Data attribution at scale

Connecting ML behavior to (training) data

Andrew Ilyas (Part I & II) andrewilyas.com
Sung Min (Sam) Park (Part III) sungminpark.com
Logan Engstrom (Part IV) loganengstrom.com
Kristian Georgiev kristian-georgiev.github.io
Aleksander Mądry madry.mit.edu

ml-data-tutorial.org | ICML 2024

Main goals

Main goals

Introduce the emerging field of **data attribution** in ML

Main goals

Introduce the emerging field of **data attribution** in ML

Disambiguate different **types** of data attribution → focus in on one (*predictive*)

→ Different applications require different notions of data attribution

Main goals

Introduce the emerging field of **data attribution** in ML

Disambiguate different **types** of data attribution → focus in on one (*predictive*)

→ Different applications require different notions of data attribution

Connect predictive data attribution to well-studied theoretical problems & techniques

→ Old problem! In simple cases, known methods with theoretical guarantees

Main goals

Introduce the emerging field of **data attribution** in ML

Disambiguate different **types** of data attribution → focus in on one (*predictive*)

→ Different applications require different notions of data attribution

Connect predictive data attribution to well-studied theoretical problems & techniques

→ Old problem! In simple cases, known methods with theoretical guarantees

Survey the state of data attribution in modern/large-scale ML

→ Steady progress being made, quantitative evaluation

Main goals

Introduce the emerging field of **data attribution** in ML

Disambiguate different **types** of data attribution → focus in on one (*predictive*)

→ Different applications require different notions of data attribution

Connect predictive data attribution to well-studied theoretical problems & techniques

→ Old problem! In simple cases, known methods with theoretical guarantees

Survey the state of data attribution in modern/large-scale ML

→ Steady progress being made, quantitative evaluation

Discuss some applications

Outline/Logistics

Part I: Data problems in ML (~30 minutes)

Corroborative, game-theoretic, and predictive data attribution

Part II: Theoretical foundations (~30 minutes)

History & theory of predictive data attribution (datamodeling)

Break (5 mins)

Part III: Scaling to modern settings (~40 minutes)

Challenges & successes in predictive data attribution for large ML systems

Part IV: Applications of data attribution (~20 minutes)

Past, present, and future applications of data attribution

Outline/Logistics

Follow along! ml-data-tutorial.org

Part I: Data problems in ML (~30 minutes)

Corroborative, game-theoretic, and predictive data attribution

Part II: Theoretical foundations (~30 minutes)

History & theory of predictive data attribution (datamodeling)

Break (5 mins)

Part III: Scaling to modern settings (~40 minutes)

Challenges & successes in predictive data attribution for large ML systems

Part IV: Applications of data attribution (~20 minutes)

Past, present, and future applications of data attribution

Outline/Logistics

Follow along! ml-data-tutorial.org

Part I: Data problems in ML (~30 minutes)

Corroborative, game-theoretic, and predictive data attribution

Part II: Theoretical foundations (~30 minutes)

History & theory of predictive data attribution (datamodeling)

Break (5 mins)

Part III: Scaling to modern settings (~40 minutes)

Challenges & successes in predictive data attribution for large ML systems

Part IV: Applications of data attribution (~20 minutes)

Past, present, and future applications of data attribution

Comments
ml-data-tutorial@mit.edu

Compliments
ml-data-tutorial@mit.edu

Complaints
madry@mit.edu

Part I: Data problems in ML

Corroborative, game-theoretic, and predictive data attribution

Chapter 1 of 4

Data problems in ML

Data problems in ML

What data should I train my language model on?

Data problems in ML

How much should I compensate content creators for their data?

What data should I train my language model on?

Data problems in ML

How much should I compensate content creators for their data?

When can I trust my model's output?

What data should I train my language model on?

Data problems in ML

How much should I compensate content creators for their data?

What data should I train my language model on?

When can I trust my model's output?

What training data is to blame for my model's error?

Data problems in ML

How much should I compensate content creators for their data?

What data should I train my language model on?

When can I trust my model's output?

What training data is to blame for my model's error?

Did my model output something copyrighted?

Data problems in ML

How much should I compensate content creators for their data?

What data should I train my language model on?

When can I trust my model's output?

What training data is to blame for my model's error?

How robust is my model to malicious training data?

Did my model output something copyrighted?

Data problems in ML

How much should I compensate content creators for their data?

What data should I train my language model on?

When can I trust my model's output?

What training data is to blame for my model's error?

How robust is my model to malicious training data?

Did my model output something copyrighted?

These are **data problems**: Qs about relating ML performance to data

Data problems in ML

Data attribution methods aim to answer these kinds of questions

Data problems in ML

Data attribution methods aim to answer these kinds of questions

*A data attribution method seeks to **connect** model **behavior** at test time **with data**.*

Data problems in ML

Data attribution methods aim to answer these kinds of questions

*A data attribution method seeks to **connect** model **behavior** at test time **with data**.*

what data?

Data problems in ML

Data attribution methods aim to answer these kinds of questions

*A data attribution method seeks to **connect** model **behavior** at test time **with data**.*

what behavior?

what data?

Data problems in ML

Data attribution methods aim to answer these kinds of questions

in what way?

A data attribution method seeks to ***connect***
model ***behavior*** *at test time* ***with data****.*

what behavior?

what data?

An *application-driven* taxonomy

How much should I compensate content creators for their data?

What data should I train my language model on?

When can I trust my model's output?

What training data is to blame for my model's error?

How robust is my model to malicious training data?

Did my model output something copyrighted?

These are **data problems**: Qs about relating ML performance to data

An *application-driven* taxonomy

Did my model output something copyrighted?

How much should I compensate content creators for their data?

What data should I train my language model on?

When can I trust my model's output?

What training data is to blame for my model's error?

How robust is my model to malicious training data?

These are **data problems**: Qs about relating ML performance to data

An *application-driven* taxonomy

Did my model output something copyrighted?

When can I trust my model's output?

Corroborative data attribution

How much should I compensate content creators for their data?

What training data is to blame for my model's error?

What data should I train my language model on?

How robust is my model to malicious training data?

These are **data problems**: Qs about relating ML performance to data

An *application-driven* taxonomy

Did my model output something copyrighted?

When can I trust my model's output?

Corroborative data attribution

How much should I compensate content creators for their data?

What training data is to blame for my model's error?

Game-theoretic data attribution

What data should I train my language model on?

How robust is my model to malicious training data?

These are **data problems**: Qs about relating ML performance to data

An *application-driven* taxonomy

Did my model output something copyrighted?

When can I trust my model's output?

Corroborative data attribution

How much should I compensate content creators for their data?

What training data is to blame for my model's error?

Game-theoretic data attribution

What data should I train my language model on?

How robust is my model to malicious training data?

Predictive data attribution

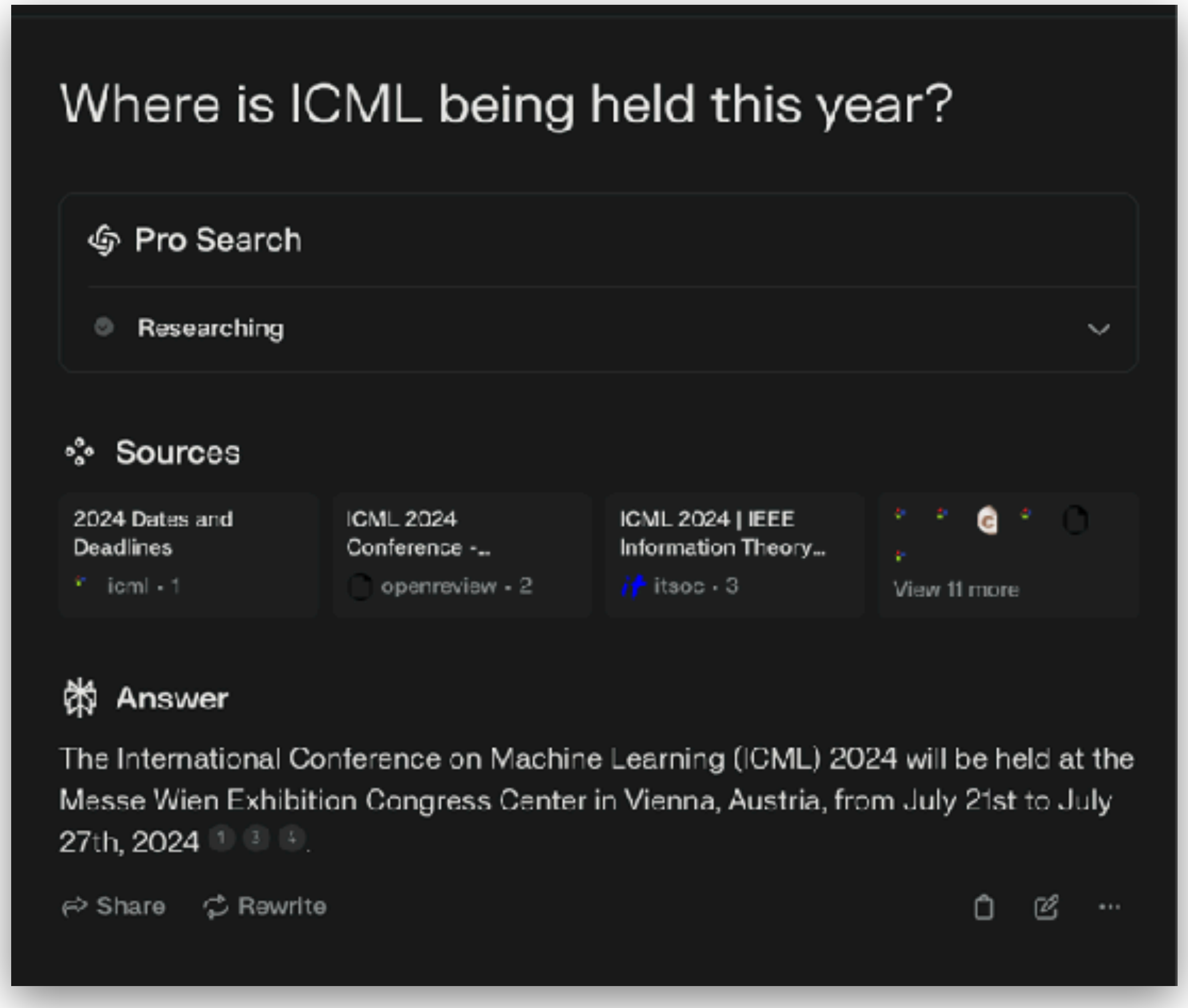
These are **data problems**: Qs about relating ML performance to data

Corroborative attribution (Evidence-finding)

Motivation [Worledge Shen Meister Winston Guestrin ’23]

Corroborative attribution (Evidence-finding)

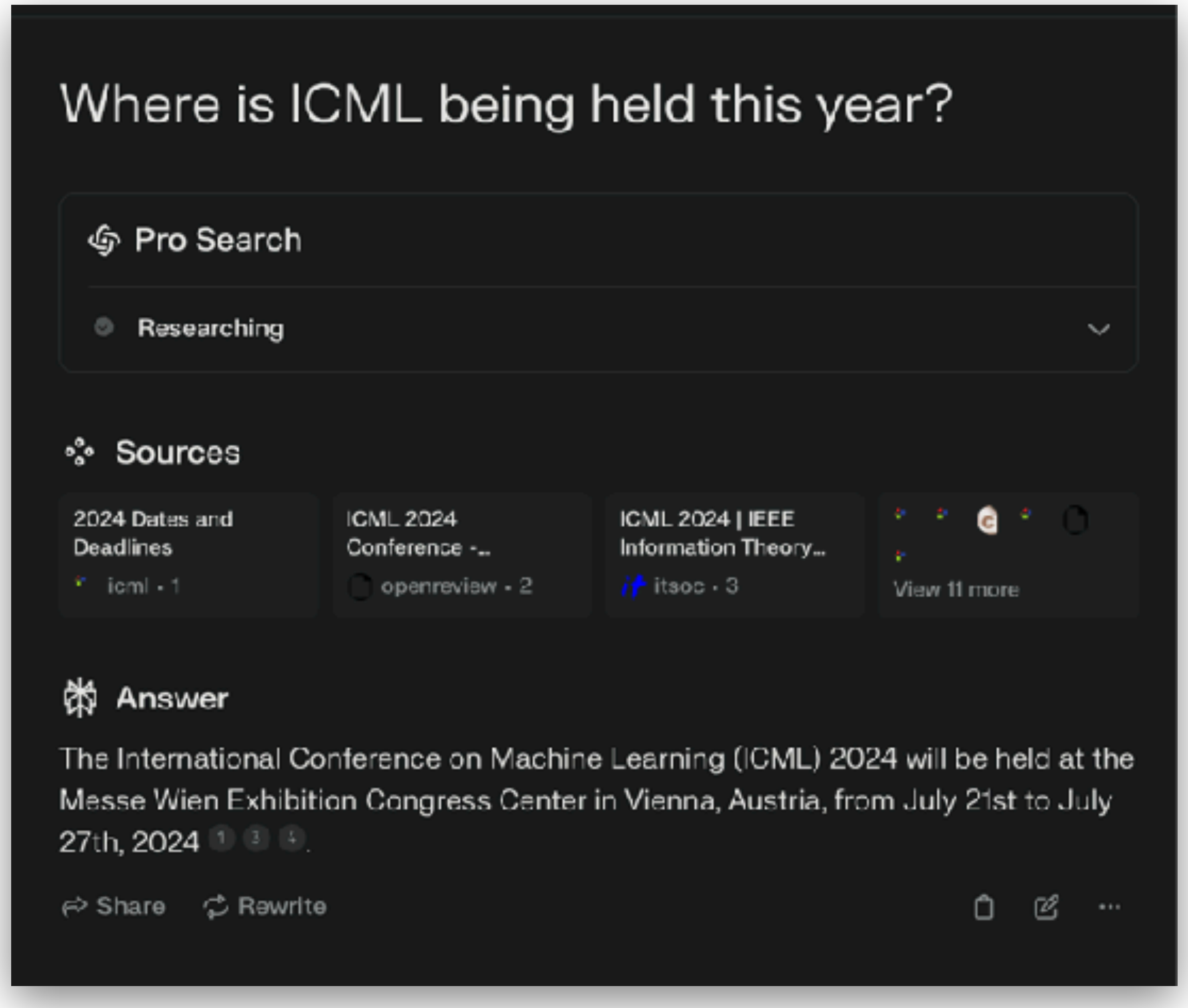
Motivation [Worledge Shen Meister Winston Guestrin '23]



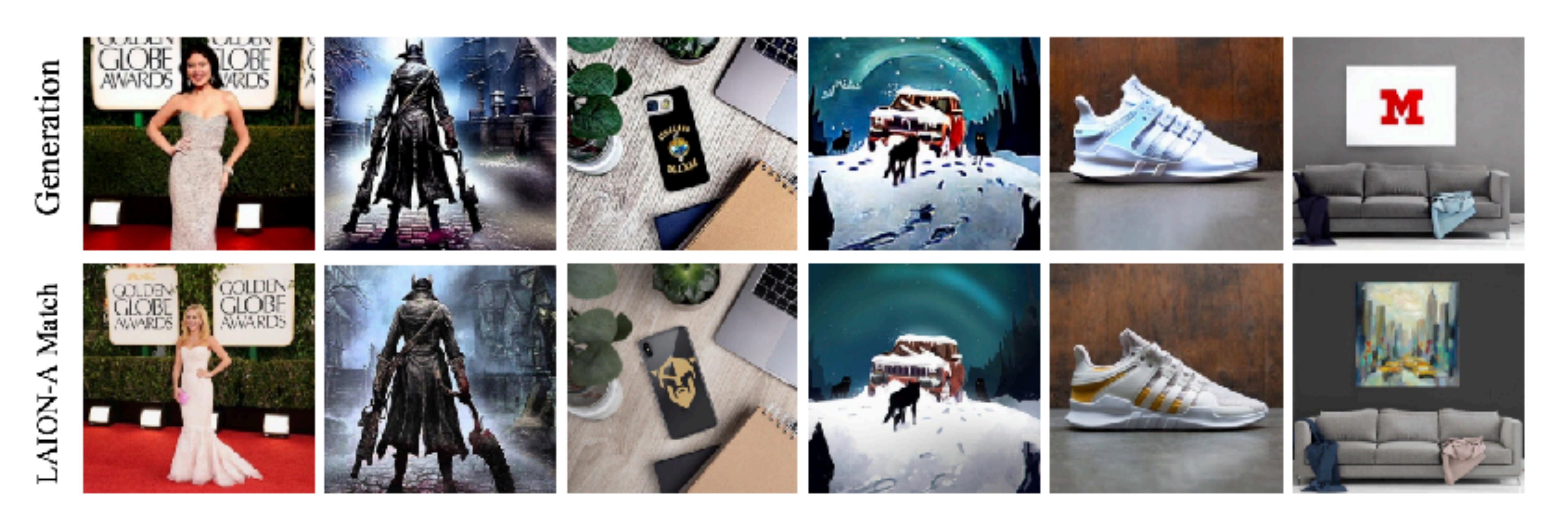
Citation

Corroborative attribution (Evidence-finding)

Motivation [Worledge Shen Meister Winston Guestrin '23]



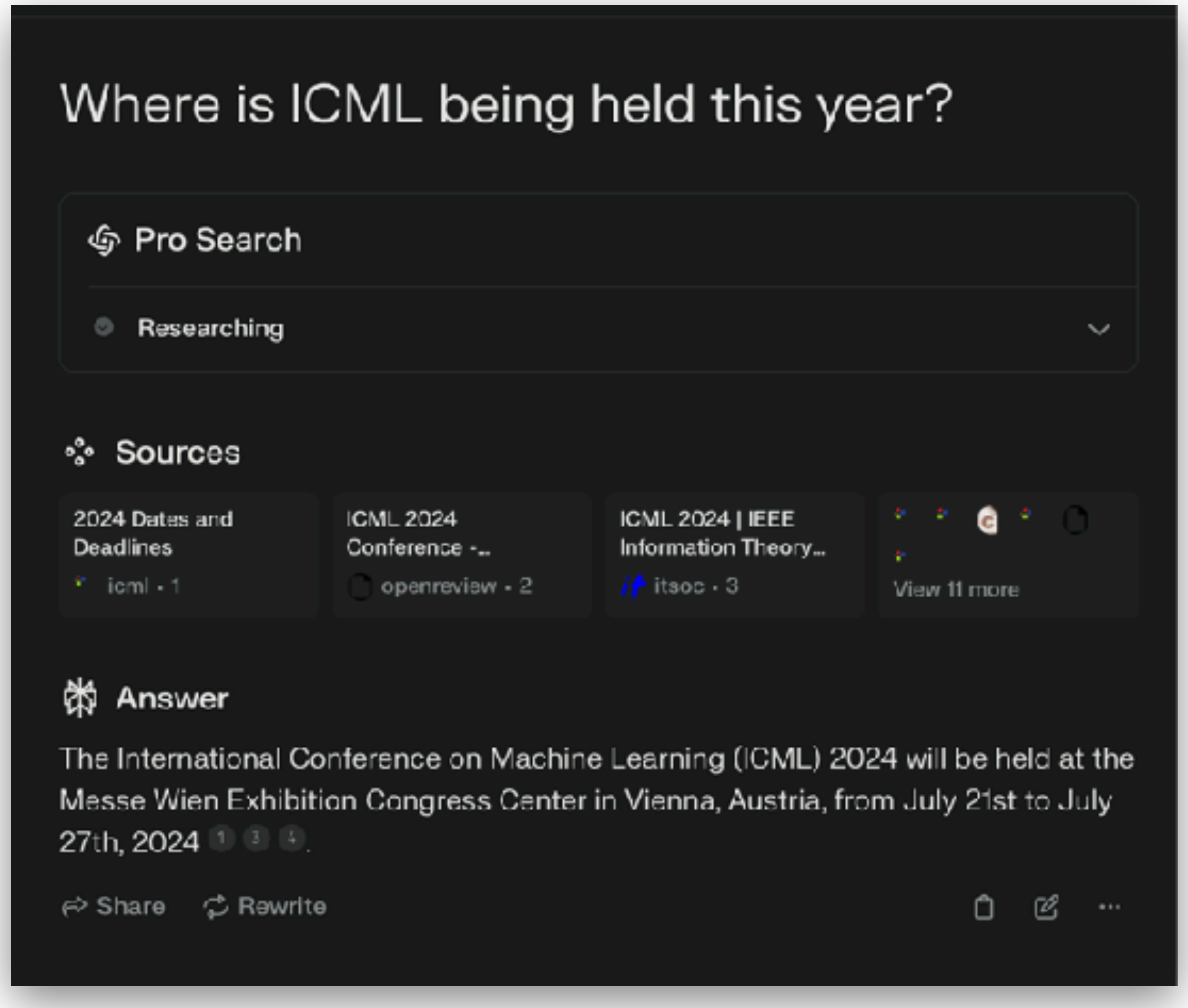
Citation



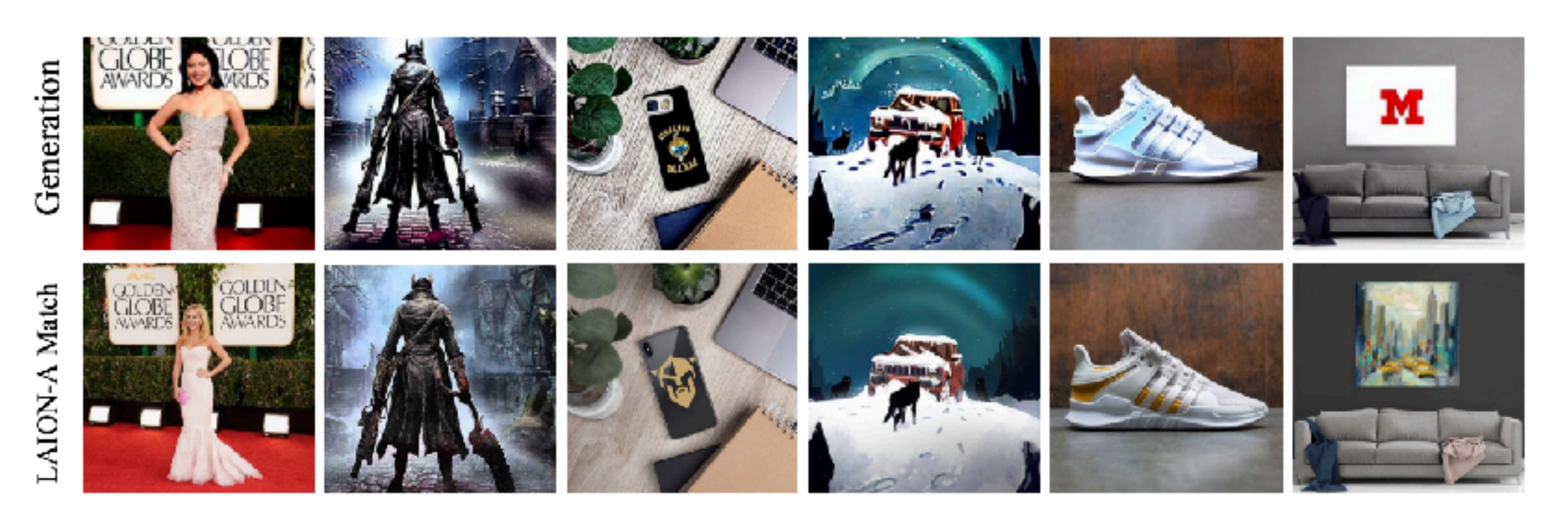
Copyright detection
[Somepalli Singla Goldblum Geiping Goldstein '22]

Corroborative attribution (Evidence-finding)

Motivation [Worledge Shen Meister Winston Guestrin ’23]



Citation



Copyright detection
[Somepalli Singla Goldblum Geiping Goldstein ’22]

We often want to know whether an ML output has any basis in a given dataset

Corroborative attribution (Evidence-finding)

Relation to data attribution

*A data attribution method seeks to **connect***
*model **behavior** at test time **with data**.*

Corroborative attribution (Evidence-finding)

Relation to data attribution

corroborative

*A ^ data attribution method seeks to **connect** model **behavior** at test time **with data**.*

Corroborative attribution (Evidence-finding)

Relation to data attribution

corroborative

*A ^ data attribution method finds **evidence** for model **behavior** at test time **with data**.*

Corroborative attribution (Evidence-finding)

Relation to data attribution

corroborative

*A ^ data attribution method finds **evidence** for model **outputs** at test time **with data**.*

Corroborative attribution (Evidence-finding)

Relation to data attribution

corroborative

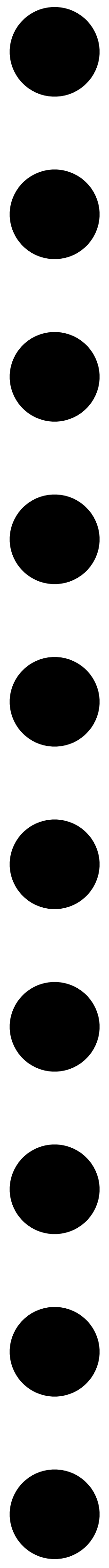
A ^ data attribution method ***finds evidence for***

model ***outputs*** *at test time* ***in a given corpus****.*

Corroborative attribution (Evidence-finding)

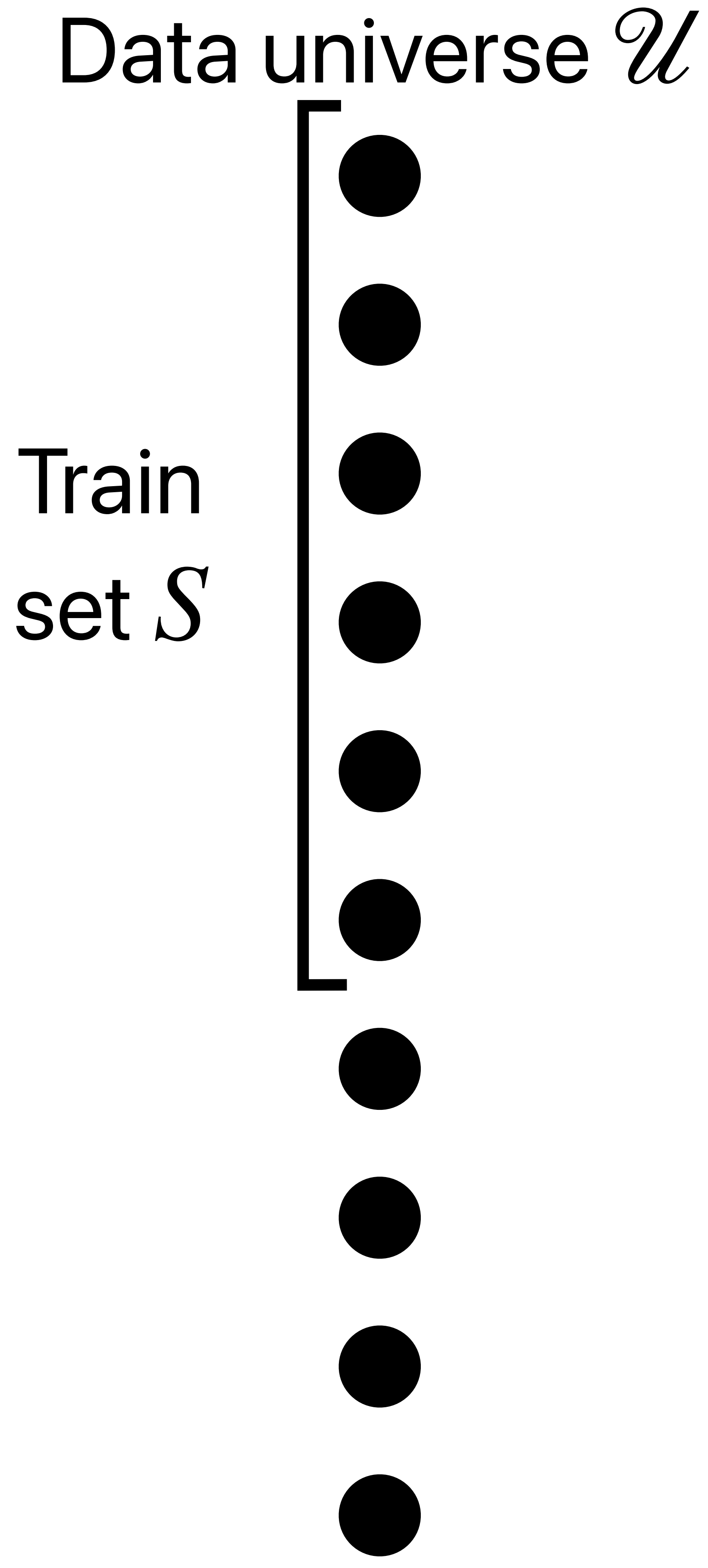
Illustration and examples

Data universe $\mathcal{U}$



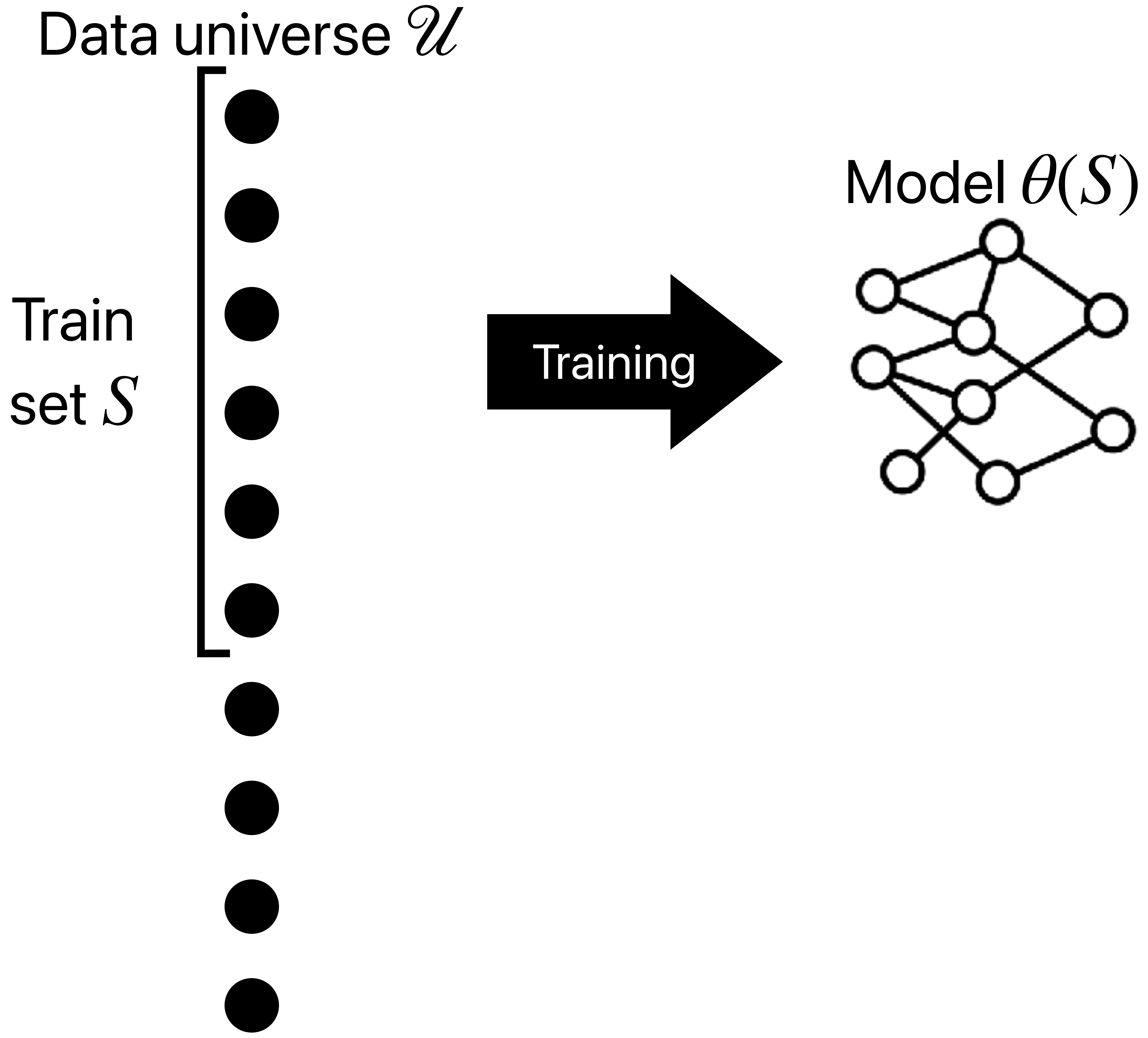
Corroborative attribution (Evidence-finding)

Illustration and examples



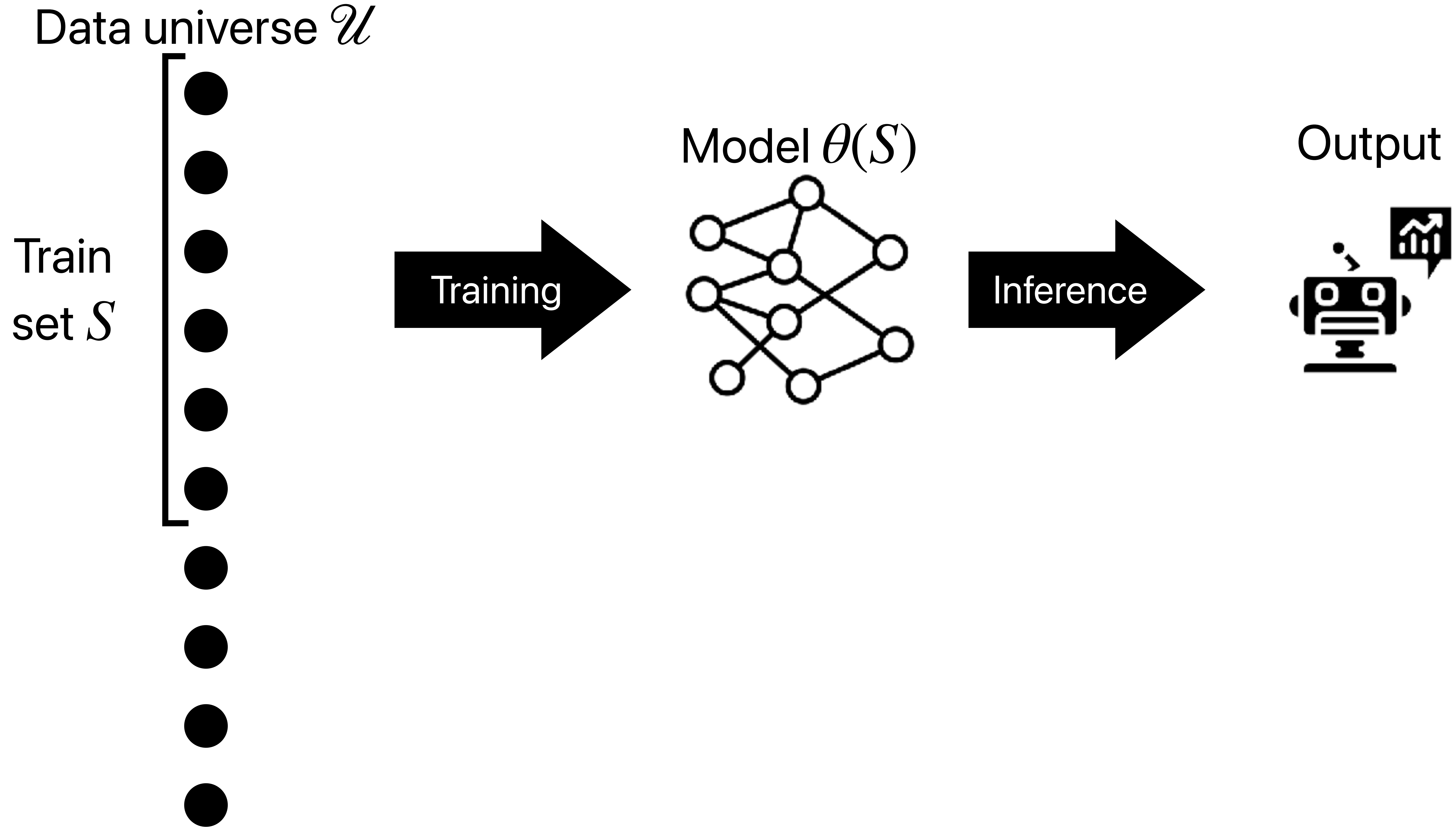
Corroborative attribution (Evidence-finding)

Illustration and examples



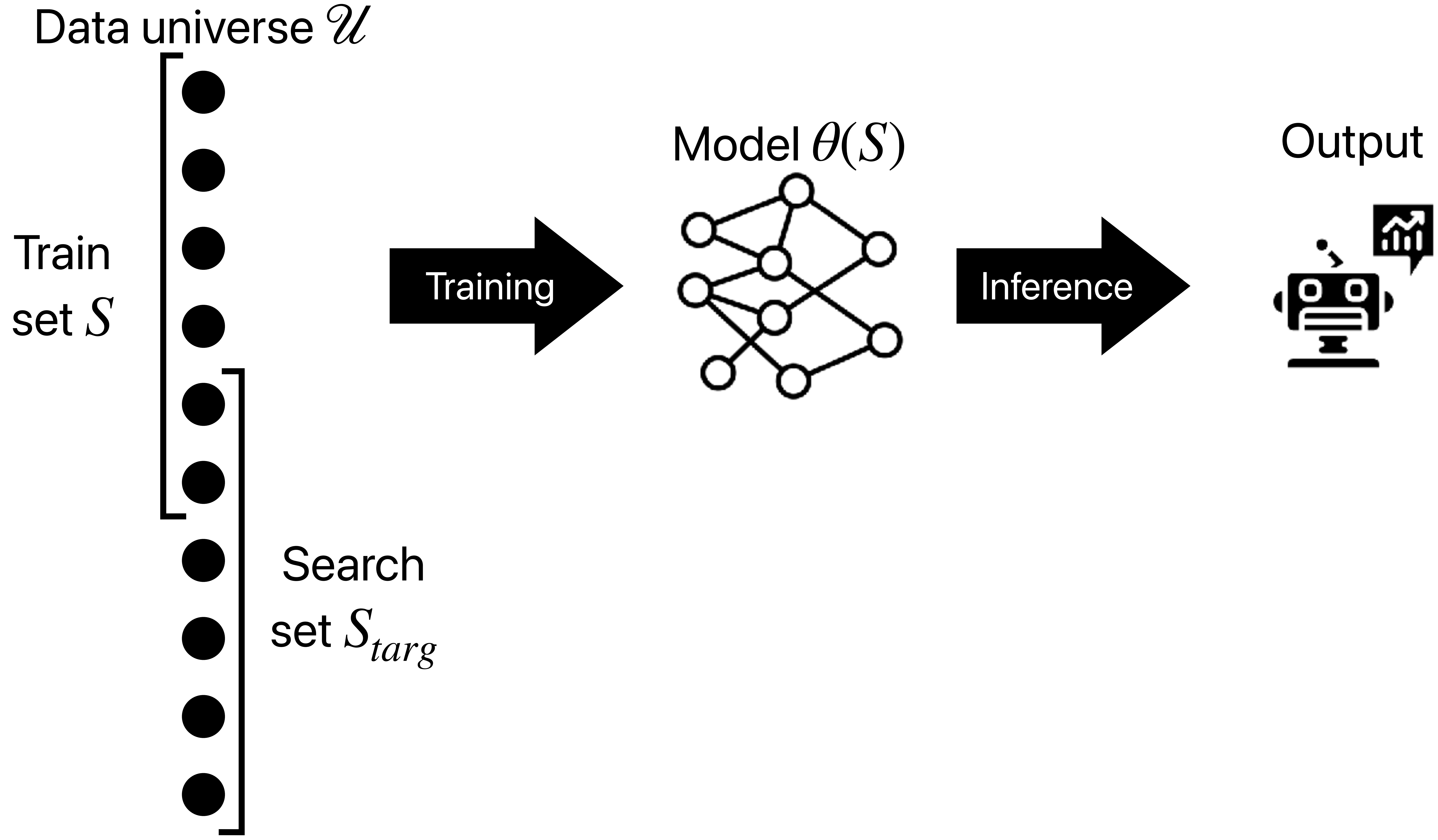
Corroborative attribution (Evidence-finding)

Illustration and examples



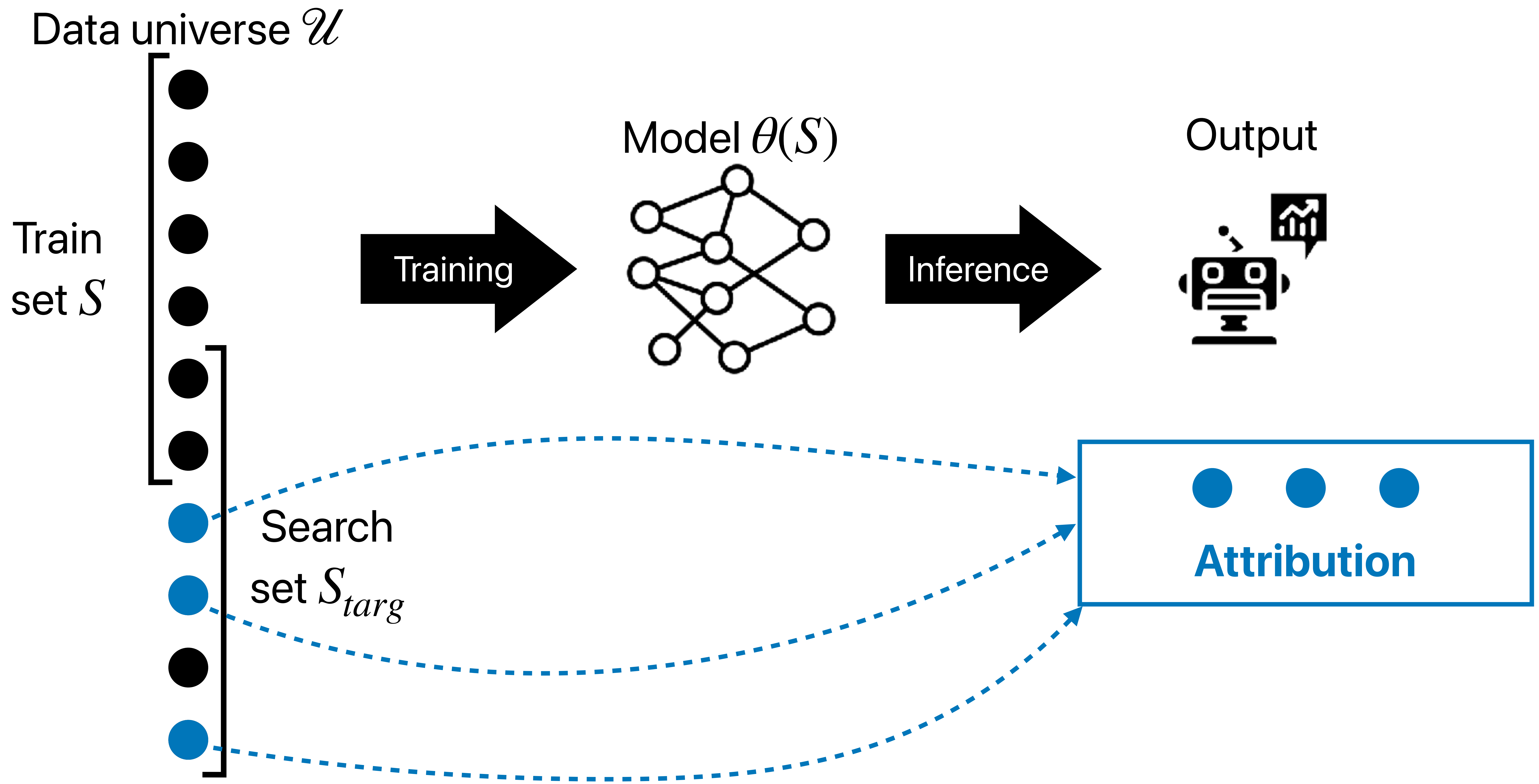
Corroborative attribution (Evidence-finding)

