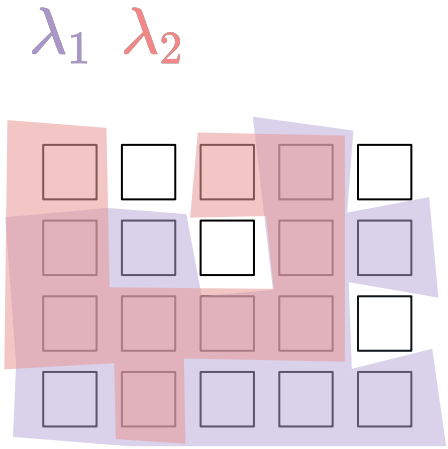
Using FMs to solve problems in Weak Supervision



Using FMs to solve problems in Weak Supervision

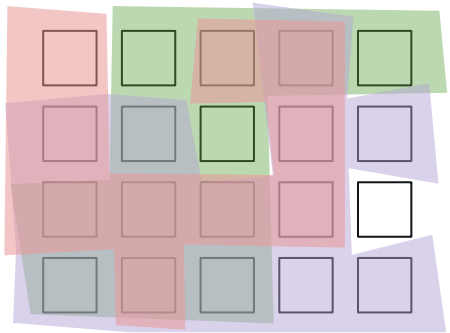


Using FMs to solve problems in Weak Supervision



Using FMs to solve problems in Weak Supervision

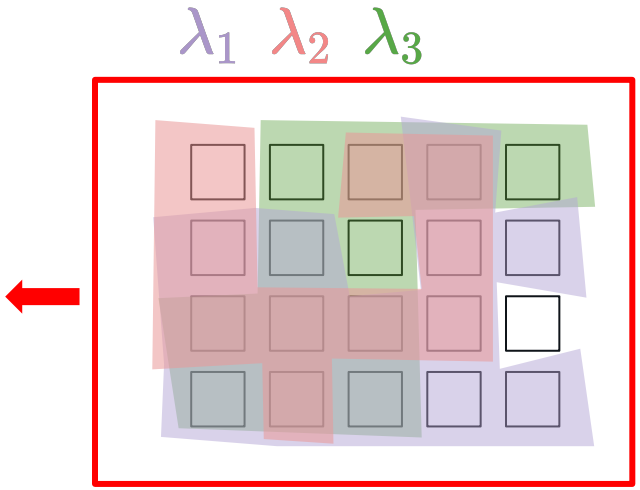
λ_1 λ_2 λ_3



Using FMs to solve problems in Weak Supervision

Coarse accuracies:
learn params θ_1 θ_2 θ_3

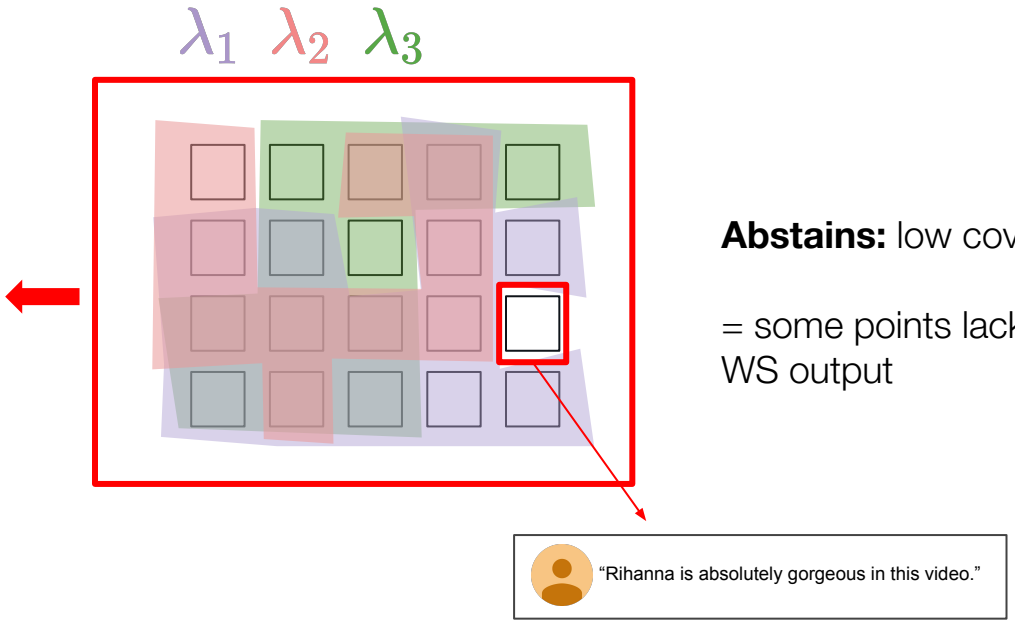
=1 parameter per LF learned over the entire dataset, not precise



Using FMs to solve problems in Weak Supervision

Coarse accuracies:
learn params θ_1 θ_2 θ_3

=1 parameter per LF learned over the entire dataset, not precise



Abstains: low coverage of LFs

= some points lack signal, bad WS output

Weak Supervision 101

Method setup

Input:

- Unlabeled dataset $\mathcal{D} = \{x_i\}_{i=1}^{n}$ with $x \in \mathcal{X}$, unknown label $y \in \{-1, 1\}$
- Weak sources $\lambda_1, \ldots, \lambda_m : \mathcal{X} \to \{-1, 0, 1\}$
- FM embedding mapping $f : \mathcal{X} \to \mathbb{R}^d$

