Dependency Structure Misspecification in Multi-Source Weak Supervision Models

Salva Rühling Cachay[1,2], Benedikt Boecking[2], Artur Dubrawski[2]

[1]Auton Lab, Carnegie Mellon University
[2]Technical University of Darmstadt

Carnegie Mellon University

Auton Lab

Successful Machine Learning methods require large amounts of labeled data

https://medium.com/syncedreview/sensetime-trains-imagenet-alexnet-in-record-1-5-minutes-e944ab049b2c

Hand labeling, however, is expensive both in terms of time and cost

https://medium.com/syncedreview/sensetime-trains-imagenet-alexnet-in-record-1-5-minutes-e944ab049b2c

Alternative:
Weak supervision and, more specifically, Data Programming [1]

[1] A. Ratner, C. De Sa, S. Wu, D. Selsam, C. Ré, “Data programming: Creating large training sets, quickly”, NeurIPS 2016.

Background

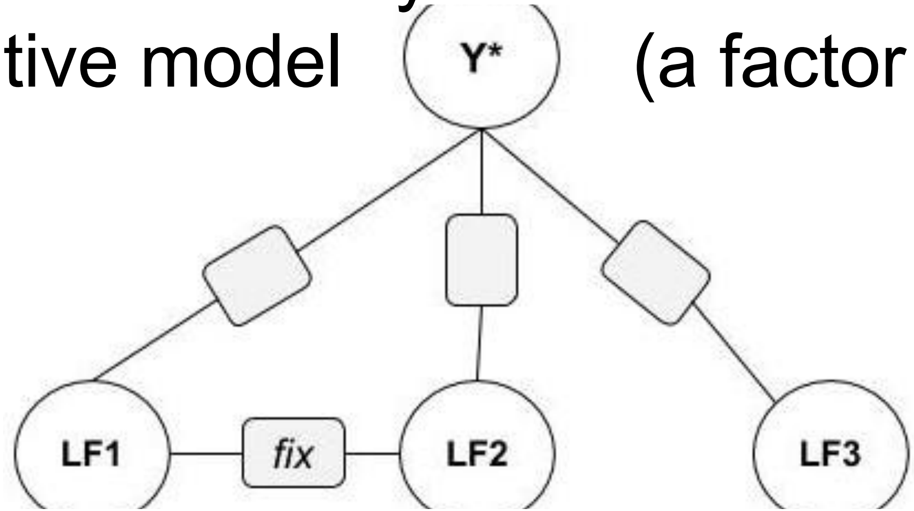
In Data Programming, the user encodes domain knowledge into *labeling functions (LF)*

```python
def labeling_function1(text_sample):
  return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```python
def labeling_function2(text_sample):
  return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

LFs can cheaply but imperfectly label subsets of data

In the Data Programming framework, the set of noisy LFs is denoised by a generative model (a factor graph), ...

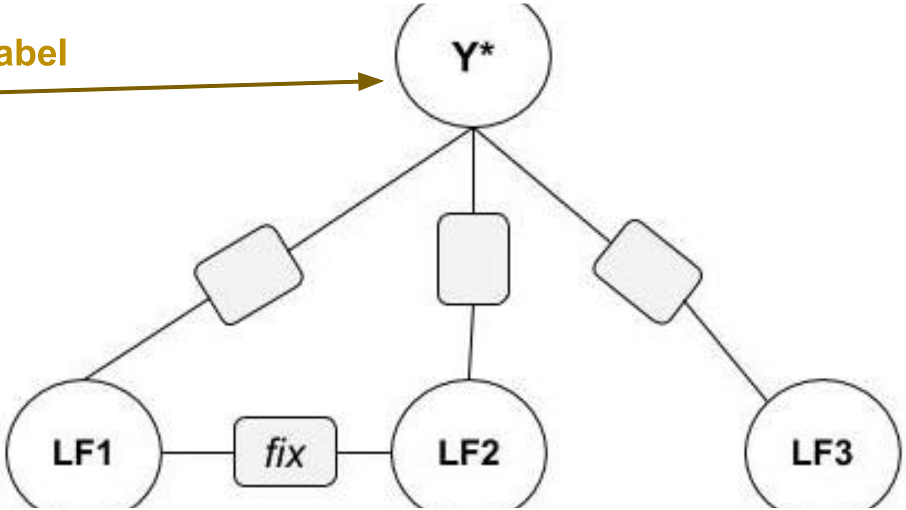