Illustration and examples



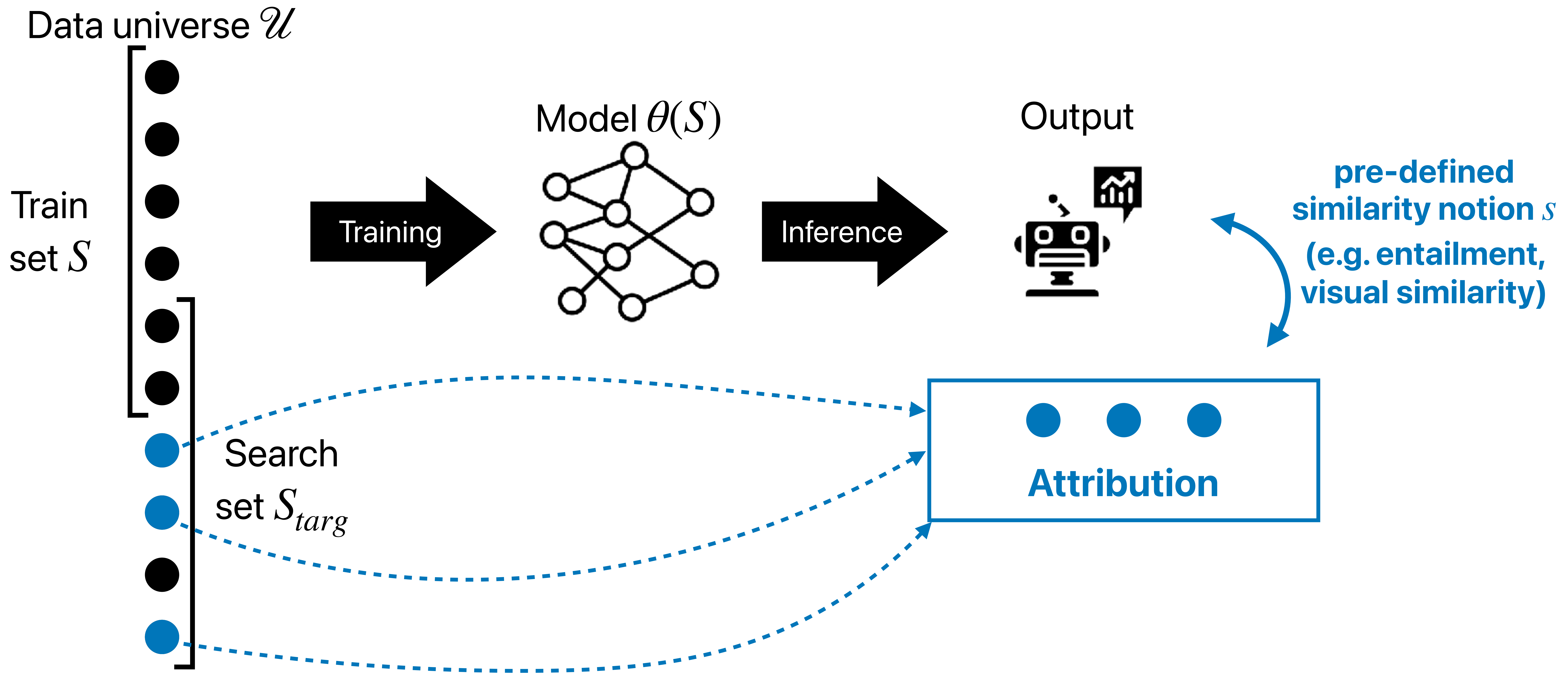
Corroborative attribution (Evidence-finding)

Illustration and examples



Corroborative attribution (Evidence-finding)

Illustration and examples



Corroborative attribution (Evidence-finding)

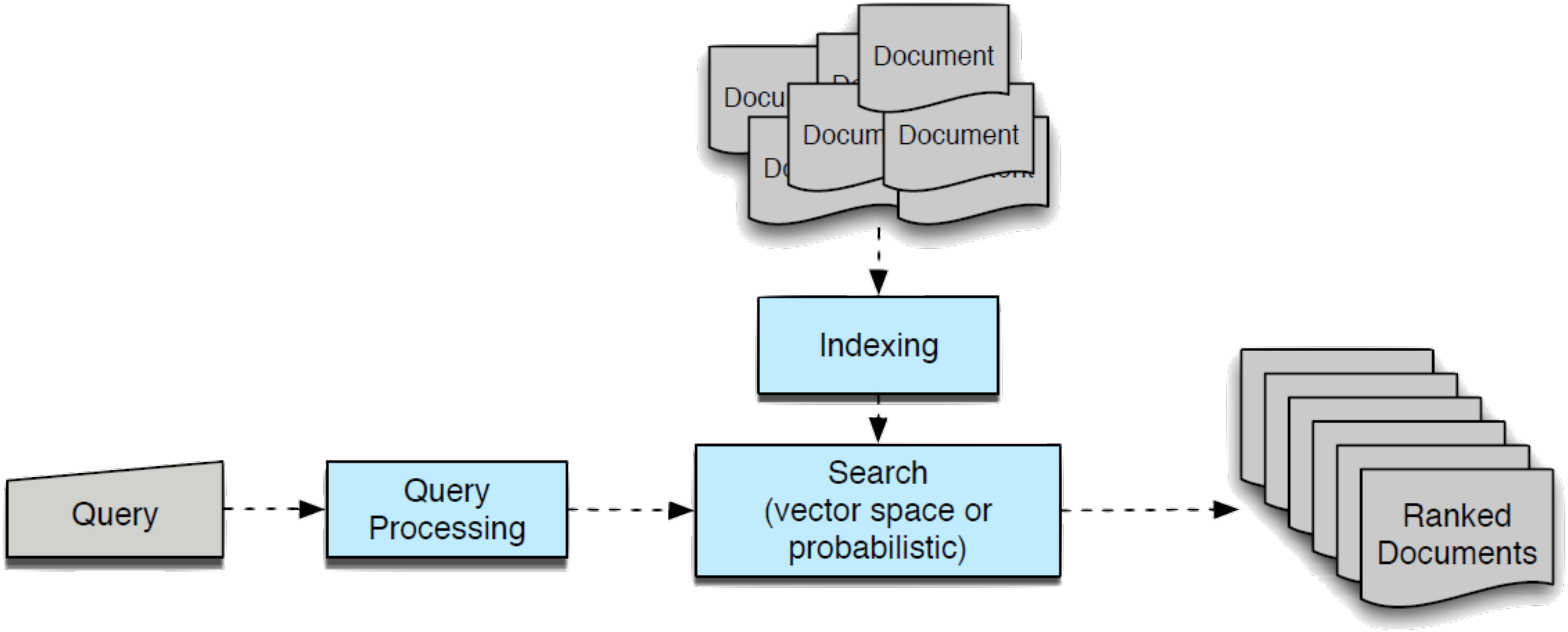
Illustration and examples

Example method: Information retrieval system

Corroborative attribution (Evidence-finding)

Illustration and examples

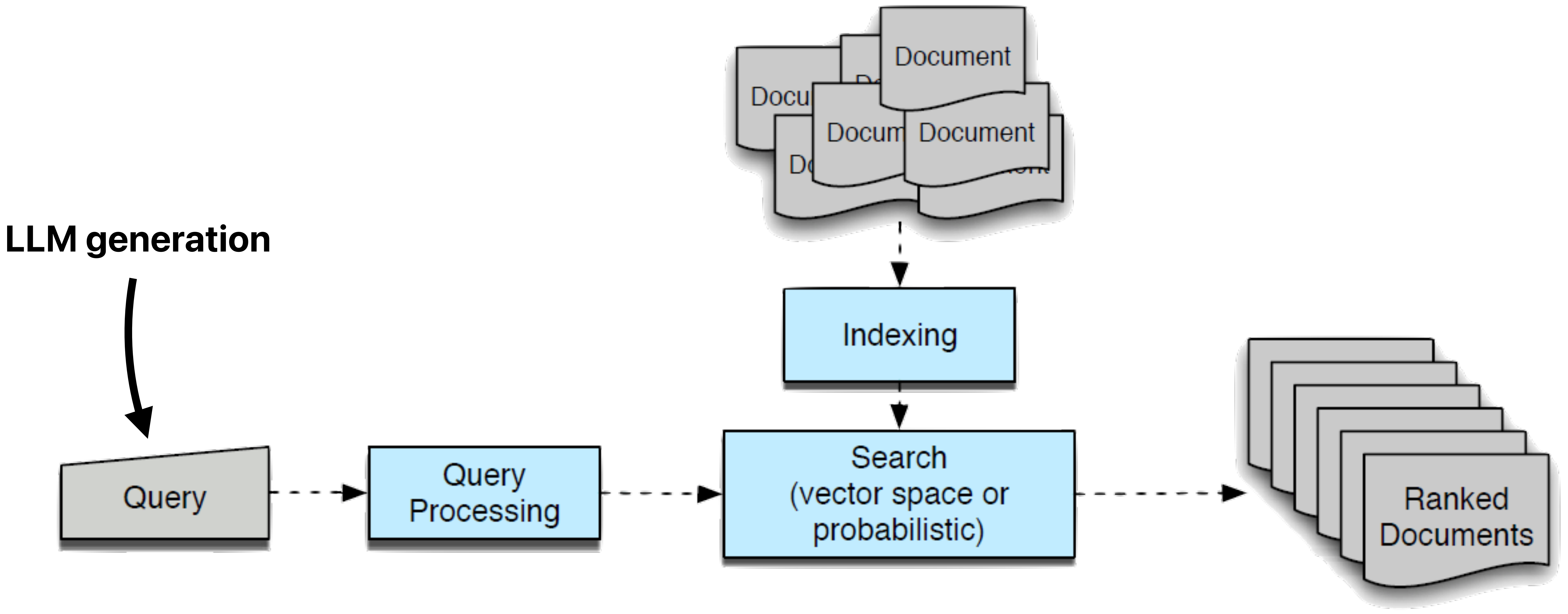
Example method: Information retrieval system



Corroborative attribution (Evidence-finding)

Illustration and examples

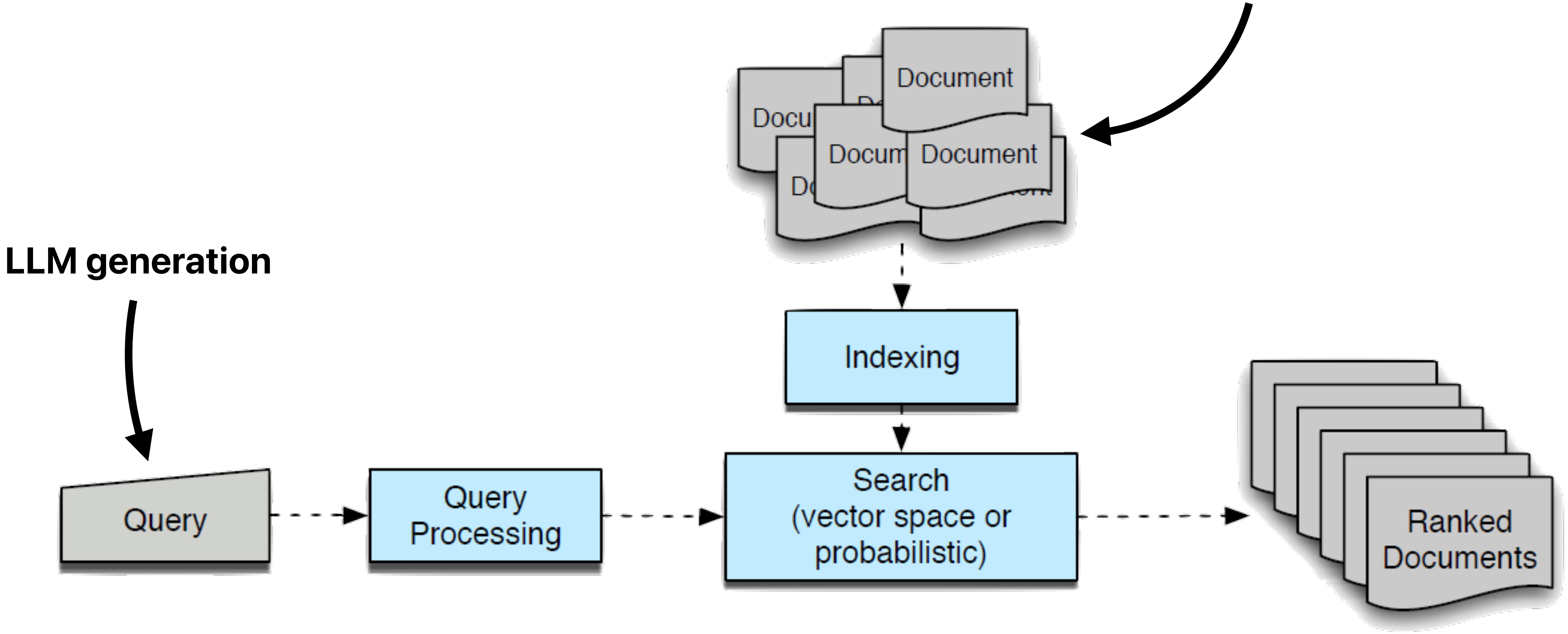
Example method: Information retrieval system



Corroborative attribution (Evidence-finding)

Illustration and examples

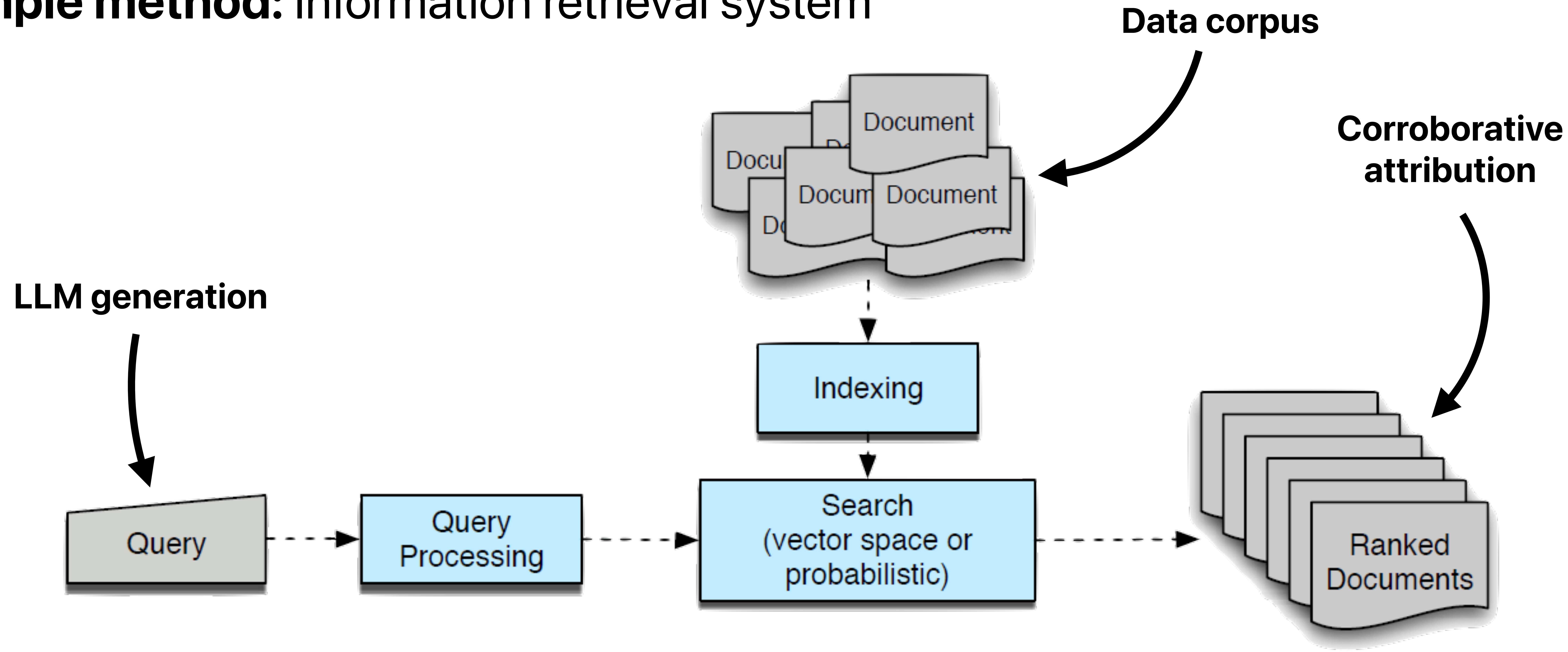
Example method: Information retrieval system



Corroborative attribution (Evidence-finding)

Illustration and examples

Example method: Information retrieval system



Game-theoretic attribution (Credit assignment)

Motivation [Ghorbani Zou ’19; Jia Dao Wang et al. ’19]

Game-theoretic attribution (Credit assignment)

Motivation [Ghorbani Zou '19; Jia Dao Wang et al. '19]

In other use cases, we care about **why** the model behaved a certain way

[WSMWG '23] call this **contributive** data attribution

Game-theoretic attribution (Credit assignment)

Motivation [Ghorbani Zou '19; Jia Dao Wang et al. '19]

In other use cases, we care about **why** the model behaved a certain way

[WSMWG '23] call this **contributive** data attribution

Game-theoretic attribution treats model training as multi-player *cooperative* game between data sources (could be individual examples or data vendors)

Goal is to assign "fair credit" to different sources for the outcome

Game-theoretic attribution (Credit assignment)

Motivation [Ghorbani Zou '19; Jia Dao Wang et al. '19]

In other use cases, we care about **why** the model behaved a certain way

[WSMWG '23] call this **contributive** data attribution

Game-theoretic attribution treats model training as multi-player *cooperative* game between data sources (could be individual examples or data vendors)

Goal is to assign "fair credit" to different sources for the outcome


AI companies are running out of data for training chatbots so Adobe keeps paying for it

Adobe is one of a few companies that pay artists for submissions to train its AI models

By Laura Bratton Published April 12, 2024

Game-theoretic attribution (Credit assignment)

Motivation [Ghorbani Zou '19; Jia Dao Wang et al. '19]

In other use cases, we care about **why** the model behaved a certain way

[WSMWG '23] call this **contributive** data attribution

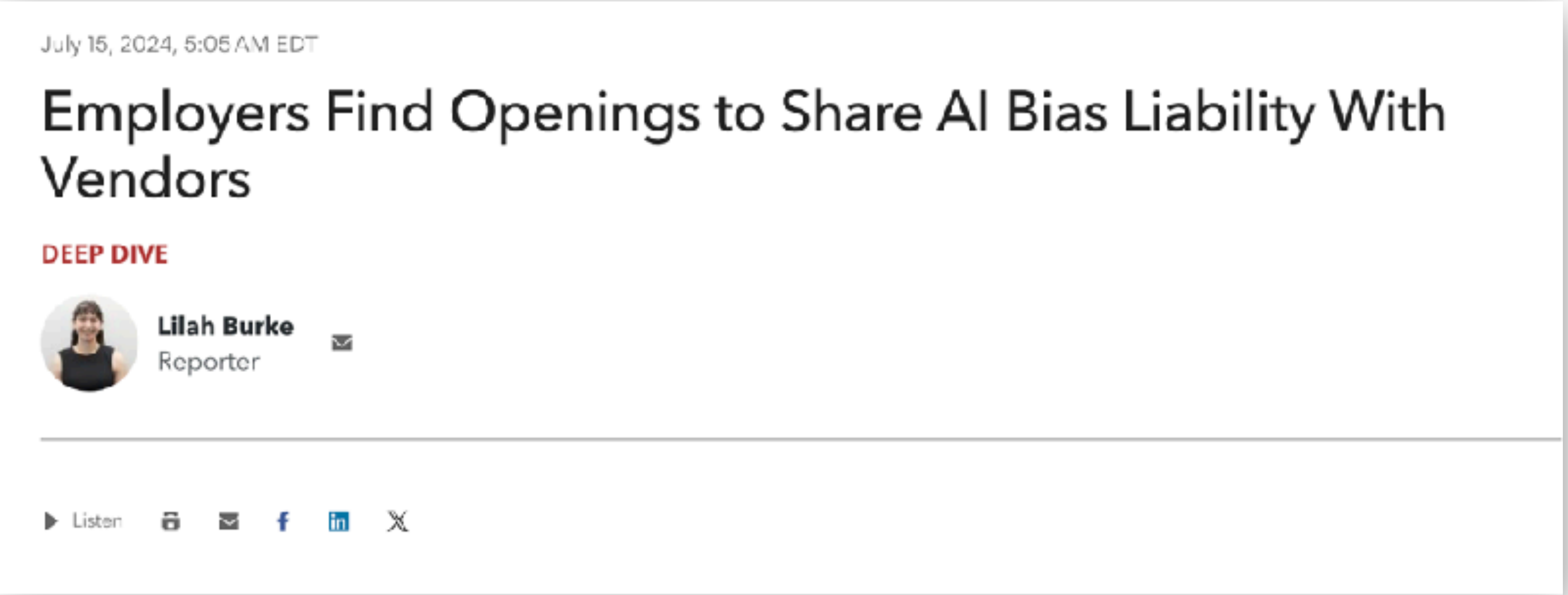
Game-theoretic attribution treats model training as multi-player *cooperative* game between data sources (could be individual examples or data vendors)

Goal is to assign "fair credit" to different sources for the outcome


AI companies are running out of data for training chatbots so Adobe keeps paying for it

Adobe is one of a few companies that pay artists for submissions to train its AI models

By Laura Bratton Published April 12, 2024


July 15, 2024, 5:05 AM EDT

Employers Find Openings to Share AI Bias Liability With Vendors

DEEP DIVE

Lilah Burke
Reporter

Listen

Game-theoretic attribution (Credit assignment)

Relation to data attribution

*A data attribution method seeks to **connect** model **behavior** at test time **with data**.*

Game-theoretic attribution (Credit assignment)

Relation to data attribution

game-theoretic

*A ^ data attribution method seeks to **connect** model **behavior** at test time **with data**.*

Game-theoretic attribution (Credit assignment)

Relation to data attribution

game-theoretic

A ^ data attribution method ***assigns fair credit for***

model ***behavior*** *at test time* ***with data****.*

Game-theoretic attribution (Credit assignment)

Relation to data attribution

game-theoretic

A^data attribution method ***assigns fair credit for***

a utility function *at test time* ***with data***.

Game-theoretic attribution (Credit assignment)

Relation to data attribution

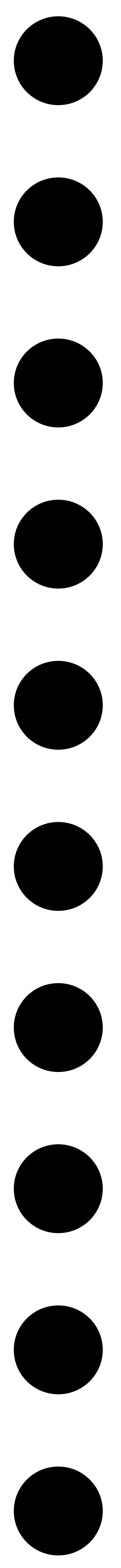
game-theoretic

A^data attribution method ***assigns fair credit for a utility function*** *at test time* ***to training data sources.***

Game-theoretic attribution (Credit assignment)

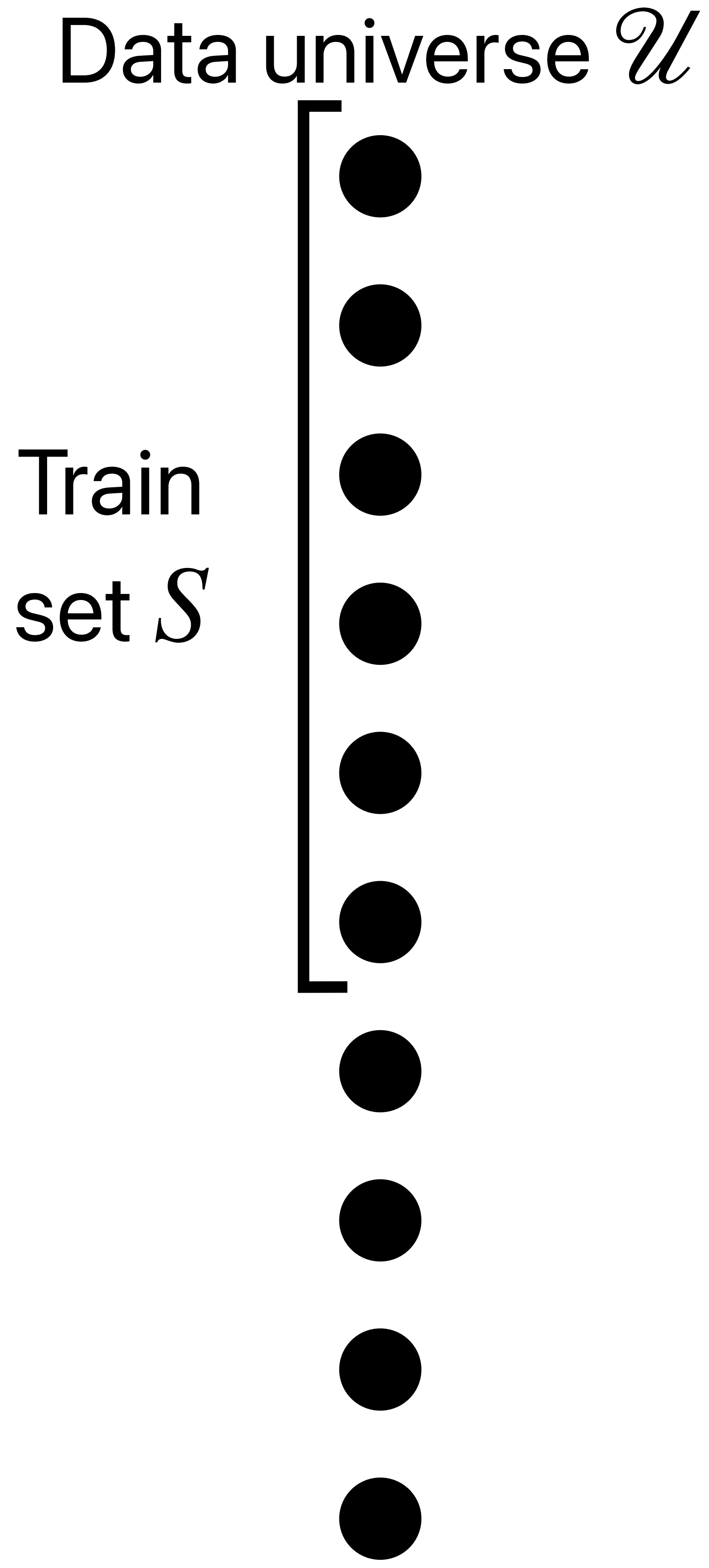
Illustration and examples

Data universe $\mathscr{U}$



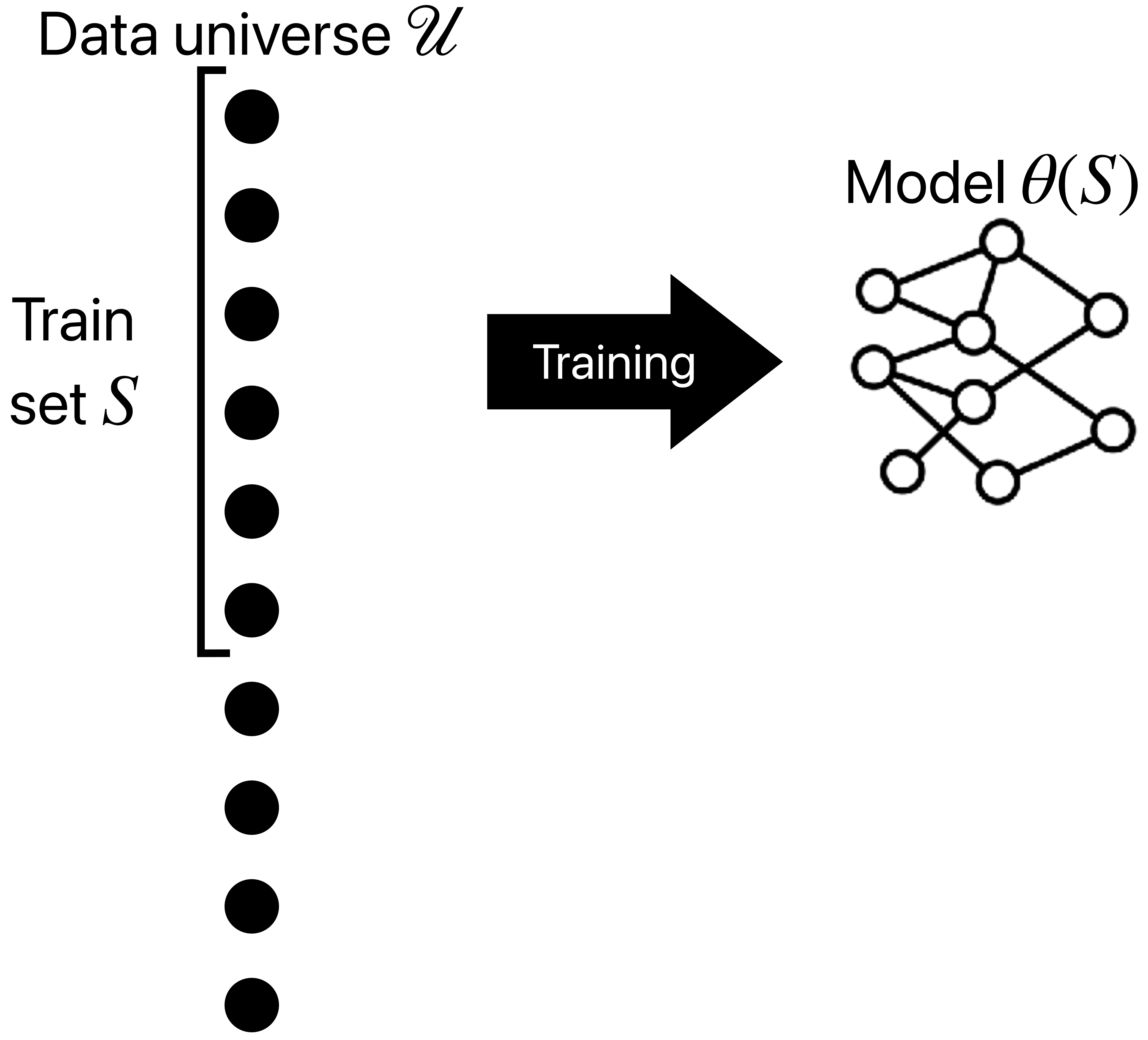
Game-theoretic attribution (Credit assignment)

Illustration and examples



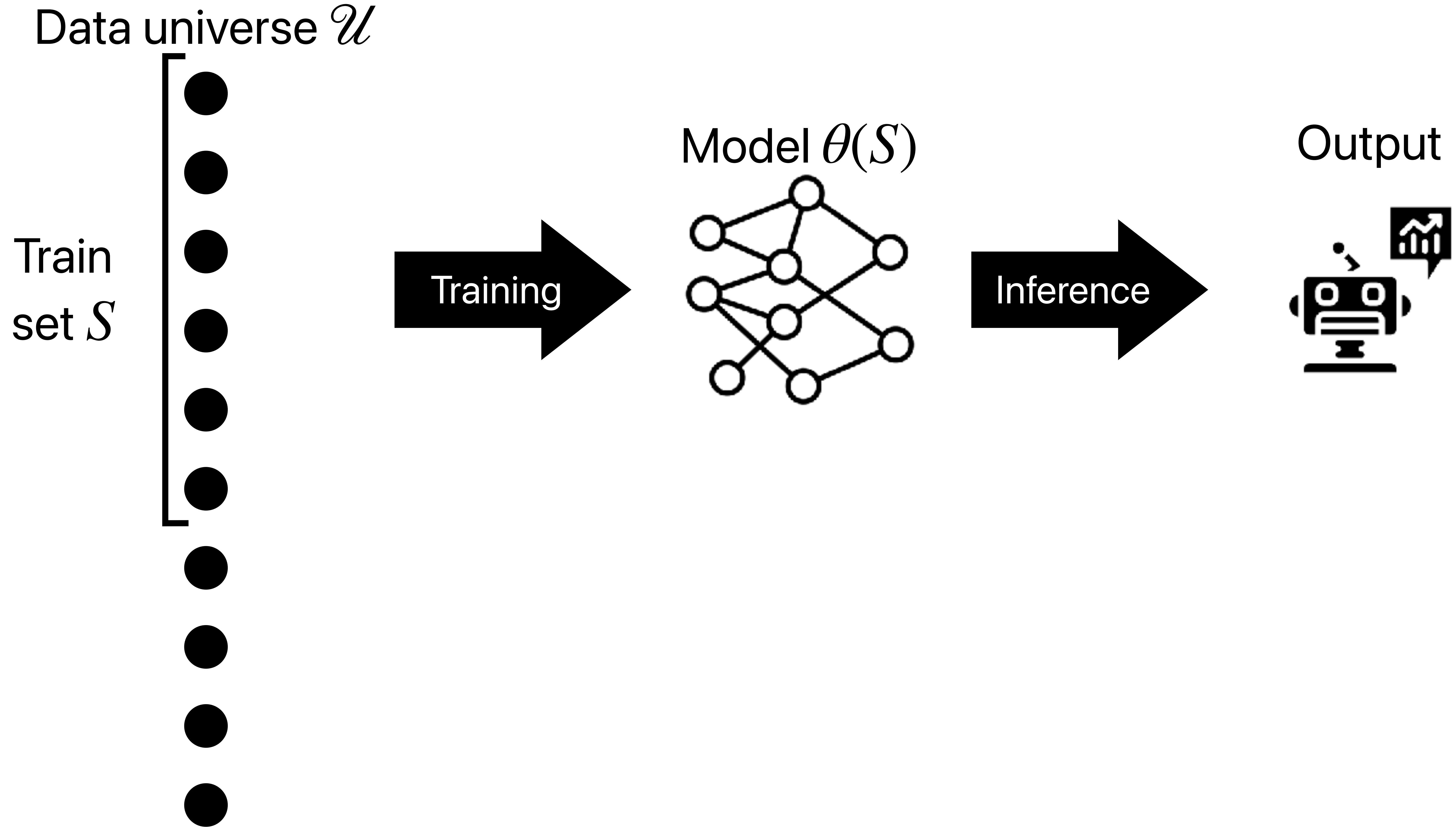
Game-theoretic attribution (Credit assignment)

Illustration and examples



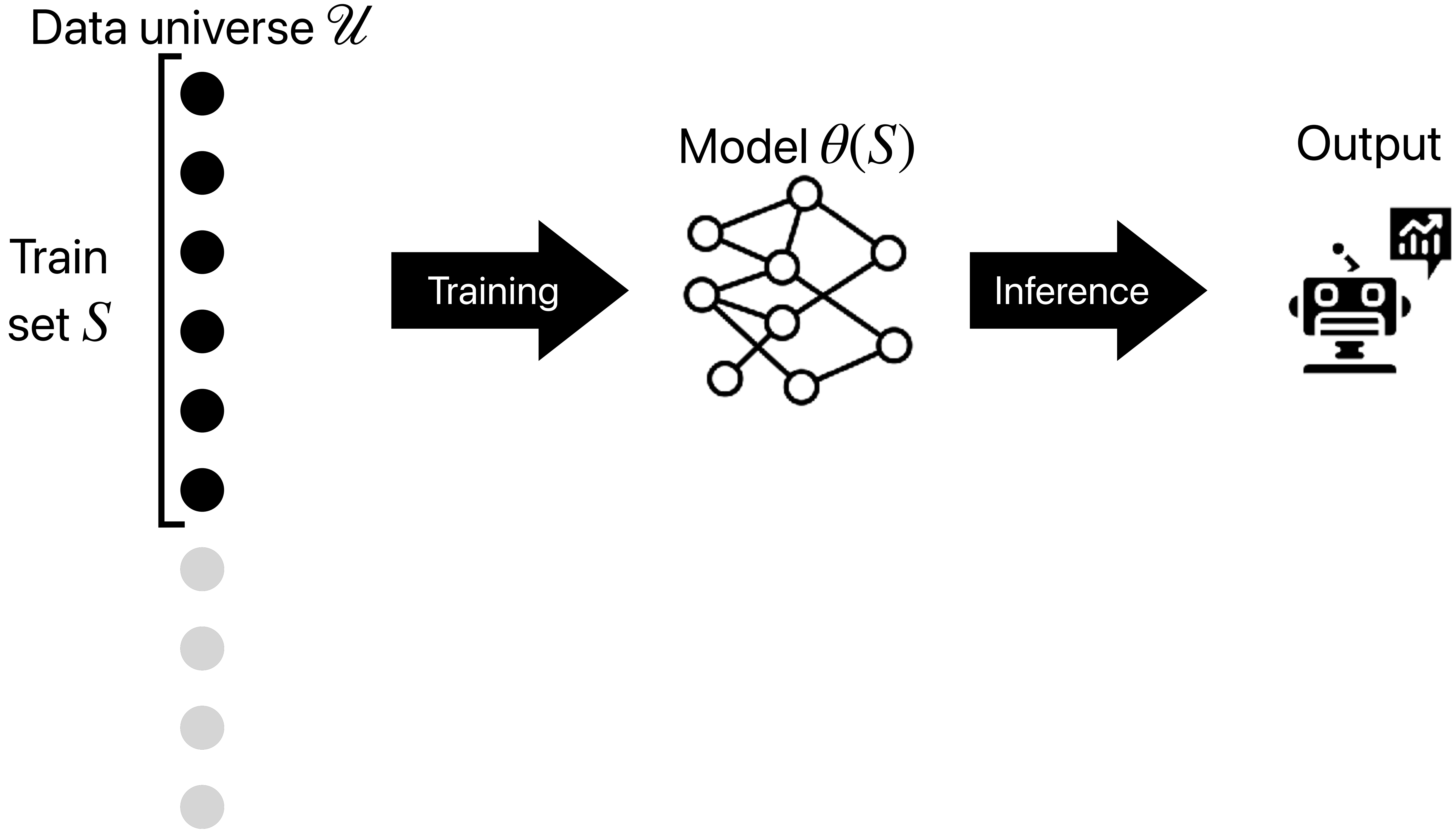
Game-theoretic attribution (Credit assignment)

Illustration and examples



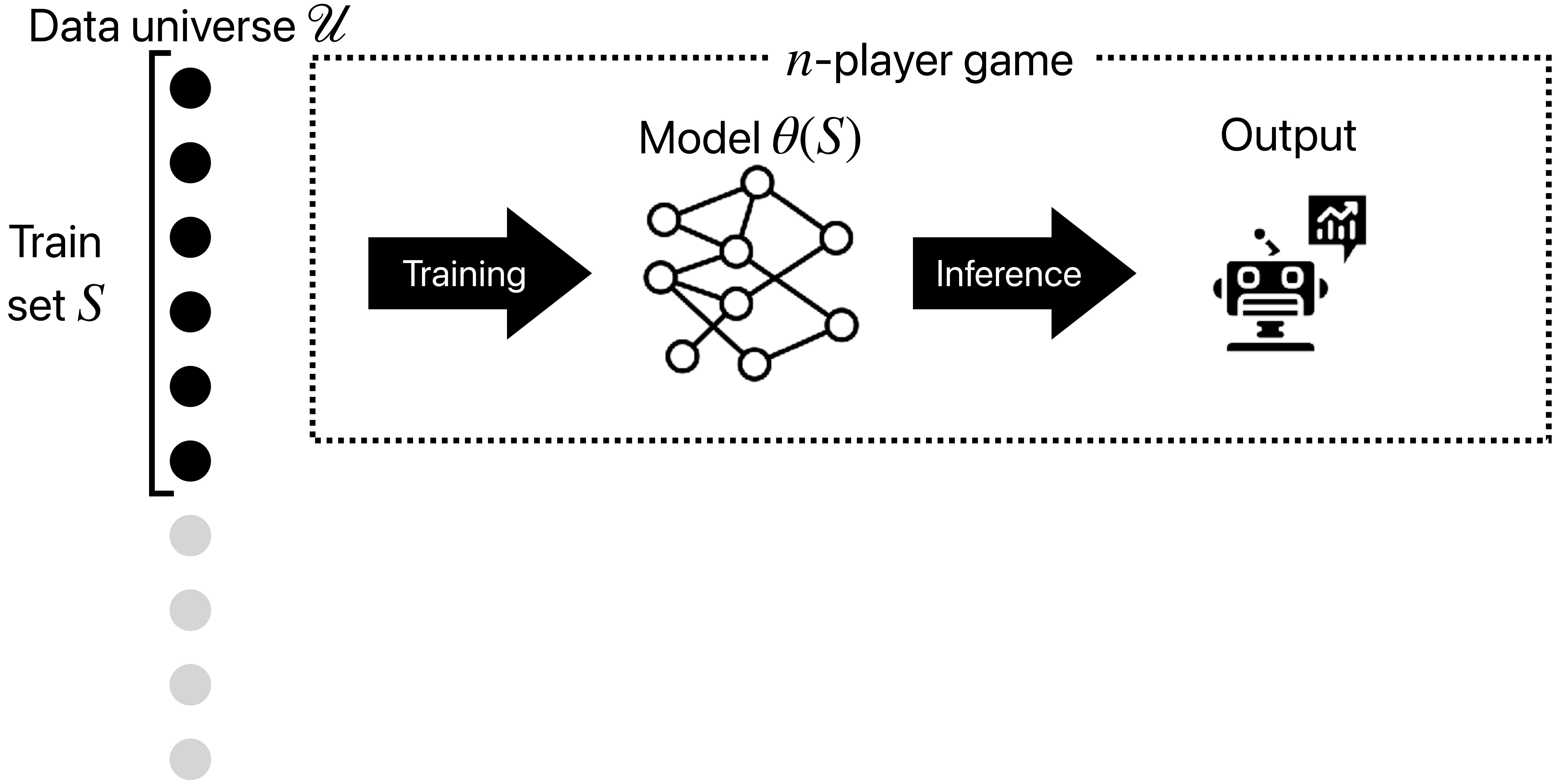
Game-theoretic attribution (Credit assignment)

Illustration and examples



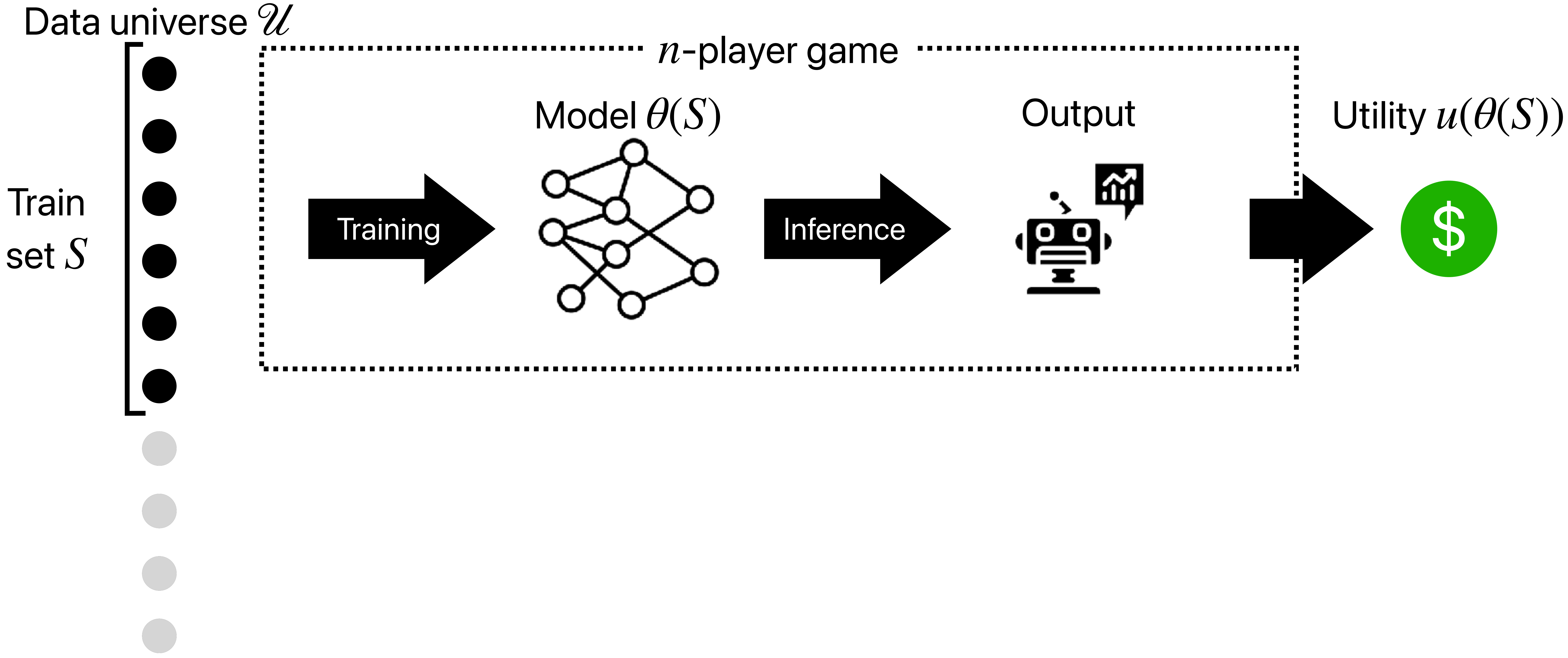
Game-theoretic attribution (Credit assignment)

Illustration and examples



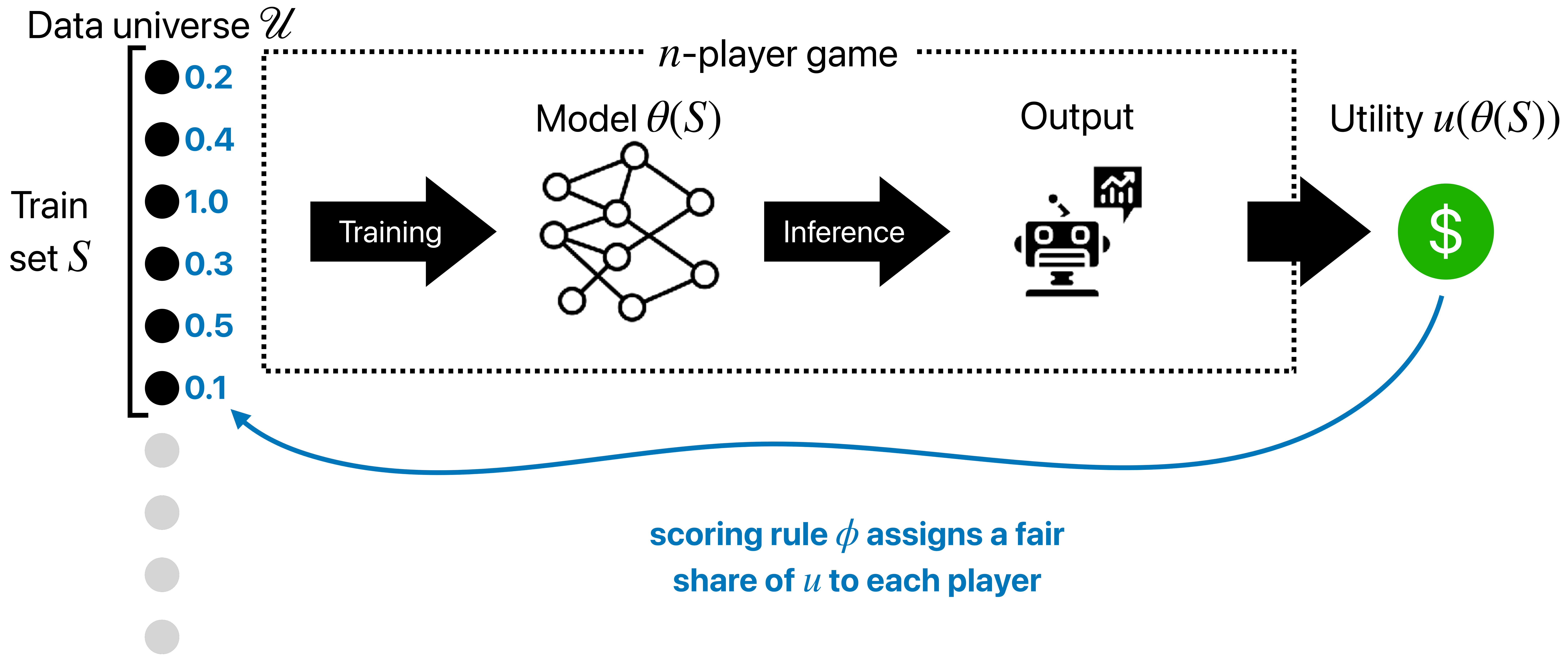
Game-theoretic attribution (Credit assignment)

Illustration and examples



Game-theoretic attribution (Credit assignment)

Illustration and examples



Game-theoretic attribution (Credit assignment)

Illustration and examples

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou ’19; Jia Dao Wang et al. ’19]

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Every possible subset not including the j-th example

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Every possible subset not including the j-th example

Marginal effect on utility of adding the example to the subset

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Every possible subset not including the j-th example

Marginal effect on utility of adding the example to the subset

Uniquely satisfies:

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Every possible subset not including the j-th example

Marginal effect on utility of adding the example to the subset

Uniquely satisfies: ✅ Symmetry

(identical data points receive identical credit)

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Every possible subset not including the j-th example

Marginal effect on utility of adding the example to the subset

Uniquely satisfies: ✅ Symmetry ✅ Linearity

(identical data points receive identical credit)

(attributing linear combo of utilities equivalent to linear combo of attributions)

Game-theoretic attribution (Credit assignment)

Illustration and examples

Example method: Data Shapley [Ghorbani Zou '19; Jia Dao Wang et al. '19]

$$v_{Shapley}(z_i) = \sum_{S' \subset S \setminus \{z_i\}} \gamma(|S'|) \cdot [u(S' \cup \{z_i\}) - u(S')]$$

Credit assigned to the j-th example

Every possible subset not including the j-th example

Marginal effect on utility of adding the example to the subset

Uniquely satisfies: ✅ Symmetry ✅ Linearity ✅ Null-sensitivity

- Symmetry: (identical data points receive identical credit)
- Linearity: (attributing linear combo of utilities equivalent to linear combo of attributions)
- Null-sensitivity: (Always-useless data points get no credit)

Predictive attribution (Datamodeling)

Motivation [Koh Liang ’17; Ilyas Park Engstrom Leclerc Madry ’22]

Predictive attribution (Datamodeling)

Motivation [Koh Liang ’17; Ilyas Park Engstrom Leclerc Madry ’22]

In some cases, we still care about causality, but not fair credit assignment

Predictive data attribution (or **datamodeling**) answers questions of the form:

> *If instead of training on my training set S, I instead trained on a different training set S', how would my model’s behavior change?*

Predictive attribution (Datamodeling)

Relation to data attribution

*A data attribution method seeks to **connect** model **behavior** at test time **with data**.*

Predictive attribution (Datamodeling)

Relation to data attribution

predictive

*A ^ data attribution method seeks to **connect** model **behavior** at test time **with data**.*

Predictive attribution (Datamodeling)

Relation to data attribution

predictive

A ^ data attribution method ***aims to predict*** '

model ***behavior*** *at test time* ***with data***.