Method setup

Input:

- Unlabeled dataset $\mathcal{D} = \{x_i\}_{i=1}^n$ with $x \in \mathcal{X}$, unknown label $y \in \{-1, 1\}$
- Weak sources $\lambda_1, \ldots, \lambda_m : \mathcal{X} \to \{-1, 0, 1\}$
- FM embedding mapping $f : \mathcal{X} \to \mathbb{R}^d$

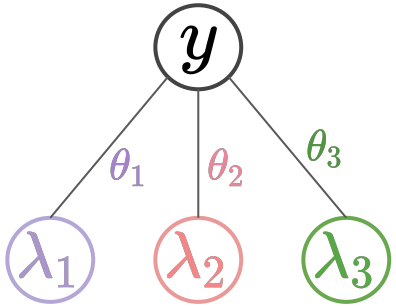
Desired output: $\Pr(y = 1 | \lambda_1, \ldots, \lambda_m, x)$

- Given a datapoint and a list of votes on its label, what is its true label?
- Intuitively: some weighted combination of votes

Standard WS Algorithm

1. Learn relationship between y and $\lambda_1, \ldots, \lambda_m$ via *graphical model*

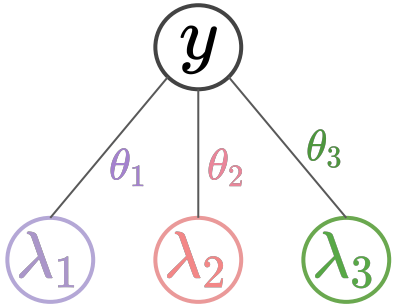
$$\Pr_\theta(y, \lambda_1, \ldots, \lambda_m) \iff$$



Standard WS Algorithm

1. Learn relationship between y and $\lambda_1, \ldots, \lambda_m$ via *graphical model*

$$\Pr_\theta(y, \lambda_1, \ldots, \lambda_m) \iff$$



- Learn accuracy parameters $\theta_i = \mathbb{E}[\lambda_i y]$
- Under the hood: algorithms for *latent variable estimation** by computing covariances between LFs, e.g. estimating $\mathbb{E}[\lambda_i \lambda_j]$ on $\mathcal{D}$ or via maximum likelihood estimation**

*Fu et. al. “Fast and three-rious: Speeding up weak supervision with triplet methods.” ICML 2020.
*Ratner et. al. “Training Complex Models with Multi-Task Weak Supervision.” AAAI 2019.
**Ratner et. al. “Data Programming: Creating Large Training Sets, Quickly”. NeurIPS 2016.

Standard WS Algorithm

2. Inference

- Given *x,* output estimate of $\mathbf{Pr}_\theta(y = 1|\lambda_1, \ldots, \lambda_m)$
- Under the hood: Bayes rule + weighing each λ_i by a function of θ_i

Standard WS Algorithm

2. Inference

No *x*!

- Given *x*, output estimate of $\mathbf{Pr}_\theta(y = 1|\lambda_1, \ldots, \lambda_m)$
- Under the hood: Bayes rule + weighing each λ_i by a function of θ_i

What's wrong with this?

- We don't use *x* directly in inference, only its votes and $\theta_1, \ldots, \theta_m$
  → coarse accuracies

Standard WS Algorithm

2. Inference

No *x*!

- Given *x*, output estimate of $\mathbf{Pr}_\theta(y = 1|\lambda_1, \ldots, \lambda_m)$
- Under the hood: Bayes rule + weighing each λ_i by a function of θ_i

What's wrong with this?

- We don't use *x* directly in inference, only its votes and $\theta_1, \ldots, \theta_m$
  → coarse accuracies

- When $\lambda_i = 0$ on *x*, the algorithm discards it and computes $\mathbf{Pr}_\theta(y = 1|\lambda_i, \ldots, \lambda_{i-1}, \lambda_{i+1}, \lambda_m)$
  → abstains result in worse probabilistic labels

Method

Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

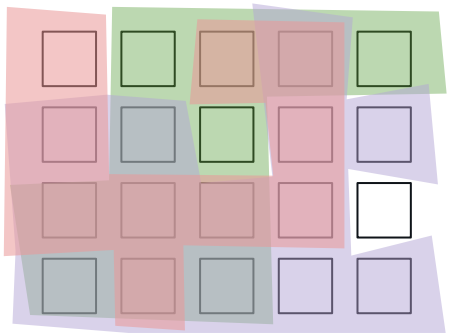
1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$

Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$

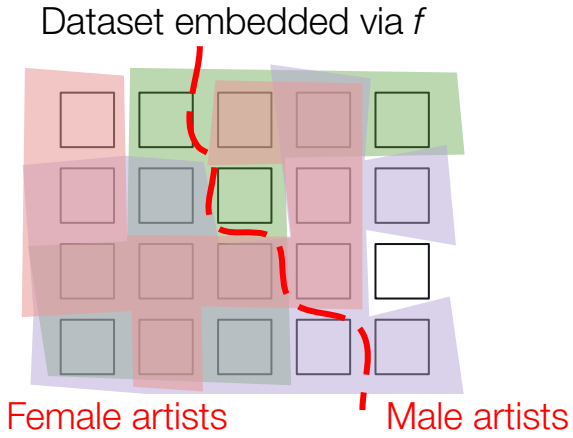
Dataset embedded via *f*



Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

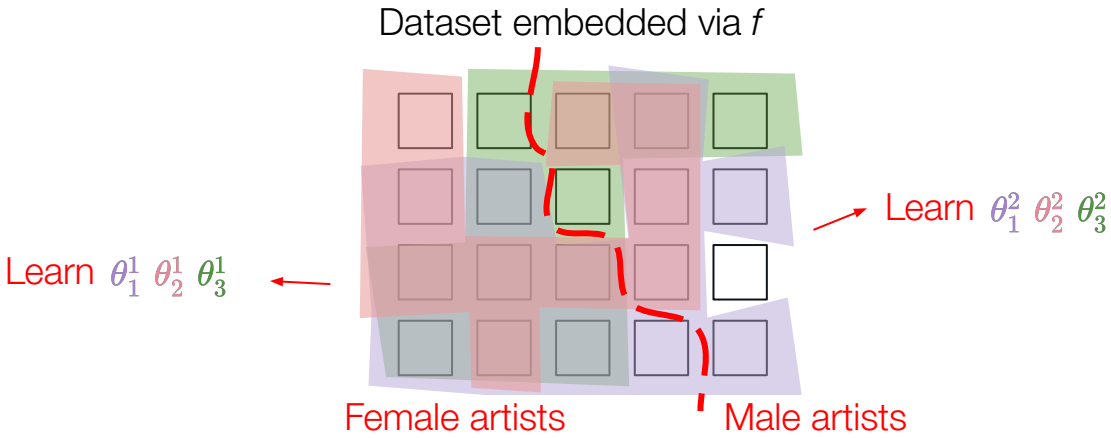
1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$



Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$



Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

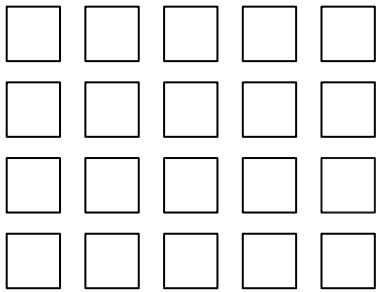
1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m \rightarrow$ fewer abstains

Dataset embedded via *f*

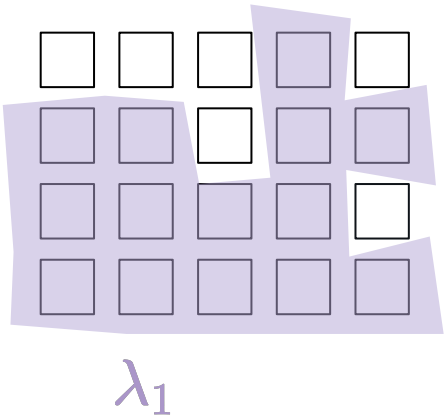


Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\mathrm{Pr}_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

Dataset embedded via *f*

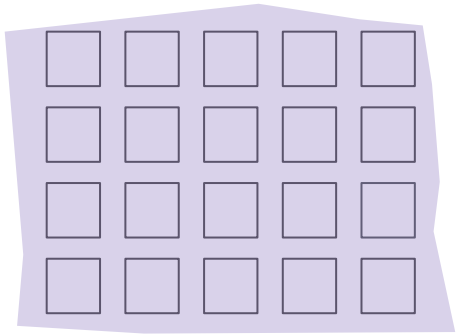


Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

Dataset embedded via *f*



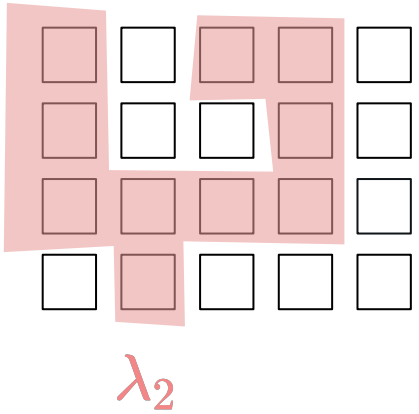
$\lambda_1 \rightarrow \bar{\lambda}_1$

Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m \rightarrow$ fewer abstains

Dataset embedded via *f*

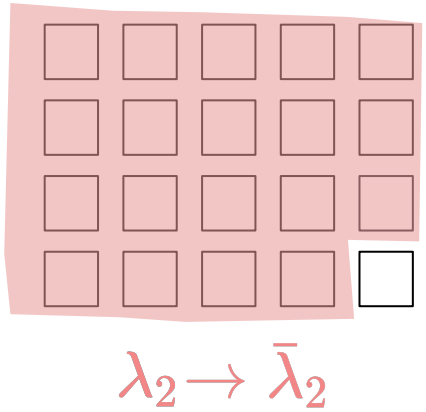


Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m \rightarrow$ fewer abstains