```
def labeling_function1(text_sample):
  return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```
def labeling_function2(text_sample):
  return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

… to produce *probabilistic labels* that can be then used to train a *downstream model*

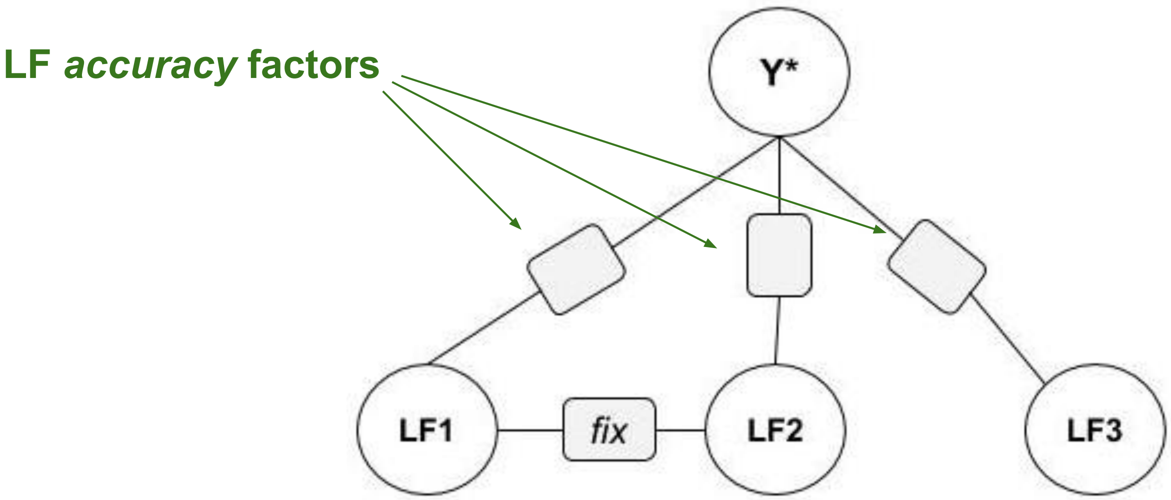
Latent true label




```
def labeling_function1(text_sample):
  return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```
def labeling_function2(text_sample):
  return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

Modeling the labeling accuracy of the LFs




```
def labeling_function1(text_sample):
  return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```
def labeling_function2(text_sample):
  return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

LFs often exhibit *statistical dependencies,*

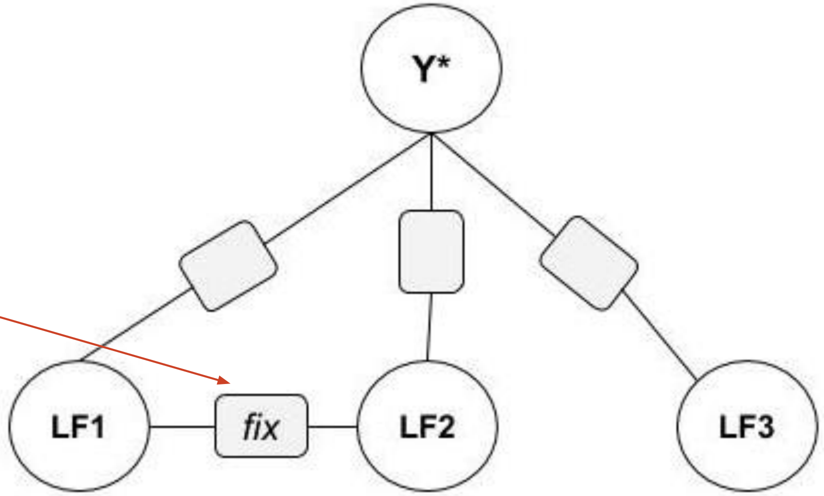
```python
def labeling_function1(text_sample):
  return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```python
def labeling_function2(text_sample):
  return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

such as LF2 *fixing* the vote of LF1 when both label

These are modeled as factors too

Fixing **factor**
(i.e. dependency)




```
def labeling_function1(text_sample):
  return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```
def labeling_function2(text_sample):
  return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

Carnegie Mellon University

Auton Lab

Examples of dependencies

LF_j	LF_k	factor type
best	great	bolstering
original	bad	priority
bad	don't waste	bolstering
worth	not worth	fixing
great	nothing great	fixing
worth	not worth	negated
special	not special	negated
recommend	terrible	priority
recommend	highly recommend	reinforcing
bad	absolutely horrible	reinforcing

New dependency types we introduce in this work

Motivation

But…

Specifying the correct dependency structure is hard!