Predictive attribution (Datamodeling)

Relation to data attribution

predictive

A ^ data attribution method ***aims to predict*** *'*

model ***outputs*** *at test time* ***with data****.*

Predictive attribution (Datamodeling)

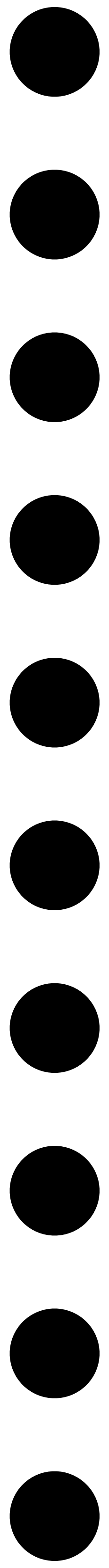
Relation to data attribution

predictive
A ^ data attribution method ***aims to predict*** '
model ***outputs*** *at test time* ***as a function of data***.

Predictive attribution (Datamodeling)

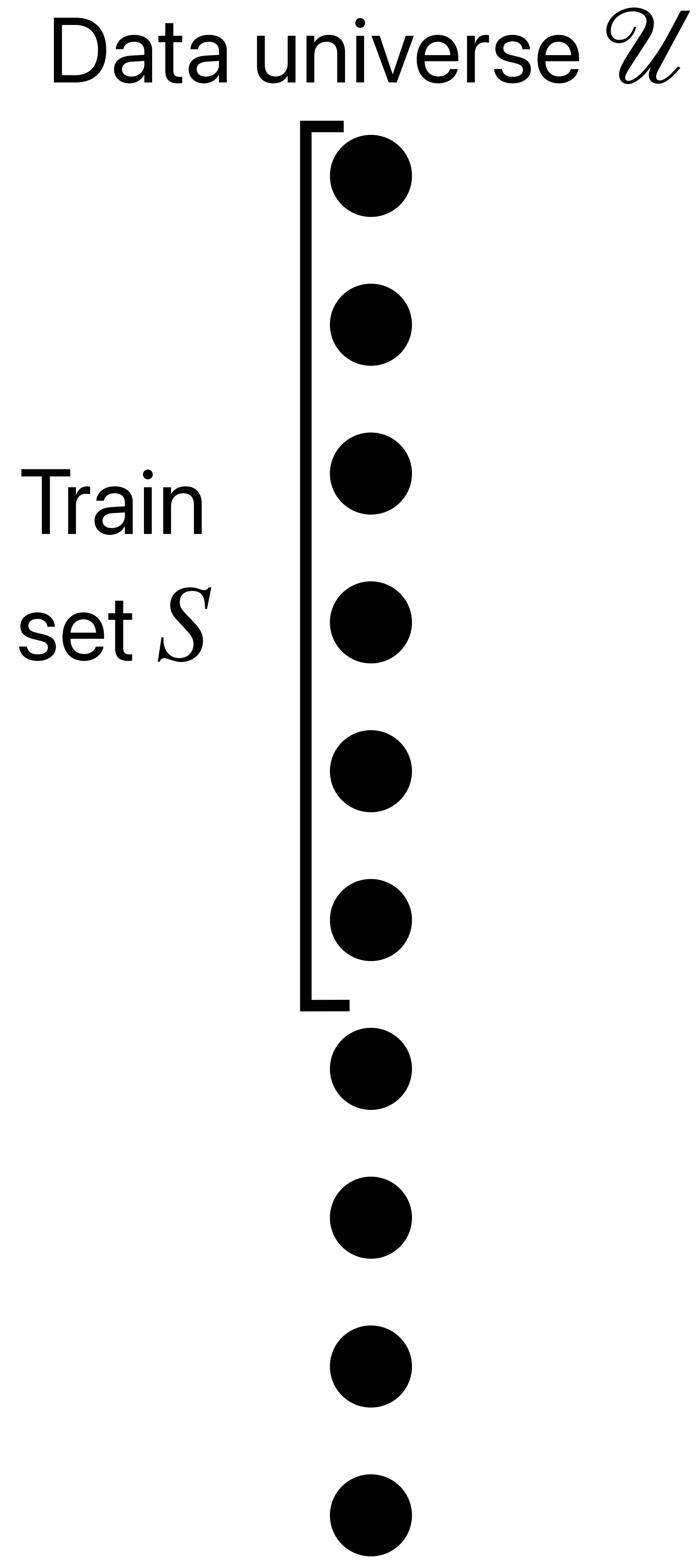
Illustration and examples

Data universe $\mathscr{U}$



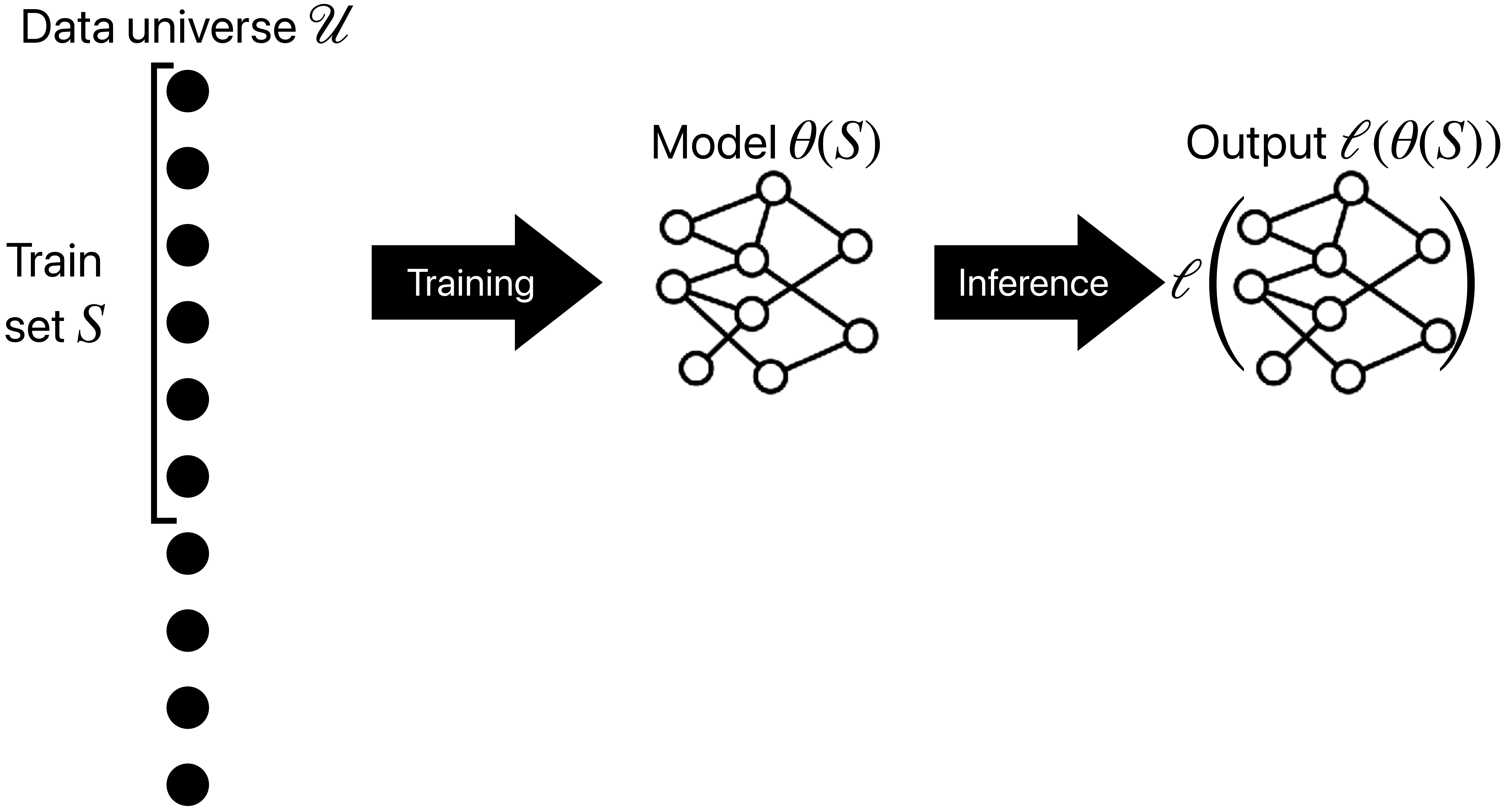
Predictive attribution (Datamodeling)

Illustration and examples



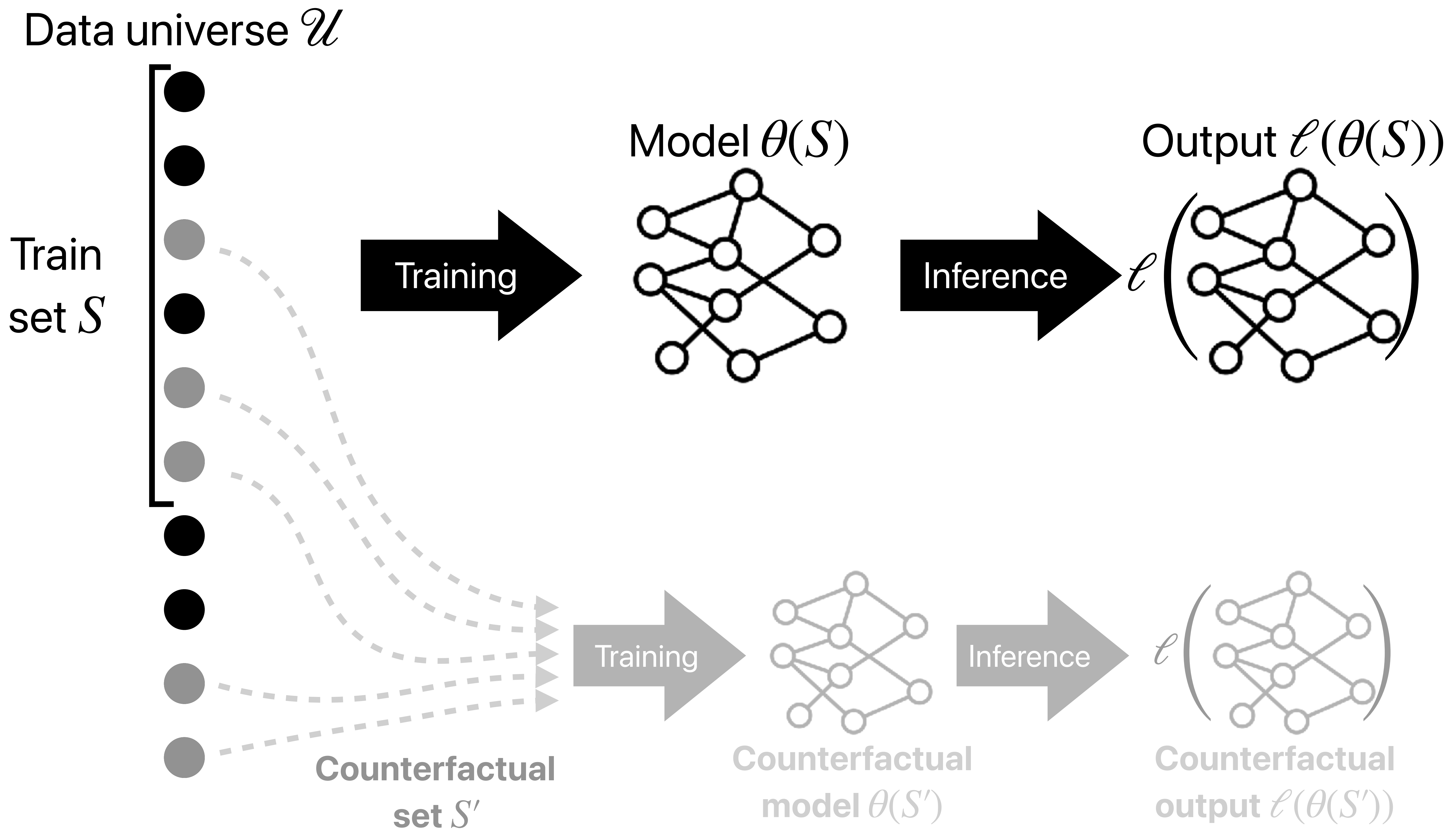
Predictive attribution (Datamodeling)

Illustration and examples



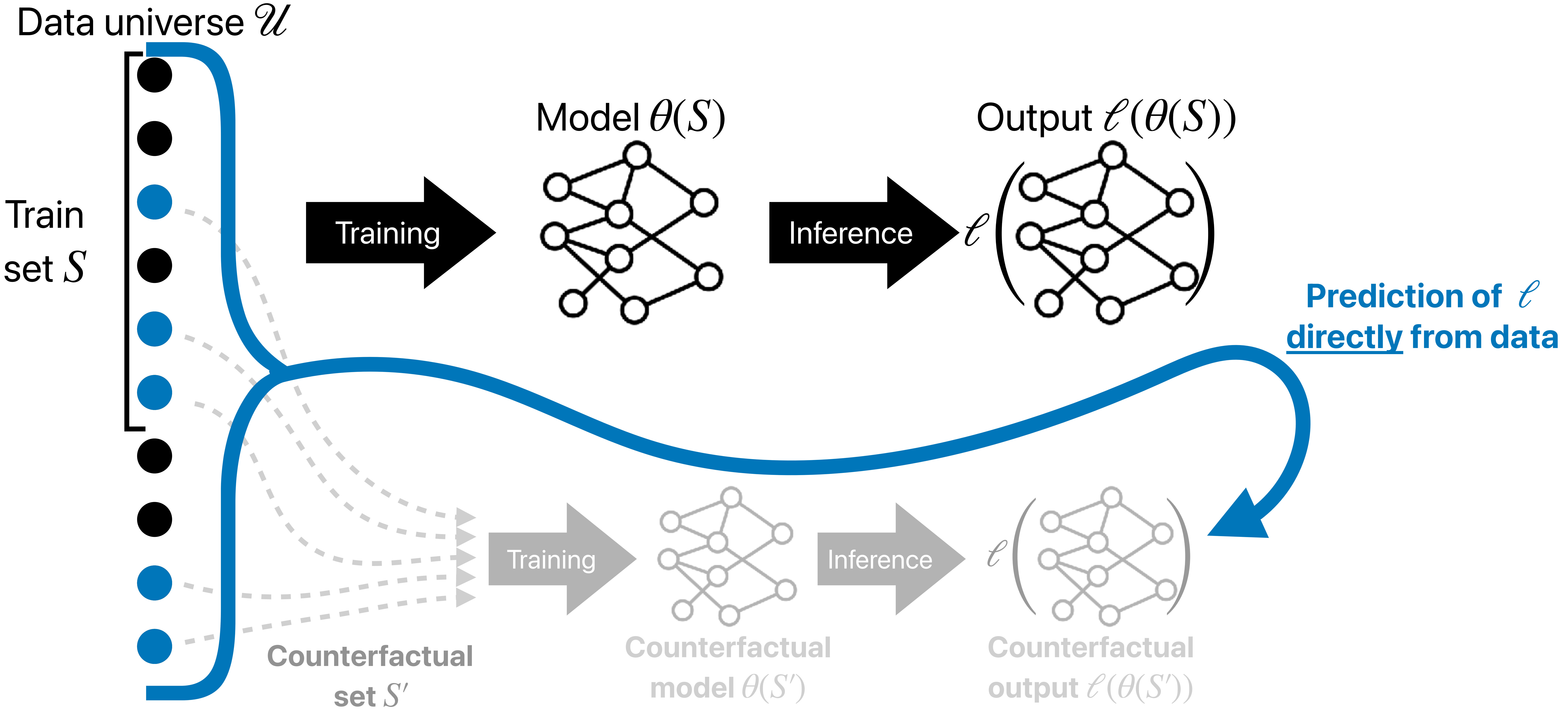
Predictive attribution (Datamodeling)

Illustration and examples



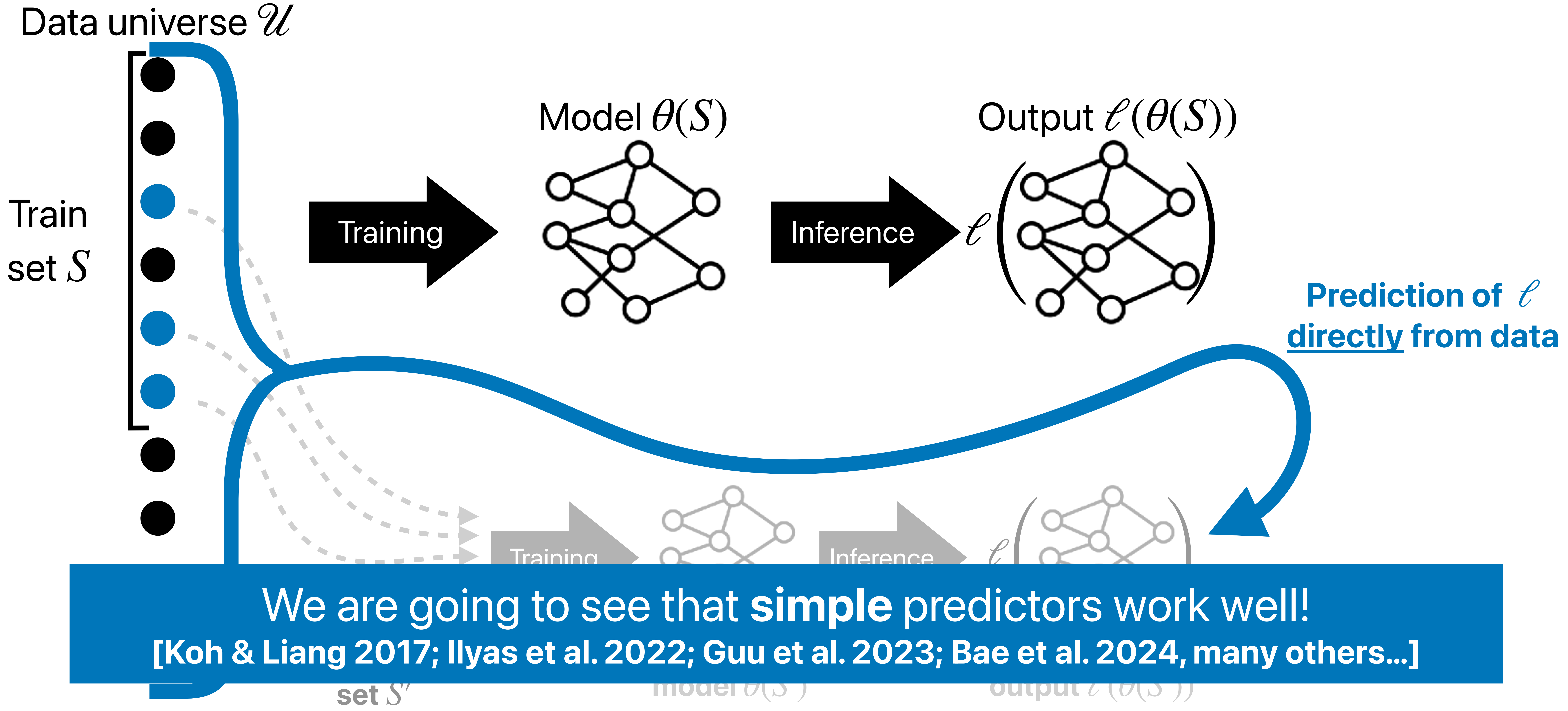
Predictive attribution (Datamodeling)

Illustration and examples



Predictive attribution (Datamodeling)

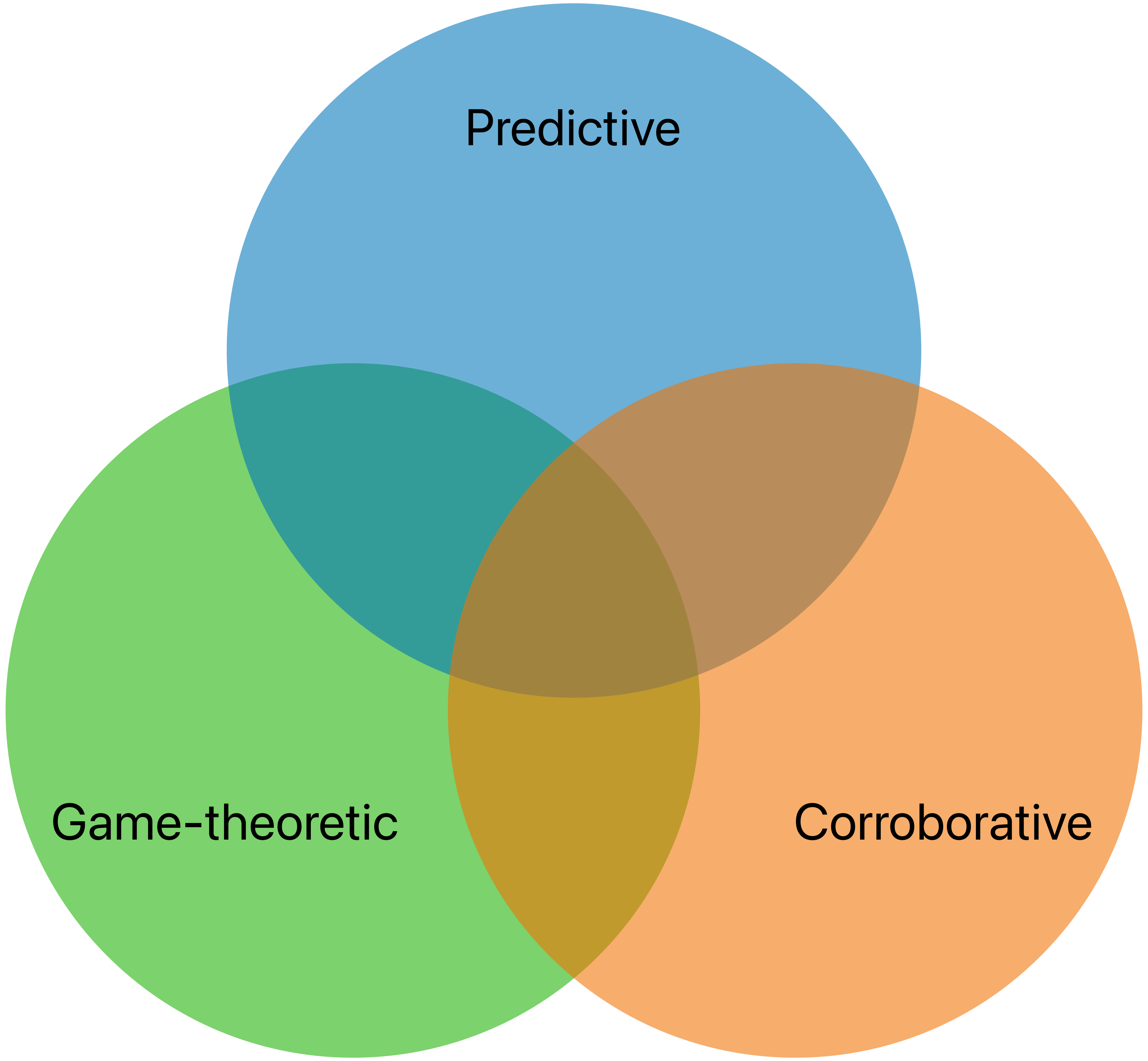
Illustration and examples



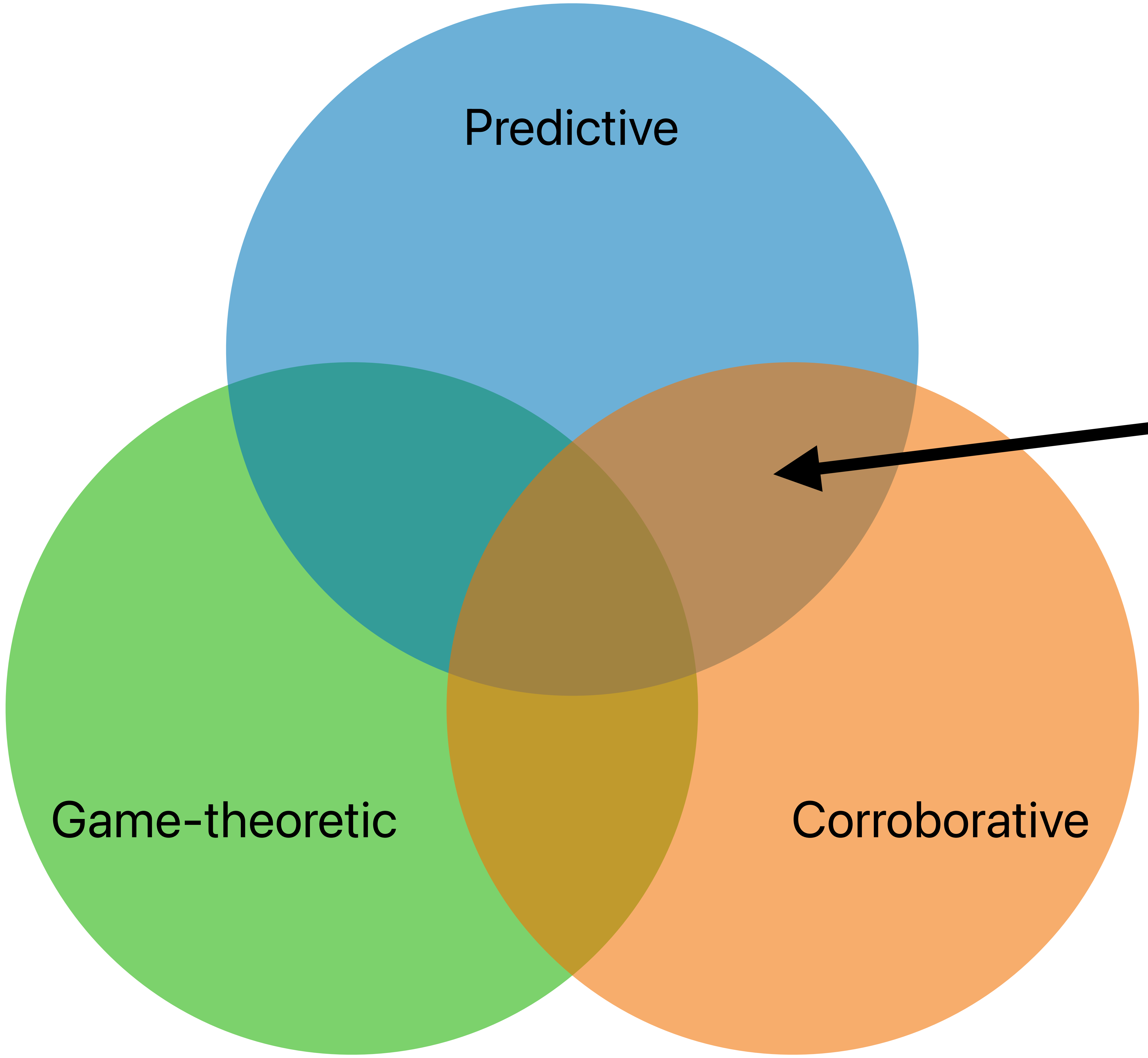
We are going to see that **simple** predictors work well!

[Koh & Liang 2017; Ilyas et al. 2022; Guu et al. 2023; Bae et al. 2024, many others…]

Comparing and contrasting perspectives

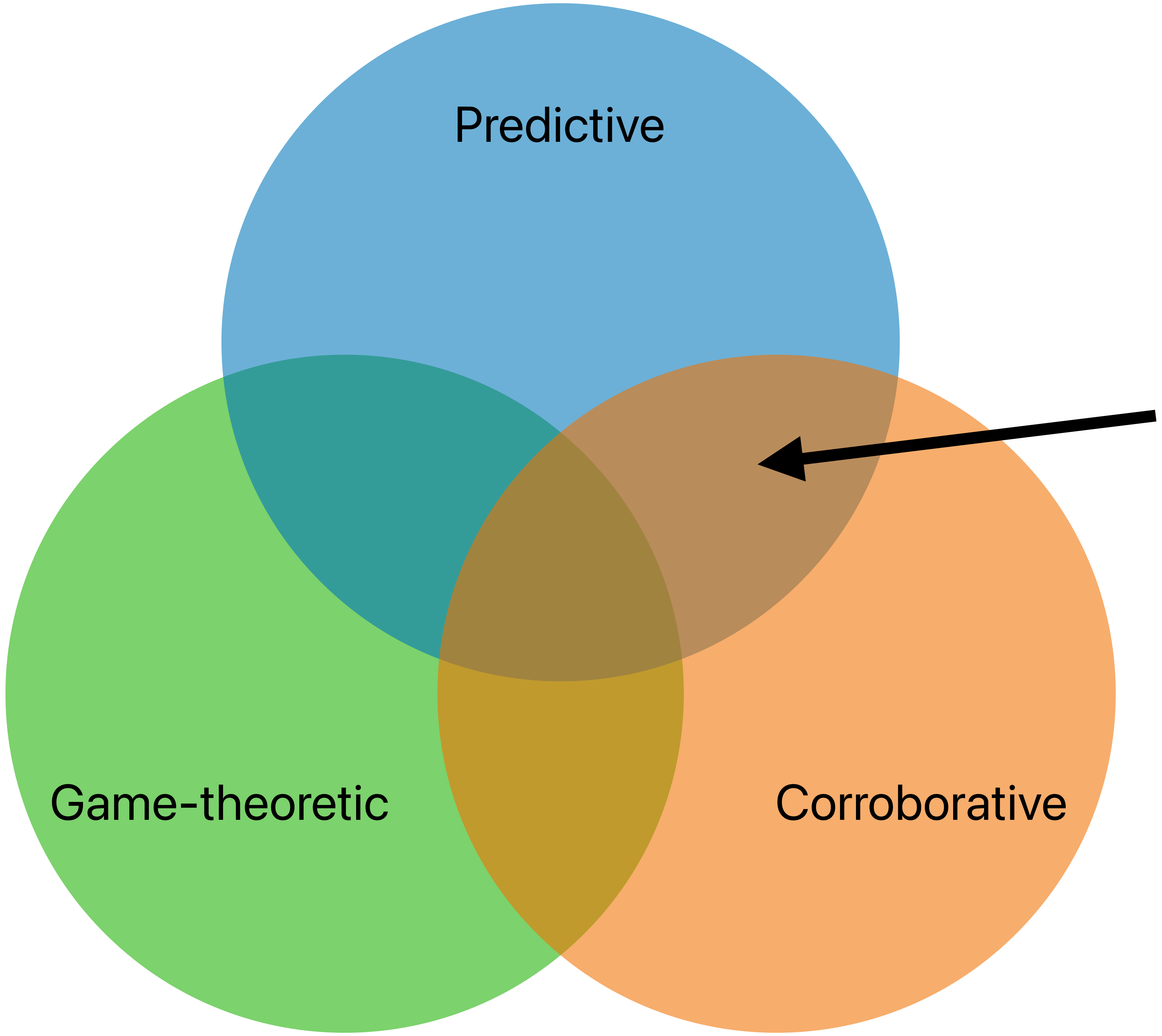


Comparing and contrasting perspectives



ML algorithm learns rules that align with similarity metric

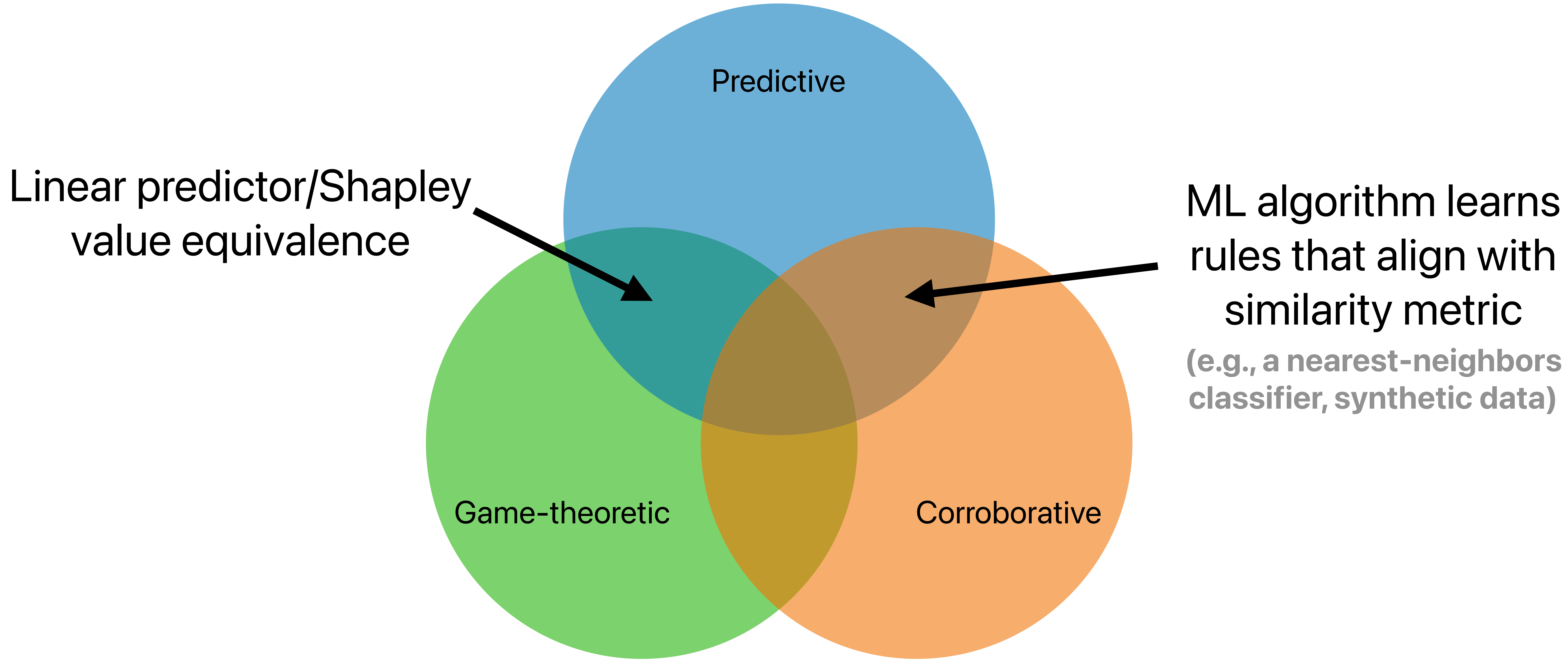
Comparing and contrasting perspectives



ML algorithm learns rules that align with similarity metric

(e.g., a nearest-neighbors classifier, synthetic data)

Comparing and contrasting perspectives

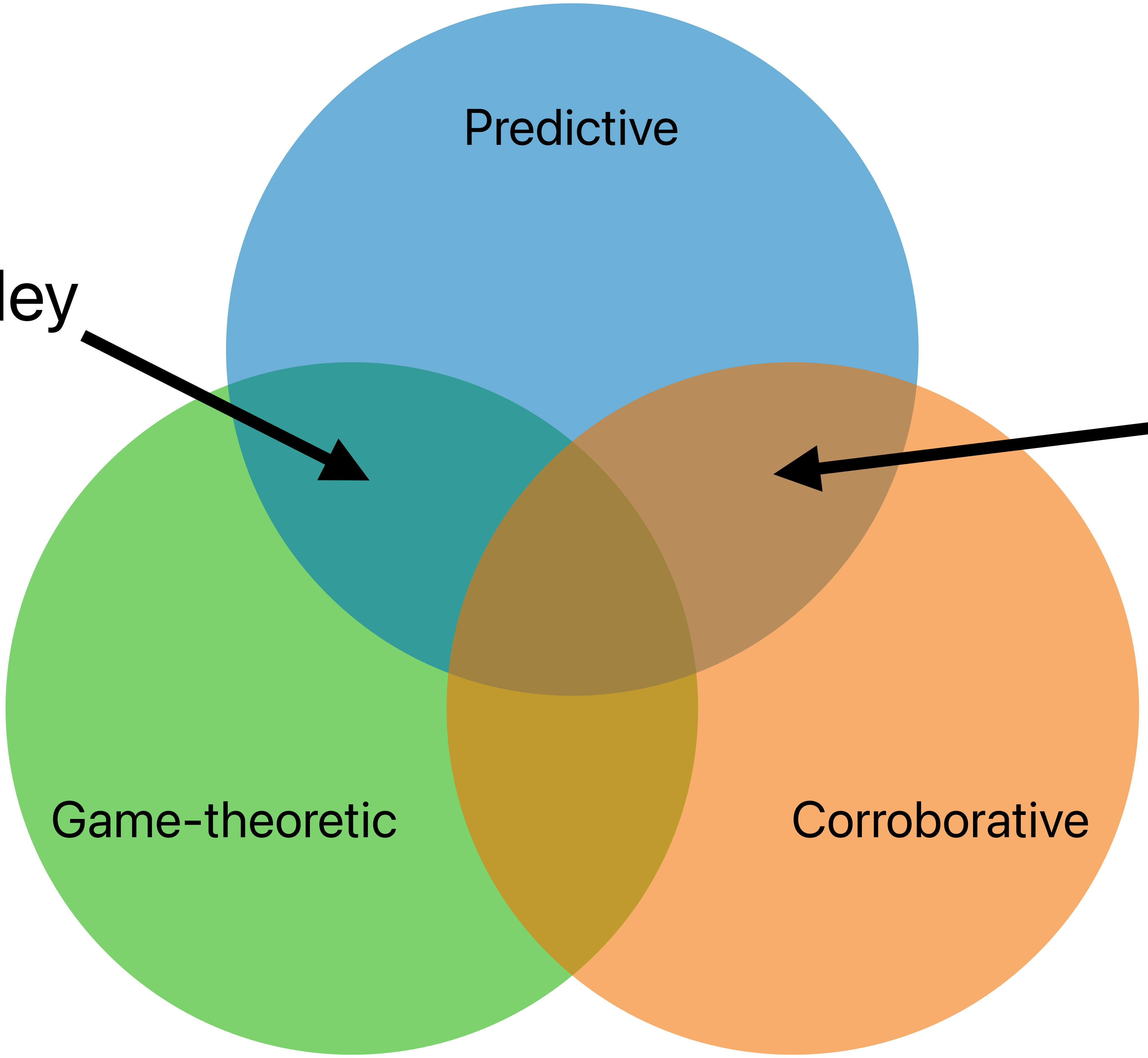


Comparing and contrasting perspectives

Linear predictor/Shapley value equivalence

A vector of Shapley values can serve as coefficients of a *linear* datamodel:

$$\hat{f}(S) = \sum_{z_i \in S} v_{Shapley}(z_i)$$



ML algorithm learns rules that align with similarity metric

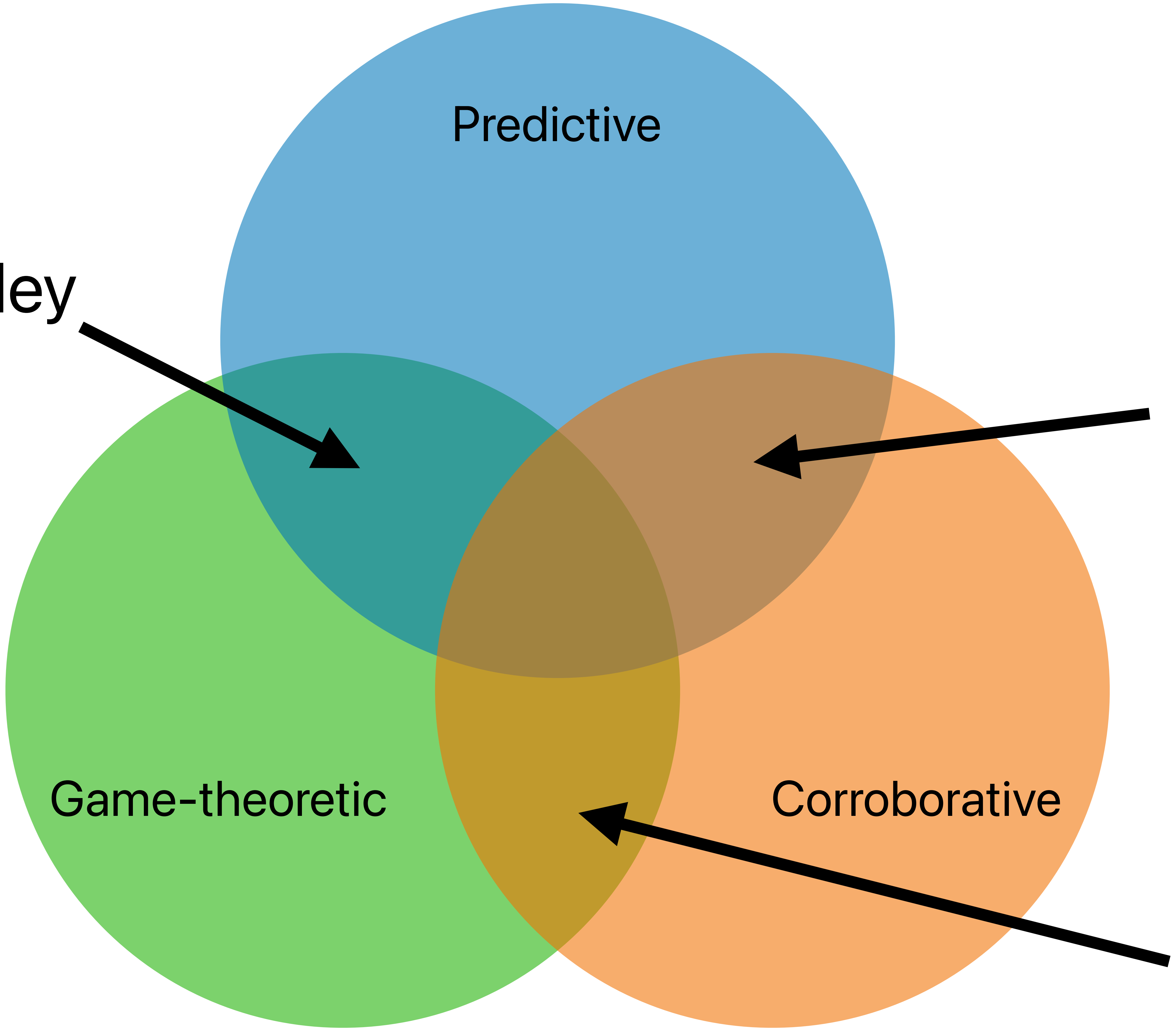
(e.g., a nearest-neighbors classifier, synthetic data)

Comparing and contrasting perspectives

Linear predictor/Shapley value equivalence

A vector of Shapley values can serve as coefficients of a *linear* datamodel:

$$\hat{f}(S) = \sum_{z_i \in S} v_{Shapley}(z_i)$$



ML algorithm learns rules that align with similarity metric

(e.g., a nearest-neighbors classifier, synthetic data)

?