Dataset embedded via *f*

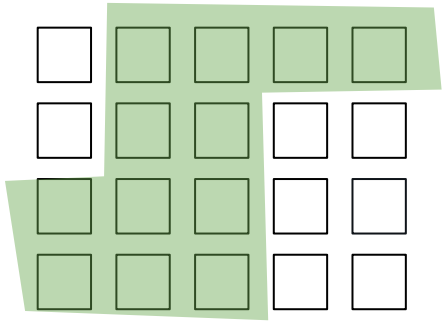


Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

Dataset embedded via *f*



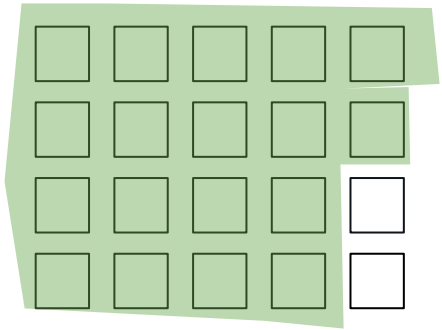
λ_3

Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m \rightarrow$ fewer abstains

Dataset embedded via *f*



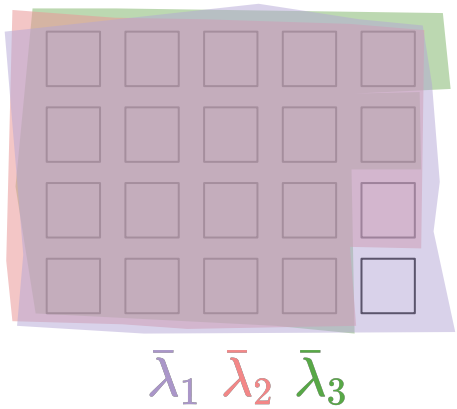
$\lambda_3 \rightarrow \bar{\lambda}_3$

Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

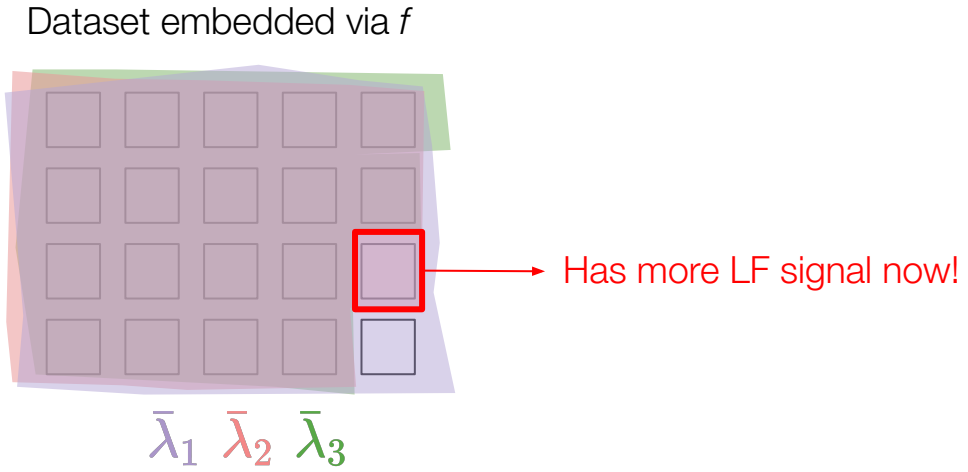
Dataset embedded via *f*



Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

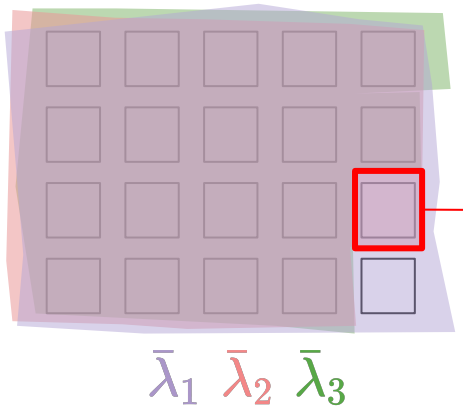


Our method, Liger 🦁🐯

Two simple modifications using FM embeddings *f*:

1. **Partition** the FM embedding space and estimate a set of parameters per part → finer grained accuracies, better approximation of $\Pr_\theta(y = 1|\lambda_1, \ldots, \lambda_m, x)$
2. **Extend** votes of LFs to points that are close by in FM embedding space to construct $\bar{\lambda}_1, \ldots \bar{\lambda}_m$→ fewer abstains

Dataset embedded via *f*

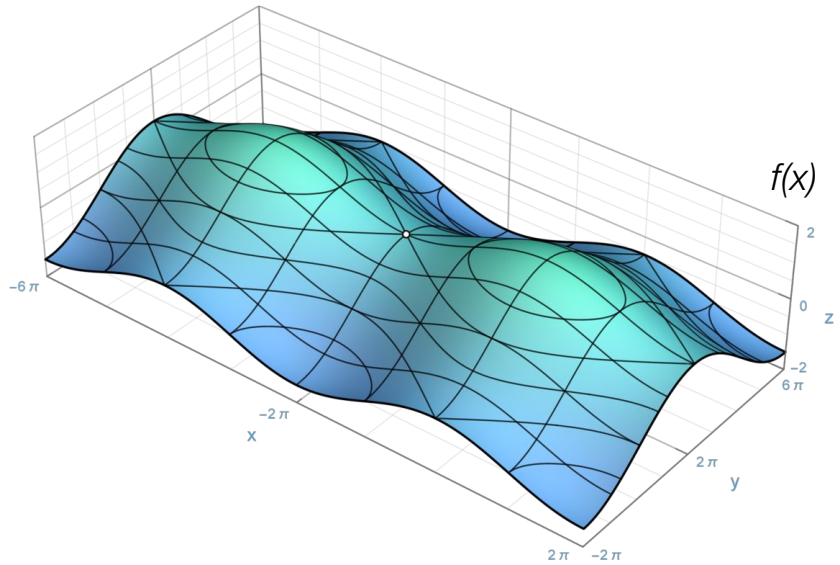


Altogether: estimate $\Pr(y = 1|\bar{\lambda}_1, \ldots, \bar{\lambda}_m, x)$ using partitioned dataset

Why does this work?