→ **What happens if we choose to model an *incorrect structure*?**

→ **Does it help to have a fairly *simple model* that only models accuracy factors/*ignores dependencies*?**

Our work & contributions

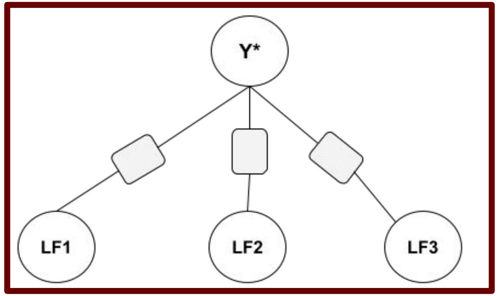
- *Theoretically bound* the downstream generalization risk under misspecification of dependencies

- *Empirically show* that specifying too many dependencies can significantly *deteriorate downstream performance*, even when the *structure appears sensible*

Theory first...

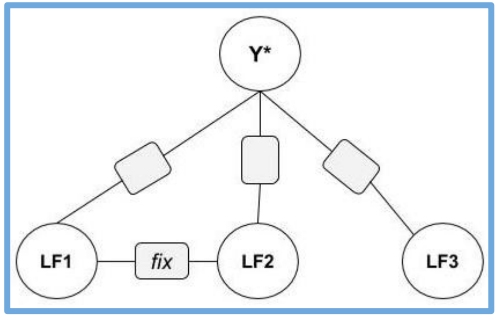
- We study two different models:

  a) ***Only accuracy factors*** **are modeled**

  b) ***Arbitrary*** **higher-order** ***dependencies*** **are modeled too**

- A priori, we do not fix any model to be the “true” one



Empirical risk minimization error (unspecific to our problem)

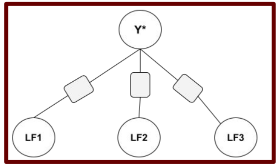
$$\text{Generalization risk} \leq \gamma + ||\Delta\text{accuracy\_parameters}||_1 + ||\text{dependency\_parameters}||_1$$

Accuracy parameter estimation error
w.r.t. to the true model

Carnegie Mellon University

Auton Lab

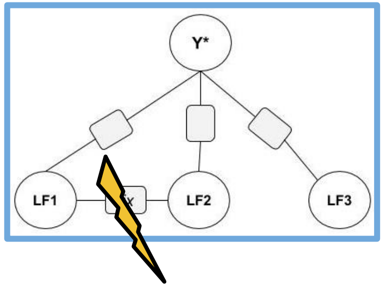
If we assume model a) to be the “true” one:



$$\text{Generalization risk} \leq \gamma + ||\Delta \text{accuracy\_parameters}||_1 + ||\text{dependency\_parameters}||_1$$

Accuracy parameter estimation error
w.r.t. to the true model

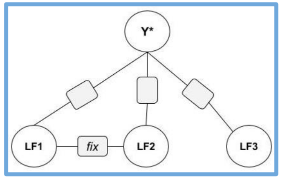
Parameter *excess* from misspecified dependencies



Carnegie Mellon University

Auton Lab

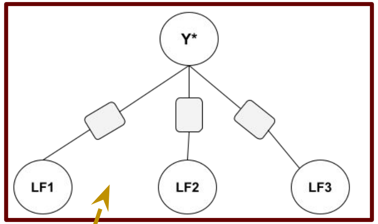
If we assume model b) to be the "true" one:



$$\text{Generalization risk} \leq \gamma + ||\Delta\text{accuracy\_parameters}||_1 + ||\text{dependency\_parameters}||_1$$

Accuracy parameter estimation error
w.r.t. to the true model

Magnitude of the dependencies that a) *failed* to model



Experiments[1]

i) select 135 real *LFs* for the IMDb Movie Review Sentiment dataset (*Positive* or *Negative*), that we expect to have helpful dependencies