Our focus: predictive data attribution

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

In this talk, we'll focus on *predictive* data attribution (i.e., datamodeling)

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

In this talk, we'll focus on *predictive* data attribution (i.e., datamodeling)

Rich history in statistics and machine learning

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

In this talk, we'll focus on *predictive* data attribution (i.e., datamodeling)

- Rich history in statistics and machine learning
- Relevance to modern problems (e.g., data selection for LLMs)

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

In this talk, we'll focus on *predictive* data attribution (i.e., datamodeling)

- Rich history in statistics and machine learning
- Relevance to modern problems (e.g., data selection for LLMs)
- Rapid progress in the last few years

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

In this talk, we'll focus on *predictive* data attribution (i.e., datamodeling)

- Rich history in statistics and machine learning
- Relevance to modern problems (e.g., data selection for LLMs)
- Rapid progress in the last few years
- **Lots** of room for improvement

Our focus: predictive data attribution

Depending on what your goal is, you may want a different notion of attribution

In this talk, we'll focus on *predictive* data attribution (i.e., datamodeling)

- Rich history in statistics and machine learning
- Relevance to modern problems (e.g., data selection for LLMs)
- Rapid progress in the last few years
- **Lots** of room for improvement

Let's begin!

Part II: Theoretical foundations

Predictive data attribution/Datamodels

Chapter 2 of 4

Problem setup

Problem setup

Setting: average loss minimization problem (also called M-estimation)

Problem setup

Setting: average loss minimization problem (also called M-estimation)

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Problem setup

Setting: average loss minimization problem (also called M-estimation)

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Loss of model θ on training sample i (strongly convex)

Problem setup

Setting: average loss minimization problem (also called M-estimation)

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Vector of n weights

Loss of model θ on training sample i (strongly convex)

Problem setup

Setting: average loss minimization problem (also called M-estimation)

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Vector of n weights

Weight on i-th training sample

Loss of model θ on training sample i (strongly convex)

Problem setup

Setting: average loss minimization problem (also called M-estimation)

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Vector of n weights

Weight on i-th training sample

Loss of model θ on training sample i (strongly convex)

Goal: predict $\theta^*(w)$ for $w \neq \mathbf{1}_n$

Problem setup

Setting: average loss minimization problem (also called M-estimation)

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Vector of n weights

Weight on i-th training sample

Loss of model θ on training sample i (strongly convex)

Goal: predict $\theta^*(w)$ for $w \neq \mathbf{1}_n$

Special case: $LOO(j) = \theta^*(\mathbf{1}_n) - \theta^*(\mathbf{1}_n - \text{Ind}_j) = \arg\min_{\theta} \sum_{i \neq j} \ell_i(\theta)$

History

History

Statistical analog of predictive data attribution

- A couple of differences: target is parameter space, dataset is known, assume unique minimizer/strong convexity)

History

Statistical analog of predictive data attribution

A couple of differences: target is parameter space, dataset is known, assume unique minimizer/strong convexity)

Variety of applications across statistics

Cross-validation [Stephenson et al. ’20; Wilson et al. ’20; Rad & Maleki ‘18], uncertainty estimation [Vovk et al. ’99; Giordano et al. ’23], many others

History

Statistical analog of predictive data attribution

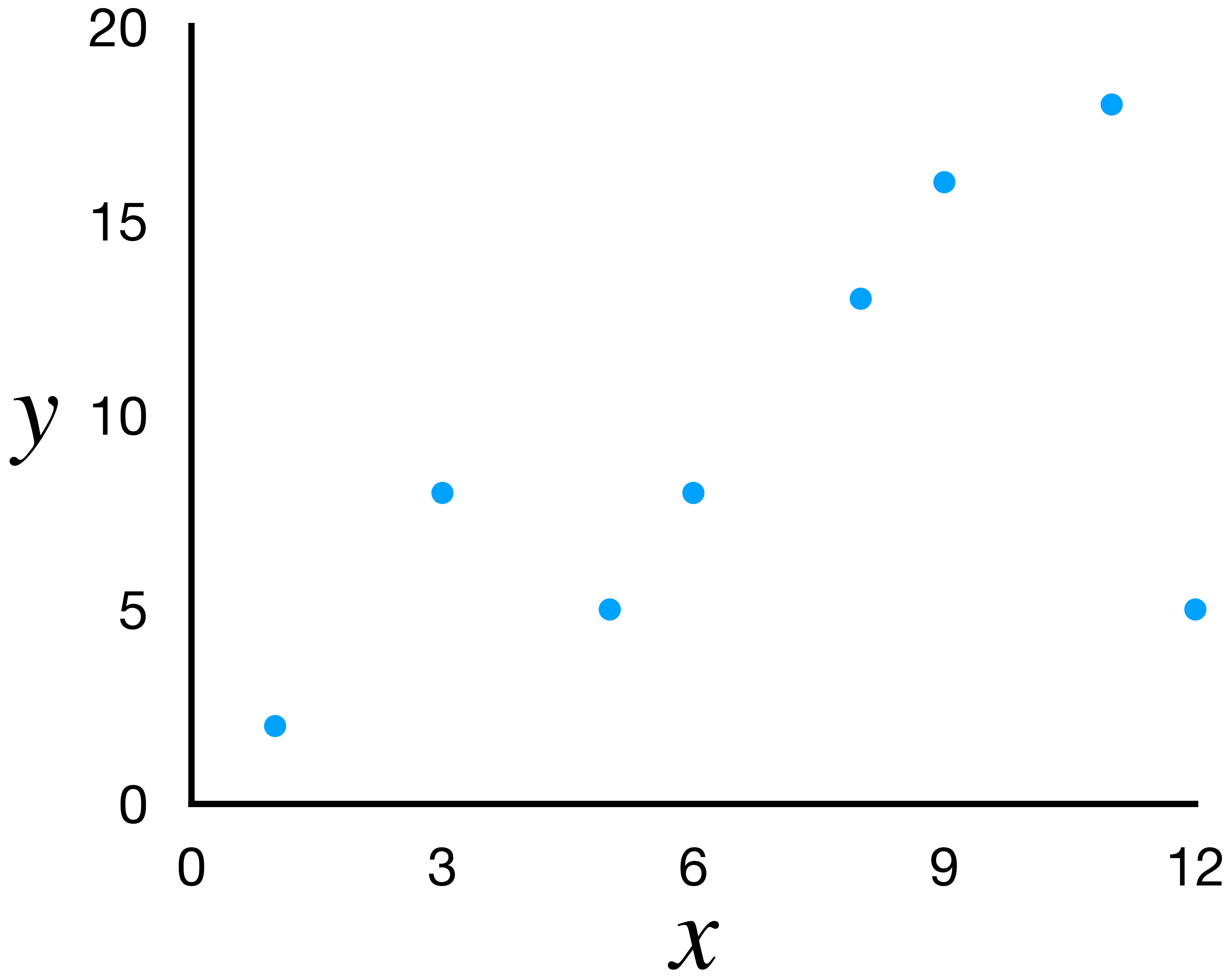
A couple of differences: target is parameter space, dataset is known, assume unique minimizer/strong convexity)

Variety of applications across statistics

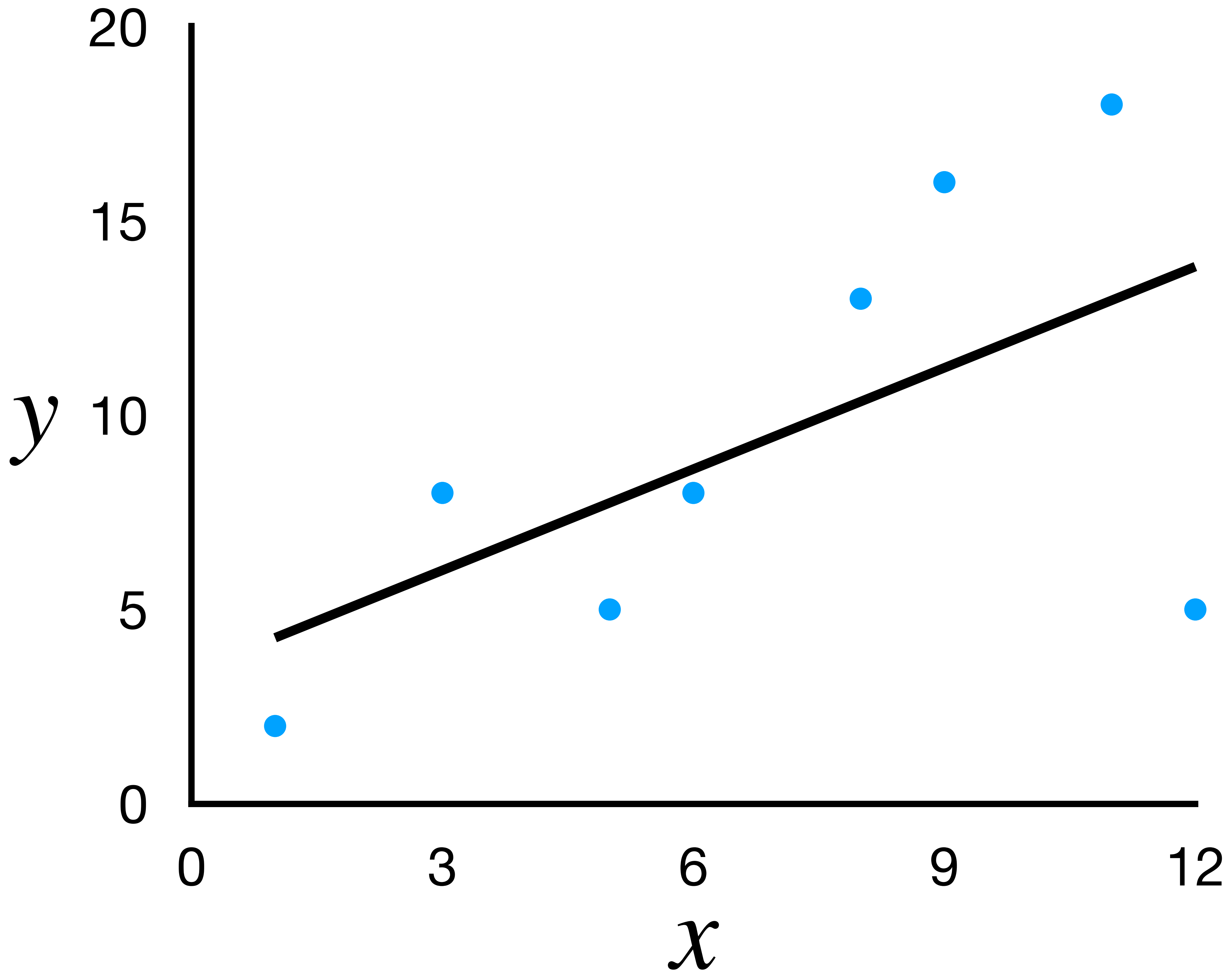
Cross-validation [Stephenson et al. '20; Wilson et al. '20; Rad & Maleki '18], uncertainty estimation [Vovk et al. '99; Giordano et al. '23], many others

Key words: von Mises calculus [von Mises '47]; infinitesimal jackknife [Jaeckel '72]; influence functions/influence curve [Hampel '74]; regression analysis [Pregibon '81]

Warmup: linear regression, leave-one-out

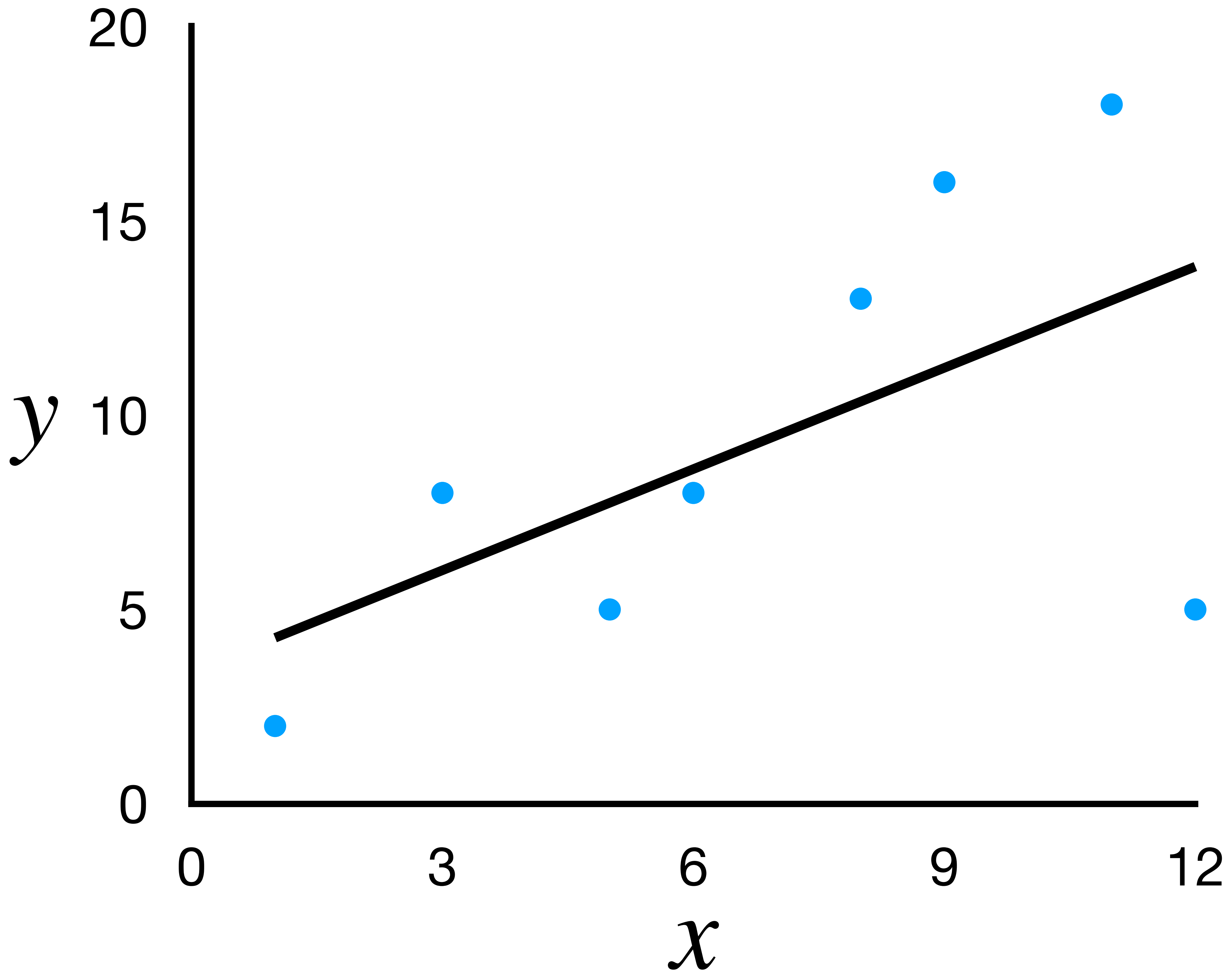


Warmup: linear regression, leave-one-out



Warmup: linear regression, leave-one-out

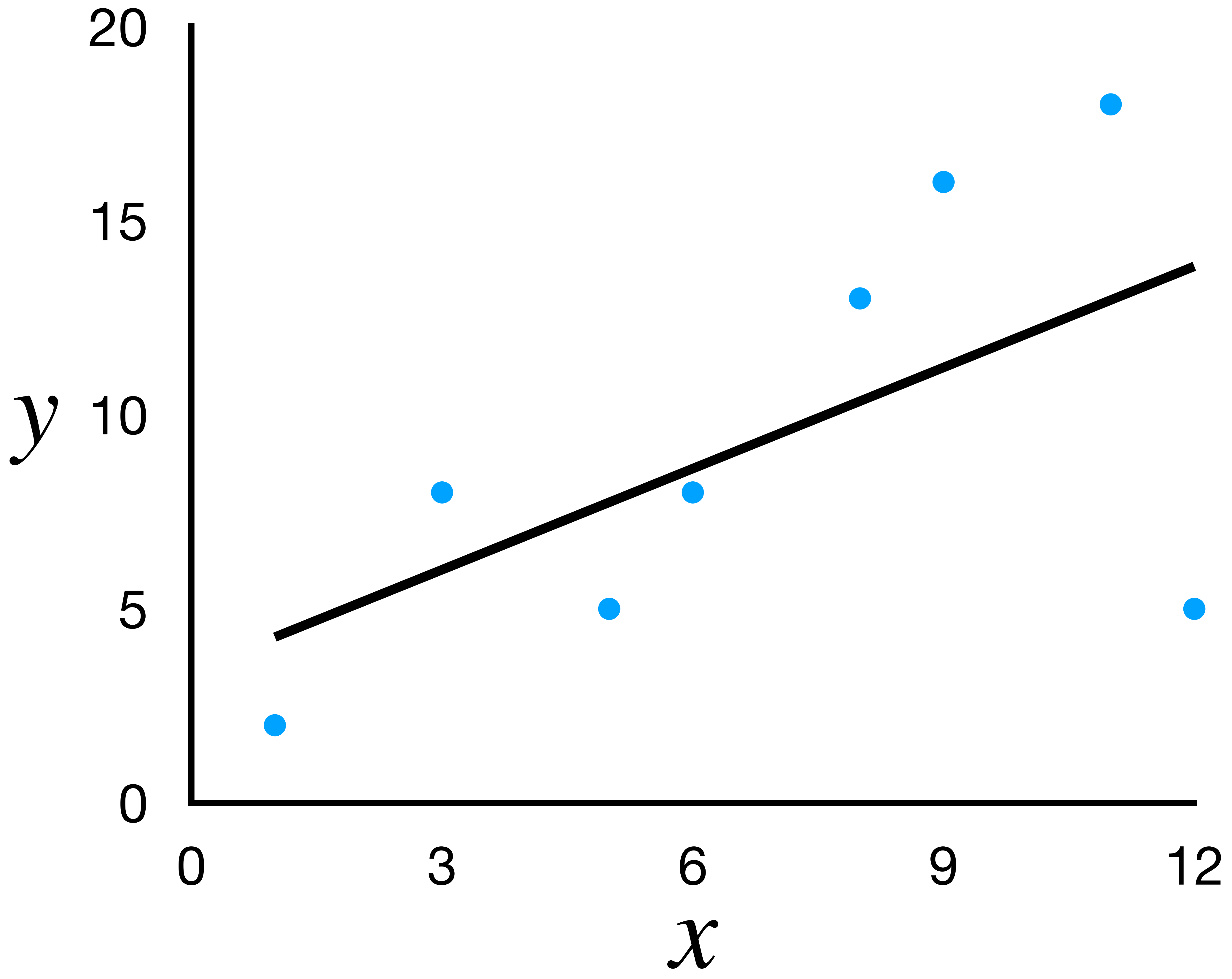
Setup:



Warmup: linear regression, leave-one-out

Setup:

Dataset $(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$

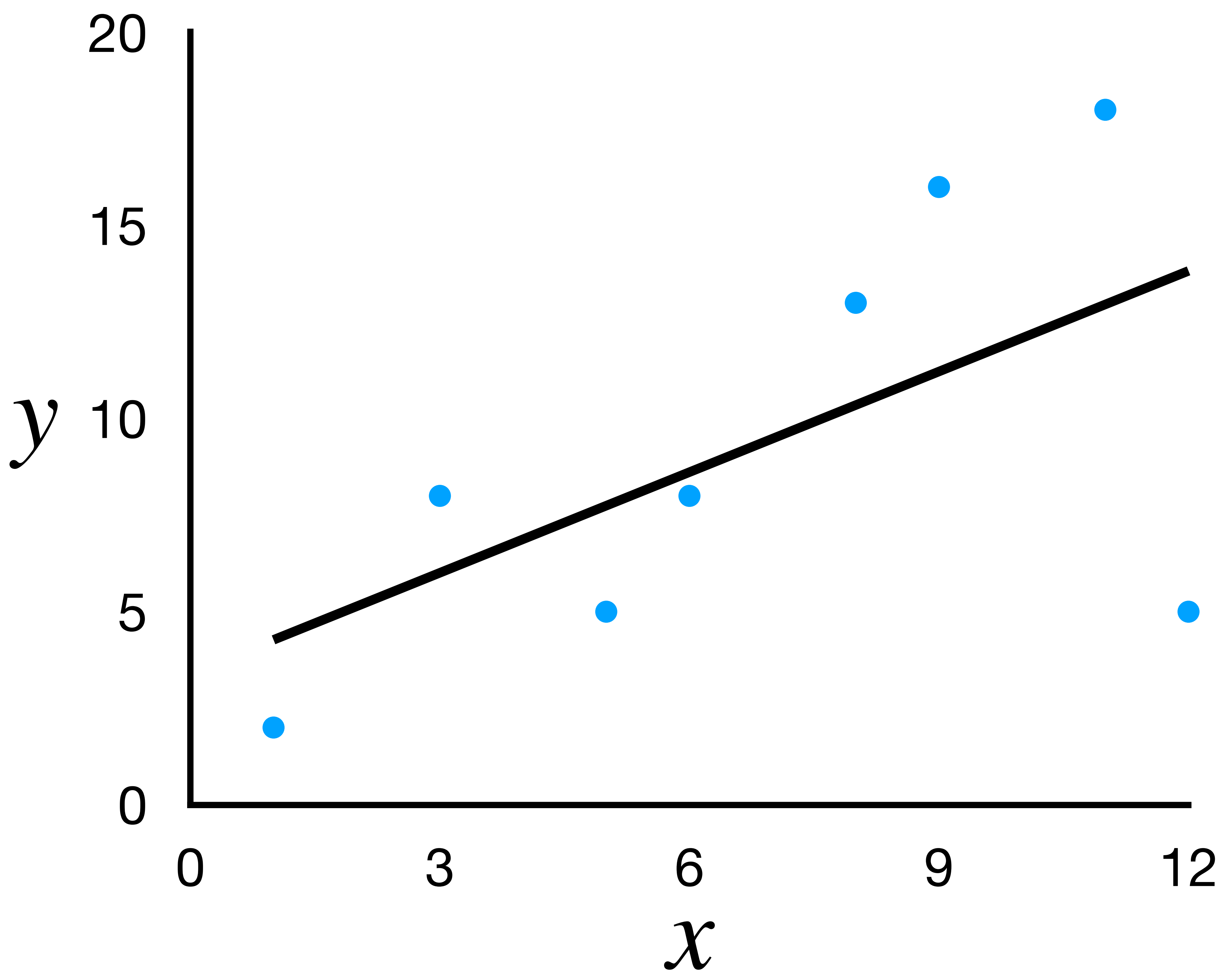


Warmup: linear regression, leave-one-out

Setup:

Dataset $(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$

Least-squares fit:



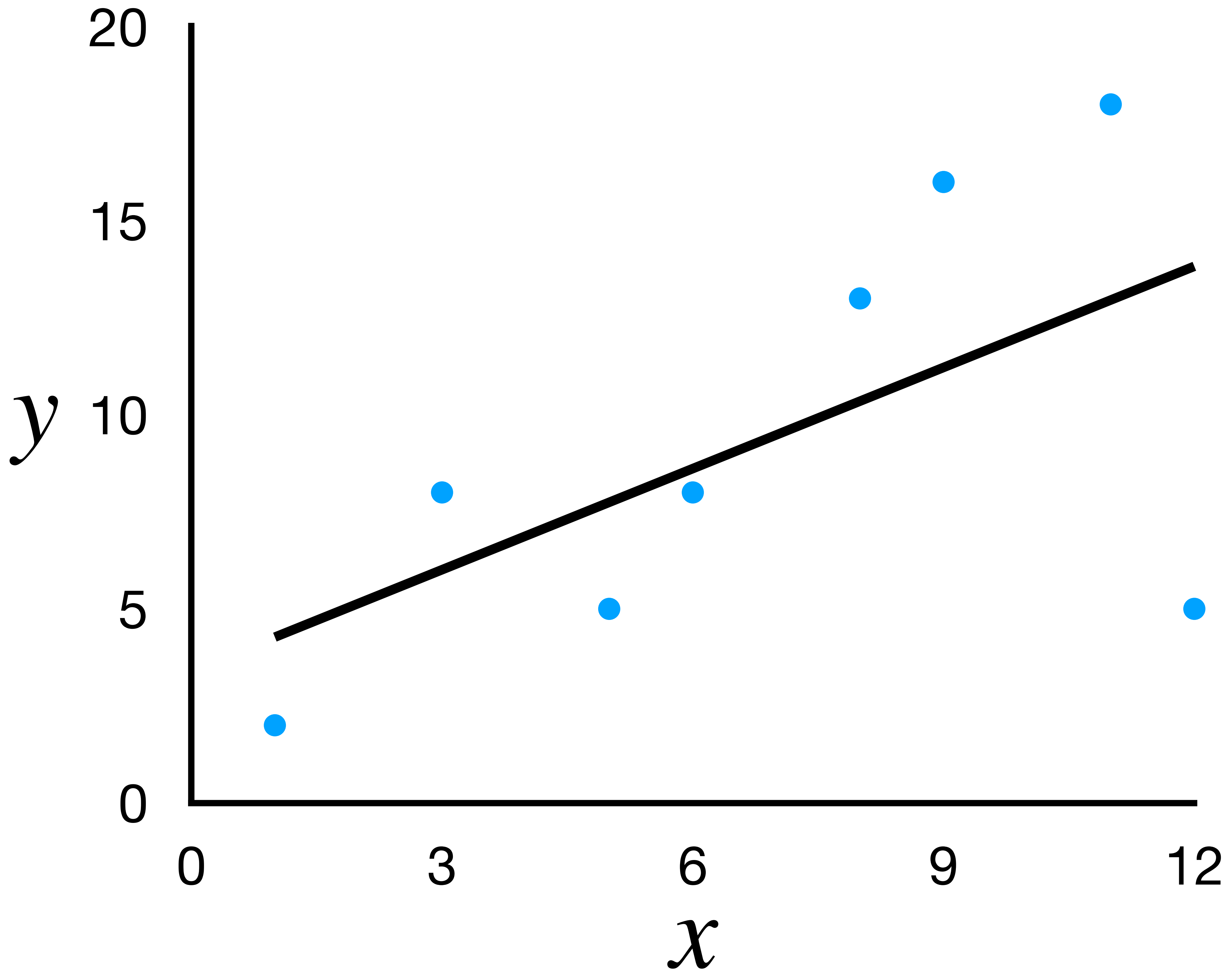
Warmup: linear regression, leave-one-out

Setup:

Dataset $(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$

Least-squares fit:

$$\theta^*(w) = \sum_{i=1}^{n} w_i \cdot (\theta^\top x_i - y_i)^2$$



Warmup: linear regression, leave-one-out

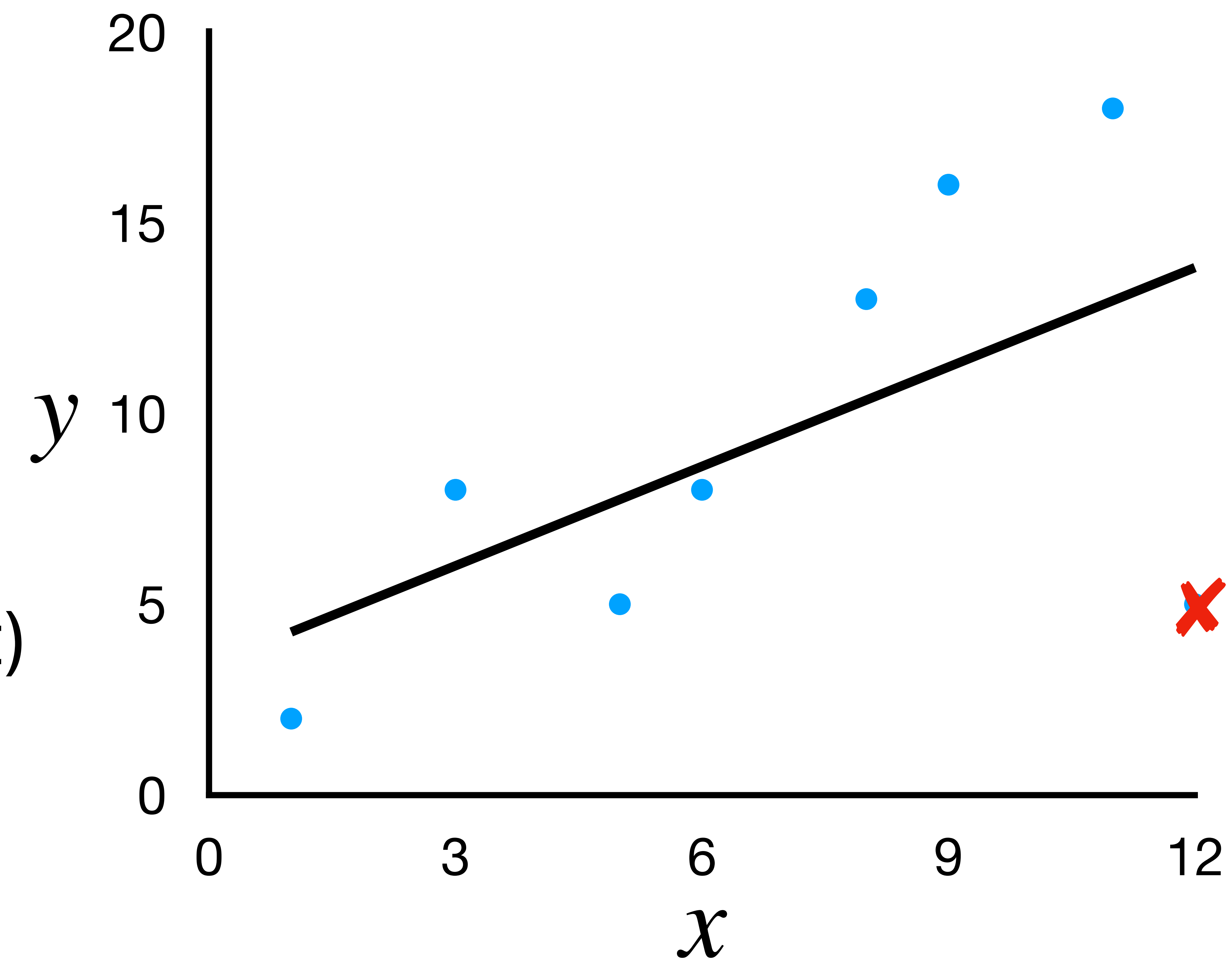
Setup:

Dataset $(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$

Least-squares fit:

$$\theta^*(w) = \sum_{i=1}^{n} w_i \cdot (\theta^\top x_i - y_i)^2$$

Target: $LOO(j)$ (leave-one-out)



Warmup: linear regression, leave-one-out

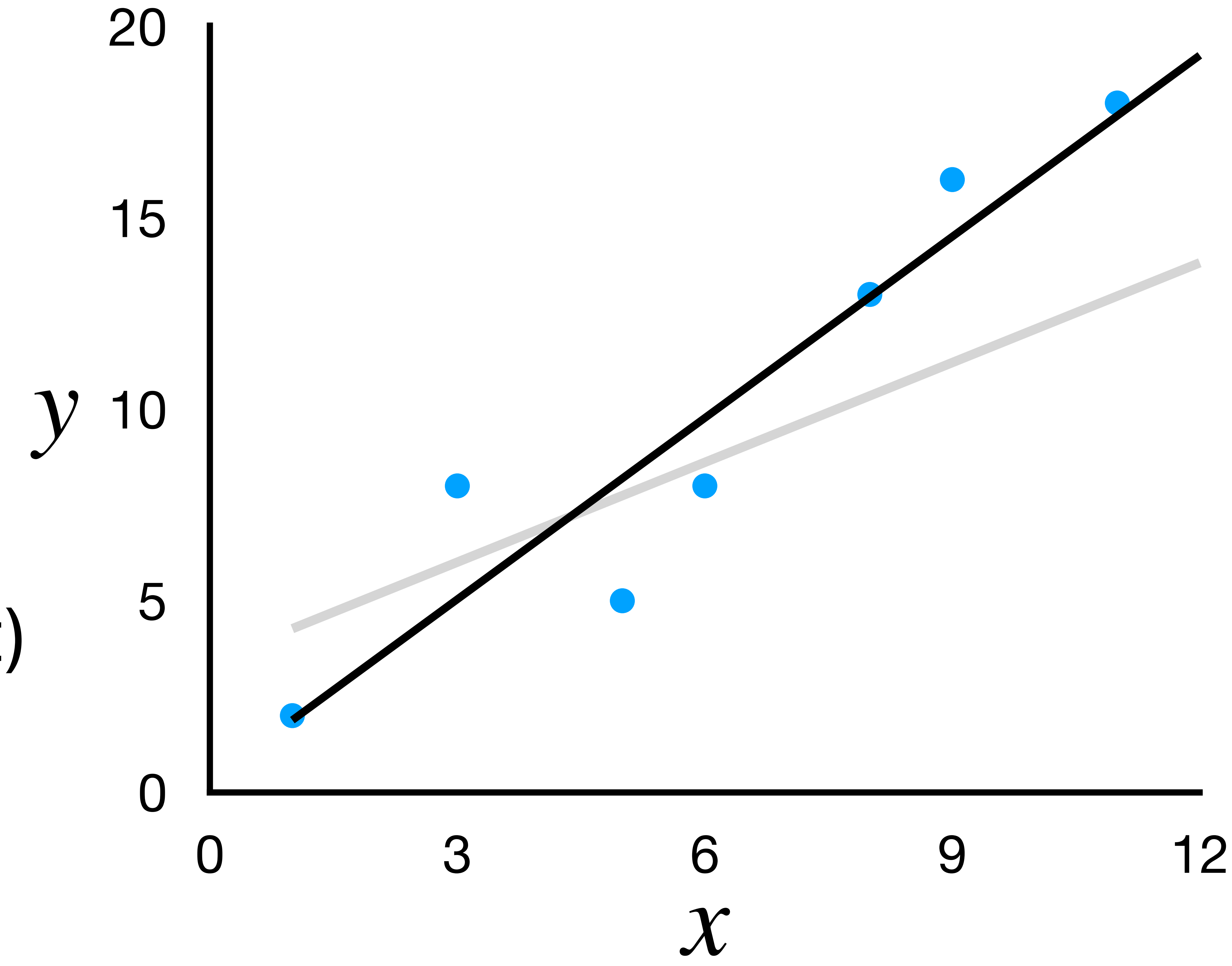
Setup:

Dataset $(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$

Least-squares fit:

$$\theta^*(w) = \sum_{i=1}^{n} w_i \cdot (\theta^\top x_i - y_i)^2$$

Target: $LOO(j)$ (leave-one-out)



Warmup: linear regression, leave-one-out

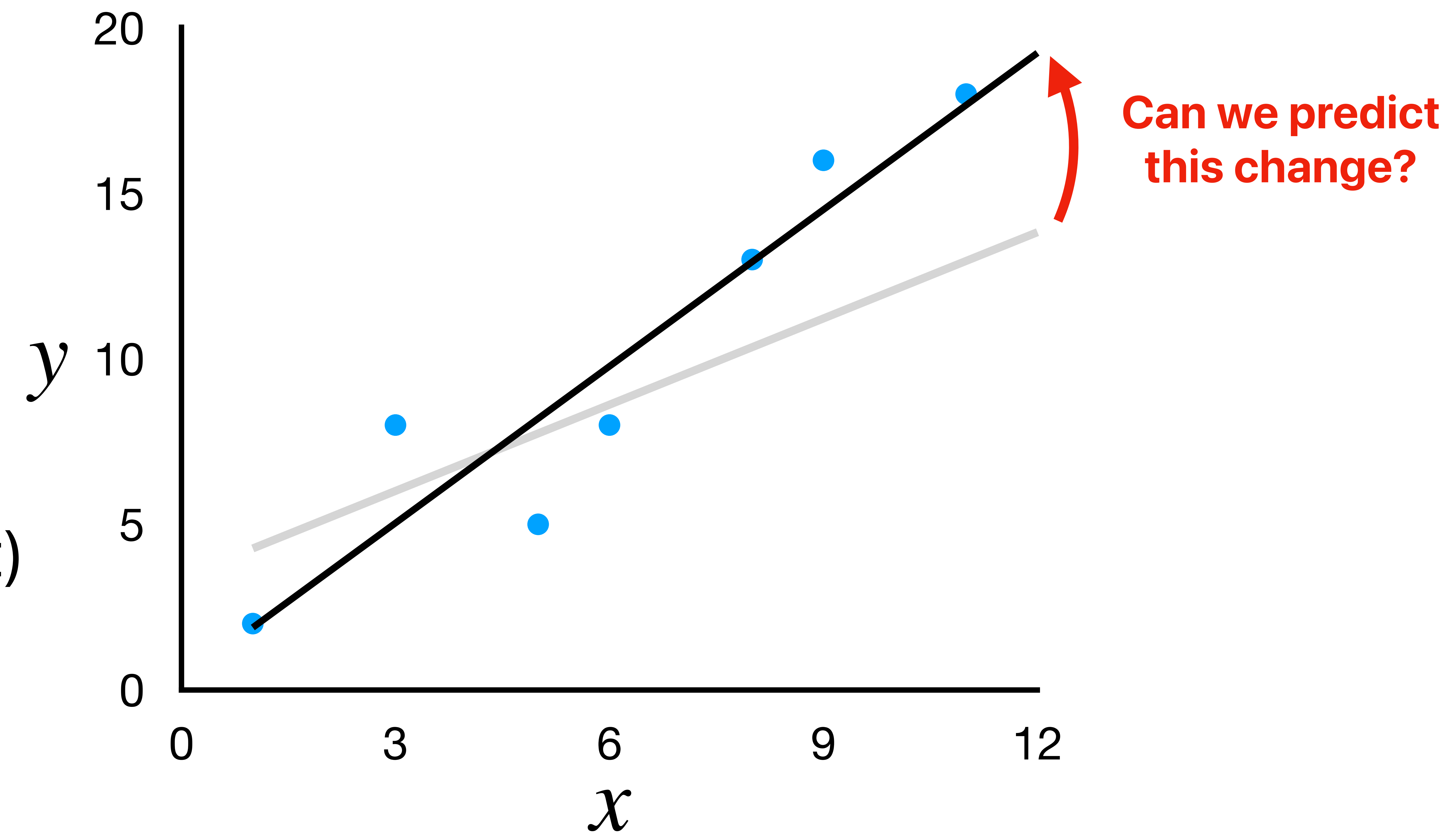
Setup:

Dataset $(x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$

Least-squares fit:

$$\theta^*(w) = \sum_{i=1}^{n} w_i \cdot (\theta^\top x_i - y_i)^2$$

Target: $LOO(j)$ (leave-one-out)



Warmup: linear regression, leave-one-out

Warmup: linear regression, leave-one-out

This is easy! Linear regression has a closed-form solution:

Warmup: linear regression, leave-one-out

This is easy! Linear regression has a closed-form solution:

$$\theta^*(w) = \left(\sum_{i=1}^{n} w_i \cdot \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum_{i=1}^{n} w_i y_i \cdot \mathbf{x_i} \right)$$

Warmup: linear regression, leave-one-out

This is easy! Linear regression has a closed-form solution:

$$\theta^*(w) = \left(\sum_{i=1}^{n} w_i \cdot \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum_{i=1}^{n} w_i y_i \cdot \mathbf{x_i} \right)$$

We compute $LOO(j) = \theta^* - \theta^*_{-j}$ via Sherman-Morrison Formula:

Warmup: linear regression, leave-one-out

This is easy! Linear regression has a closed-form solution:

$$\theta^*(w) = \left(\sum_{i=1}^{n} w_i \cdot \mathbf{x_i}\mathbf{x_i}^\top \right)^{-1} \left(\sum_{i=1}^{n} w_i y_i \cdot \mathbf{x_i} \right)$$

We compute $LOO(j) = \theta^* - \theta^*_{-j}$ via Sherman-Morrison Formula:

$$LOO(j) = \frac{\left(\mathbf{x_j}^\top \theta^* - y_j \right) \left(\sum_{i=1}^{n} \mathbf{x_i}\mathbf{x_i}^\top \right)^{-1} \mathbf{x_j}}{1 - \mathbf{x_j}^\top \left(\sum_{i=1}^{n} \mathbf{x_i}\mathbf{x_i}^\top \right)^{-1} \mathbf{x_j}}$$

Warmup: linear regression, leave-one-out

This is easy! Linear regression has a closed-form solution:

$$\theta^*(w) = \left(\sum_{i=1}^{n} w_i \cdot \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum_{i=1}^{n} w_i y_i \cdot \mathbf{x_i} \right)$$

We compute $LOO(j) = \theta^* - \theta^*_{-j}$ via Sherman-Morrison Formula:

Residual error

$$LOO(j) = \frac{\left(\mathbf{x_j}^\top \theta^* - y_j\right) \left(\sum_{i=1}^{n} \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \mathbf{x_j}}{1 - \mathbf{x_j}^\top \left(\sum_{i=1}^{n} \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \mathbf{x_j}}$$

Warmup: linear regression, leave-one-out

This is easy! Linear regression has a closed-form solution:

$$\theta^*(w) = \left(\sum_{i=1}^{n} w_i \cdot \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum_{i=1}^{n} w_i y_i \cdot \mathbf{x_i} \right)$$

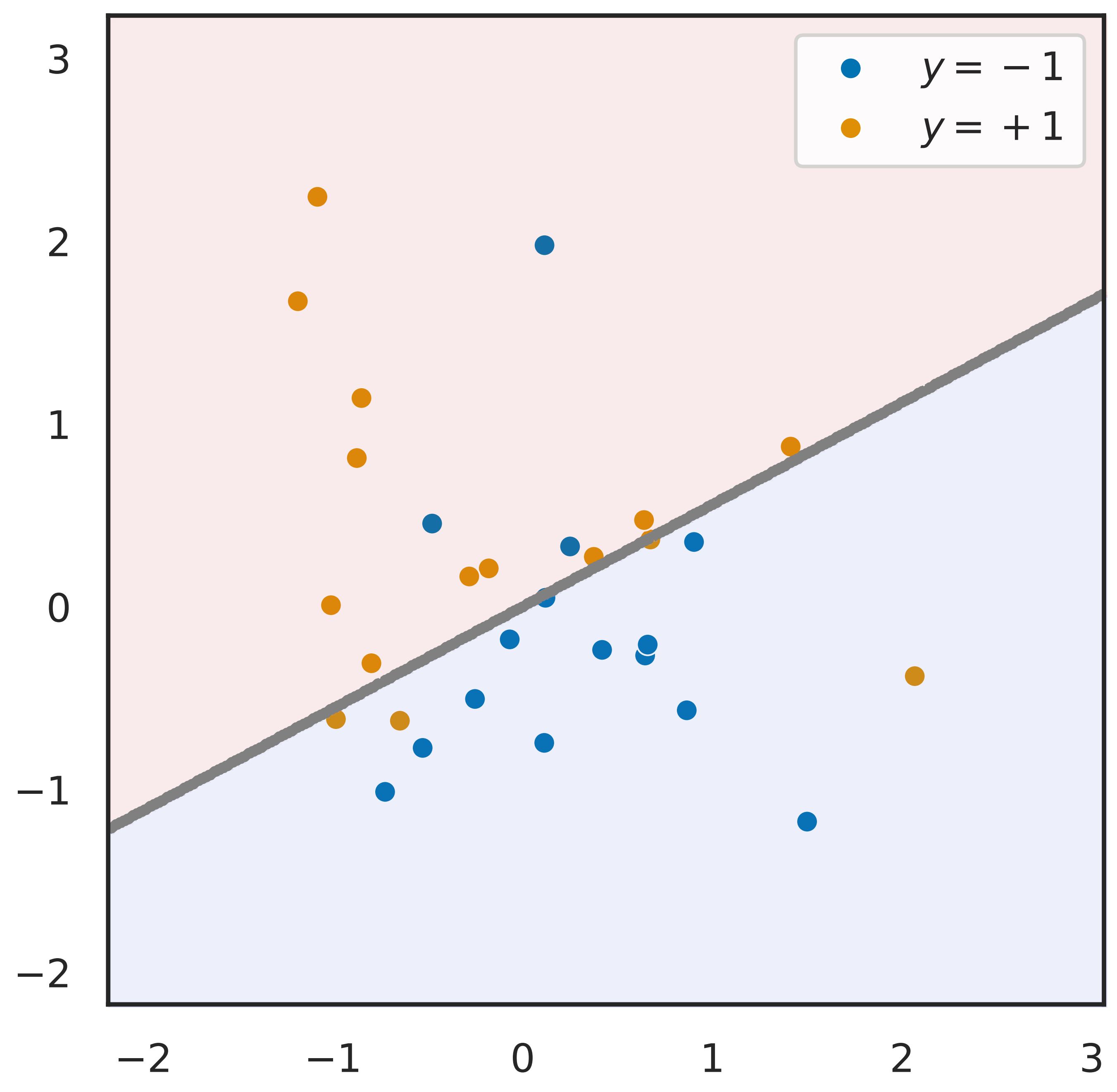
We compute $LOO(j) = \theta^* - \theta^*_{-j}$ via Sherman-Morrison Formula:

Residual error

(Already computed while estimating θ^*)

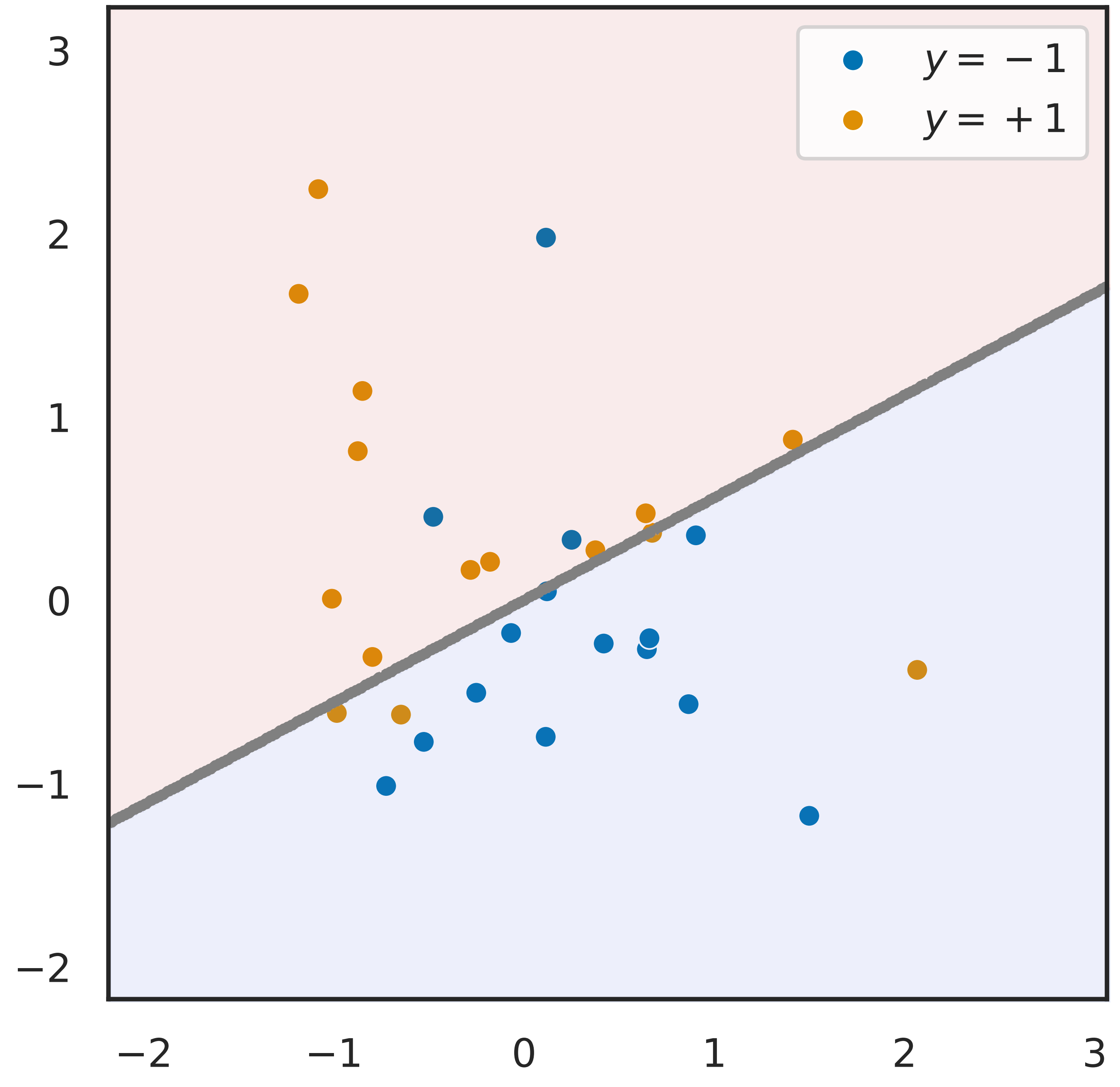
$$LOO(j) = \frac{\left(\mathbf{x_j}^\top \theta^* - y_j \right) \left(\sum_{i=1}^{n} \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \mathbf{x_j}}{1 - \mathbf{x_j}^\top \left(\sum_{i=1}^{n} \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \mathbf{x_j}}$$

LOO for more complex (linear) models



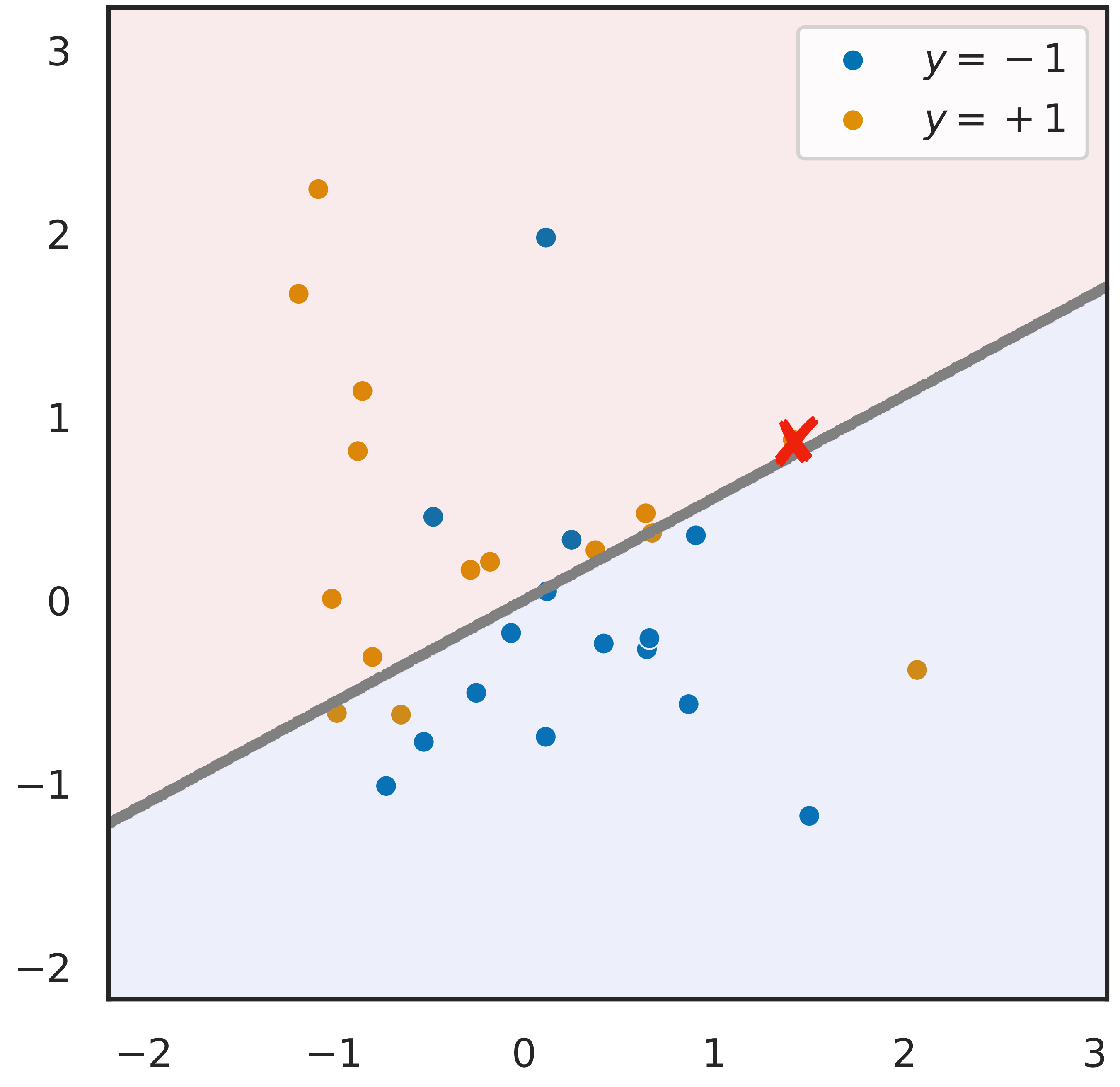
LOO for more complex (linear) models

What about for a more complex (but still linear) model class (e.g., GLMs)?



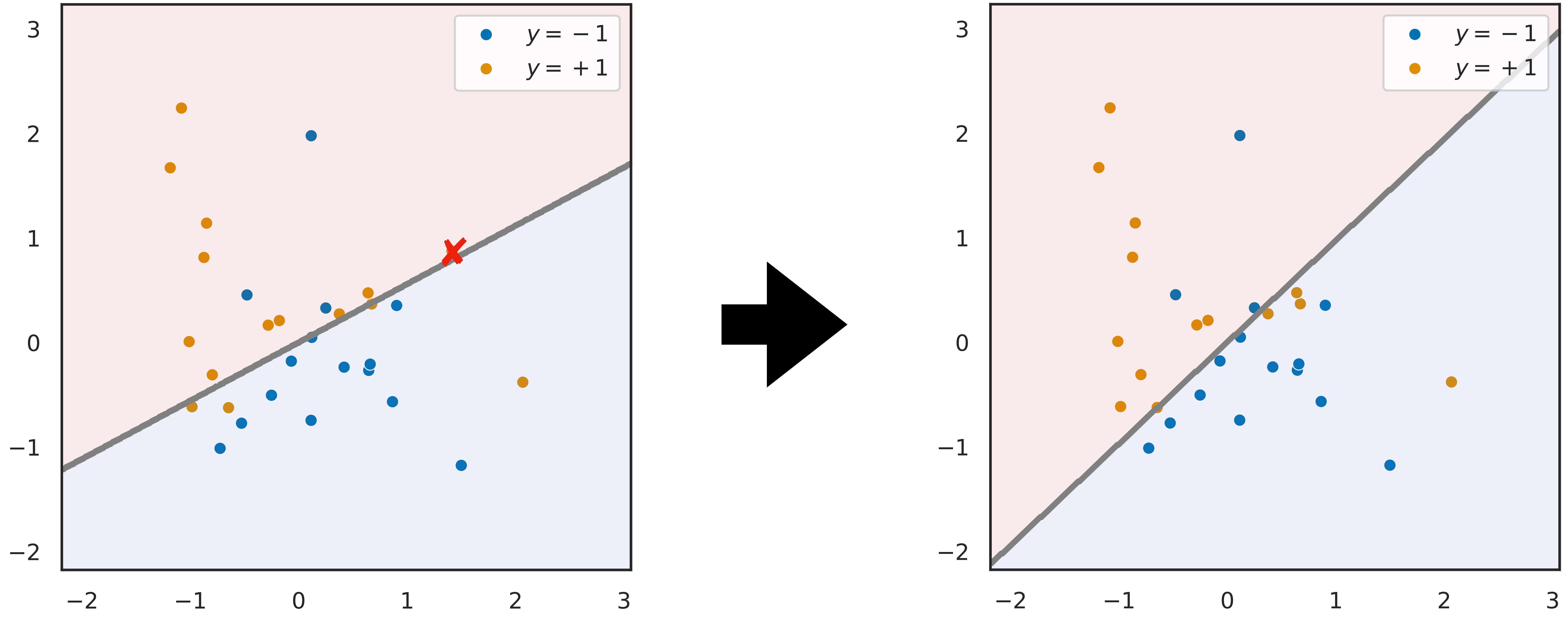
LOO for more complex (linear) models

What about for a more complex (but still linear) model class (e.g., GLMs)?



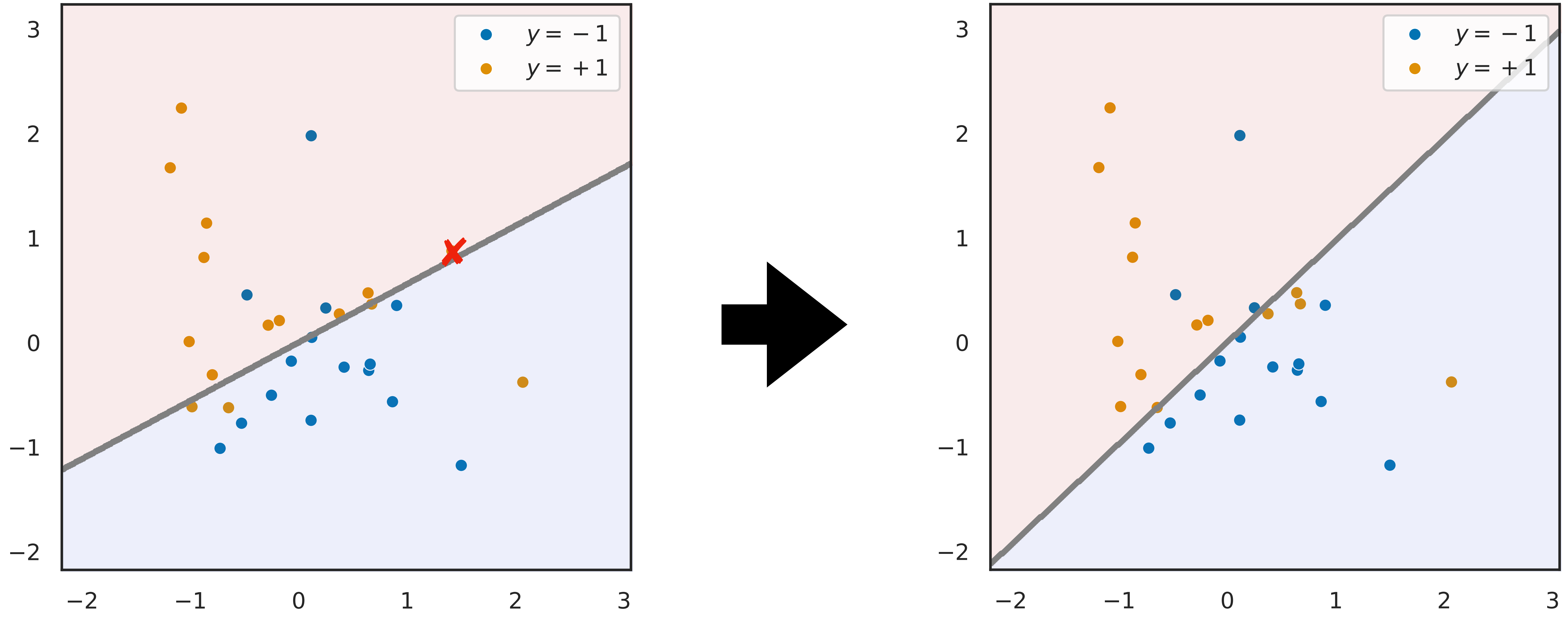
LOO for more complex (linear) models

What about for a more complex (but still linear) model class (e.g., GLMs)?



LOO for more complex (linear) models

What about for a more complex (but still linear) model class (e.g., GLMs)?



Even for logistic regression, previous calculation doesn't work (no closed form)

LOO for more complex (linear) models

[Pregibon ’81; Shao & Tu ’12; Rad & Maleki ’20]

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Clever idea: quadratic approximation

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Clever idea: quadratic approximation

Assume **linear** losses $\ell_i(\theta) = \mathscr{L}_i(\theta^\top x_i)$ (true for logistic regression, LASSO, etc.)

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Clever idea: quadratic approximation

Assume **linear** losses $\ell_i(\theta) = \mathscr{L}_i(\theta^\top x_i)$ (true for logistic regression, LASSO, etc.)

Let $L_{-j}(\theta) = \sum_{j \neq i} \mathscr{L}_i(\theta^\top x)$, so that $\theta^*_{-j} = \arg\min_\theta L_{-j}(\theta)$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Clever idea: quadratic approximation

Assume **linear** losses $\ell_i(\theta) = \mathcal{L}_i(\theta^\top x_i)$ (true for logistic regression, LASSO, etc.)

Let $L_{-j}(\theta) = \sum_{j \neq i} \mathcal{L}_i(\theta^\top x)$, so that $\theta^*_{-j} = \arg\min_\theta L_{-j}(\theta)$

Form a quadratic approximation of $L_{-j}(\theta)$ around the point $\theta^*(\mathbf{1}_n)$, use the (closed-form) solution as an estimate of $\theta^*(\mathbf{1}_n - \mathrm{Ind}_n[j])$

LOO for more complex (linear) models

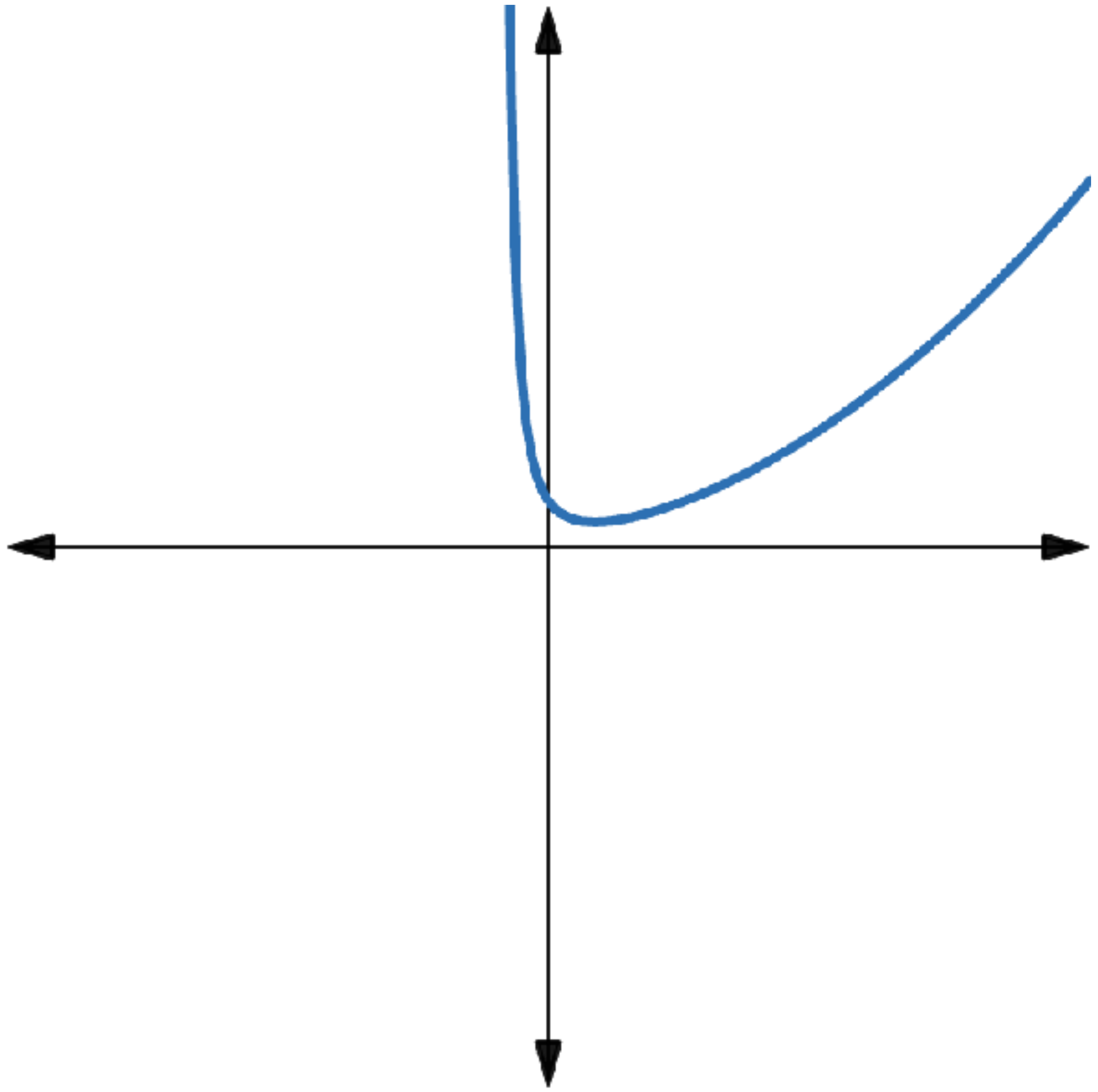
[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

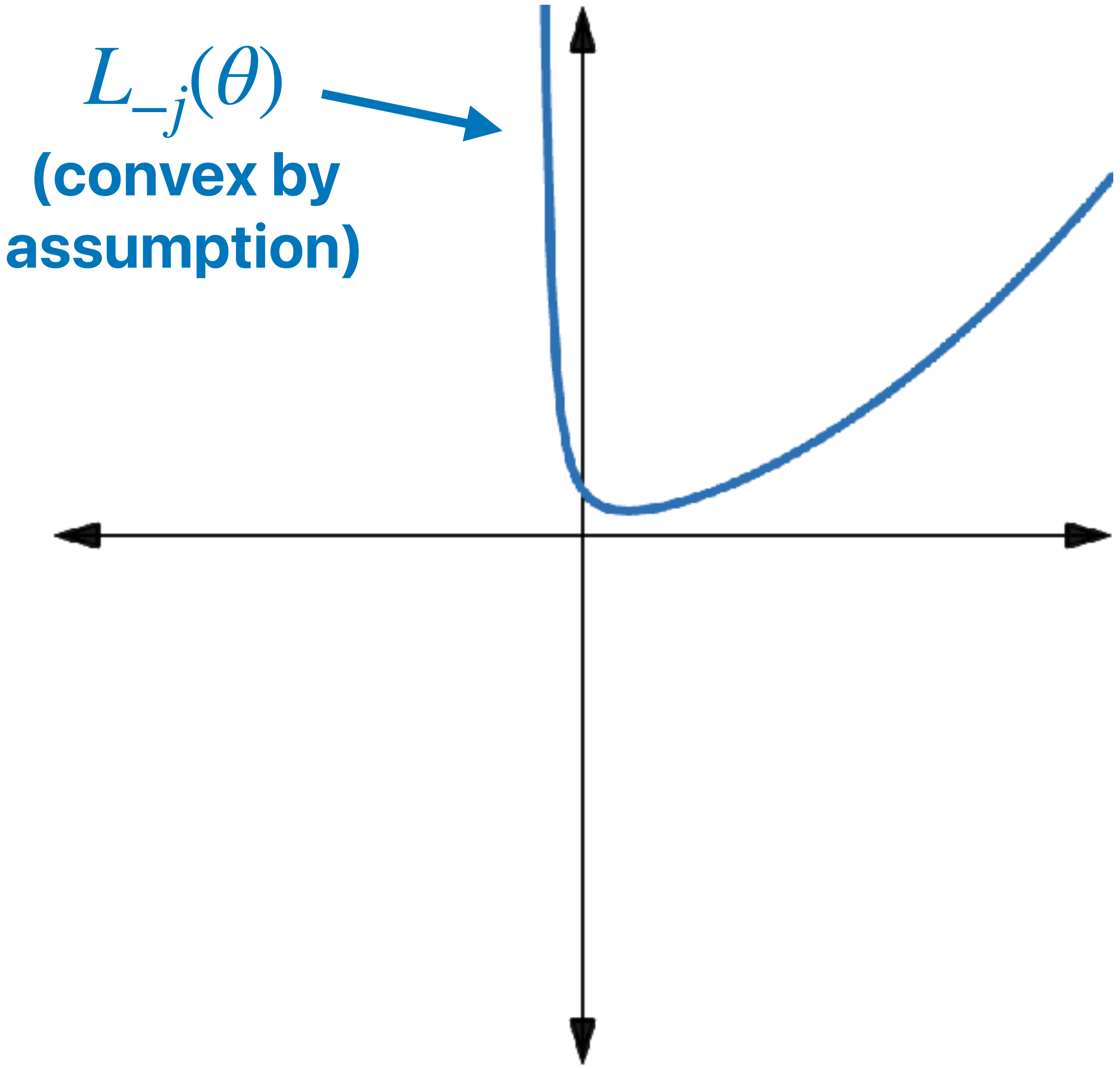
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

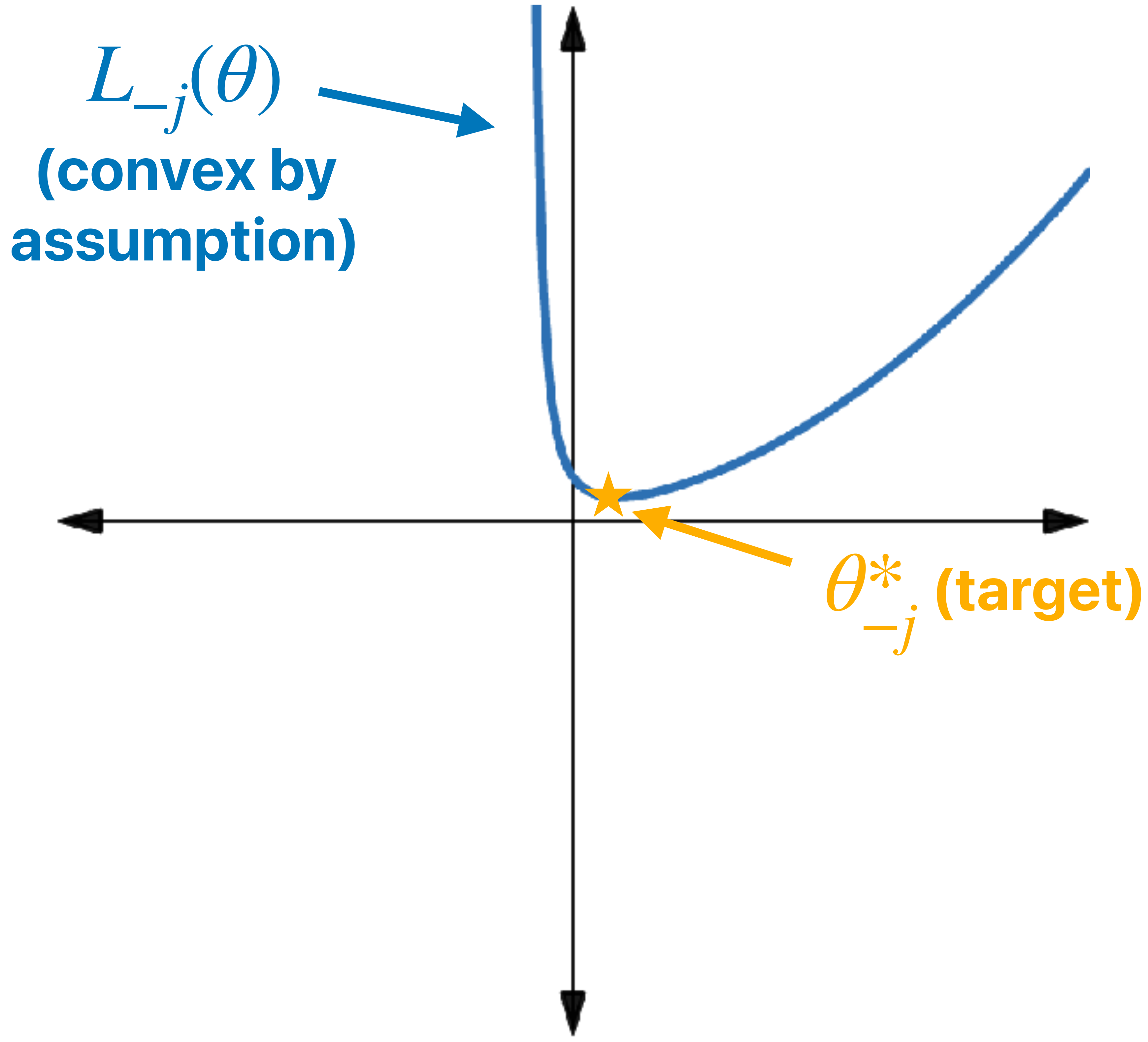
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

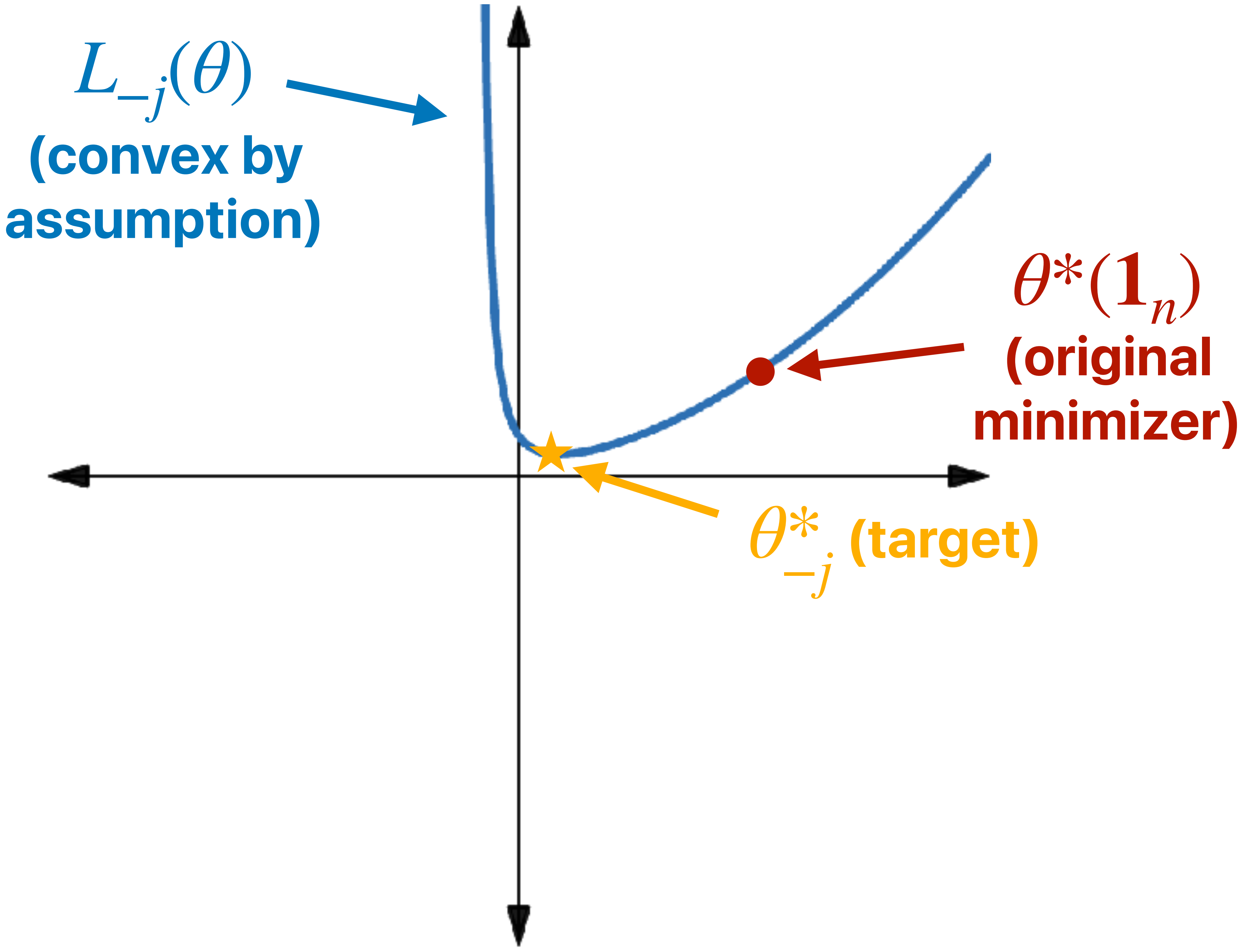
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

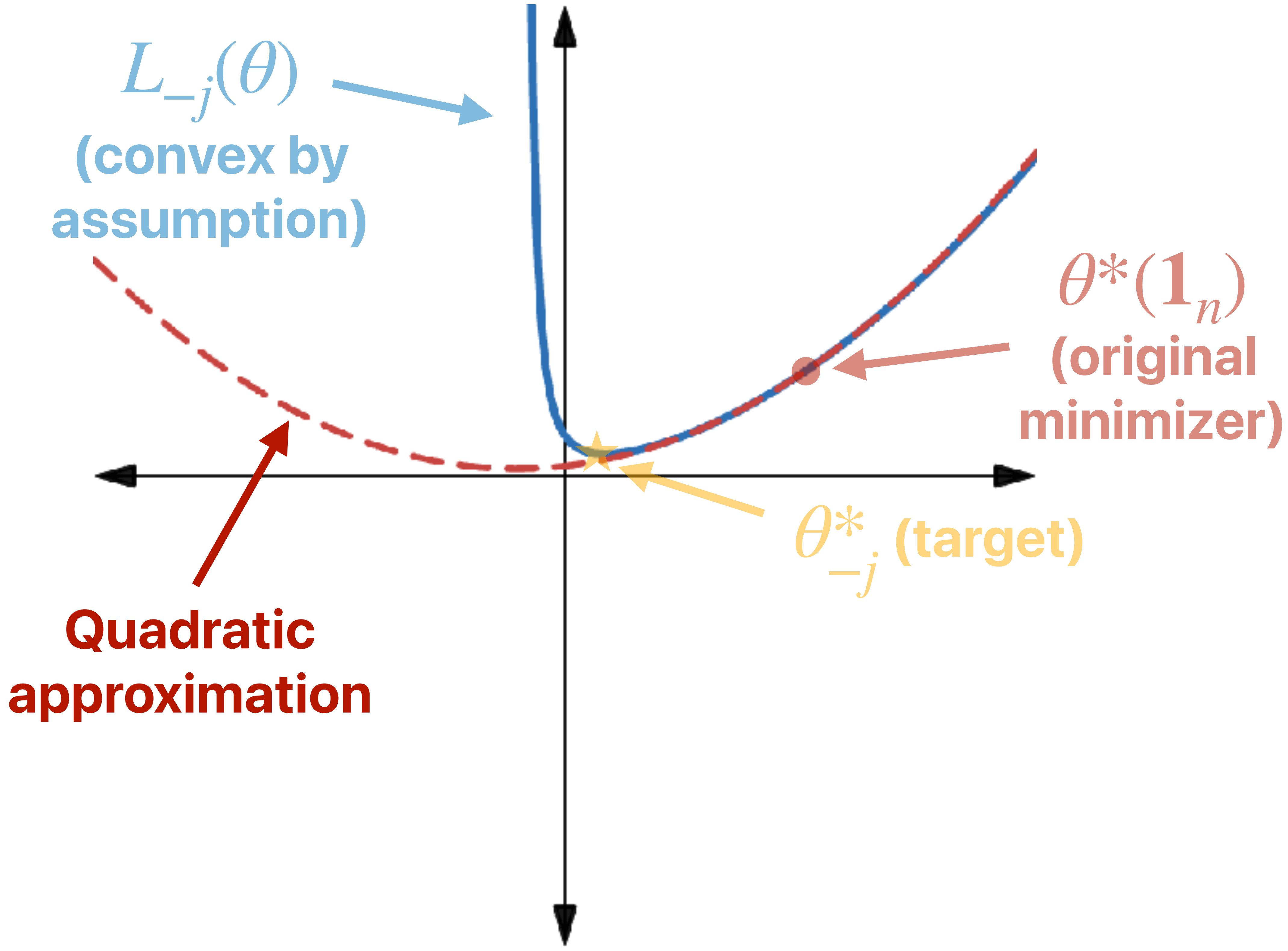
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

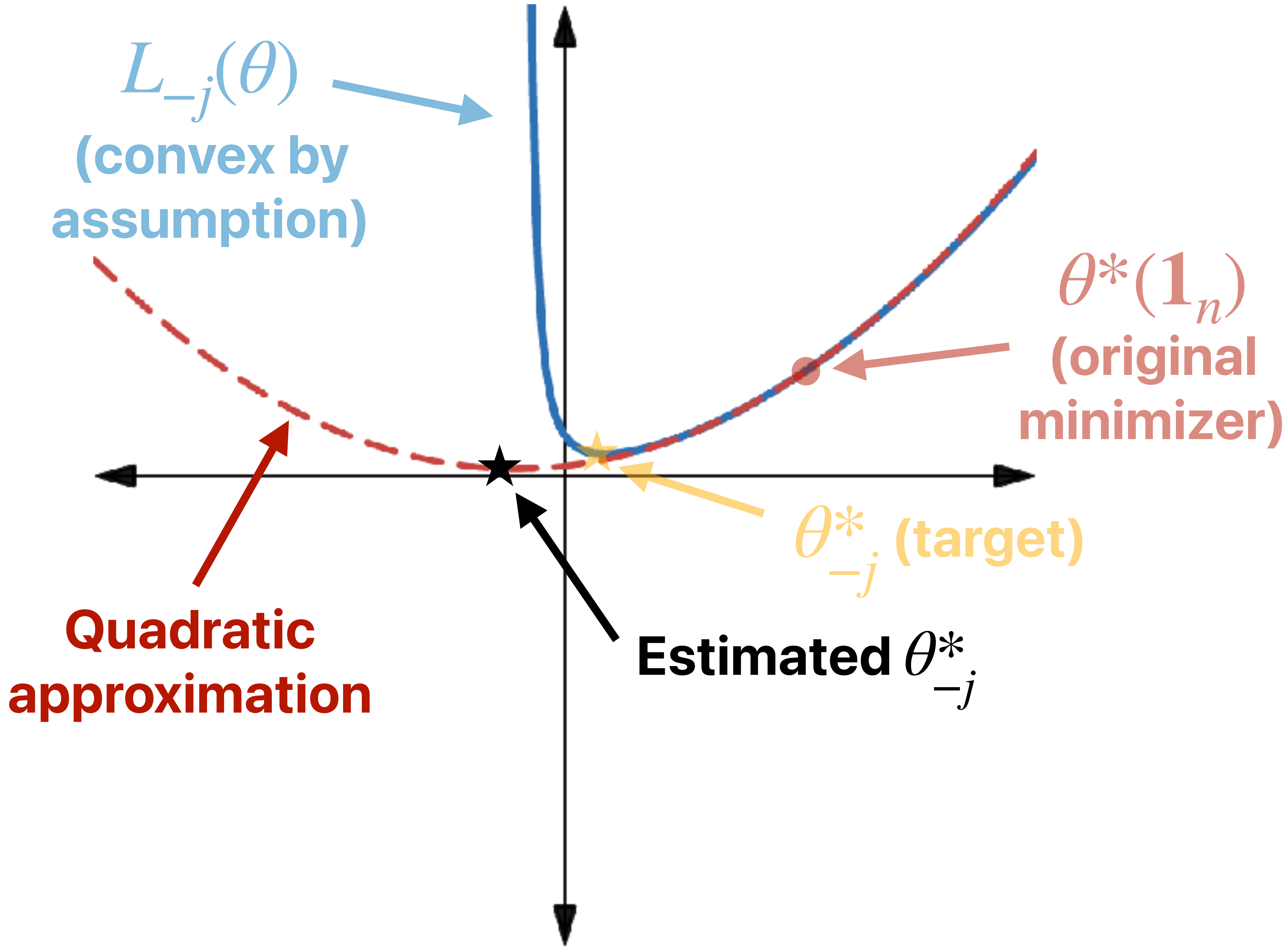
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

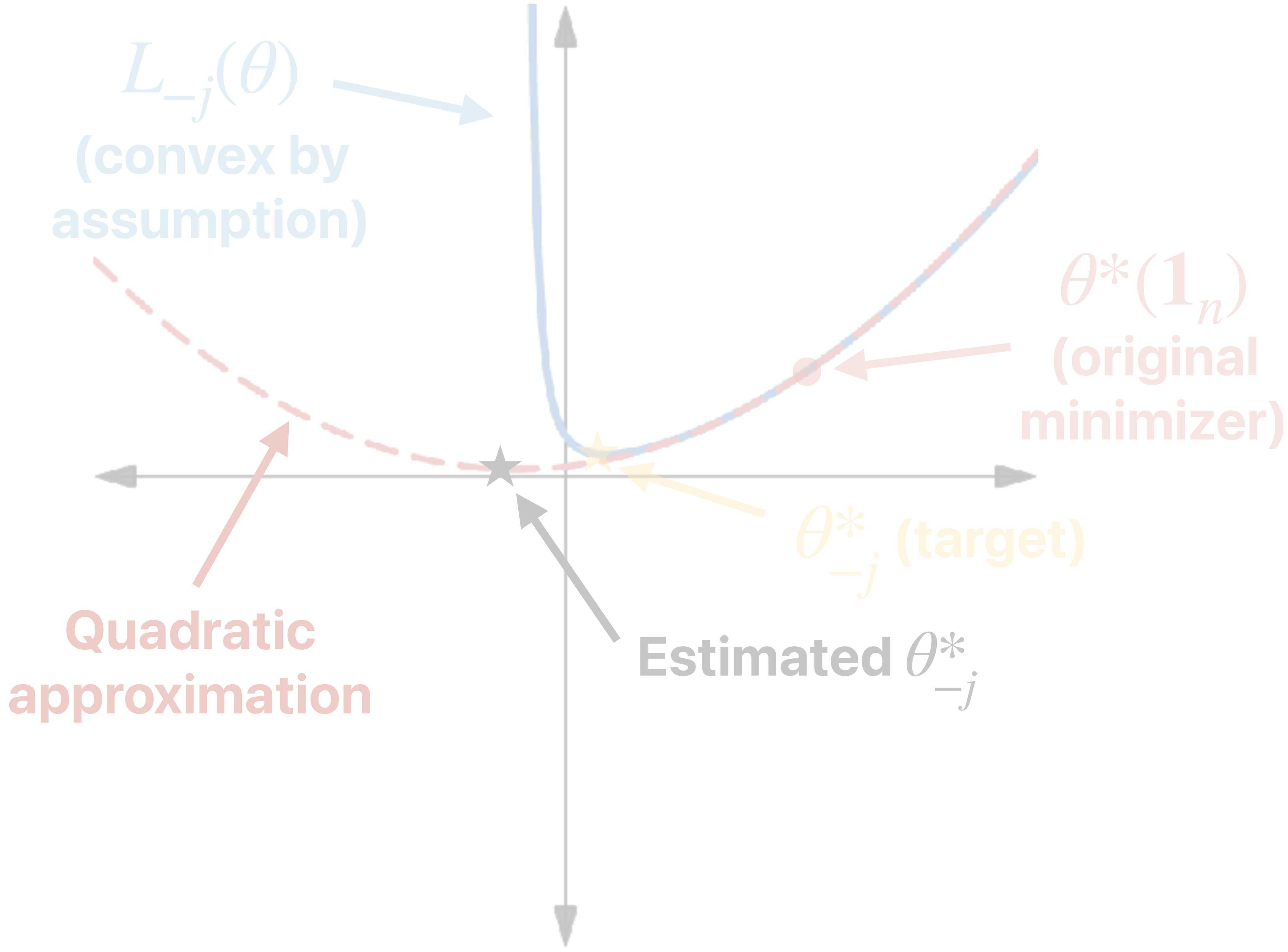
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

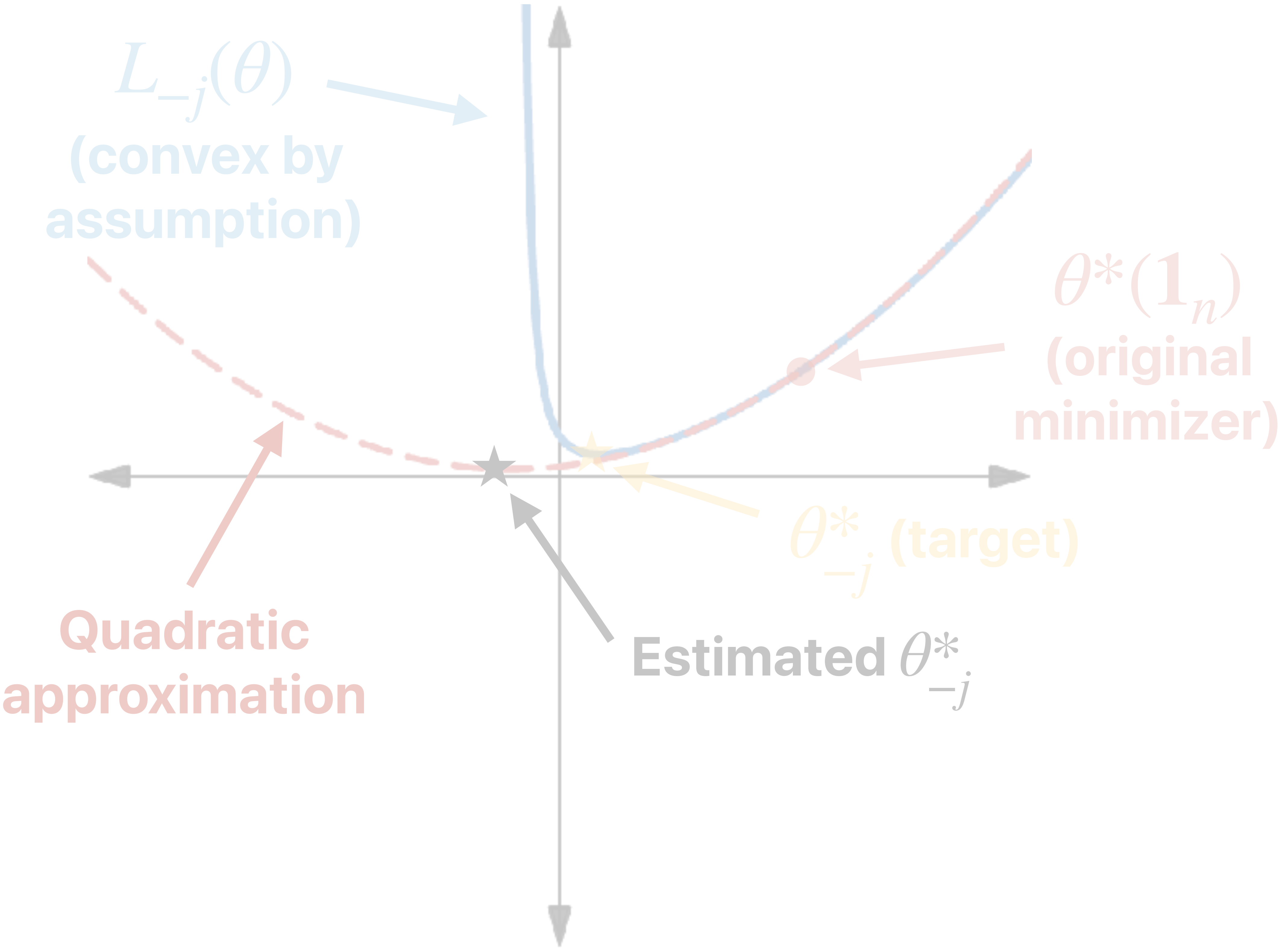
Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