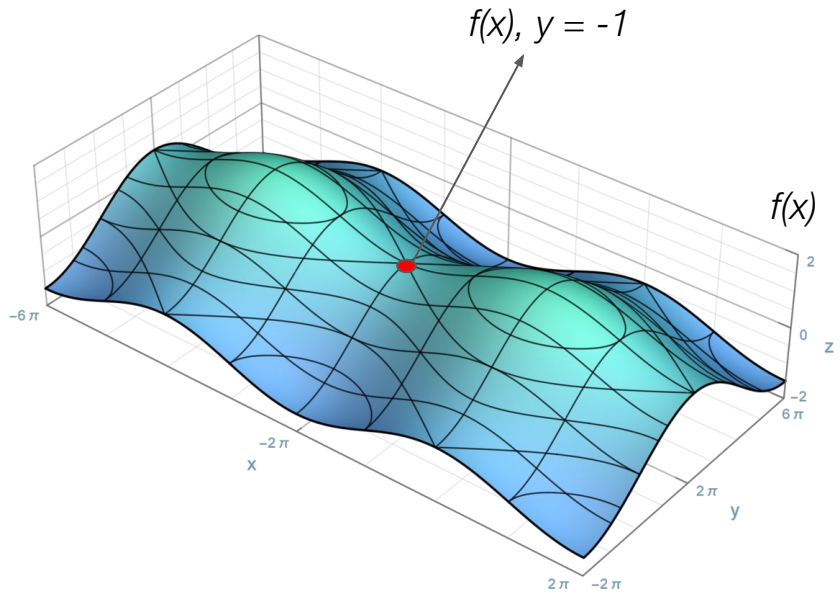
Theory: why does this work?

Liger relies on local *smoothness* of labels in the FM embedding space.



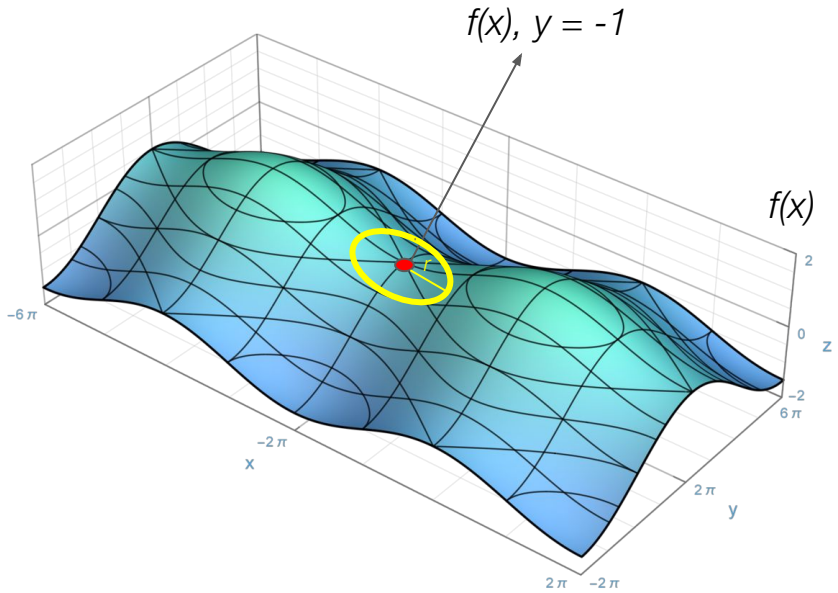
Theory: why does this work?

Liger relies on local *smoothness* of labels in the FM embedding space.



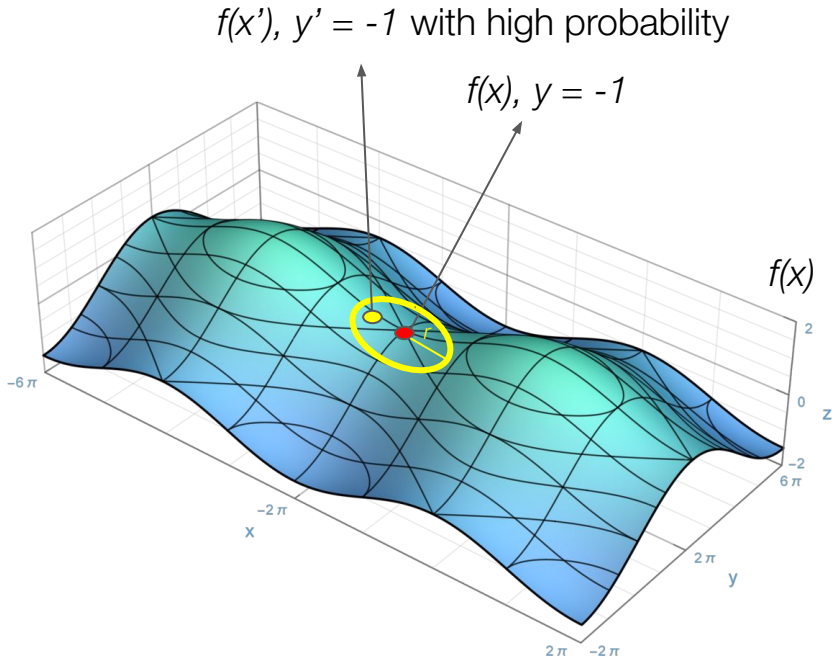
Theory: why does this work?

