Alternative Specifications in Machine Learning

Percy Liang

Jacob Steinhardt, Fereshte Khani

FLI Workshop

January 5, 2016
Foundations of machine learning

Train $p_0(x)$
Foundations of machine learning

Train $p_0(x)$

Test $p_1(x)$

Classic statistical learning theory:

training distribution = test distribution
Foundations of machine learning

Train $p_0(x)$

Test $p_1(x)$

Classic statistical learning theory:

training distribution = test distribution

Relaxation: domain adaptation, mild adversaries

training distribution $\approx$ test distribution
Foundations of machine learning

Train \( p_0(x) \)  
Test \( p_1(x) \)

Classic statistical learning theory:

\[
\text{training distribution} = \text{test distribution}
\]

Relaxation: domain adaptation, mild adversaries

\[
\text{training distribution} \approx \text{test distribution}
\]

Issue:

doesn't address large changes (disasters, adversaries)
Changes and changes
Changes and changes

Long-term risks of AI: unknown unknowns
What’s the right specification?

**Specification:** standard machine learning

**Input:** training data

**Output:** model that does obtains low *expected* test error

Is *expected* test error enough?
What’s the right specification?

**Specification:** standard machine learning

**Input:** training data

**Output:** model that does obtains low *expected* test error

Is *expected* test error enough?

**Scenario:**

- Err on 1% on instances
- Agents maximize, adversaries minimize, could drive us there!
New specification 1/2

[ACL 2016]

Specification: selective prediction

Input: training data

Output: model that outputs correct answer or “don’t know”

Previous work: Chow (1970); Tortorella (2000); El-Yaniv & Wiener (2010); Balsubramani (2016)
Unanimous prediction

Assumption: exists mapping with zero error
Unanimous prediction

\[ S = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
\end{bmatrix} \quad M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \quad T = \begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
\end{bmatrix} \]
Unanimous prediction

Models consistent with training data:

\[ C = \{ M \geq 0 : SM = T \} \]
Unanimous prediction

Models consistent with training data:

\[ C = \{ M \geq 0 : SM = T \} \]

Challenge:

Checking all consistent \( M \in C \) is slow...
Fast two point scheme

• Choose $M_1, M_2 \in C$ randomly enough

• Return "don't know" iff $M_1$ and $M_2$ disagree

Choose $M_1, M_2 \in C$ randomly enough

Return "don't know" iff $M_1$ and $M_2$ disagree
Experimental results

• GeoQuery semantic parsing dataset (800 train, 280 test)

What is the population of Texas?

![Graph showing the relationship between percentage of data and recall. The graph includes two lines: one for precision (LS) and another for recall (LS). The x-axis represents the percentage of data, ranging from 0 to 1, and the y-axis represents recall, ranging from 0 to 1. The precision line is a horizontal line at 1, while the recall line increases as the percentage of data increases.]
New specification 2/2

[NIPS 2016]

Specification: unsupervised risk estimation

Input: unlabeled examples and model
Output: estimate of labeled accuracy

Previous work: Donmez et al. (2010); Dawid/Skene (1979); Zhang et al. (2014); Jaffe et al. (2015); Balasubramanian et al. (2011)
Is this possible?

Compute $\mathbb{E}[\text{loss}(x, y; \theta)]$
Assumptions

Conditional independence:

\[ y \perpend x_1 \perpend x_2 \perpend x_3 \]
Assumptions

Conditional independence:

Loss function decomposes:

\[ A(x; \theta) - f_1(x_1, y; \theta) - f_2(x_2, y; \theta) - f_3(x_3, y; \theta) \]
Assumptions

Conditional independence:

Loss function decomposes:

\[ A(x; \theta) - f_1(x_1, y; \theta) - f_2(x_2, y; \theta) - f_3(x_3, y; \theta) \]

only conditional independence structure
Intuition

Three views agree → (probably) low error
Intuition

Three views disagree → high error
Tensor factorization

\( y \)

\( x_1 \quad x_2 \quad x_3 \)

\( (k \text{ labels, views } v = 1, 2, 3) \)

\[
\begin{bmatrix}
  f_v(x, 1) \\
  \ldots \\
  f_v(x, k)
\end{bmatrix}
\]

[Anandkumar et al., 2013]
Tensor factorization

(k labels, views $v = 1, 2, 3$)

\[
M_v = \begin{bmatrix}
E[f_v(x, 1) | y = 1] & \ldots & E[f_v(x, 1) | y = k] \\
\ldots & \ldots & \ldots \\
E[f_v(x, k) | y = 1] & \ldots & E[f_v(x, k) | y = k]
\end{bmatrix}
\]

[Anandkumar et al., 2013]
Tensor factorization

\[ y \]

\[ x_1 \quad x_2 \quad x_3 \]

\( (k \text{ labels}, \text{ views } v = 1, 2, 3) \)

- Observe \( \mathbb{E}[f_1(x, a)f_2(x, b)] \)
Tensor factorization

- Observe $\mathbb{E}[f_1(x, a)f_2(x, b)f_3(x, c)]$

($k$ labels, views $v = 1, 2, 3$)
Tensor factorization

(k labels, views $v = 1, 2, 3$)

• Observe $\mathbb{E}[f_1(x, a)f_2(x, b)f_3(x, c)]$

• Perform tensor factorization to obtain

$$M_{vba} = \mathbb{E}[f_v(x, b) \mid y = a]$$

[Anandkumar et al., 2013]
Tensor factorization

\[ \begin{align*}
\text{Observe } & \mathbb{E}[f_1(x, a)f_2(x, b)f_3(x, c)] \\
\text{Perform tensor factorization to obtain } & M_{vba} = \mathbb{E}[f_v(x, b) | y = a] \\
\text{Use to compute risk (up to label permutation) } & \mathbb{E}[A(x; \theta) - f_1(x_1, y; \theta) - f_2(x_2, y; \theta) - f_3(x_3, y; \theta)]
\end{align*} \]

(k labels, views \( v = 1, 2, 3 \))
Results

![Images of digit reconstructions](image1.png)

![Graph showing estimated risk vs distortion](graph.png)
• Maximize expected accuracy ⇒ selective prediction, unsupervised risk estimation

• Key question: Can we weaken the assumptions?
Code and data

worksheets.codalab.org

Collaborators

Fereshte Khani  Jacob Steinhardt

Thank you!