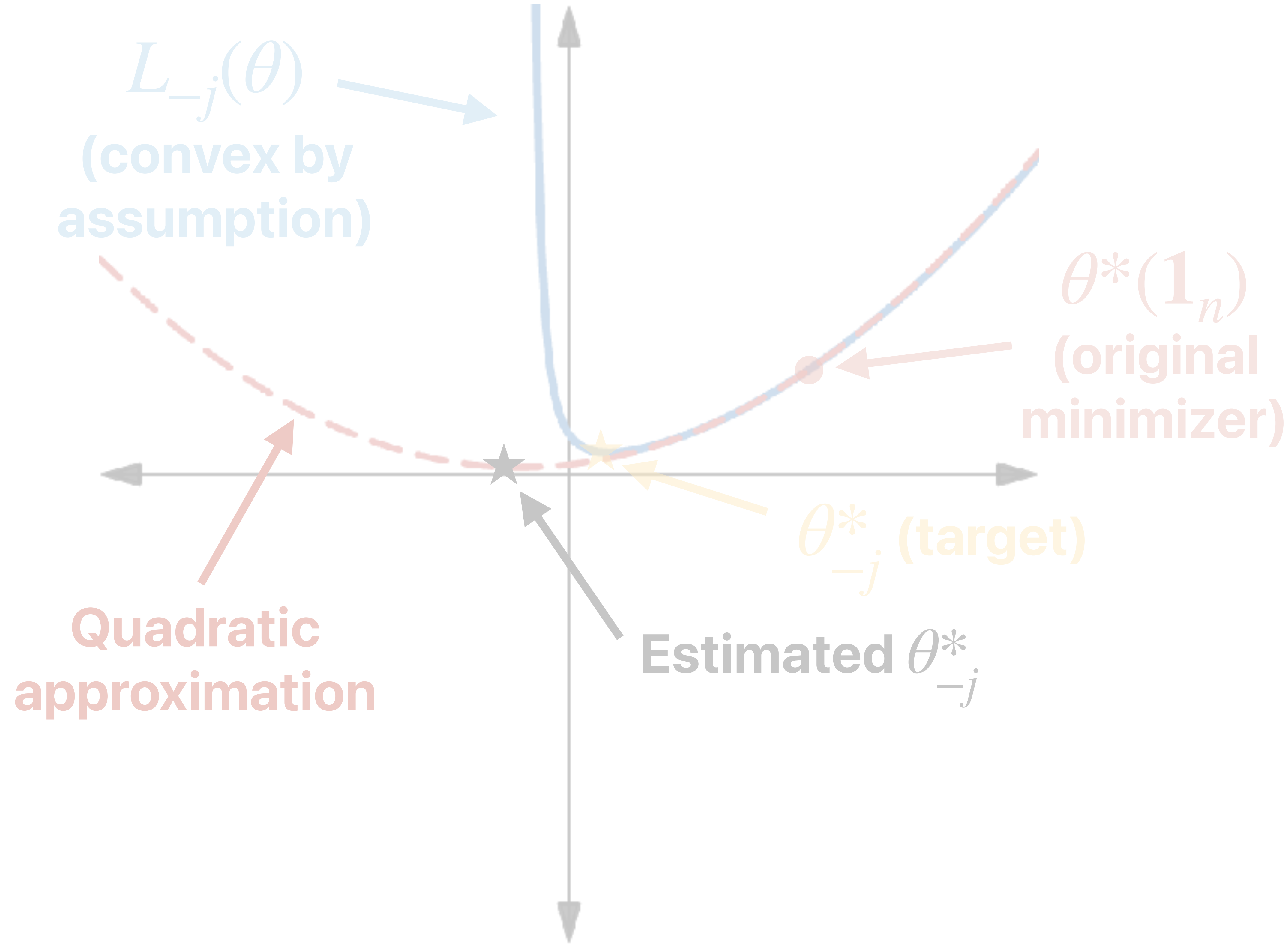


$$\theta^*_{-j} \approx \theta^* - \left(\sum_{i \neq j} \nabla^2 \ell_i(\theta) \right)^{-1} \left(\sum_{i \neq j} \nabla \ell_i(\theta) \right)$$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



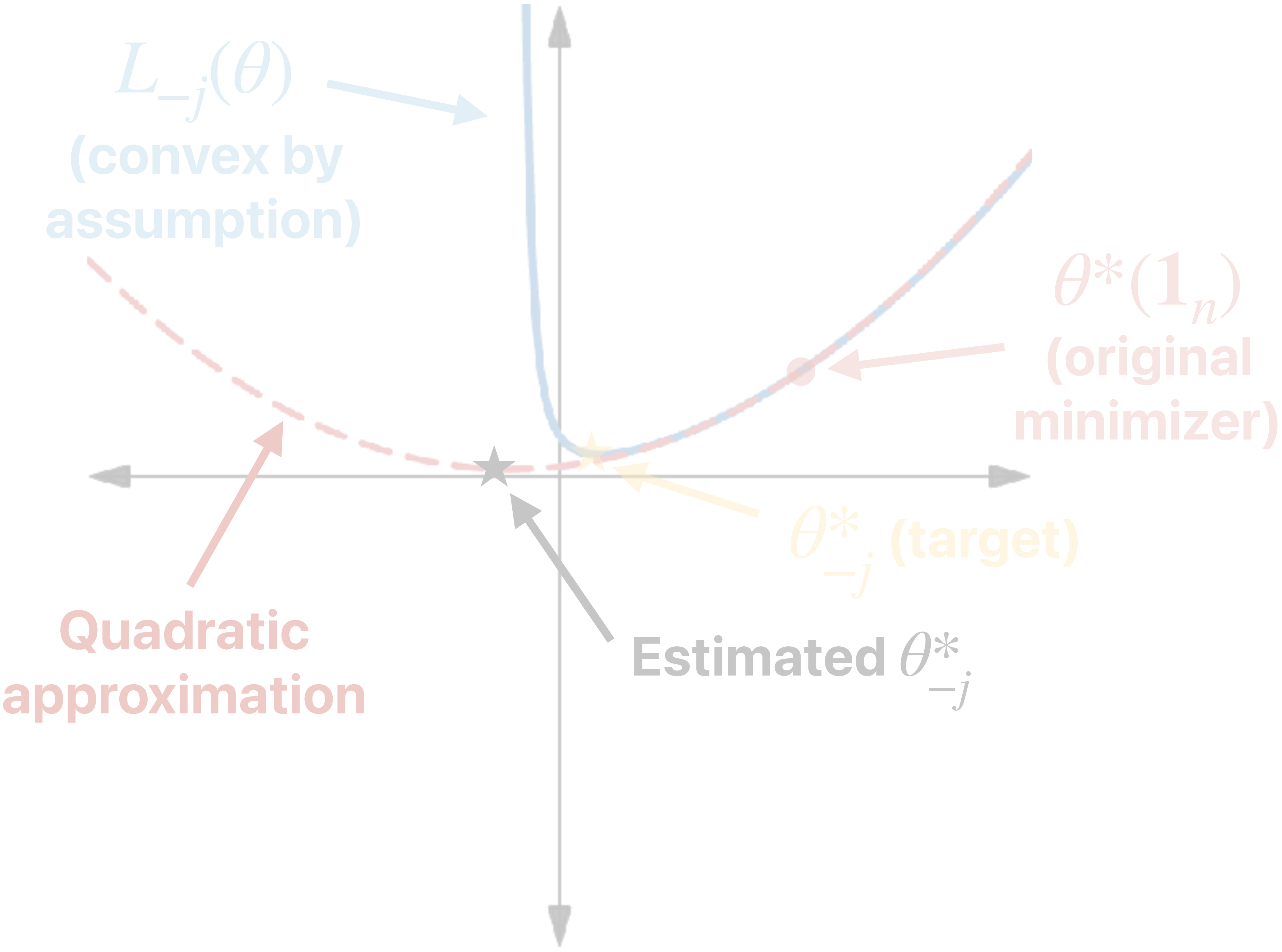
$$\theta^*_{-j} \approx \theta^* - \left(\sum_{i \neq j} \nabla^2 \ell_i(\theta) \right)^{-1} \left(\sum_{i \neq j} \nabla \ell_i(\theta) \right)$$

(Linearity)

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



$$\theta^*_{-j} \approx \theta^* - \left(\sum_{i \neq j} \nabla^2 \ell_i(\theta)\right)^{-1} \left(\sum_{i \neq j} \nabla \ell_i(\theta)\right)$$

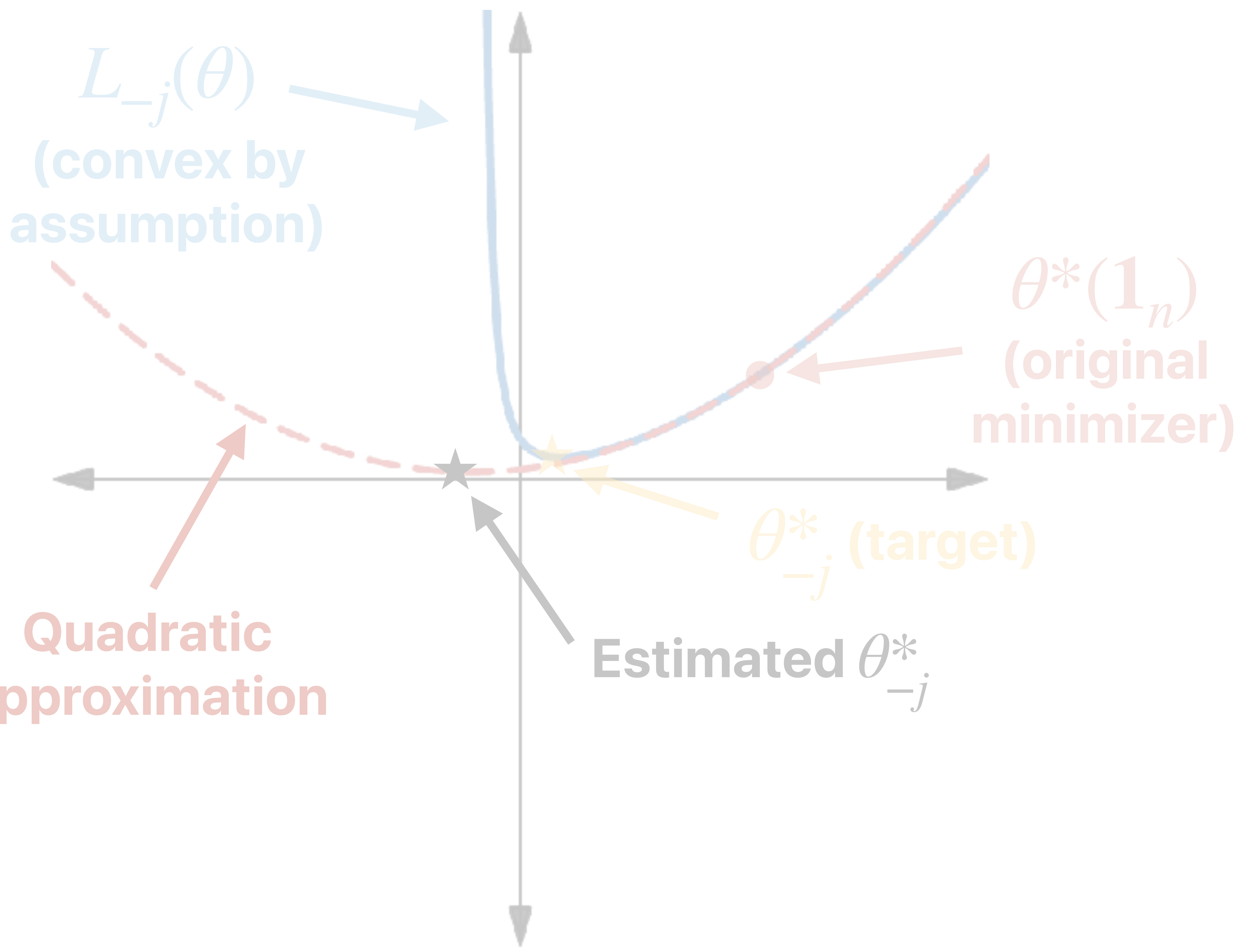
(Linearity)

$$= \left(\sum_{i \neq j} \mathcal{L}''_i(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top\right)^{-1} \left(\sum_{i \neq j} \mathcal{L}'_i(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i}\right)$$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



$$\theta^*_{-j} \approx \theta^* - \left(\sum_{i \neq j} \nabla^2 \ell_i(\theta) \right)^{-1} \left(\sum_{i \neq j} \nabla \ell_i(\theta) \right)$$

(Linearity)

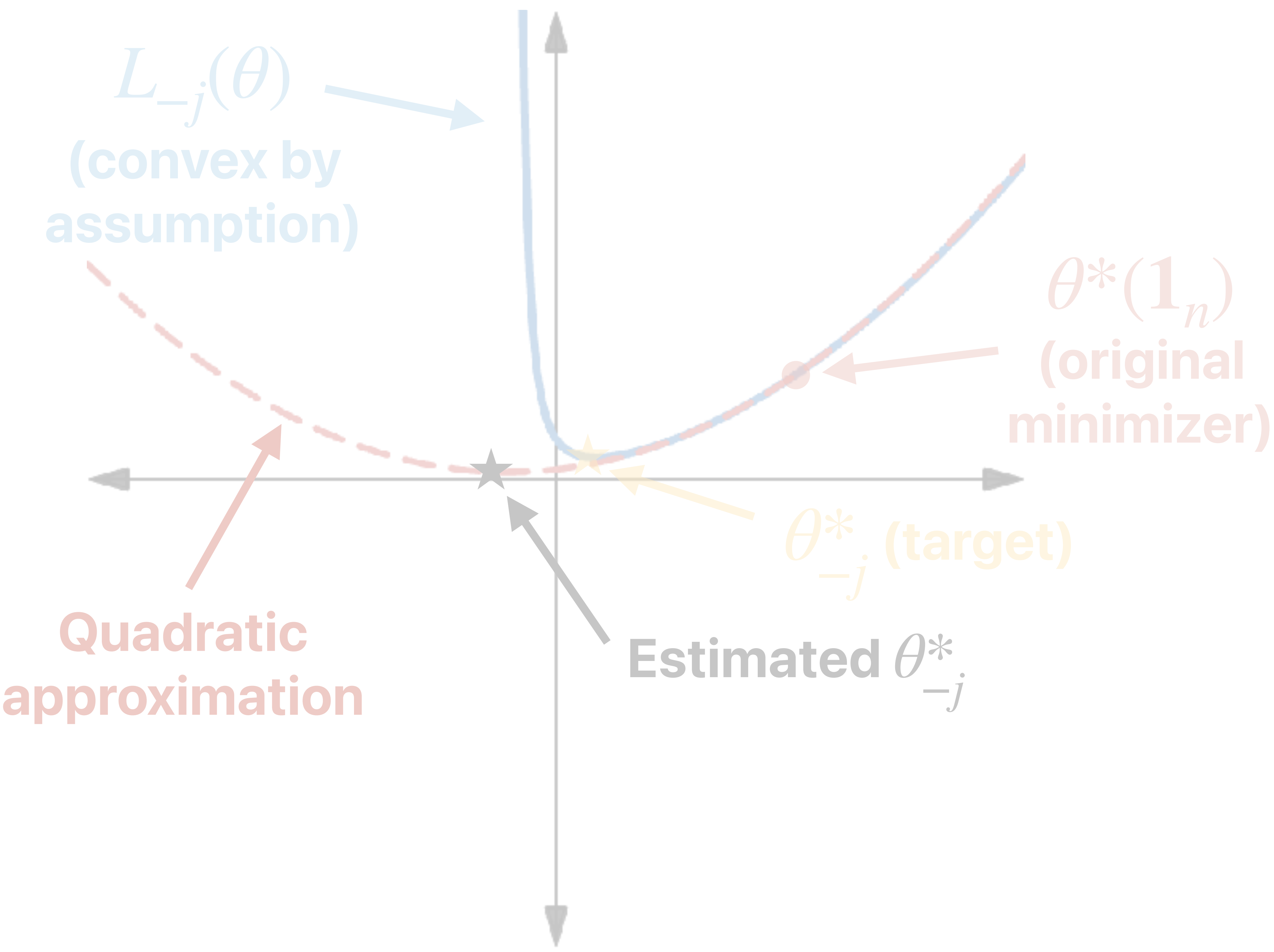
$$= \left(\sum_{i \neq j} \mathcal{L}_i''(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum_{i \neq j} \mathcal{L}_i'(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i} \right)$$

(Sherman Morrison formula)

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



$$\theta^*_{-j} \approx \theta^* - \left(\sum_{i \neq j} \nabla^2 \ell_i(\theta) \right)^{-1} \left(\sum_{i \neq j} \nabla \ell_i(\theta) \right)$$

(Linearity)

$$= \left(\sum_{i \neq j} \mathcal{L}_i''(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum_{i \neq j} \mathcal{L}_i'(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i} \right)$$

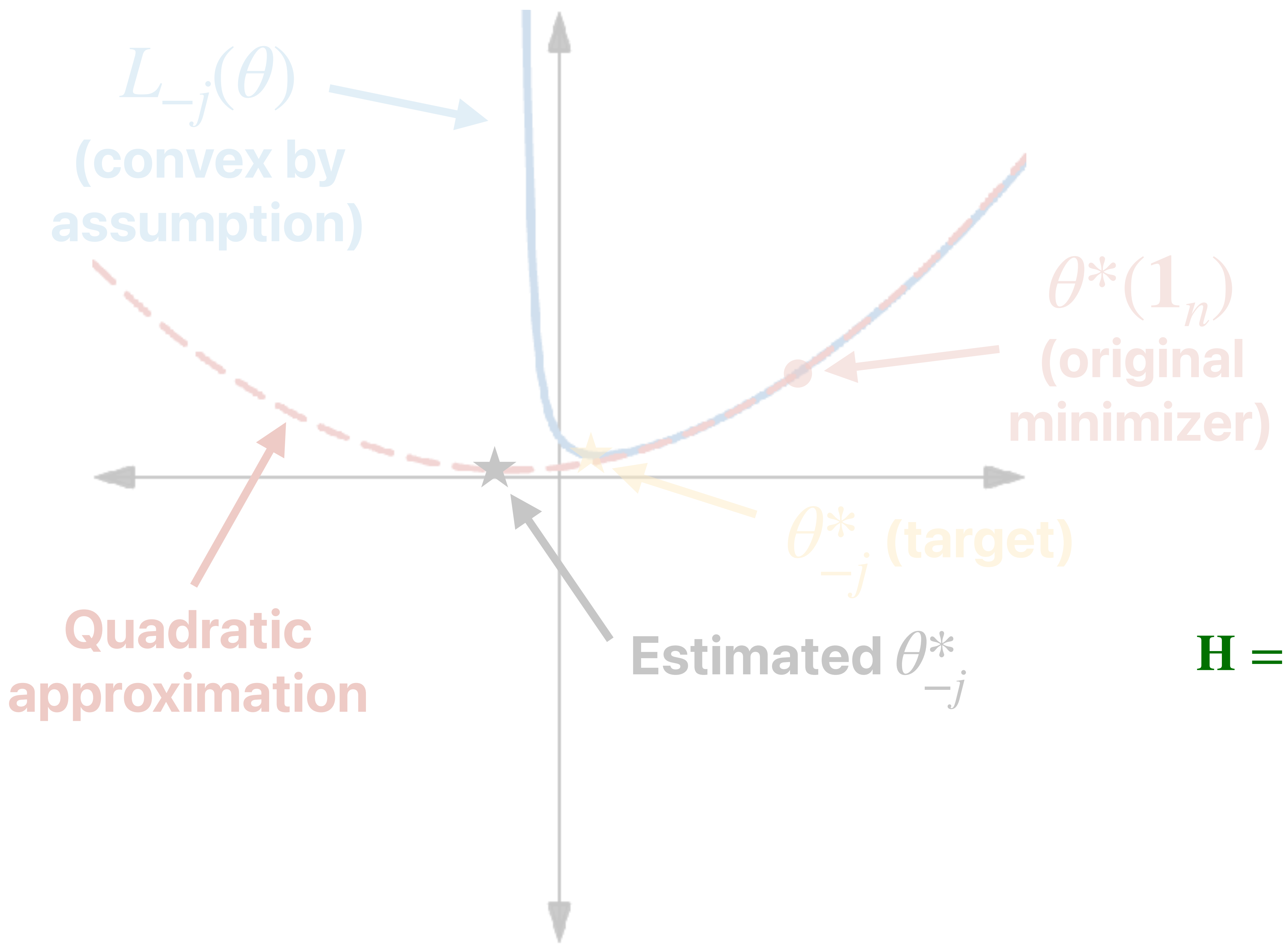
(Sherman Morrison formula)

$$LOO(j) \approx \frac{\mathbf{H}^{-1} \left(\mathcal{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j} \right)}{1 - \mathcal{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$



$$\theta^*_{-j} \approx \theta^* - \left(\sum_{i \neq j} \nabla^2 \ell_i(\theta)\right)^{-1} \left(\sum_{i \neq j} \nabla \ell_i(\theta)\right)$$

(Linearity)

$$= \left(\sum_{i \neq j} \mathcal{L}_i''(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i}\mathbf{x_i}^\top\right)^{-1} \left(\sum_{i \neq j} \mathcal{L}_i'(\theta^\top \mathbf{x_i}) \cdot \mathbf{x_i}\right)$$

(Sherman Morrison formula)

$$\mathbf{H} = \sum_{i=1}^{n} \mathcal{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i}\mathbf{x_i}^\top$$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathcal{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathcal{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}, \qquad \mathbf{H} := \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top$$

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}, \qquad \mathbf{H} := \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top$$

Linear regression

LOO for more complex (linear) models

[Pregibon ’81; Shao & Tu ’12; Rad & Maleki ’20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}, \qquad \mathbf{H} := \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top$$

Linear regression

→ we recover closed form!

LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathcal{L}_j'(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathcal{L}_j''(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}, \qquad \mathbf{H} := \sum_{i=1}^{n} \mathcal{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top$$

Linear regression

→ we recover closed form!

Logistic regression

LOO for more complex (linear) models

[Pregibon ’81; Shao & Tu ’12; Rad & Maleki ’20]

Idea: form a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta_*^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}, \qquad \mathbf{H} := \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top$$

Linear regression

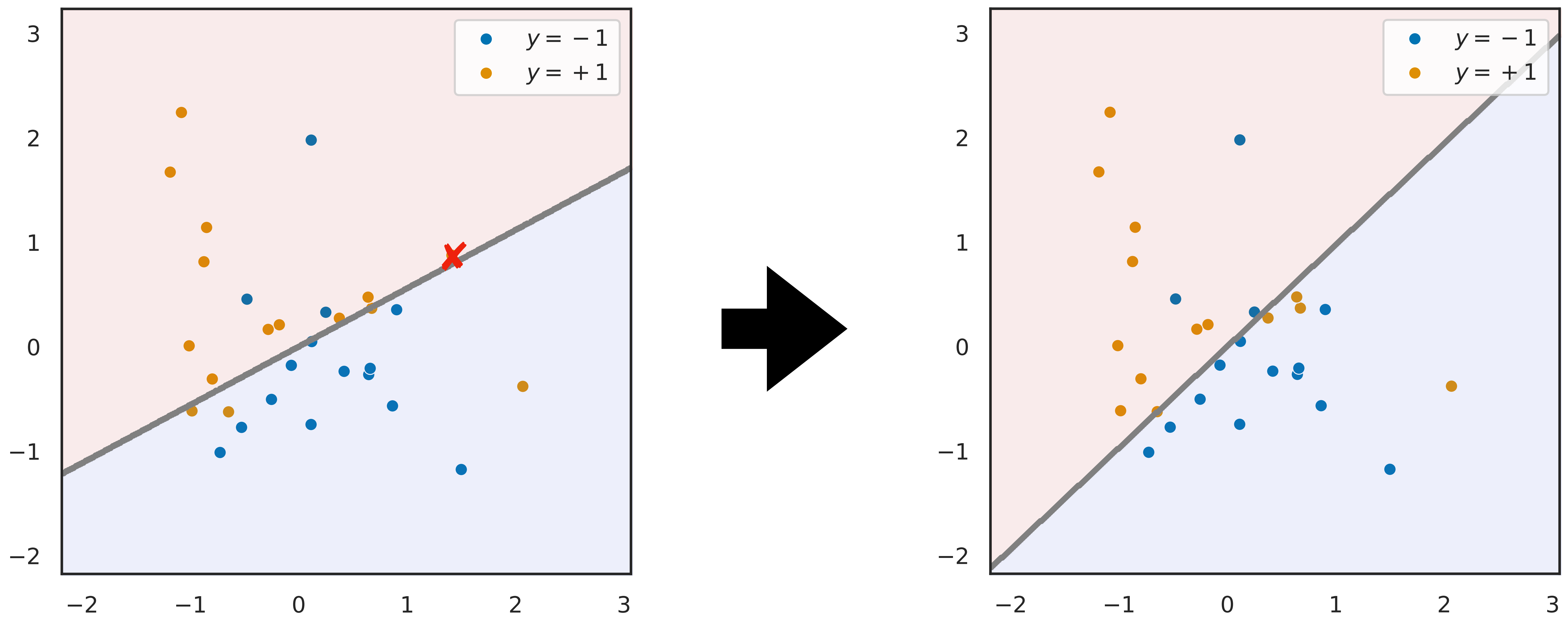
→ we recover closed form!

Logistic regression

→ does this work?

LOO for more complex (linear) models

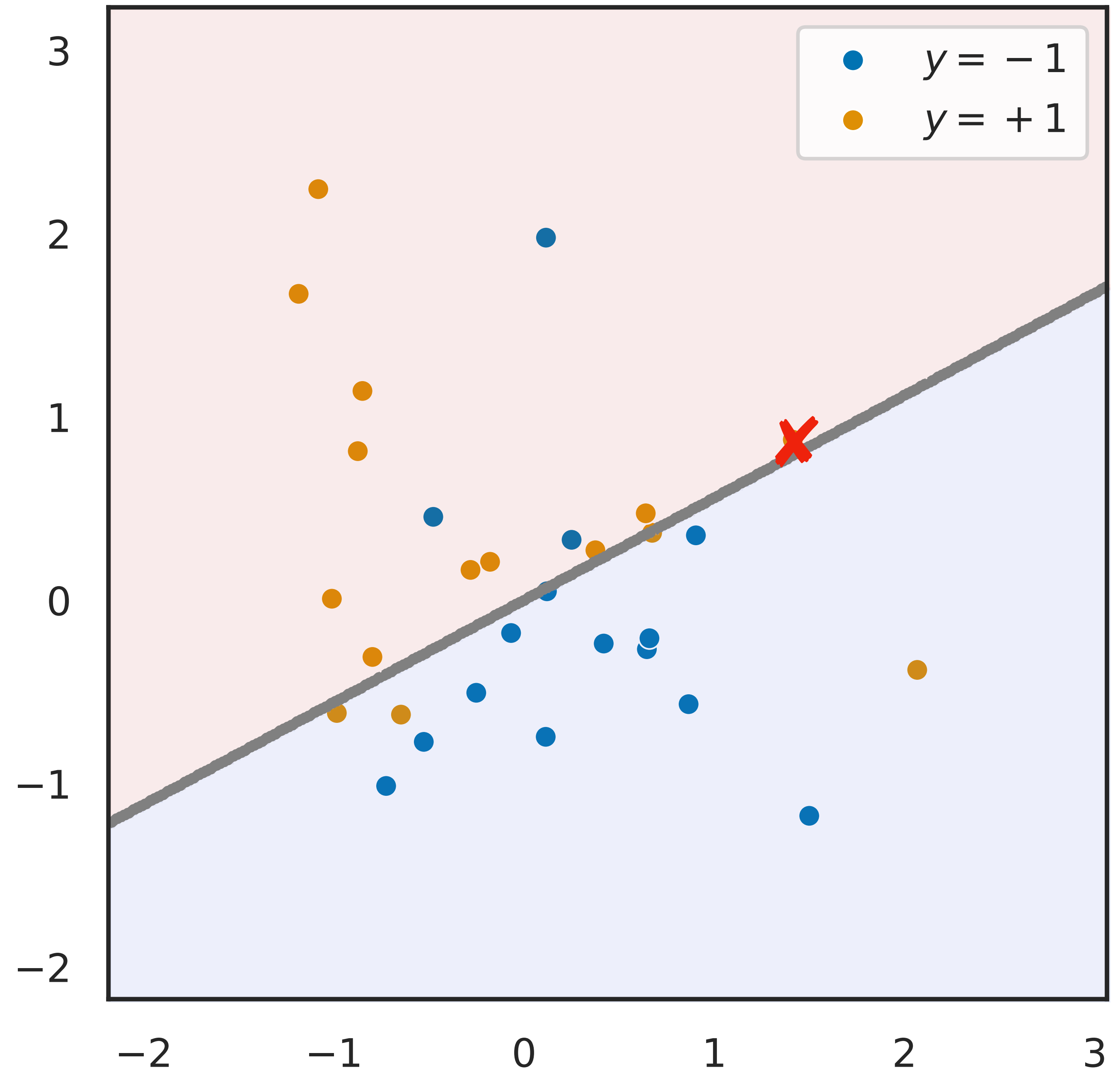
[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

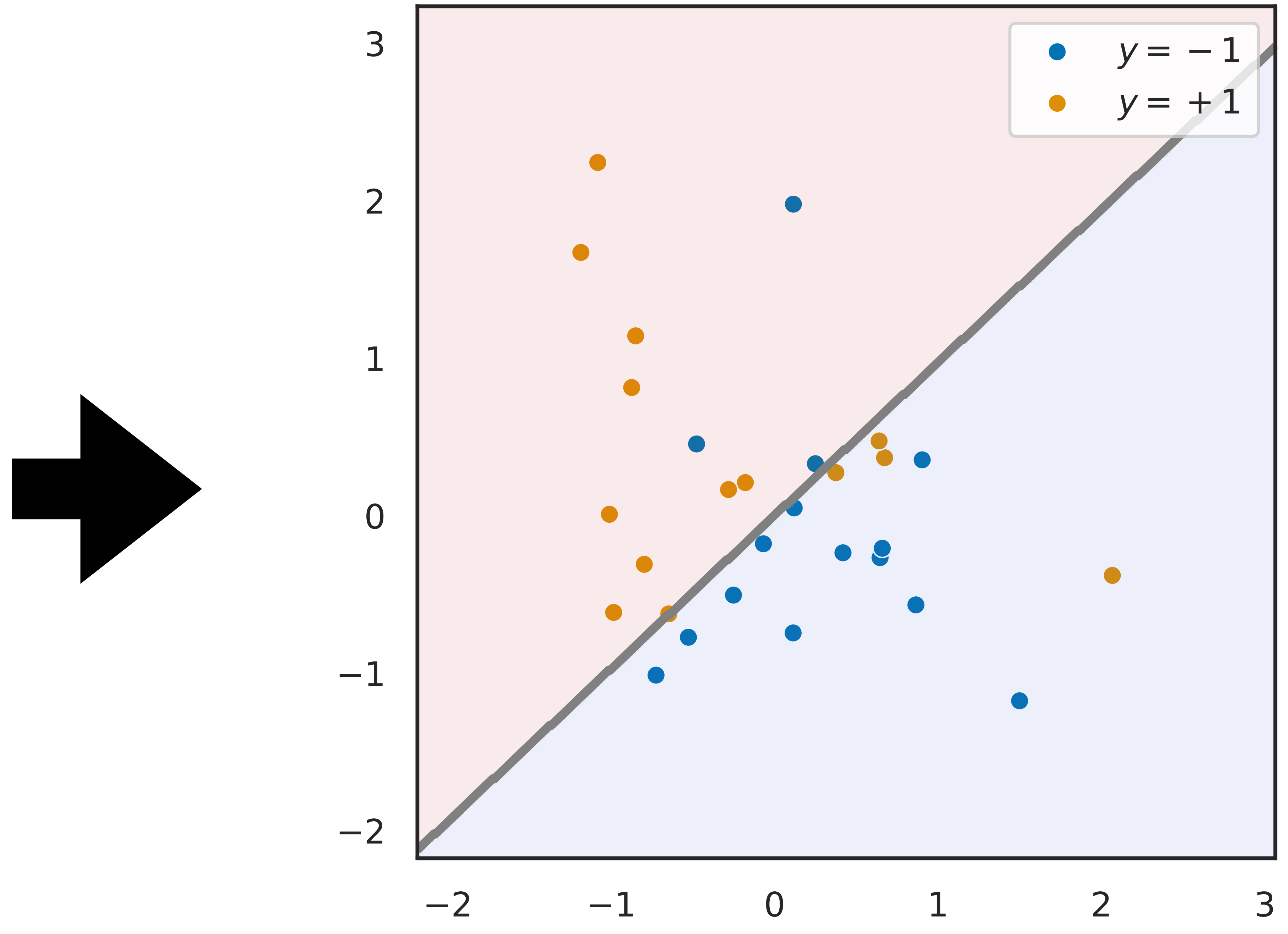


LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Applying our approximation to the logistic regression example from before:

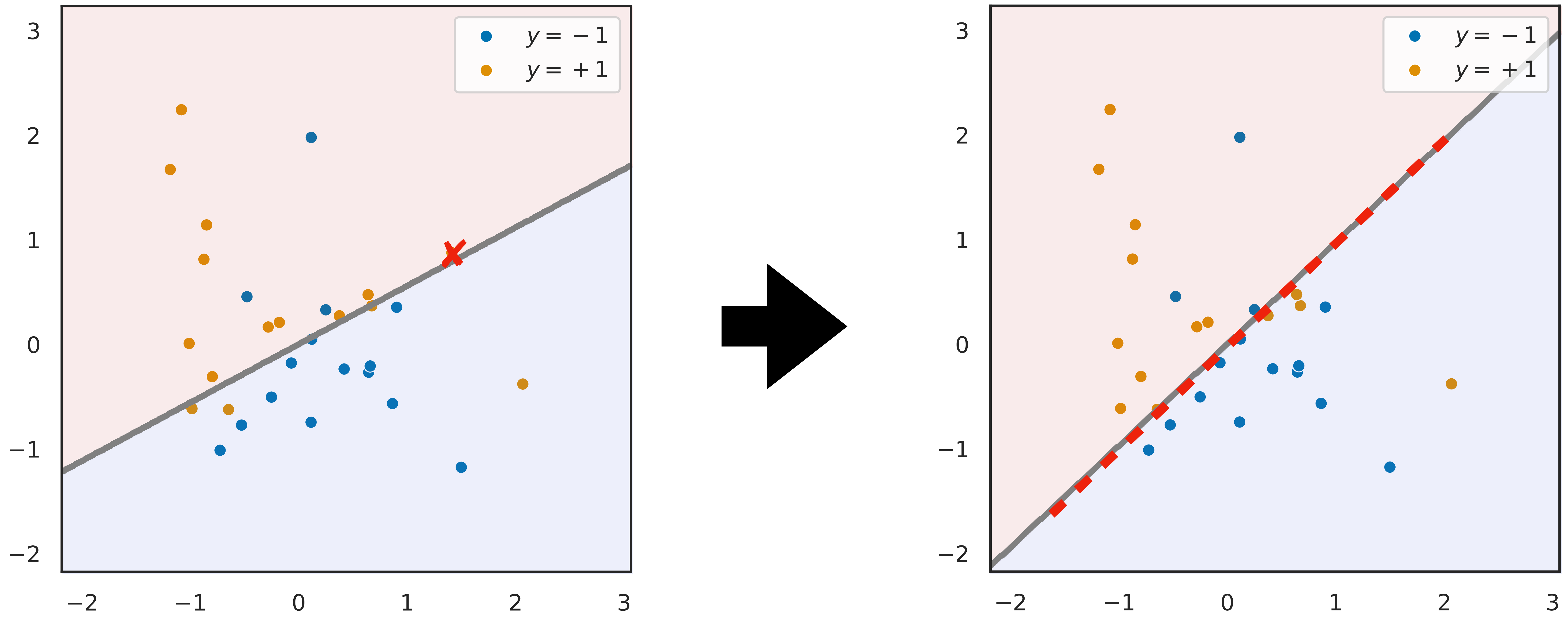




LOO for more complex (linear) models

[Pregibon ’81; Shao & Tu ’12; Rad & Maleki ’20]

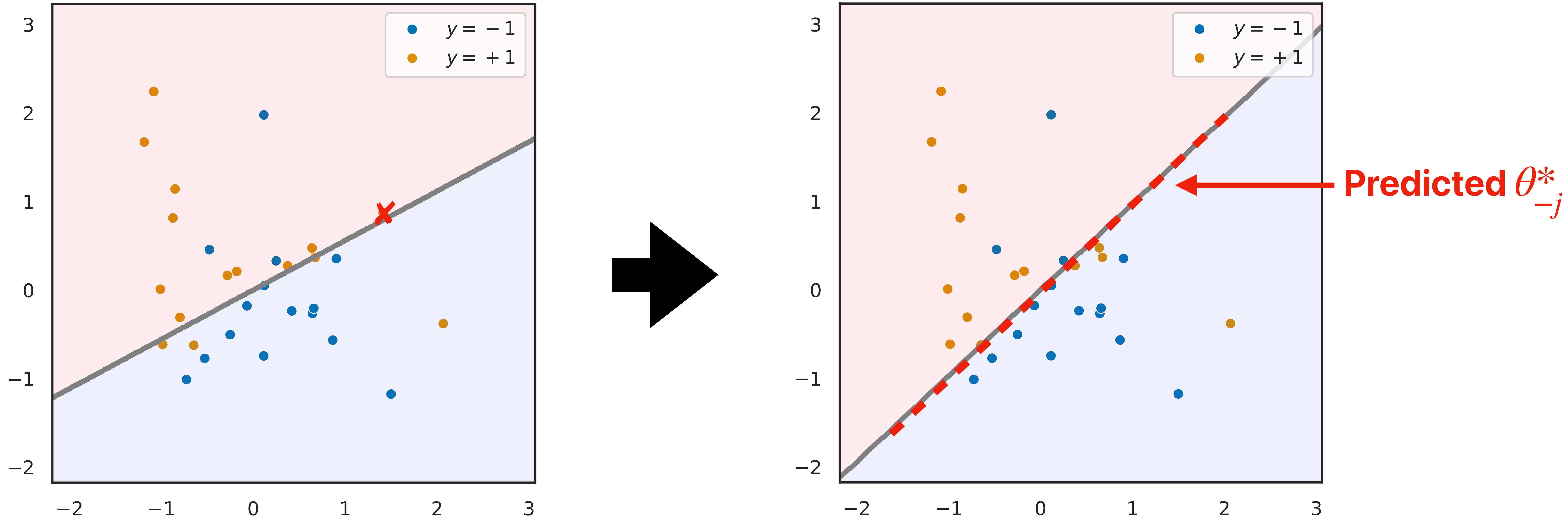
Applying our approximation to the logistic regression example from before:



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

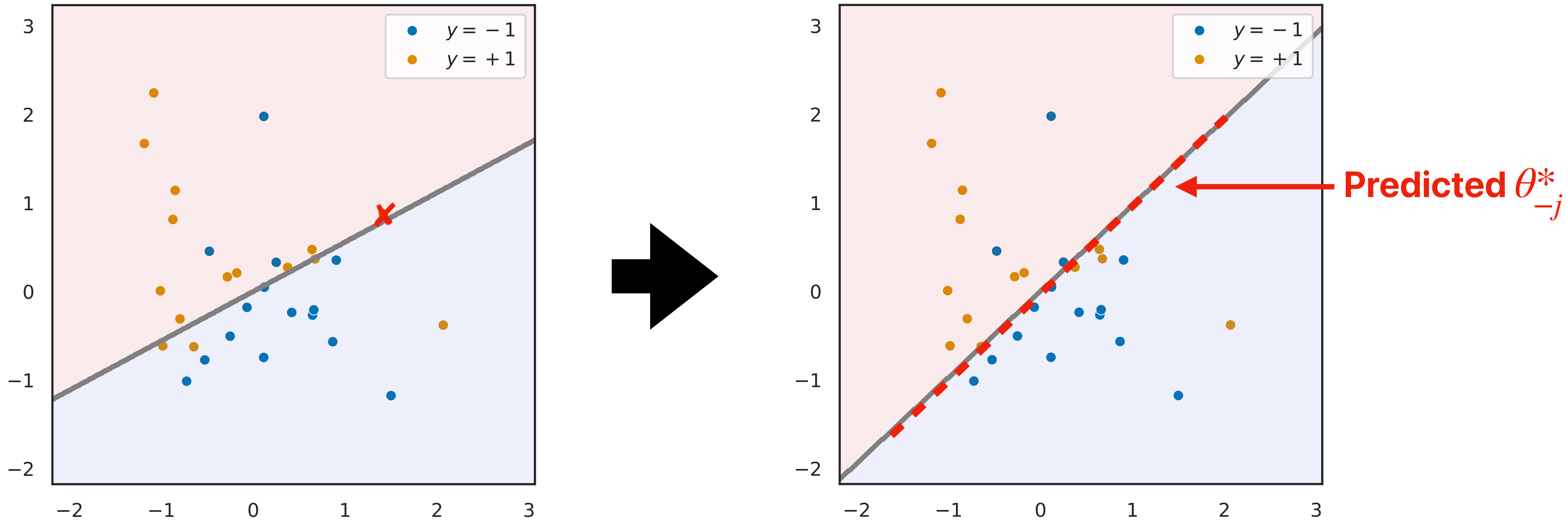
Applying our approximation to the logistic regression example from before:



LOO for more complex (linear) models

[Pregibon '81; Shao & Tu '12; Rad & Maleki '20]

Applying our approximation to the logistic regression example from before:



We *successfully* predict the effect of dropping the sample!

Beyond linear models

[Jaeckel ’72; Hampel ’74; Koh Liang ’17; Giordano Stephenson Liu Jordan Broderick ’18]

Beyond linear models

[Jaeckel ’72; Hampel ’74; Koh Liang ’17; Giordano Stephenson Liu Jordan Broderick ’18]

Everything so far only works for linear models—what about more general models?

Beyond linear models

[Jaeckel ’72; Hampel ’74; Koh Liang ’17; Giordano Stephenson Liu Jordan Broderick ’18]

Everything so far only works for linear models—what about more general models?

Another clever idea: use a Taylor approximation!

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Everything so far only works for linear models—what about more general models?

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Beyond linear models

[Jaeckel ’72; Hampel ’74; Koh Liang ’17; Giordano Stephenson Liu Jordan Broderick ’18]

Everything so far only works for linear models—what about more general models?

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

How do we compute this?

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Everything so far only works for linear models—what about more general models?

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

How do we compute this?

Start with optimality condition for $\theta^*(\mathbf{w})$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Everything so far only works for linear models—what about more general models?

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

How do we compute this?

Start with optimality condition for $\theta^*(\mathbf{w})$

$$\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Everything so far only works for linear models—what about more general models?