Liger relies on local *smoothness* of labels in the FM embedding space.



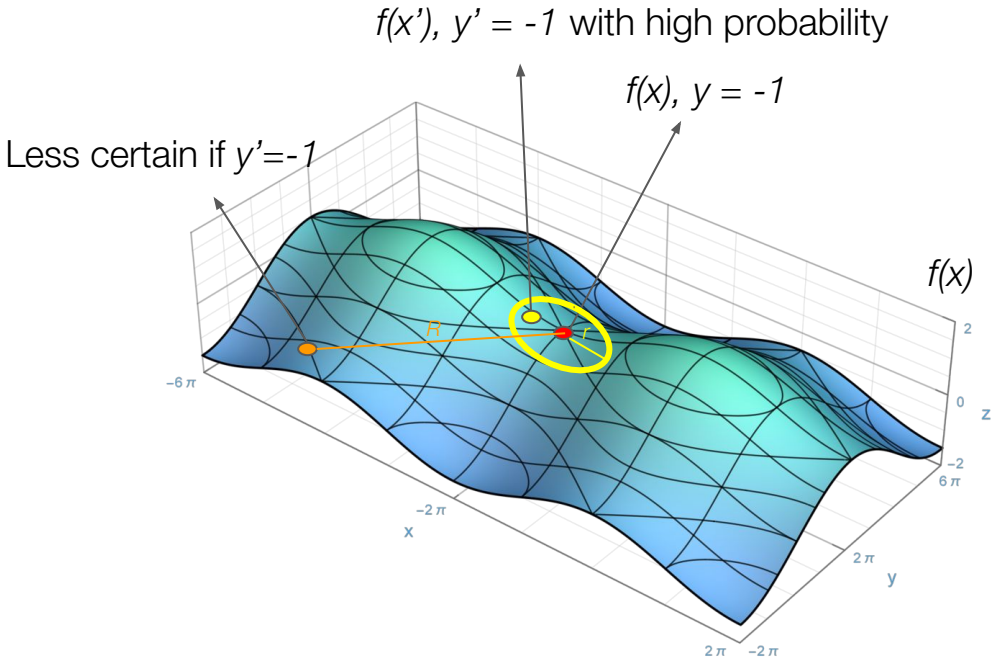
Theory: why does this work?

Liger relies on local *smoothness* of labels in the FM embedding space.



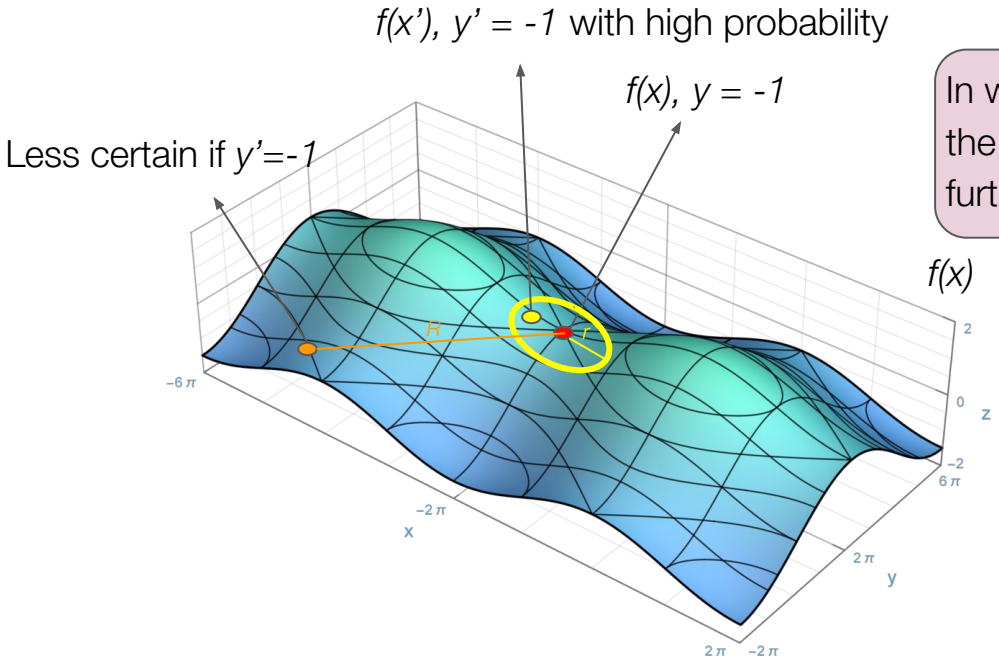
Theory: why does this work?

Liger relies on local *smoothness* of labels in the FM embedding space.



Theory: why does this work?