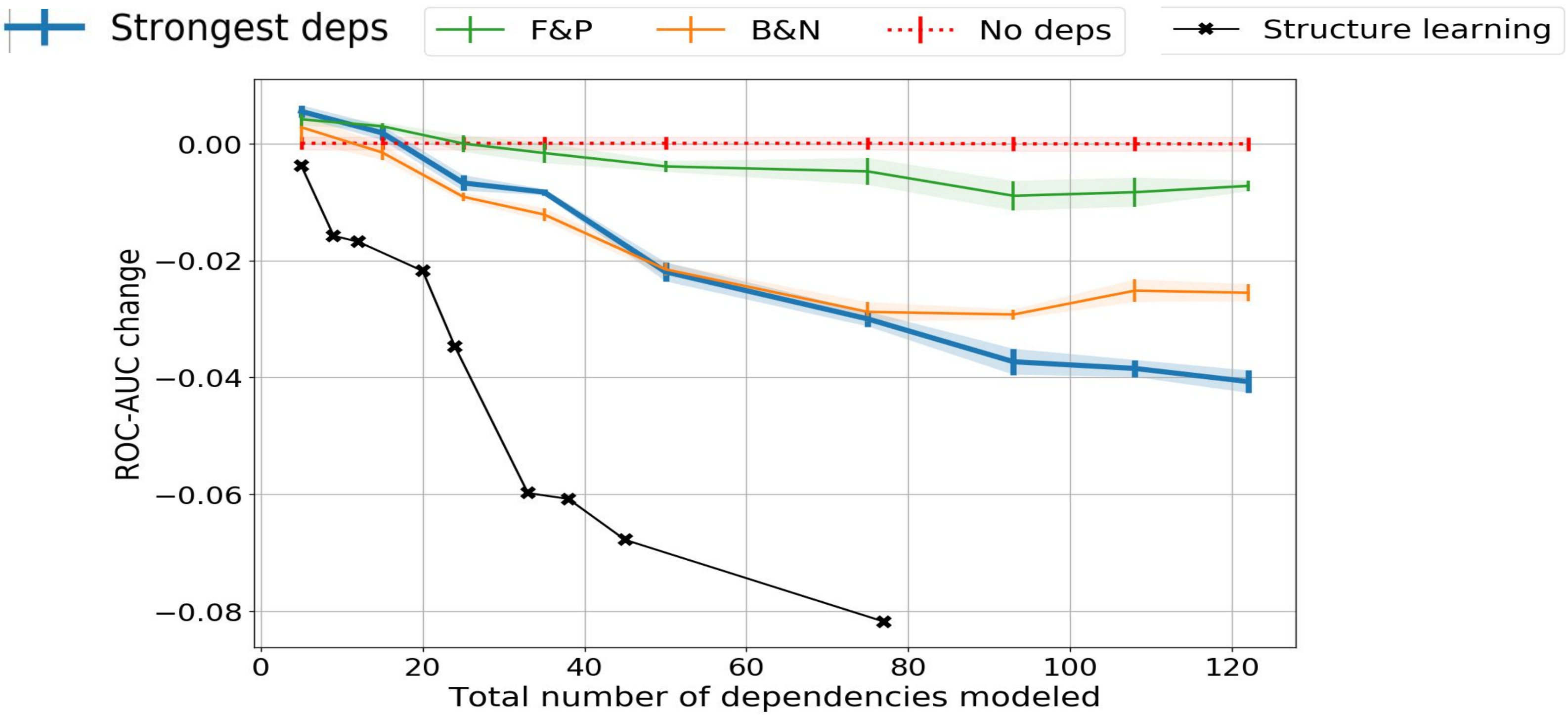
ii) select 85 real *LFs* for the Bias in Bios dataset, where we aim to differentiate between occupations (teacher vs professor)

[1] we report the test set performance of a simple 3-layer neural network trained on the probabilistic labels, averaged out over 100 runs

We use ground truth labels to obtain evidence of true dependencies

LF_j	LF_k	factor type l	factor value $v^l_{j,k}$
best	great	bolstering	801
original	bad	priority	327
bad	don't waste	bolstering	110
worth	not worth	fixing	238
great	nothing great	fixing	15
worth	not worth	negated	219
special	not special	negated	8
recommend	terrible	priority	53
recommend	highly recommend	reinforcing	226
bad	absolutely horrible	reinforcing	7

IMDB: ***Modeling more than a few dependencies worsens performance*** *by up to 4 AUC points!*