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = \frac{d}{d\mathbf{w}_j} 0$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

Beyond linear models

[Jaeckel ’72; Hampel ’74; Koh Liang ’17; Giordano Stephenson Liu Jordan Broderick ’18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j}\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\nabla \ell_j(\theta^*(\mathbf{w})) + \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\,\frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = 0$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w})) \frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\nabla \ell_j(\theta^*(\mathbf{w}))$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j}\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\left[\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right]\frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\nabla \ell_j(\theta^*(\mathbf{w}))$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\left[\sum_{i=1}^{n} \mathbf{w}_i \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right]^{-1} \left[\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right] \frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\left[\sum_{i=1}^{n} \mathbf{w}_i \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right]^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\left[\sum_{i=1}^{n} \mathbf{w}_i \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right]^{-1} \left[\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right] \frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\left[\sum_{i=1}^{n} \mathbf{w}_i \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right]^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j} \sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\left[\sum_{i=1}^{n} \mathbf{w}_i \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right]^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Beyond linear models

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

Another clever idea: use a Taylor approximation!

$$LOO(j) = \theta^* - \theta^*_{-j} \approx -\frac{d\theta}{d\mathbf{w}_j}$$

Start with optimality condition for $\theta^*(\mathbf{w})$ → differentiate both sides

$$\frac{d}{d\mathbf{w}_j}\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla \ell_i(\theta^*(\mathbf{w})) = 0$$

$$\frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$\frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$\frac{d\theta^*(\mathbf{w})}{d\mathbf{w}_j} = -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

Compare this to the quadratic approximation estimator:

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

Compare this to the quadratic approximation estimator:

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

Compare this to the quadratic approximation estimator:

$$\mathbf{H} = \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i} \mathbf{x_i}^\top$$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

Compare this to the quadratic approximation estimator:

$$\mathbf{H} = \sum_{i=1}^{n} \mathcal{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i}\mathbf{x_i}^\top$$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathcal{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathcal{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

Compare this to the quadratic approximation estimator:

$$\mathbf{H} = \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i}\mathbf{x_i}^\top$$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

Final estimator (Taylor's version)

[Jaeckel '72; Hampel '74; Koh Liang '17; Giordano Stephenson Liu Jordan Broderick '18]

$$LOO(j) \approx -\left(\sum_{i=1}^{n} \mathbf{w}_i \cdot \nabla^2 \ell_i(\theta^*(\mathbf{w}))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{w}))$$

Theoretical guarantees provided by [Giordano Stephenson Liu Jordan Broderick '18]

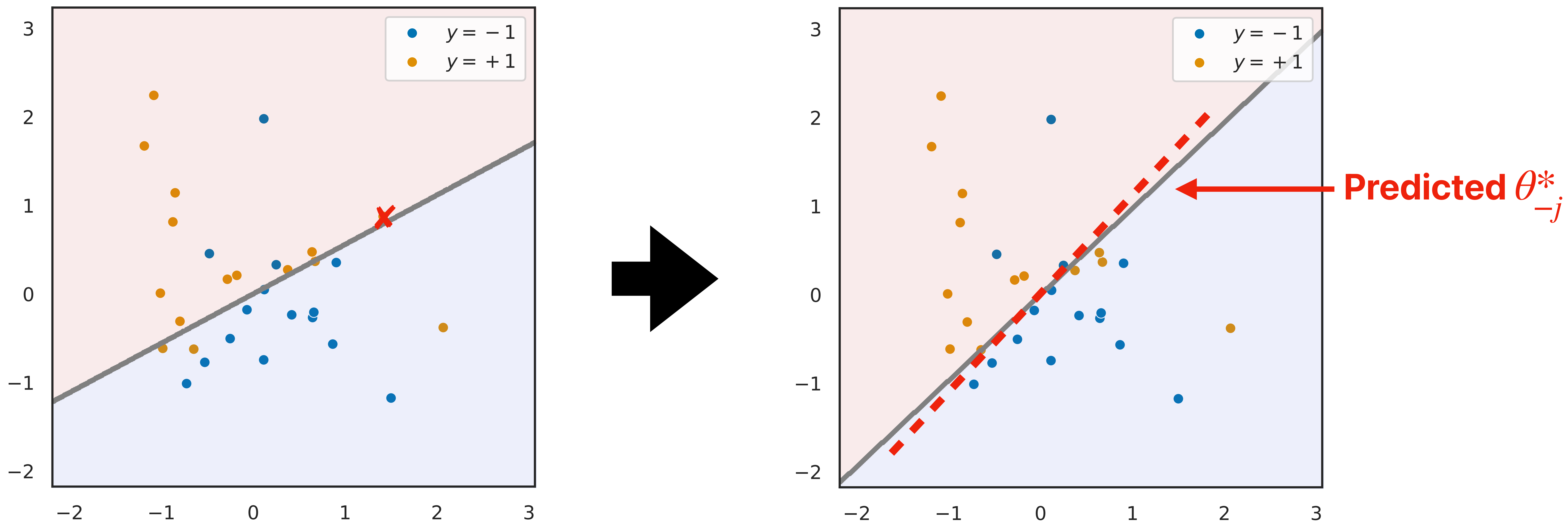
Compare this to the quadratic approximation estimator:

$$\mathbf{H} = \sum_{i=1}^{n} \mathscr{L}_i''(\theta_*^\top \mathbf{x_i}) \cdot \mathbf{x_i}\mathbf{x_i}^\top$$

$$LOO(j) \approx \frac{\mathbf{H}^{-1}\left(\mathscr{L}_j'(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}\right)}{1 - \mathscr{L}_j''(\theta^\top \mathbf{x_j}) \cdot \mathbf{x_j}^\top \mathbf{H}^{-1} \mathbf{x_j}}$$

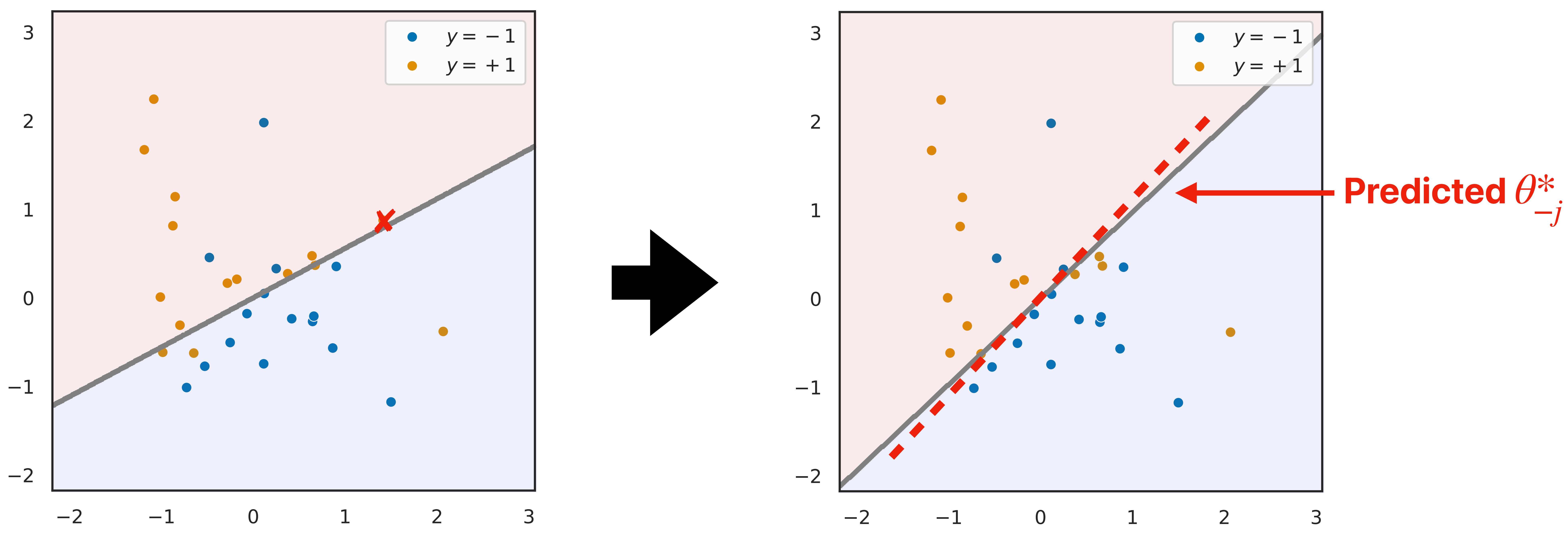
Missing! New estimator does <u>not</u> recover least squares closed form (but still effective in practice)

Final estimator (Taylor's version)



Final estimator (Taylor's version)

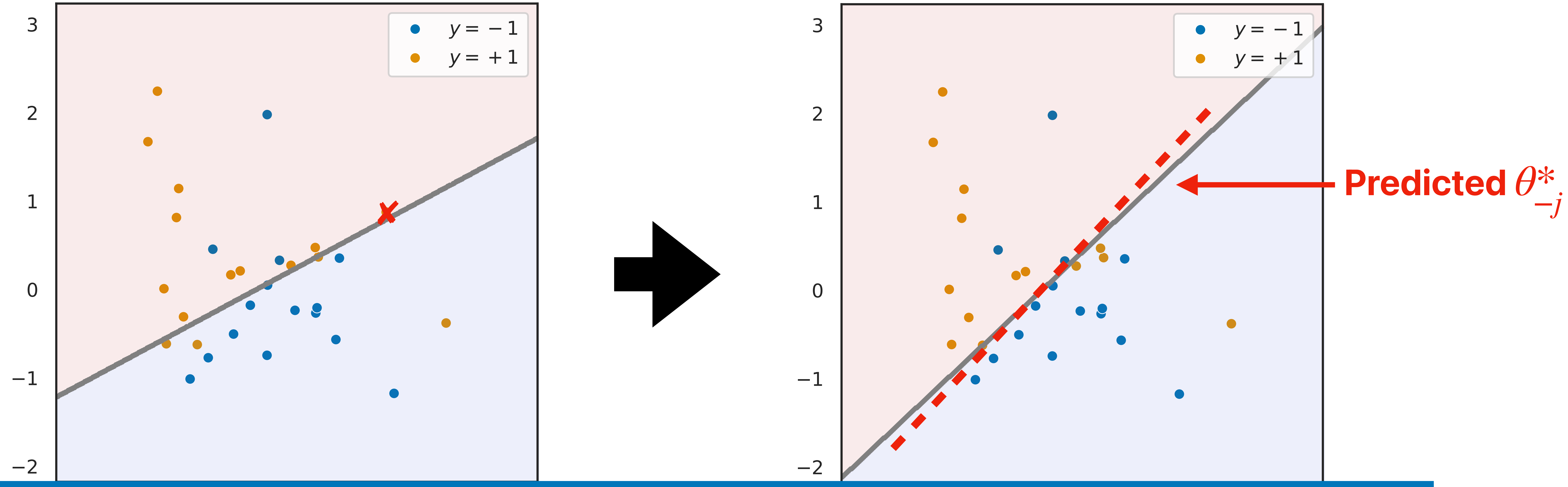
Applying our approximation to the logistic regression example from before:



We *successfully* (but less accurately) predict the effect of dropping the sample!

Final estimator (Taylor's version)

Applying our approximation to the logistic regression example from before:



We *su*

Bonus: Estimator also (trivially) applies beyond LOO estimation
See the notes for details! [GSLJB '18; KATL '19]

Takeaways

Takeaways

Well-studied statistical analog to datamodeling

Estimation of parameters under weighted re-fitting

Takeaways

Well-studied statistical analog to datamodeling

Estimation of parameters under weighted re-fitting

Variety of rigorous estimators depending on problem setting

Linear model, least-squares → closed form solution

Linear model, convex loss → quadratic approximation in parameter (θ) space

Non-linear model, convex loss → linear approximation in data weight ($\mathbf{w}$) space

Takeaways

Well-studied statistical analog to datamodeling

Estimation of parameters under weighted re-fitting

Variety of rigorous estimators depending on problem setting

Linear model, least-squares → closed form solution

Linear model, convex loss → quadratic approximation in parameter (θ) space

Non-linear model, convex loss → linear approximation in data weight ($\mathbf{w}$) space

Up next: What about non-linear, <u>non-convex</u> models?

5 minute break!

ml-data-tutorial.org

Predictive attribution in practice

Scaling to deep learning

Chapter 3 of 4

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry ’22]

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry '22]

Fix:

Data universe $\mathcal{U}$

e.g., images from Flickr,
or text from Wikipedia

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry '22]

Fix:

Data universe $\mathcal{U}$

e.g., images from Flickr, or text from Wikipedia

ML algorithm $\theta(\cdot)$

e.g., training a neural network

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry '22]

Fix:

Data universe $\mathcal{U}$

e.g., images from Flickr, or text from Wikipedia

ML algorithm $\theta(\,\cdot\,)$

e.g., training a neural network

Specific output $\ell(\,\cdot\,)$

e.g., loss of a model on a specific input

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry '22]

Fix:

Data universe $\mathcal{U}$	ML algorithm $\theta(\,\cdot\,)$	Specific output $\ell(\,\cdot\,)$
e.g., images from Flickr, or text from Wikipedia	e.g., training a neural network	e.g., loss of a model on a specific input

For any dataset S selected from $\mathcal{U}$, define $f(S)$ as the result of:

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry ’22]

Fix:

Data universe $\mathcal{U}$	ML algorithm $\theta(\,\cdot\,)$	Specific output $\ell(\,\cdot\,)$
e.g., images from Flickr, or text from Wikipedia	e.g., training a neural network	e.g., loss of a model on a specific input

For any dataset S selected from $\mathcal{U}$, define $f(S)$ as the result of:

1. Training a model $\theta(S)$ on S using the specified learning algorithm

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry '22]

Fix:

Data universe $\mathcal{U}$	ML algorithm $\theta(\cdot)$	Specific output $\ell(\cdot)$
e.g., images from Flickr, or text from Wikipedia	e.g., training a neural network	e.g., loss of a model on a specific input

For any dataset S selected from $\mathcal{U}$, define $f(S)$ as the result of:

1. Training a model $\theta(S)$ on S using the specified learning algorithm
2. Evaluating the output of interest $\ell(\cdot)$ on the trained model

Recall (Part I): Predictive data attribution

Formal definition [Ilyas Park Engstrom Leclerc Madry '22]

Fix:

Data universe $\mathcal{U}$	ML algorithm $\theta(\cdot)$	Specific output $\ell(\cdot)$
e.g., images from Flickr, or text from Wikipedia	e.g., training a neural network	e.g., loss of a model on a specific input

For any dataset S selected from $\mathcal{U}$, define $f(S)$ as the result of:

1. Training a model $\theta(S)$ on S using the specified learning algorithm
2. Evaluating the output of interest $\ell(\cdot)$ on the trained model

Datamodel: Given any dataset $S \subset \mathcal{U}$, return a prediction $\hat{f}(S) \approx f(S)$

Recall (Part II): Influence function

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n - \text{one-hot}_n(j))$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n)$

$\theta^*(\mathbf{1}_n - \textbf{one-hot}_n(j))$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n)$

$\theta^*(\mathbf{1}_n - \textbf{one-hot}_n(j))$

We found an **efficient closed-form approximation** via Taylor expansion:

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n)$

$\theta^*(\mathbf{1}_n - \textbf{one-hot}_n(j))$

We found an **efficient closed-form approximation** via Taylor expansion:

$$\theta^*_{-j} \approx \theta^* + \frac{d\theta^*(\mathbf{w})}{dw_j}$$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n)$

$\theta^*(\mathbf{1}_n - \textbf{one-hot}_n(j))$

We found an **efficient closed-form approximation** via Taylor expansion:

$$\theta^*_{-j} \approx \theta^* + \frac{d\theta^*(\mathbf{w})}{dw_j} = \theta^* + \left(\sum_{i=1}^{n} \nabla^2 \ell_i(\theta^*(\mathbf{1}_n))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{1}_n))$$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n)$

$\theta^*(\mathbf{1}_n - \textbf{one-hot}_n(j))$

We found an **efficient closed-form approximation** via Taylor expansion:

$$\theta^*_{-j} \approx \theta^* + \frac{d\theta^*(\mathbf{w})}{dw_j} = \theta^* + \left(\underbrace{\sum_{i=1}^{n} \nabla^2 \ell_i(\theta^*(\mathbf{1}_n))}_{\mathbf{H}}\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{1}_n))$$

Recall (Part II): Influence function

We focused on strongly convex weighted loss minimization:

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{n} w_i \cdot \ell_i(\theta)$$

(strongly convex) loss on the i-th training example

Special case: **leave-one-out (LOO)** estimation

$$LOO(j) := \theta^* - \theta^*_{-j}$$

$\theta^*(\mathbf{1}_n)$

$\theta^*(\mathbf{1}_n - \textbf{one-hot}_n(j))$

We found an **efficient closed-form approximation** via Taylor expansion:

$$\theta^*_{-j} \approx \theta^* + \frac{d\theta^*(\mathbf{w})}{dw_j} = \theta^* + \left(\sum_{i=1}^{n} \nabla^2 \ell_i(\theta^*(\mathbf{1}_n))\right)^{-1} \nabla \ell_j(\theta^*(\mathbf{1}_n))$$

$\mathbf{H}$

$\mathbf{g_j}$

First idea: direct translation (convex → ML)

[Koh Liang ’17]

First idea: direct translation (convex → ML)

[Koh Liang ’17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

First idea: direct translation (convex → ML)

[Koh Liang '17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

Fix:

Data universe $\mathcal{U}$	ML algorithm A	Specific output $\ell(\cdot)$
Dataset S	$\theta(S) = \arg\min_\theta \sum_{z_i \in S}^{n} \ell_i(\theta)$	Test loss $\ell(\theta) = \theta$

First idea: direct translation (convex → ML)

[Koh Liang '17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

Fix:

Data universe $\mathcal{U}$	ML algorithm A	Specific output $\ell(\cdot)$
Dataset S	$\theta(S) = \arg\min_\theta \sum_{z_i \in S}^{n} \ell_i(\theta)$	Test loss $\ell(\theta) = \theta$

Datamodeling: Given example z_j, predict $f(\mathcal{U} \setminus \{z_j\})$ (exactly the LOO problem!)

First idea: direct translation (convex → ML)

[Koh Liang '17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

Fix:

Data universe $\mathcal{U}$	ML algorithm A	Specific output $\ell(\cdot)$
Dataset S	$\theta(S) = \arg\min_\theta \sum_{z_i \in S}^{n} \ell_i(\theta)$	Test loss $\ell(\theta) = \theta$

Datamodeling: Given example z_j, predict $f(\mathcal{U} \setminus \{z_j\})$ (exactly the LOO problem!)

Taylor-based estimator **(Influence function)**: $\hat{f}(S \setminus \{z_j\}) := f(S) - \mathbf{H}^{-1}\mathbf{g_j}$

First idea: direct translation (convex → ML)

[Koh Liang '17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

Fix:

Data universe $\mathcal{U}$	ML algorithm A	Specific output $\ell(\cdot)$
Dataset S	$\theta(S) =$ SGD on a DNN	Test loss $\ell(\theta) = \theta$

Datamodeling: Given example z_j, predict $f(\mathcal{U} \setminus \{z_j\})$ (exactly the LOO problem!)

Taylor-based estimator **(Influence function)**: $\hat{f}(S \setminus \{z_j\}) := f(S) - \mathbf{H}^{-1}\mathbf{g_j}$

First idea: direct translation (convex → ML)

[Koh Liang '17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

Fix:

Data universe $\mathscr{U}$	ML algorithm A	Specific output $\ell(\cdot)$
Dataset S	$\theta(S) =$ SGD on a DNN	Loss on a specific example

Datamodeling: Given example z_j, predict $f(\mathscr{U} \setminus \{z_j\})$ (exactly the LOO problem!)

Taylor-based estimator **(Influence function)**: $\hat{f}(S \setminus \{z_j\}) := f(S) - \mathbf{H}^{-1}\mathbf{g_j}$

First idea: direct translation (convex → ML)

[Koh Liang '17]

Note: LOO is a **special case** of the predictive data attribution/datamodeling problem

Fix:

Data universe $\mathcal{U}$ | ML algorithm A | Specific output $\ell(\cdot)$

Dataset S | $\theta(S) =$ SGD on a DNN | Loss on a specific example

Datamodeling: Given example z_j, predict $f(\mathcal{U} \setminus \{z_j\})$ (exactly the LOO problem!)

Taylor-based estimator **(Influence function)**: $\hat{f}(S \setminus \{z_j\}) := f(S) - \mathbf{H}^{-1}\mathbf{g_j}$

Does this work for NNs? (**Strong convexity** & **convergence** **both violated!**)

First idea: direct translation (convex → ML)

[Koh Liang '17]

First idea: direct translation (convex → ML)

[Koh Liang '17]

Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \setminus \{z_j\}) := f(S) - \nabla \ell(\theta)^\top \mathbf{H}^{-1} \mathbf{g_j}$$

First idea: direct translation (convex → ML)

[Koh Liang '17]

Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \backslash \{z_j\}) := f(S) - \nabla \ell(\theta)^\top \mathbf{H}^{-1} \mathbf{g_j}$$

Train one model

First idea: direct translation (convex → ML)

[Koh Liang ’17]

Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \setminus \{z_j\}) := f(S) - \nabla \ell(\theta)^\top \mathbf{H}^{-1} \mathbf{g_j}$$

Train one model

Compute its Hessian

First idea: direct translation (convex → ML)

[Koh Liang '17]

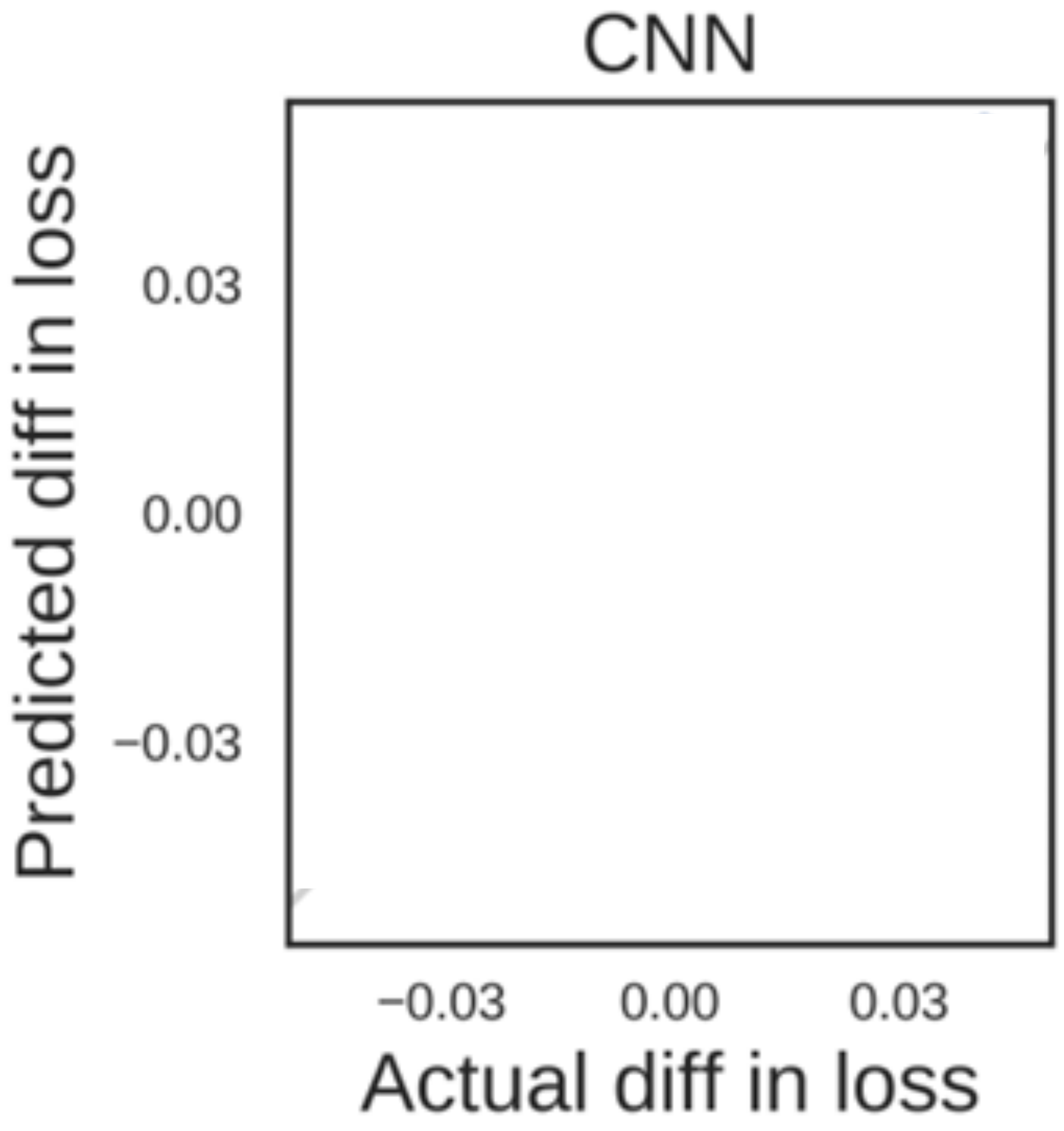
Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \setminus \{z_j\}) := f(S) - \nabla \ell(\theta)^\top \mathbf{H}^{-1} \mathbf{g_j}$$

Train one model

Compute its Hessian



First idea: direct translation (convex → ML)

[Koh Liang '17]

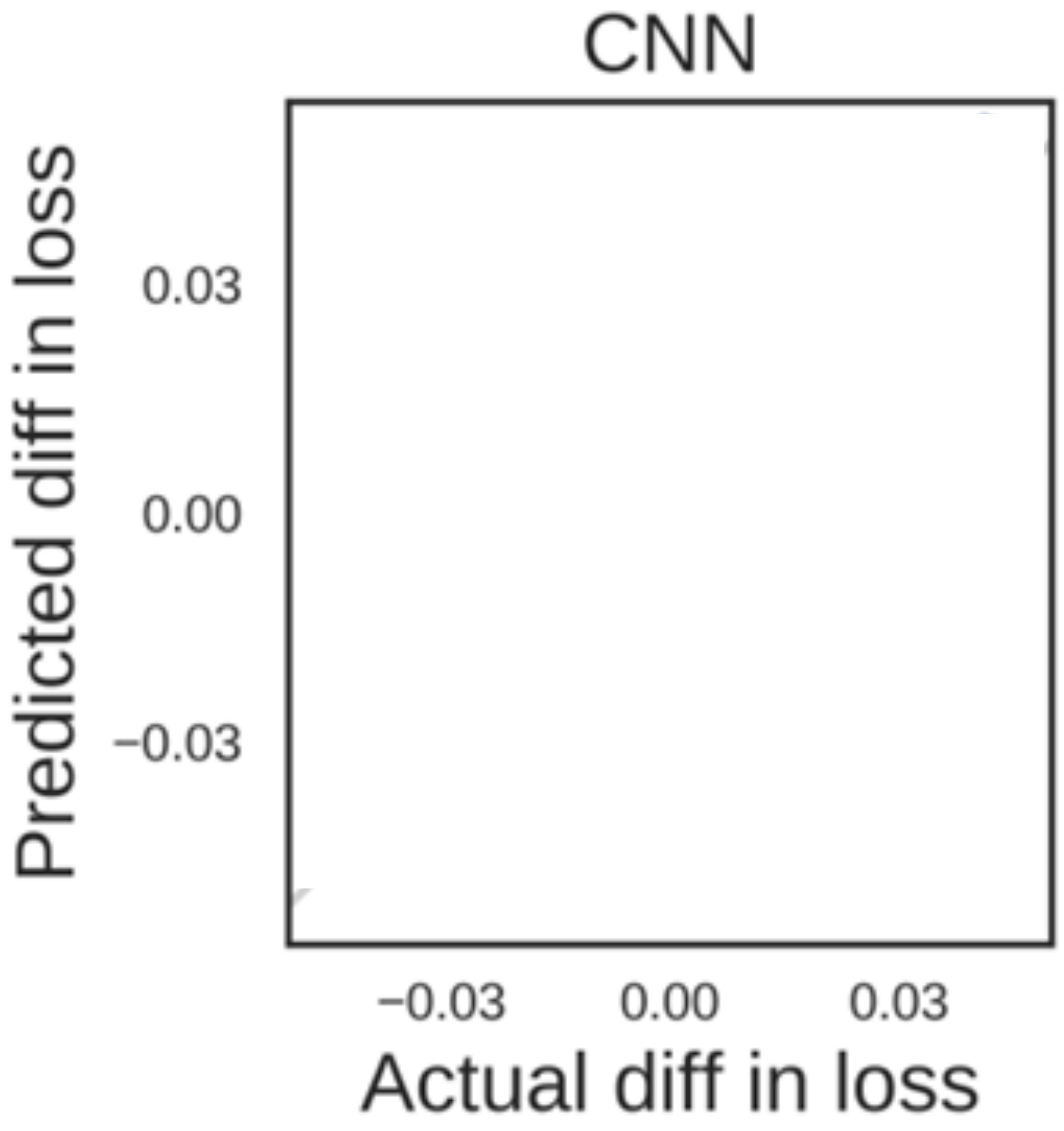
Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \setminus \{z_j\}) := f(S) - \nabla \ell(\theta)^{\top} \mathbf{H}^{-1} \mathbf{g_j}$$

Train one model

Compute its Hessian



Re-training

First idea: direct translation (convex → ML)

[Koh Liang '17]

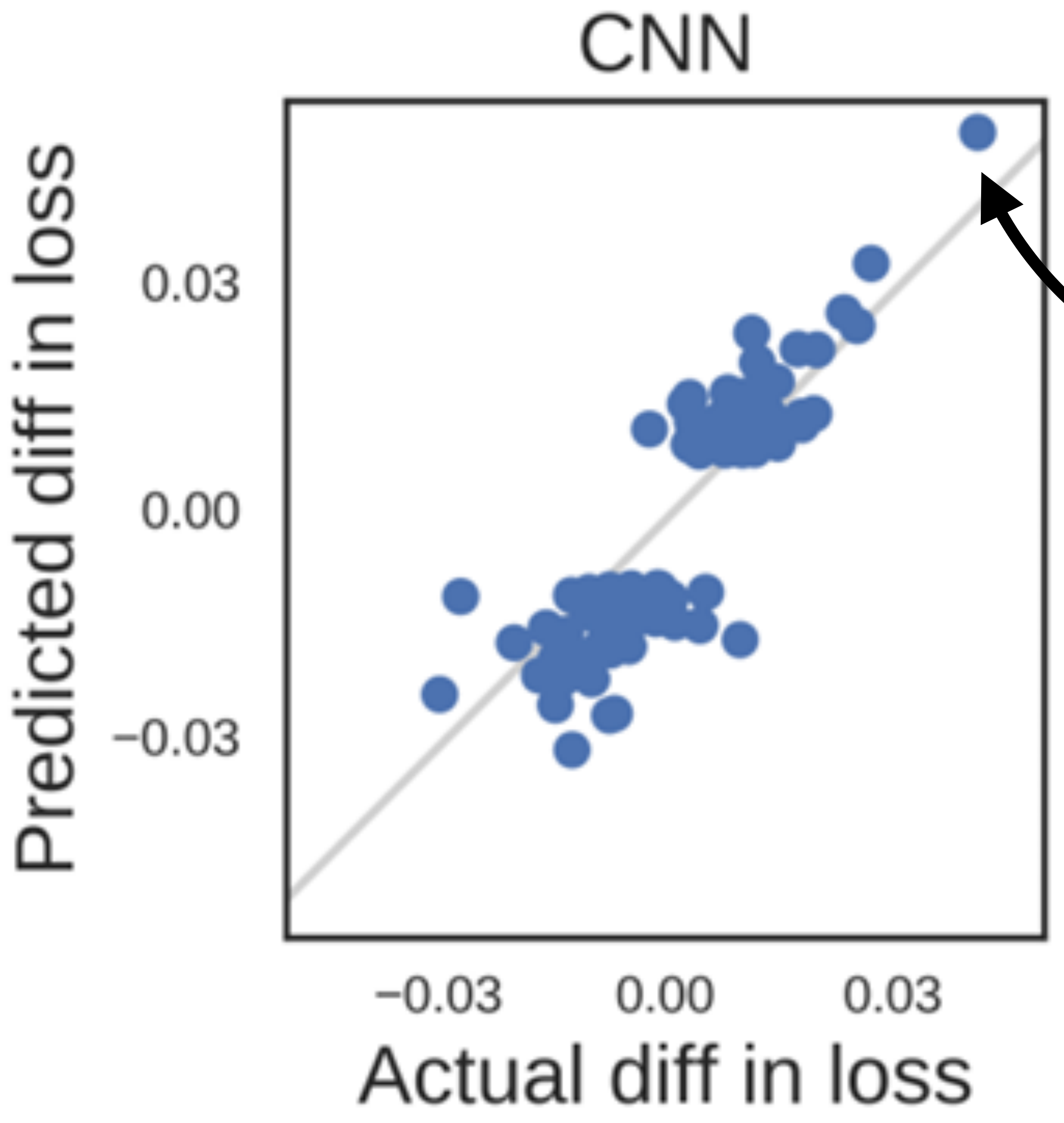
Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \backslash \{z_j\}) := f(S) - \nabla \ell(\theta)^\top \mathbf{H}^{-1} \mathbf{g_j}$$

Train one model

Compute its Hessian



Re-training

Each ● is a single training example

First idea: direct translation (convex → ML)

[Koh Liang '17]

Result: Seems to work in some settings (small CNNs)

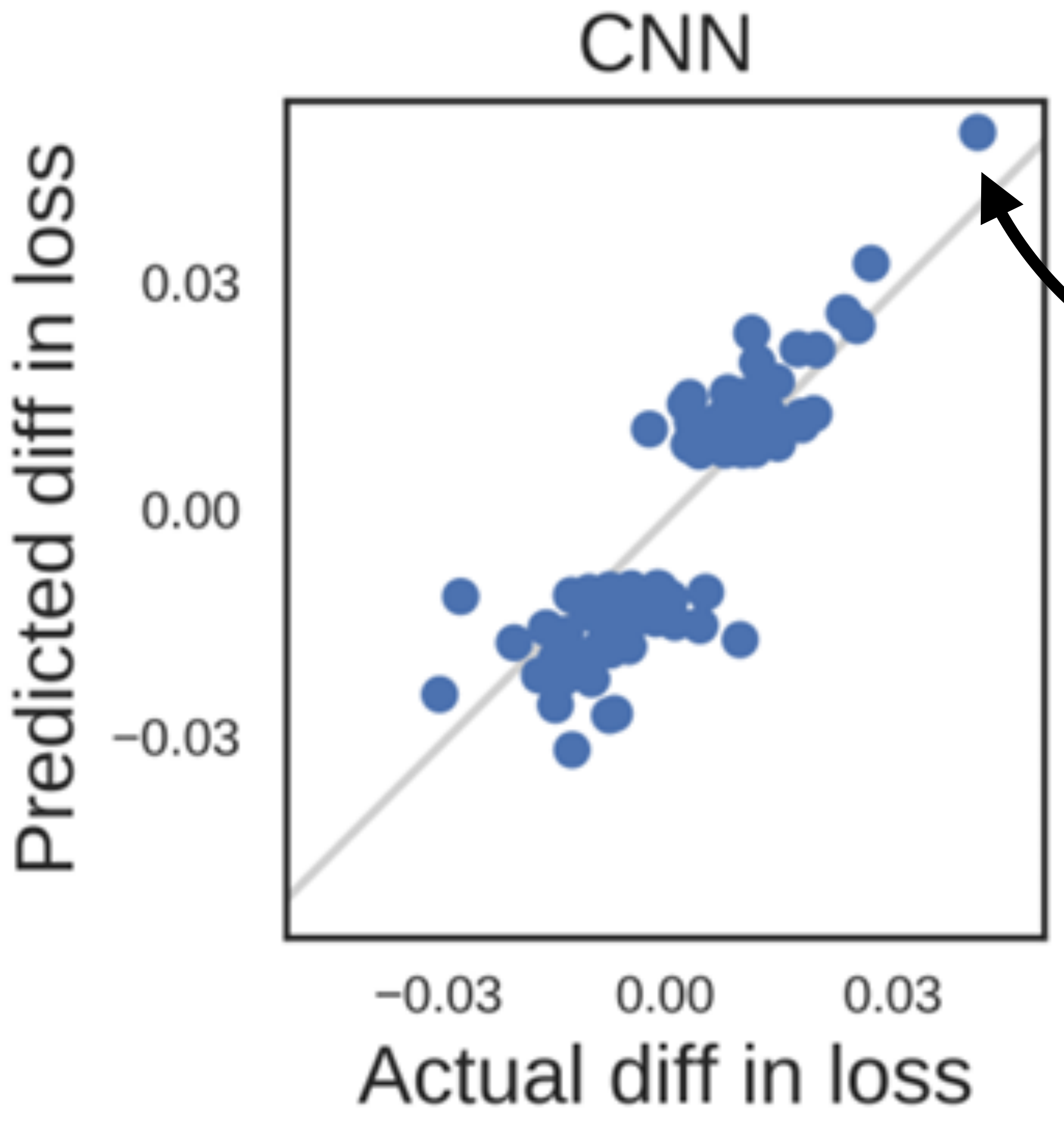
Influence function (IF) estimator

(Taylor-based LOO in loss space)

$$\hat{f}(S \setminus \{z_j\}) := f(S) - \nabla \ell(\theta)^\top \mathbf{H}^{-1} \mathbf{g_j}$$

Train one model

Compute its Hessian



Each ● is a single training example

Re-training

First idea: direct translation (convex → ML)

[Koh Liang ’17; Schioppa, Zablotskaia, Vilar, Sokolov ’22]

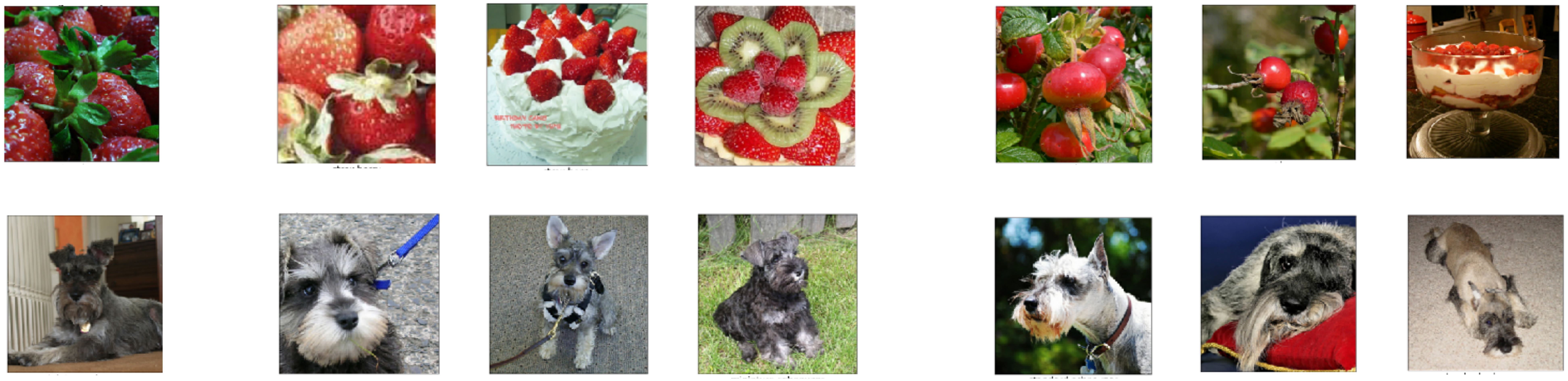
Qualitatively, do the examples with largest LOO effects “make sense”?

First idea: direct translation (convex → ML)

[Koh Liang '17; Schioppa, Zablotskaia, Vilar, Sokolov '22]

Qualitatively, do the examples with largest LOO effects "make sense"?

Target | Positively influencing | Negatively influencing

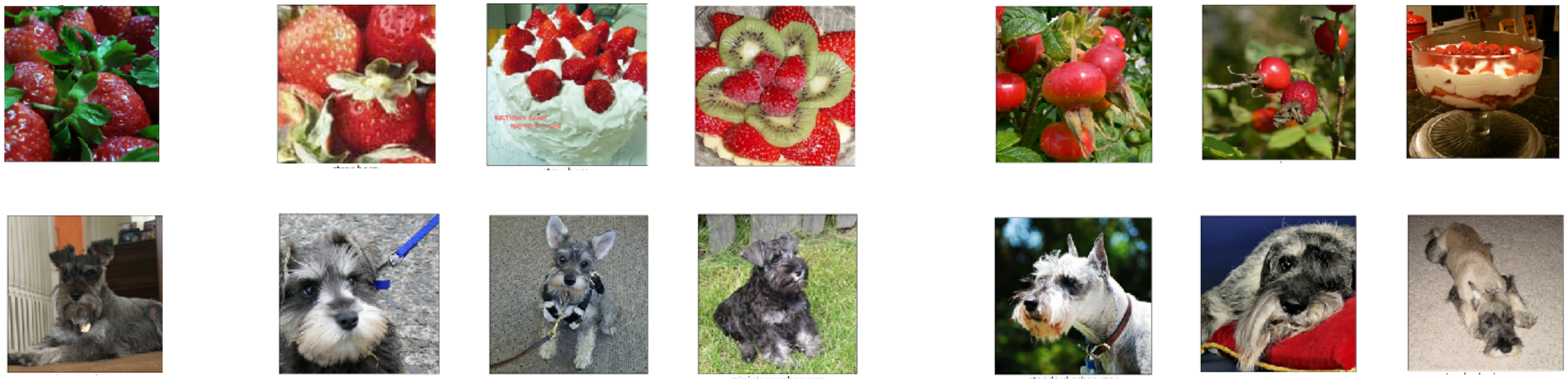


First idea: direct translation (convex → ML)

[Koh Liang ’17; Schioppa, Zablotskaia, Vilar, Sokolov ’22]

Qualitatively, do the examples with largest LOO effects “make sense”?

Target | Positively influencing | Negatively influencing



Promising sign: training examples with biggest predicted LOO effect on loss resemble the “target” test examples!

Does the influence function work reliably?

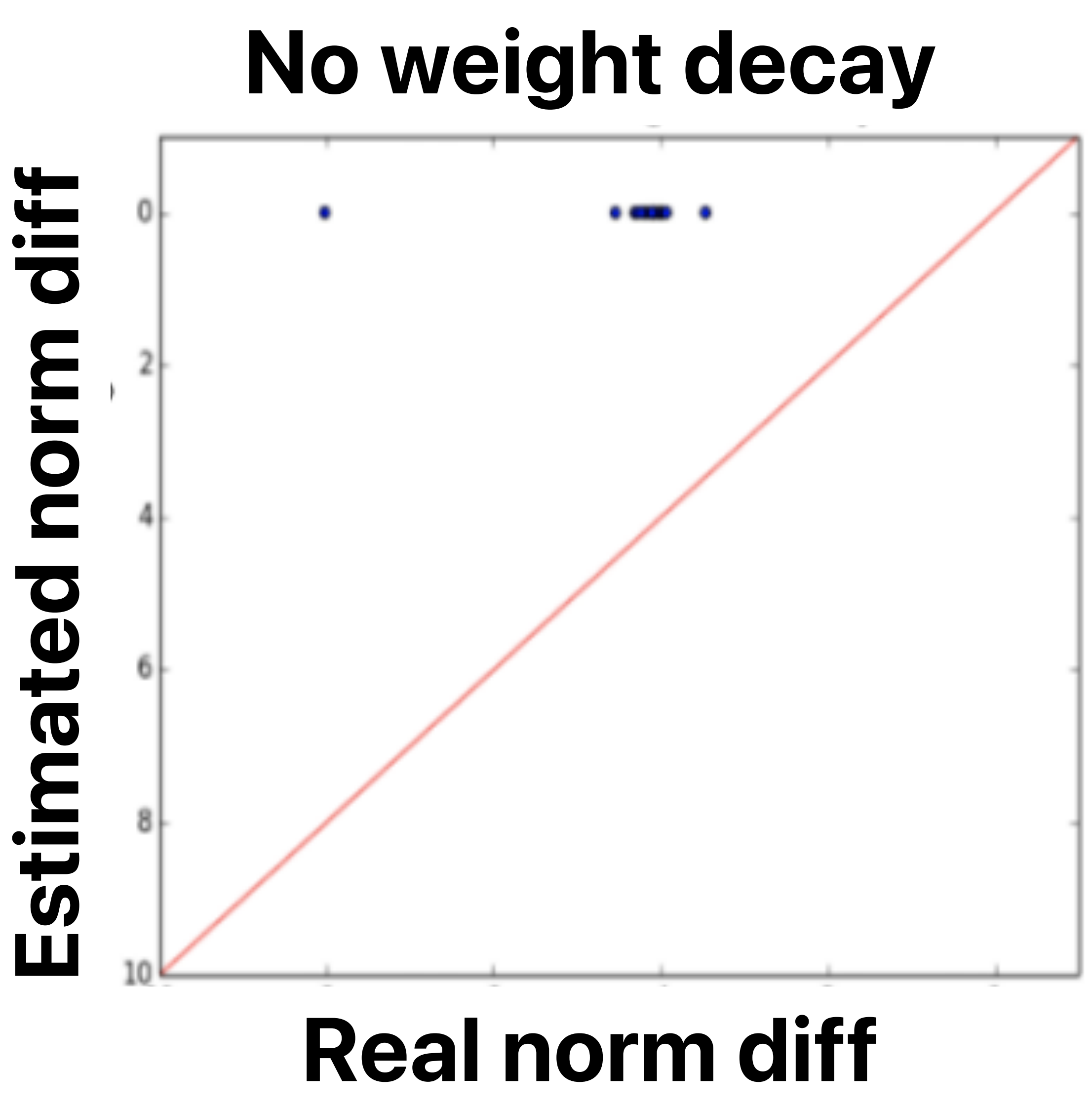
Drawbacks of the direct approach

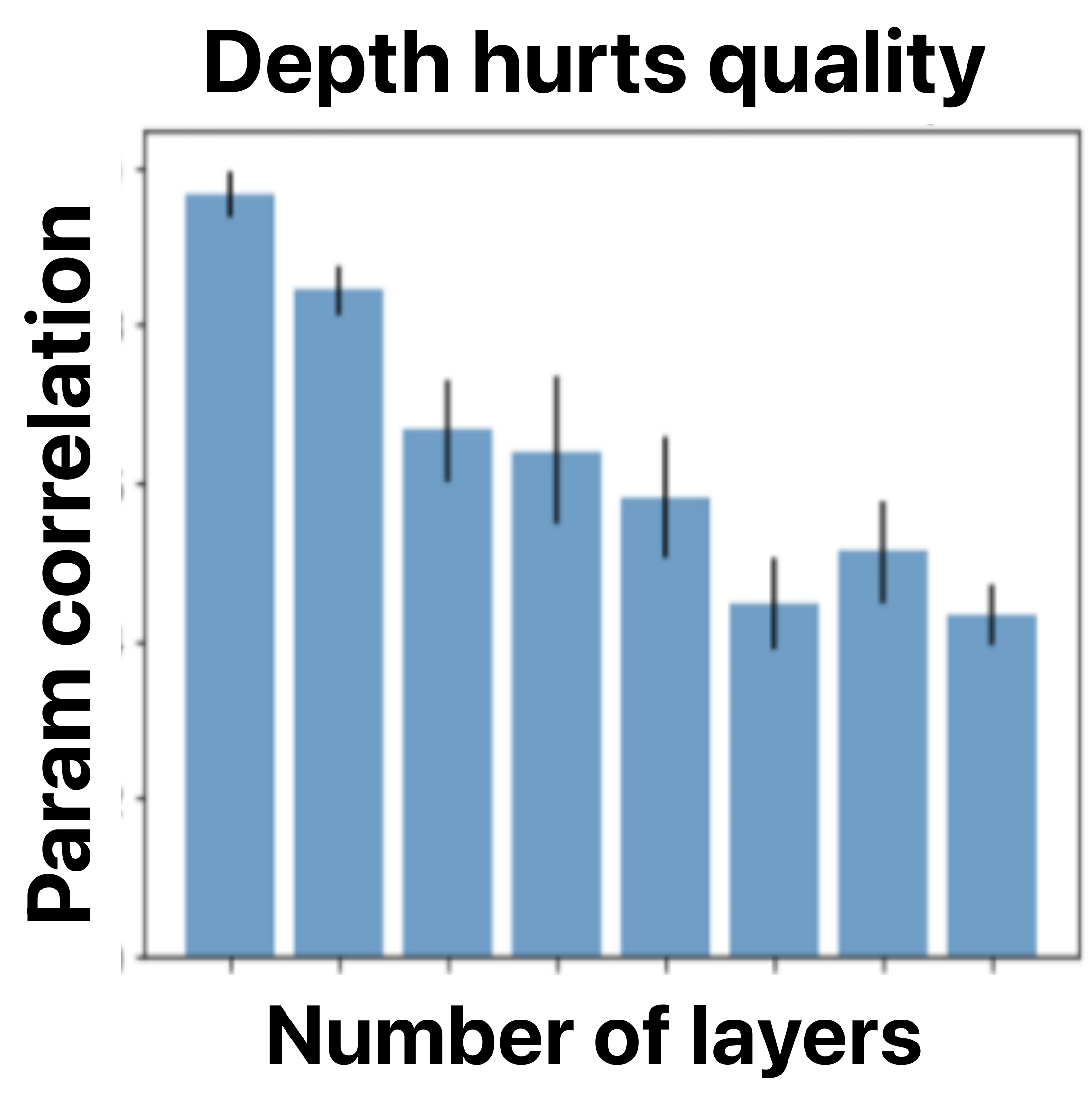
Does the influence function work reliably?