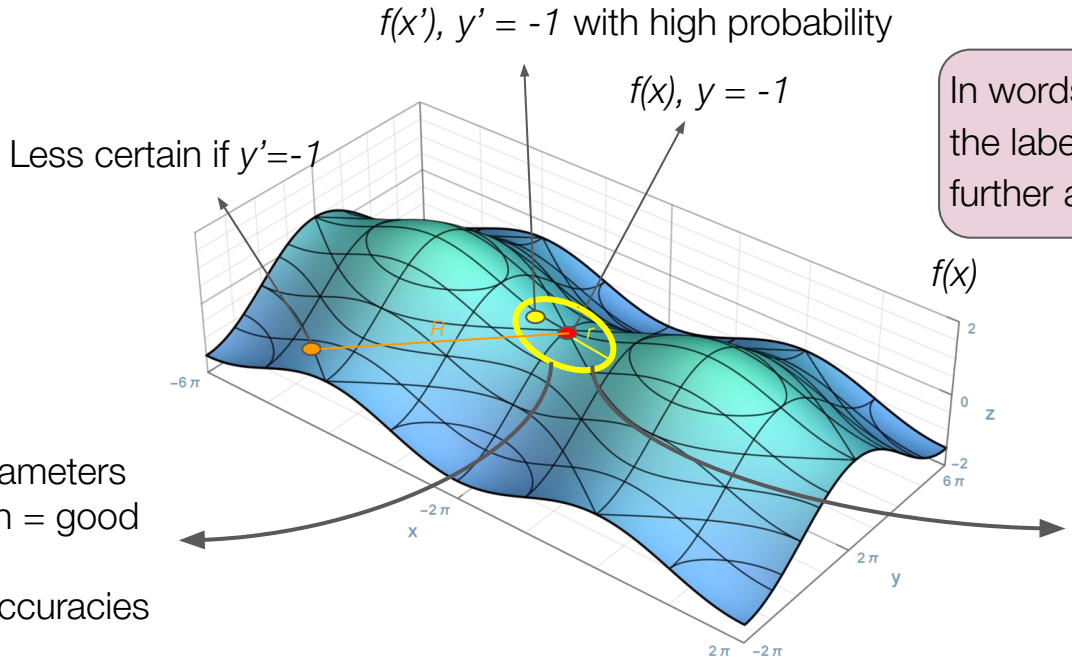
Liger relies on local *smoothness* of labels in the FM embedding space.



In words: smoothness = how unlikely the label changes as you move further away from a point

Theory: why does this work?

Liger relies on local *smoothness* of labels in the FM embedding space.



In words: smoothness = how unlikely the label changes as you move further away from a point

Estimating parameters over this region = good estimates of finer-grained accuracies

Extending LF votes in this region = LF vote likely to be correct

Theory: why does this work?

-  Partitioning into too many sets = good local estimates, high variance
-  Extending too far in embedding space = LF votes become incorrect because true label changes value
- Need to control these depending on how smooth FM is!

Theory: why does this work?

-  Partitioning into too many sets = good local estimates, high variance
-  Extending too far in embedding space = LF votes become incorrect because true label changes value
- Need to control these depending on how smooth FM is!

Theoretical results: improvement over standard WS depends on FM smoothness, amount of data, and above choices.

Theoretical results

Thm 1 (informal). Suppose we partition data into *s* sets, and *d* is the average diameter of a set in embedding space. *K* is a smoothness constant (lower *K* = more smooth). Then, Liger's error (no extensions) is:

$$Error \leq K \cdot d + \mathcal{O}\left(\frac{ms}{n}\right) + H(y|\lambda_1, \ldots \lambda_m, x)$$

Bias — Variance — Irreducible Error (conditional entropy)

Theoretical results

Thm 1 (informal). Suppose we partition data into *s* sets, and *d* is the average diameter of a set in embedding space. *K* is a smoothness constant (lower *K* = more smooth). Then, Liger's error (no extensions) is:

$$Error \leq K \cdot d + \mathcal{O}\left(\frac{ms}{n}\right) + H(y|\lambda_1, \ldots \lambda_m, x)$$

Bias — Variance — Irreducible Error (conditional entropy)

- Bias-variance tradeoff in size/number of partitions, depending on smoothness
- Irreducible error: amount of randomness in *y* after observing *x* and LF votes

Theoretical Results

Next, what does extending and using $\bar{\lambda}$ do?

- Increases bias (larger diameter due to extending)
- Decreases variance (more coverage = more points to estimate on)
- Irreducible error: $H(y|\lambda, x) \rightarrow H(y|\bar{\lambda}, x)$ unclear!