B&N - We only model *bolstering* and *negated* dependencies/factors
F&P - We only model *fixing* and *priority* dependencies/factors

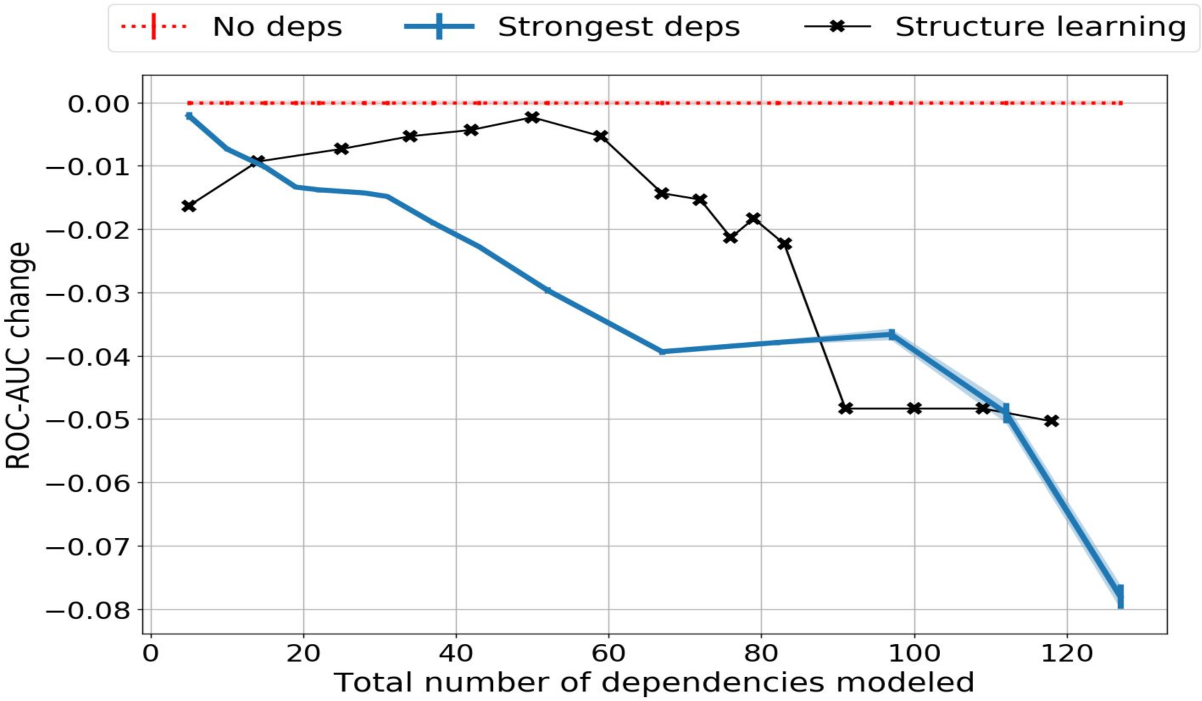
Carnegie Mellon University

Auton Lab

Again, even the weakest dependencies, **semantically, make sense**

LF_j	LF_k	factor type l	factor value $v_{j,k}^l$
best	great	bolstering	801
original	bad	priority	327
bad	don't waste	bolstering	110
worth	not worth	fixing	238
great	nothing great	fixing	15
worth	not worth	negated	219
special	not special	negated	8
recommend	terrible	priority	53
recommend	highly recommend	reinforcing	226
bad	absolutely horrible	reinforcing	7

Similar behavior observed in our Bias in Bios[3] experiment



[3] M. De-Arteaga et al., "Bias in bios: A case study of semantic representation bias in a high-stakes setting", In Proceedings of the Conference on Fairness, Accountability, and Transparency (2019)

Discussion & Conclusion

- Do the observed performance losses occur
  - due to the true model being (close to) the independent model?
  - due to higher sample complexity of learning additional parameters?

- In any case, our results and insights are highly relevant for practitioners
  - The behaviors we observed and studied have not been yet broadly recognized nor researched

- We conclude that
  - *Ignoring* potential *dependencies* often is a *reasonable baseline* for practitioners
  - Use *dependencies/structure learning carefully*

Thank you

Salva Rühling Cachay

Benedikt Boecking

Artur Dubrawski

This work was made possible by the CMU Robotics Institute Summer Scholars (RISS) program and the German Academic Exchange Service (DAAD)