Theoretical Results

Thm 2 (informal). Suppose we extend one LF λ_i by *r.* Define p_i as the proportion of the region where λ_i is extended, and $p(\lambda_{-i})$ as the model accuracy when using all LFs but λ_i. Let $M(r)$ be a function describing smoothness (lower = smoother). Then,

$$H(y|\lambda, x) - H(y|\bar{\lambda}, x) \geq 2p_i(1 - p(\lambda_{-i}))^2(\theta_i - M(r))^2$$

Theoretical Results

Thm 2 (informal). Suppose we extend one LF λ_i by *r.* Define p_i as the proportion of the region where λ_i is extended, and $p(\lambda_{-i})$ as the model accuracy when using all LFs but λ_i. Let $M(r)$ be a function describing smoothness (lower = smoother). Then,

$$H(y|\lambda, x) - H(y|\bar{\lambda}, x) \geq 2p_i(1 - p(\lambda_{-i}))^2(\theta_i - M(r))^2$$

- Tradeoff in how much we extend (p_i vs $M(r)$), depending on smoothness
- Improvement when $\bar{\lambda}_i$ has better-than-random accuracy
- Improvement is less when other LFs already are highly accurate

Results

Empirical Results

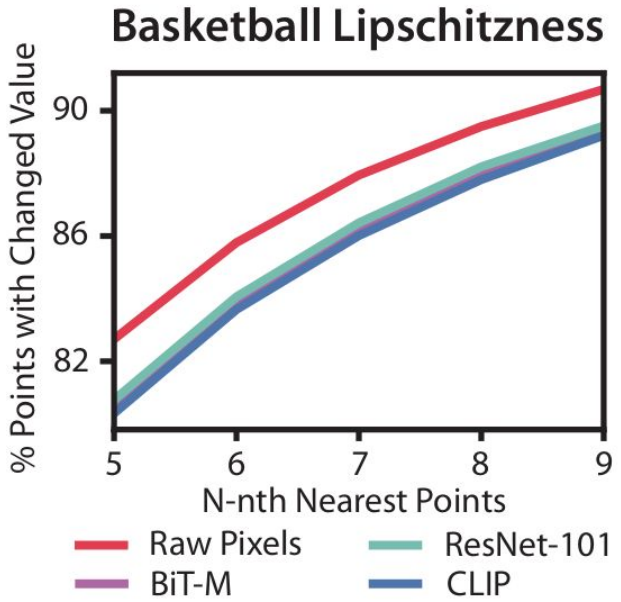
Weak Supervision Datasets + GPT-3 embeddings (for text), CLIP embeddings (for video)

		Weak Sources Only				
	Task	**WS-kNN**	**WS-Adapter**	**WS-LM**	**LIGER**	**ΔCoverage**
NLP	**Spam**	72.8	92.3	83.6	**95.0**	+45.5
	Weather	62.0	86.0	78.0	**98.0**	+90.2
	Spouse	16.9	17.1	47.0	**52.2**	+12.1
Video	**Basketball**	33.3	48.9	27.9	**69.6**	+8.3
	Commercial	84.7	92.8	88.4	**93.5**	+18.8
	Tennis	83.0	**83.8**	82.0	83.3	+32.5

A closer look at smoothness

Embedding	F1-score
Raw pixel	19.3
RN-101	31.1
BiT-M	42.5
CLIP	**69.6**

Smoother

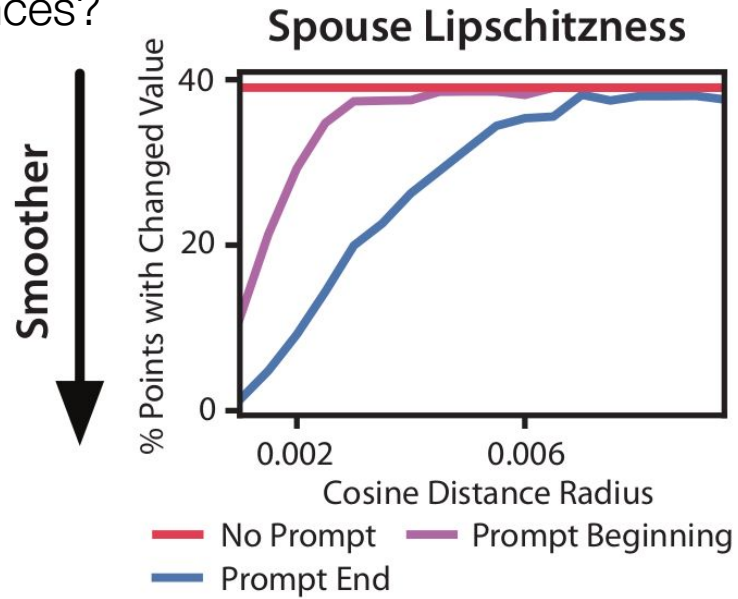


Matches theory that more smooth FM embeddings = better performance

A closer look at smoothness

What's the best way to embed sentences?

Prompting	F1-score
No Prompt	48.5
Prompt Beginning	50.2
Prompt End	**52.2**



Matches theory that more smooth FM embeddings = better performance

Summary

- Unlabeled data + alternative forms of knowledge lying around: weak sources, foundation models
- Liger smartly combines weak supervision with foundation models with two simple steps that exploit FM embedding *smoothness*
  - **Partition:** Estimate finer-grained accuracy parameters
  - **Extend:** Improve coverage of labeling functions
- Improves over standard WS and simple baselines based on FM smoothness

Future Directions

View FMs as blackbox sources of information with easy (partial) access.

How do we understand and best utilize them in inexpensive ways?

- Simple training methods on their embeddings (e.g. adapters)
- How to choose a good FM for your task?
- Can we combine multiple FMs themselves a la Weak Supervision?

Thank you!

Contact: mfchen@stanford.edu, danfu@cs.stanford.edu

Shoring Up the Foundations: Fusing Model Embeddings and Weak Supervision
https://arxiv.org/abs/2203.13270