Drawbacks of the direct approach

IF seems to be fragile/limited to small models [Basu Pope Feizi '17]

Sensitive to hyperparameters; worse for deeper & wider models



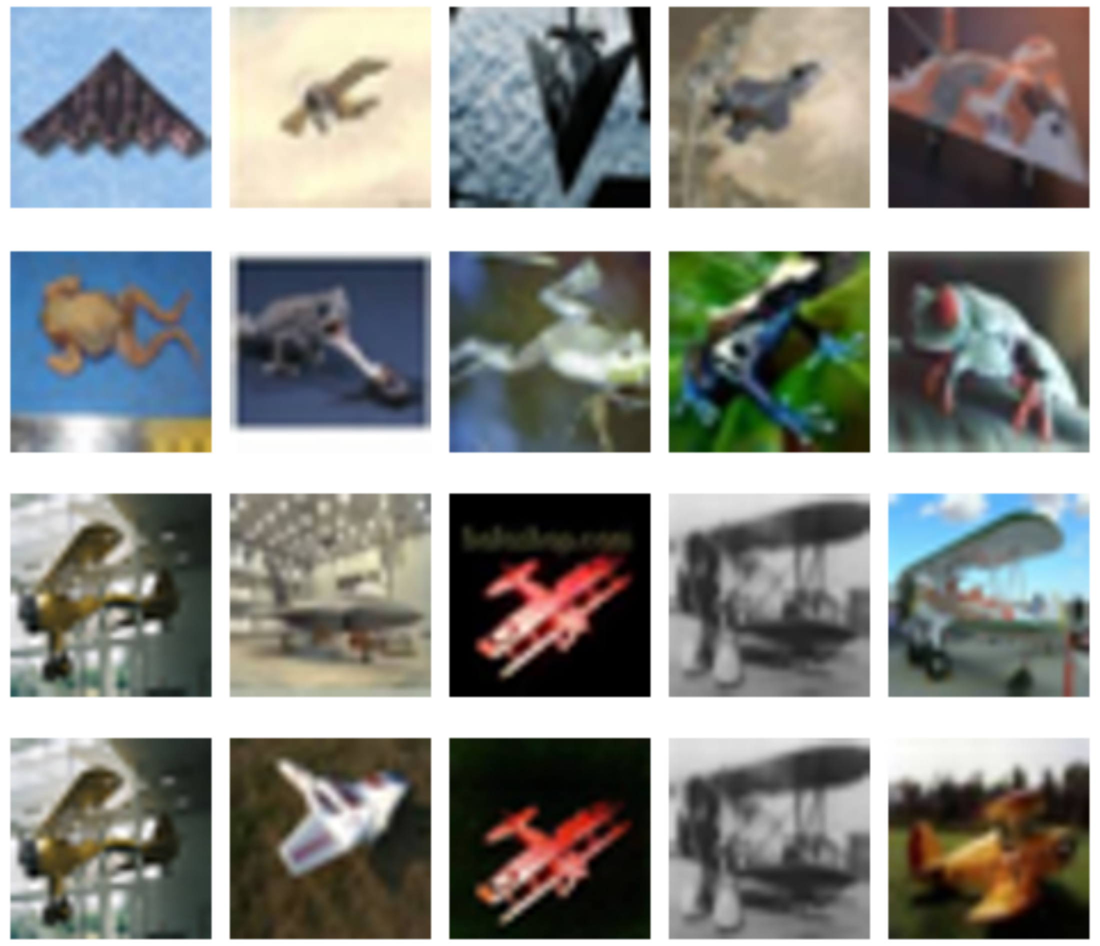


Does the influence function work reliably?

Drawbacks of the direct approach

IF can't distinguish examples from different dataset [Hammoudeh Lowd '22]

- **Setup**: training set S consisting of both CIFAR-10 and MNIST examples
- Apply LOO estimator (and related ones) to $\ell(\theta)$ = loss on a single example
- Visualize training examples with the highest LOO effect



Influence function

Representer point

TracInCP

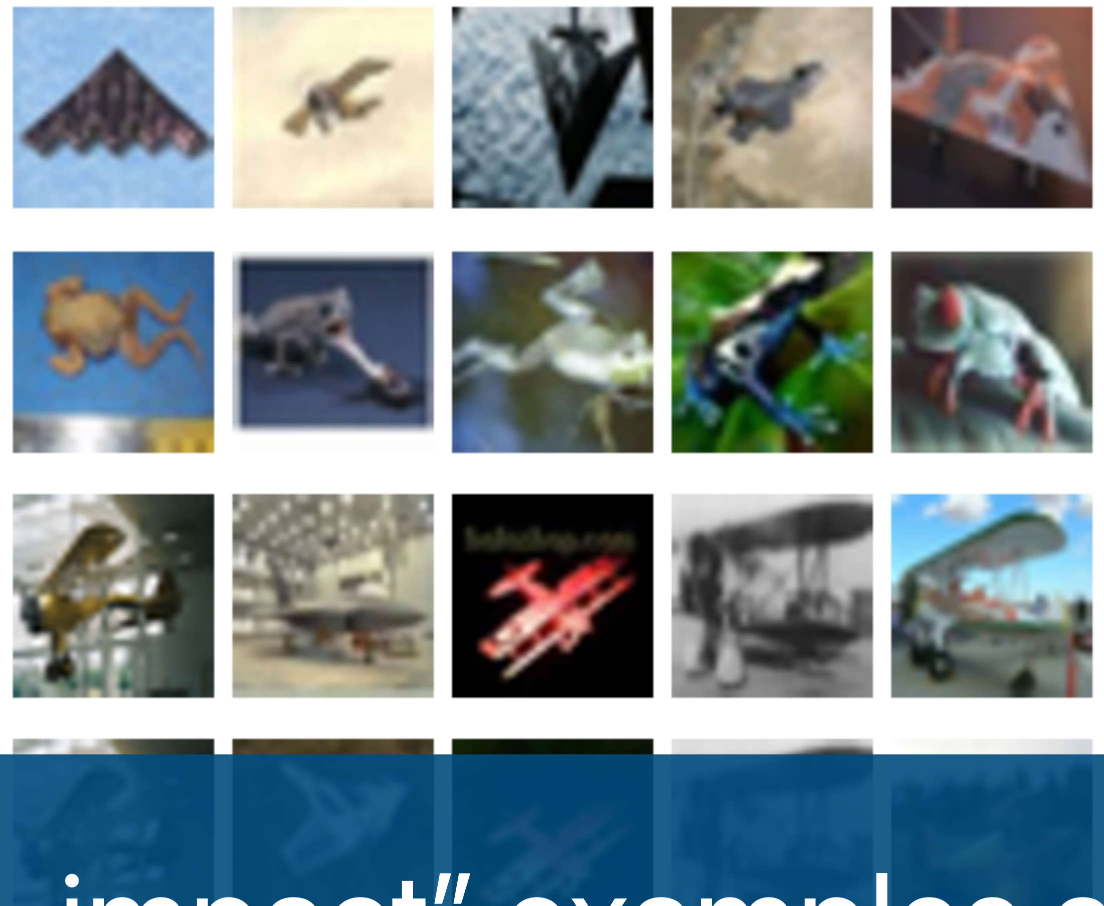
TracIn

Does the influence function work reliably?

Drawbacks of the direct approach

IF can't distinguish examples from different dataset [Hammoudeh Lowd '22]

- **Setup**: training set S consisting of both CIFAR-10 and MNIST examples
- Apply LOO estimator (and related ones) to $\ell(\theta)$ = loss on a single example
- Visualize training examples with the highest LOO effect



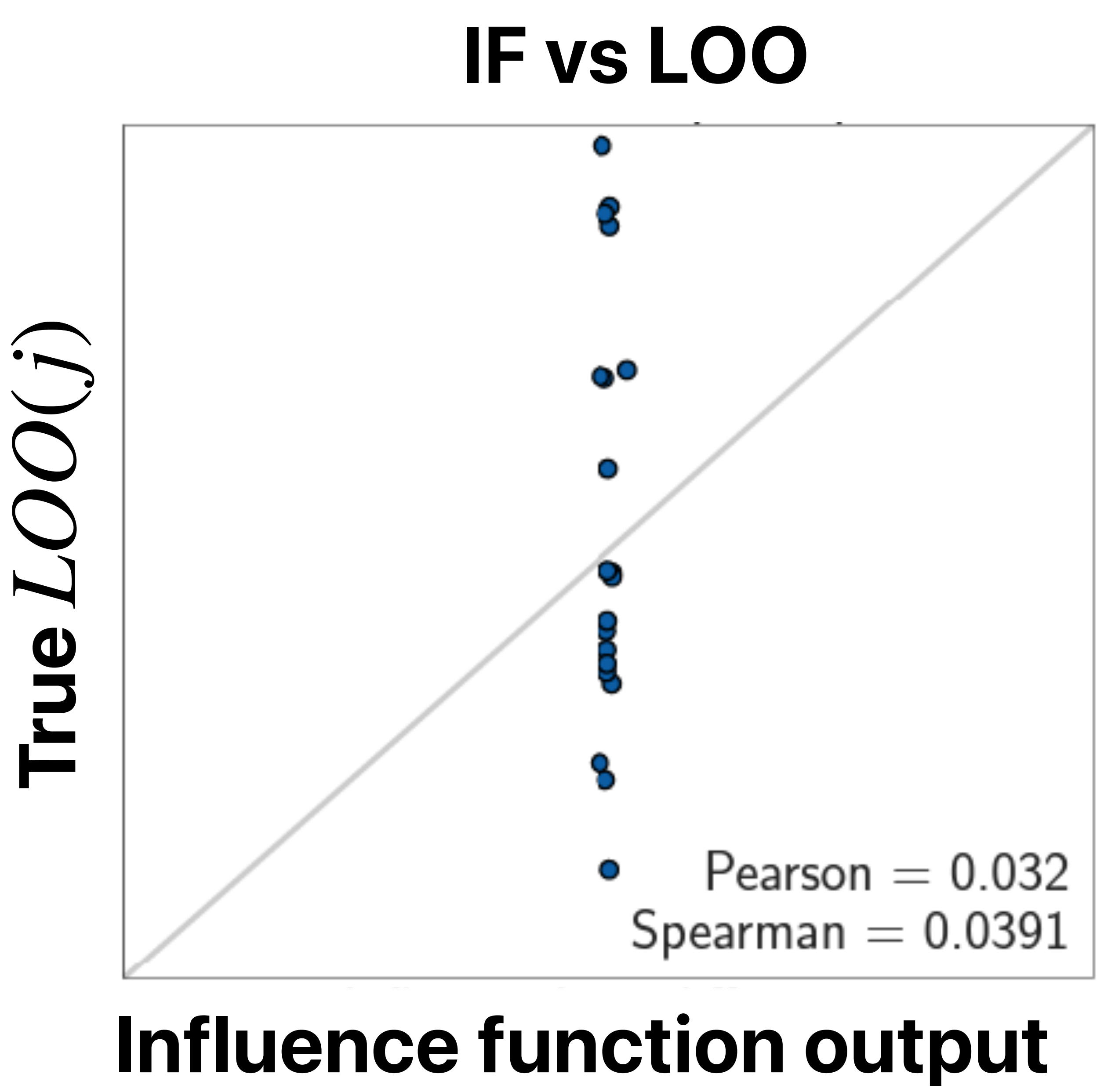
Failed sanity check: "highest-impact" examples are from other dataset!

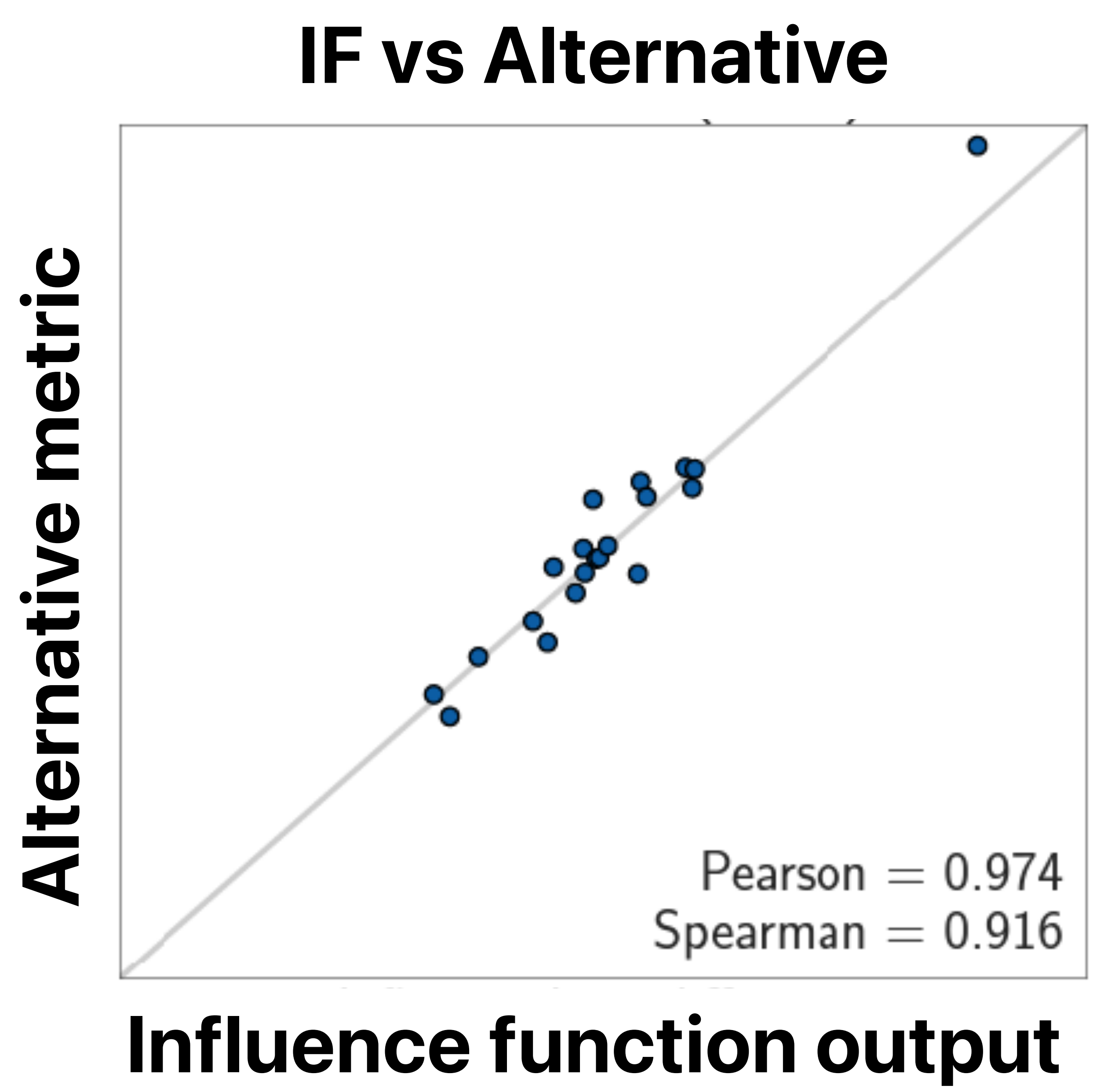
Does the influence function work reliably?

Drawbacks of the direct approach

IF does not actually approximate re-training [Bae Ng Lo Ghassemi Grosse '22]

Lack of convexity, convergence, etc. mean that influence functions do **not** actually approximate LOO (unlike in the convex case)





Where do we go from here?

So far: Datamodeling/predictive attribution problem

Where do we go from here?

So far: Datamodeling/predictive attribution problem

Promising method from statistics → tried direct application in ML

Where do we go from here?

So far: Datamodeling/predictive attribution problem

Promising method from statistics → tried direct application in ML

Ran into some issues:

- Parameter-space updates are fragile [BPF17]
- “Most impactful” training examples can fail basic sanity checks [HL22]
- Even conceptually, might not be an LOO approximation [BNLGG22]

Where do we go from here?

So far: Datamodeling/predictive attribution problem

Promising method from statistics → tried direct application in ML

Ran into some issues:

- Parameter-space updates are fragile [BPF17]
- “Most impactful” training examples can fail basic sanity checks [HL22]
- Even conceptually, might not be an LOO approximation [BNLGG22]

What now?

- Try new ways to adapt classical (statistical) LOO estimator?
- Give up and try ML-specific techniques for estimating LOO?

Where do we go from here?

So far: Datamodeling/predictive attribution problem

Promising method from statistics → tried direct application in ML

Ran into some issues:

- Parameter-space updates are fragile [BPF17]
- "Most impactful" training examples can fail basic sanity checks [HL22]
- Even conceptually, might not be an LOO approximation [BNLGG22]

What now?

- Try new ways to adapt classical (statistical) LOO estimator?
- Give up and try ML-specific techniques for estimating LOO?

Clearly room for improvement—but how will we measure progress?

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Recall: output of predictive data attribution is a datamodel function $\hat{f}$ such that

$$\hat{f}(S) \approx \ell(\mathscr{A}(S)) \text{ for any } S \subset \mathscr{U}$$

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Recall: output of predictive data attribution is a datamodel function $\hat{f}$ such that

$$\hat{f}(S) \approx \ell(\mathcal{A}(S)) \text{ for any } S \subset \mathcal{U}$$

Idea: Treat this as a learning problem, use *population error* to evaluate success:

$$\text{LDS}_{\mathcal{D}} := \mathbb{E}_{S^{(1)}\ldots S^{(m)}\sim\mathcal{D}}\left[\text{Correlation}\left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m}\right)\right]$$

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Recall: output of predictive data attribution is a datamodel function $\hat{f}$ such that

$$\hat{f}(S) \approx \ell(\mathcal{A}(S)) \text{ for any } S \subset \mathcal{U}$$

Idea: Treat this as a learning problem, use *population error* to evaluate success:

$$\text{LDS}_{\mathcal{D}} := \mathbb{E}_{S^{(1)}\ldots S^{(m)}\sim\mathcal{D}}\left[\text{Correlation}\left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m}\right)\right]$$

distribution over subsets

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Recall: output of predictive data attribution is a datamodel function $\hat{f}$ such that

$$\hat{f}(S) \approx \ell(\mathcal{A}(S)) \text{ for any } S \subset \mathcal{U}$$

Idea: Treat this as a learning problem, use *population error* to evaluate success:

$$\text{LDS}_{\mathcal{D}} := \mathbb{E}_{S^{(1)}\ldots S^{(m)} \sim \mathcal{D}} \left[\text{Correlation} \left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m} \right) \right]$$

distribution over subsets

subsets sampled from the distribution

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Recall: output of predictive data attribution is a datamodel function $\hat{f}$ such that

$$\hat{f}(S) \approx \ell(\mathcal{A}(S)) \text{ for any } S \subset \mathcal{U}$$

Idea: Treat this as a learning problem, use *population error* to evaluate success:

$$\text{LDS}_{\mathcal{D}} := \mathbb{E}_{S^{(1)}\dots S^{(m)} \sim \mathcal{D}} \left[\text{Correlation} \left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m} \right) \right]$$

distribution over subsets

subsets sampled from the distribution

true output of model trained on subset

Linear datamodeling score

Making predictive data attribution quantitative [KAT+19; IPE+22; PGI+23]

Recall: output of predictive data attribution is a datamodel function $\hat{f}$ such that

$$\hat{f}(S) \approx \ell(\mathcal{A}(S)) \text{ for any } S \subset \mathcal{U}$$

Idea: Treat this as a learning problem, use *population error* to evaluate success:

$$\text{LDS}_{\mathcal{D}} := \mathbb{E}_{S^{(1)}\ldots S^{(m)}\sim\mathcal{D}}\left[\text{Correlation}\left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m}\right)\right]$$

distribution over subsets

subsets sampled from the distribution

true output of model trained on subset

predicted output of model trained on subset

Linear datamodeling score

Making predictive data attribution quantitative

Linear datamodeling score

Making predictive data attribution quantitative

Example (Influence function estimator):

Linear datamodeling score

Making predictive data attribution quantitative

Example (Influence function estimator):

1. Sample many random 50% subsets $S^{(1)}, S^{(2)}, \ldots, S^{(m)} \subset U$

Linear datamodeling score

Making predictive data attribution quantitative

Example (Influence function estimator):

1. Sample many random 50% subsets $S^{(1)}, S^{(2)}, \ldots, S^{(m)} \subset U$

2. Apply learning algorithm $\mathscr{A}$ to each $S^{(m)}$ to obtain $f(S^{(i)})$

Linear datamodeling score

Making predictive data attribution quantitative

Example (Influence function estimator):

1. Sample many random 50% subsets $S^{(1)}, S^{(2)}, \ldots, S^{(m)} \subset U$

2. Apply learning algorithm $\mathscr{A}$ to each $S^{(m)}$ to obtain $f(S^{(i)})$

3. Construct predicted output using LOO estimates (e.g., using the IF):

$$\hat{f}(S') := f(S) - \sum_{i \in S \setminus S'} LOO(i)$$

Linear datamodeling score

Making predictive data attribution quantitative

Example (Influence function estimator):

1. Sample many random 50% subsets $S^{(1)}, S^{(2)}, \ldots, S^{(m)} \subset U$

2. Apply learning algorithm $\mathscr{A}$ to each $S^{(m)}$ to obtain $f(S^{(i)})$

3. Construct predicted output using LOO estimates (e.g., using the IF):

$$\hat{f}(S') := f(S) - \sum_{i \in S \setminus S'} LOO(i)$$

4. Measure Correlation $\left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m} \right)$

Linear datamodeling score

Making predictive data attribution quantitative

Example (Influence function estimator):

1. Sample many random 50% subsets $S^{(1)}, S^{(2)}, \ldots, S^{(m)} \subset U$

2. Apply learning algorithm $\mathscr{A}$ to each $S^{(m)}$ to obtain $f(S^{(i)})$

Can pre-compute just once!
Evaluate any new estimates

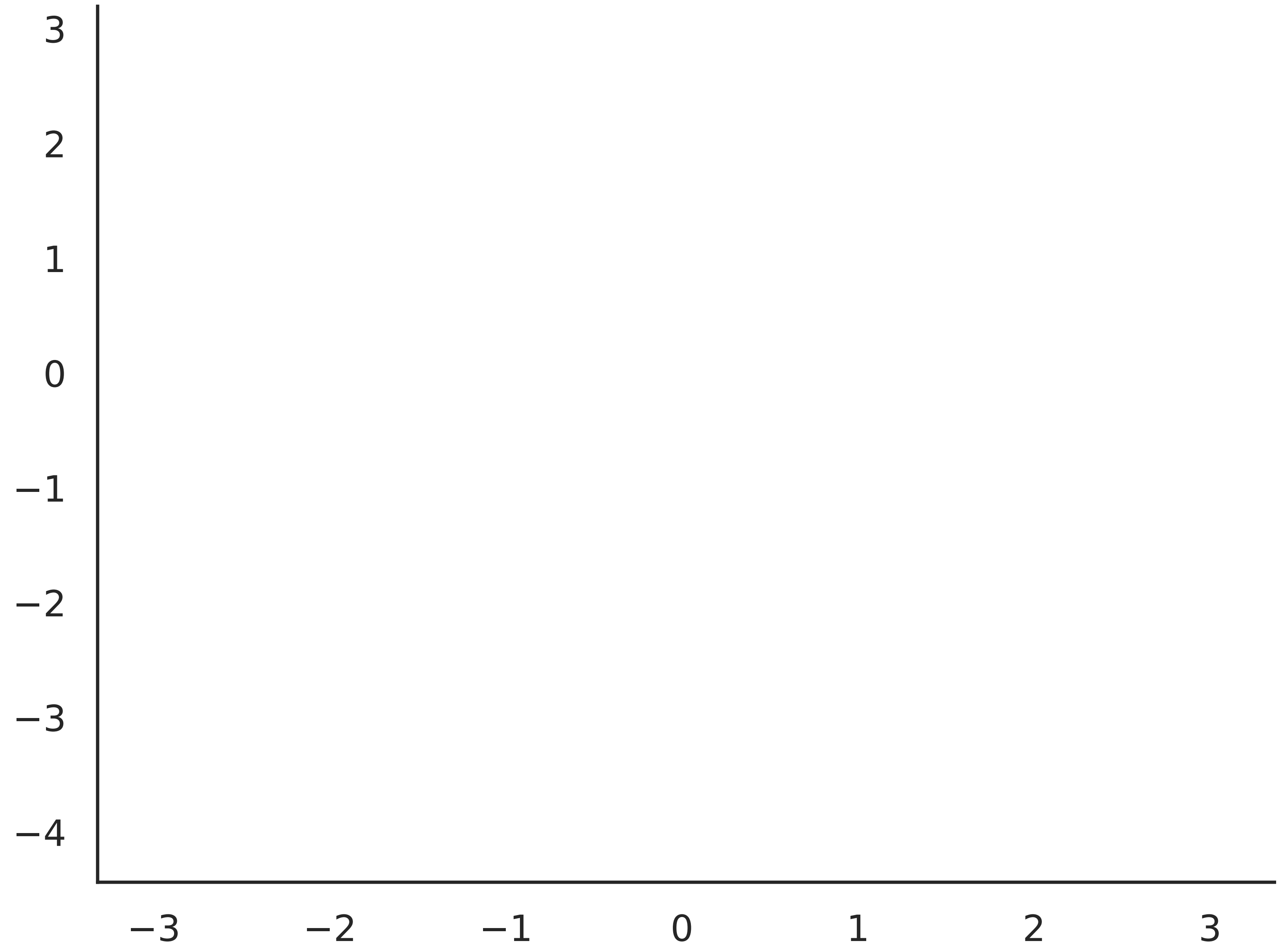
3. Construct predicted output using LOO estimates (e.g., using the IF):

$$\hat{f}(S') := f(S) - \sum_{i \in S \setminus S'} LOO(i)$$

4. Measure Correlation $\left(\{f(S^{(i)})\}_{i=1}^{m}, \{\hat{f}(S^{(i)})\}_{i=1}^{m} \right)$

Results: influence function estimator

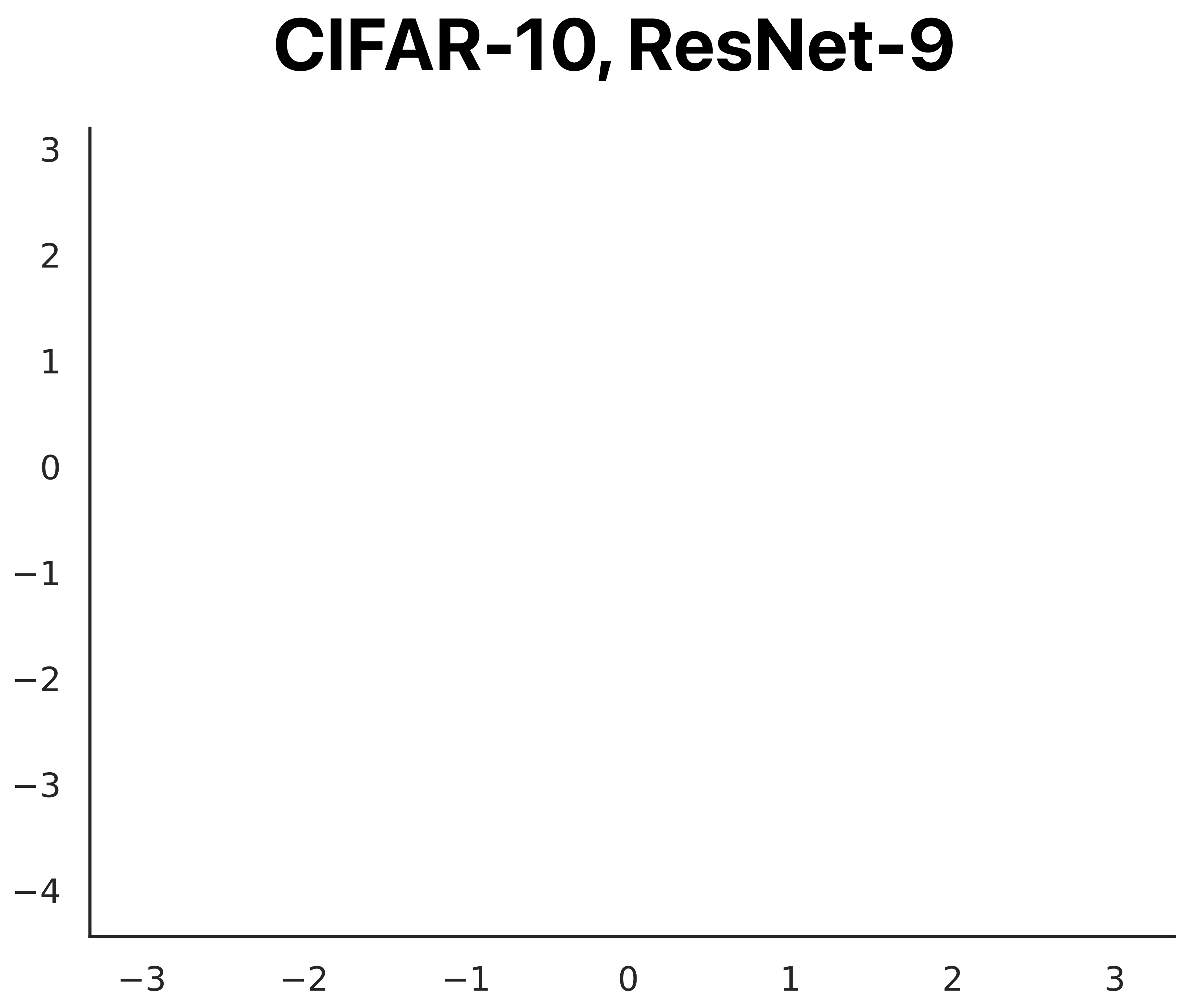
CIFAR-10, ResNet-9



Results: influence function estimator

True output

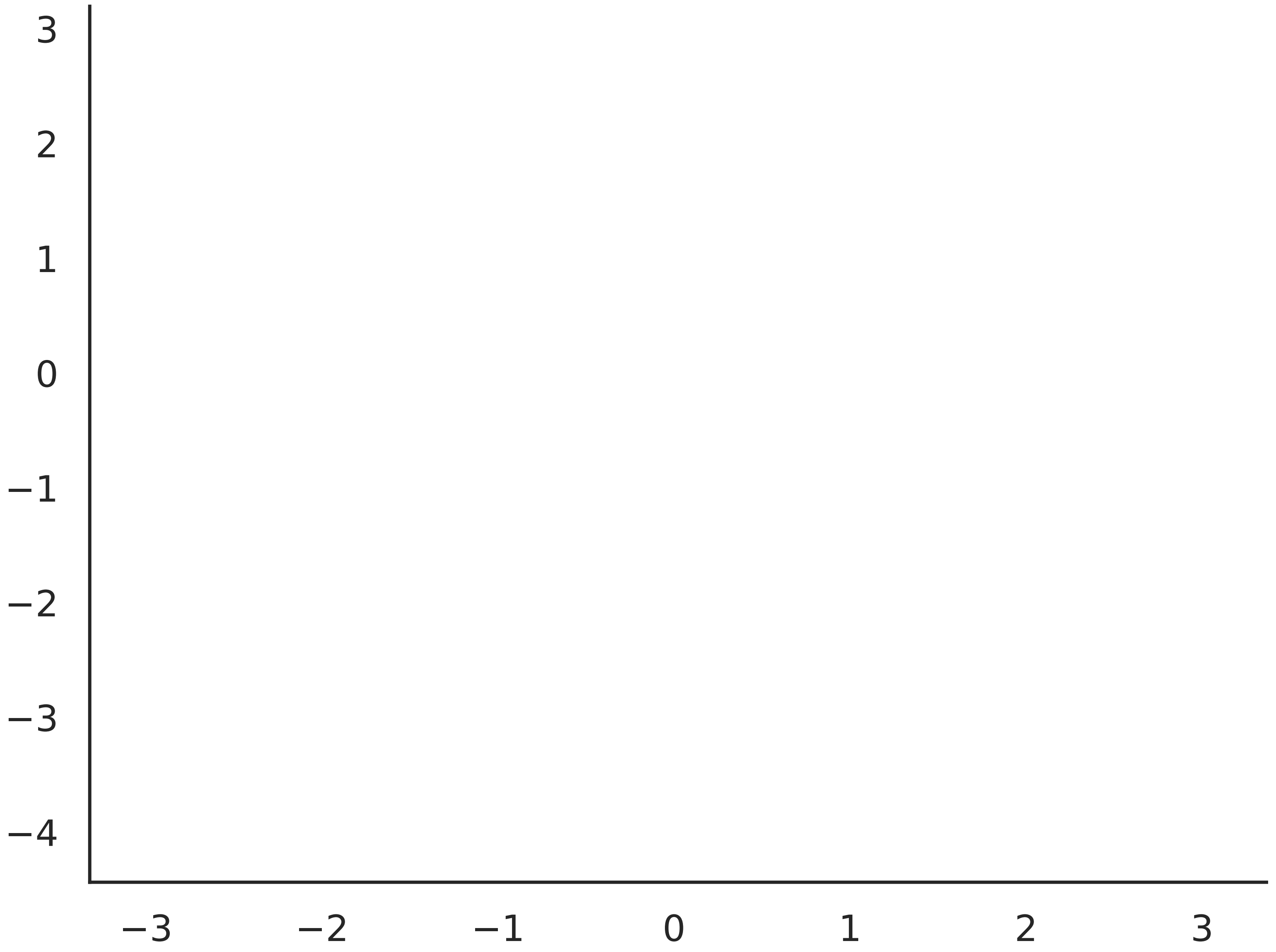
$\mathbb{E}[f(S_i)]$



Results: influence function estimator

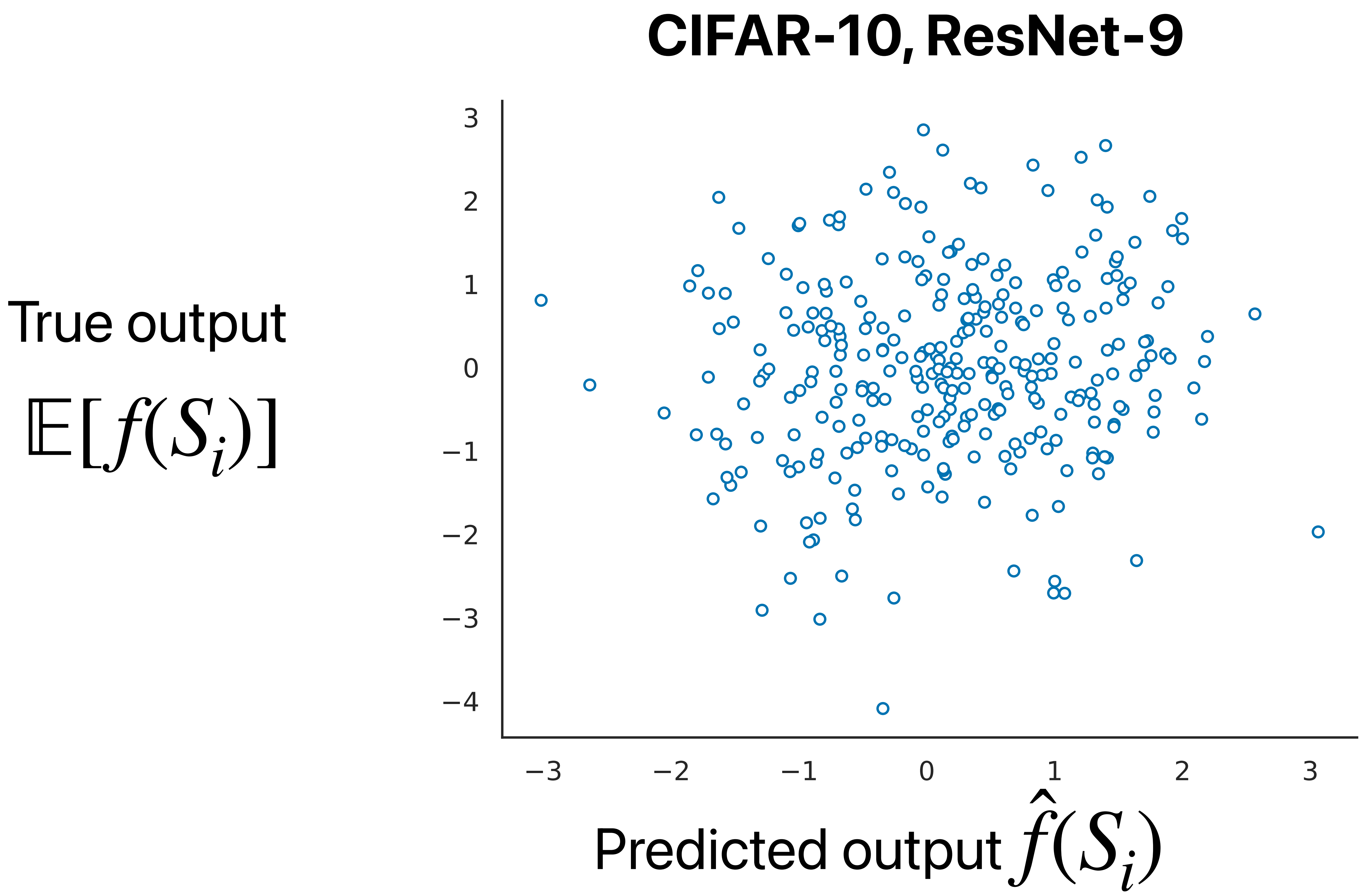
CIFAR-10, ResNet-9

True output $\mathbb{E}[f(S_i)]$



Predicted output $\hat{f}(S_i)$

Results: influence function estimator



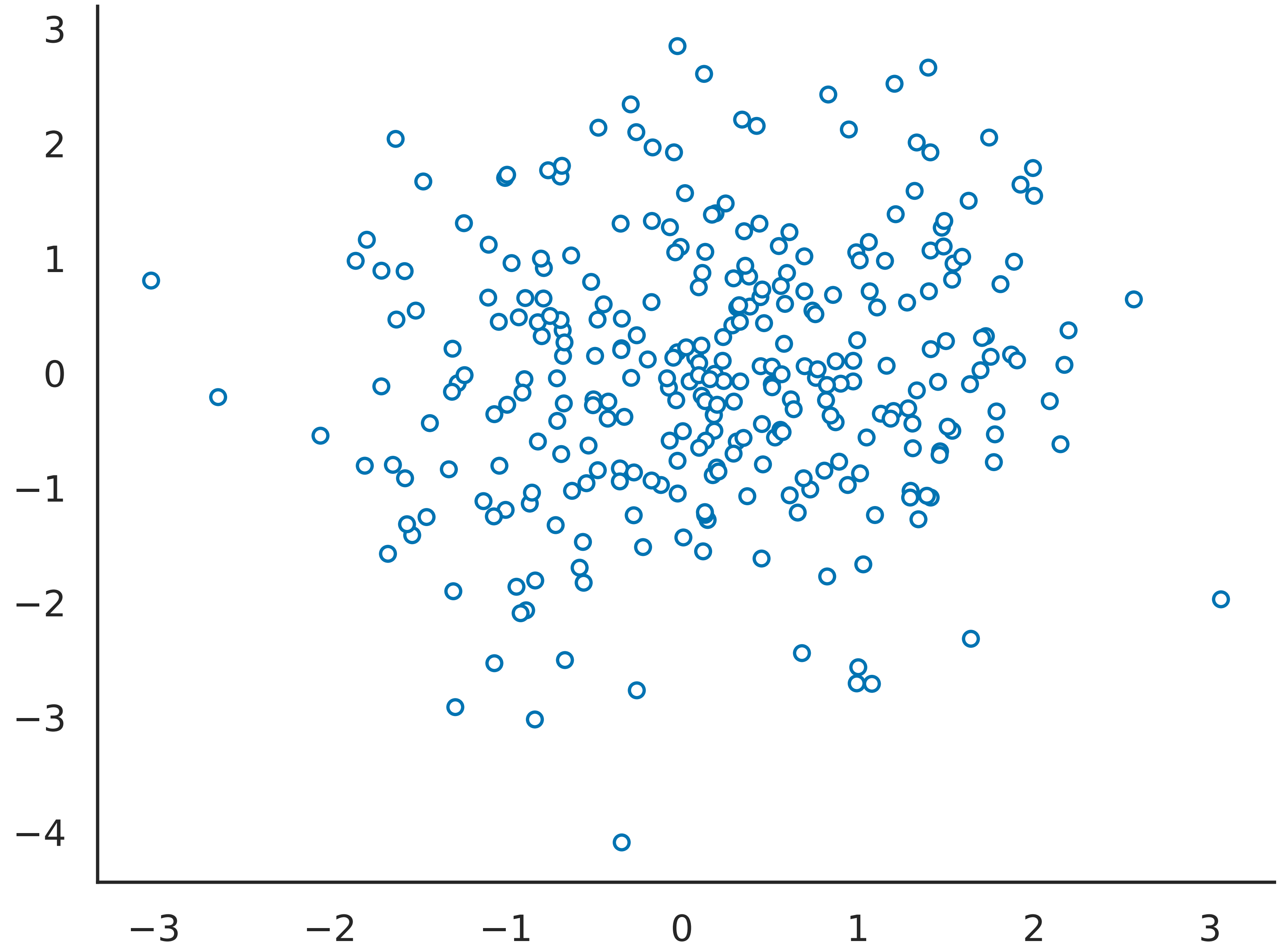
$\rho = 0.05$

Results: influence function estimator

CIFAR-10, ResNet-9

True output

$\mathbb{E}[f(S_i)]$



Predicted output $\hat{f}(S_i)$

$\rho = 0.05$

Confirms intuition that the IF estimator does not scale

Is there a good linear datamodel?

Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

Linear function of included/
excluded datapoints

Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

Fixed, additive effect of training point i

Linear function of included/excluded datapoints

Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

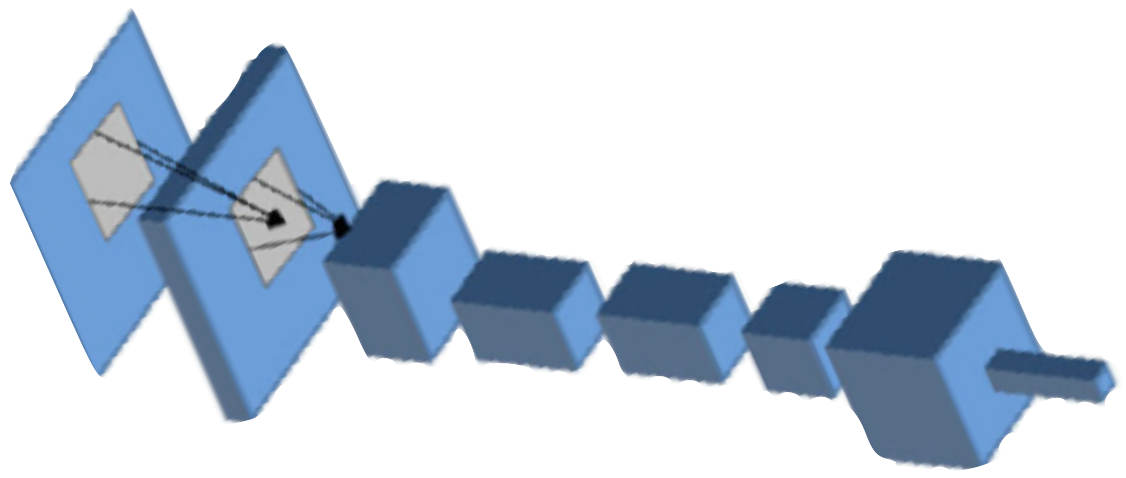
$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

Is there a good linear datamodel?

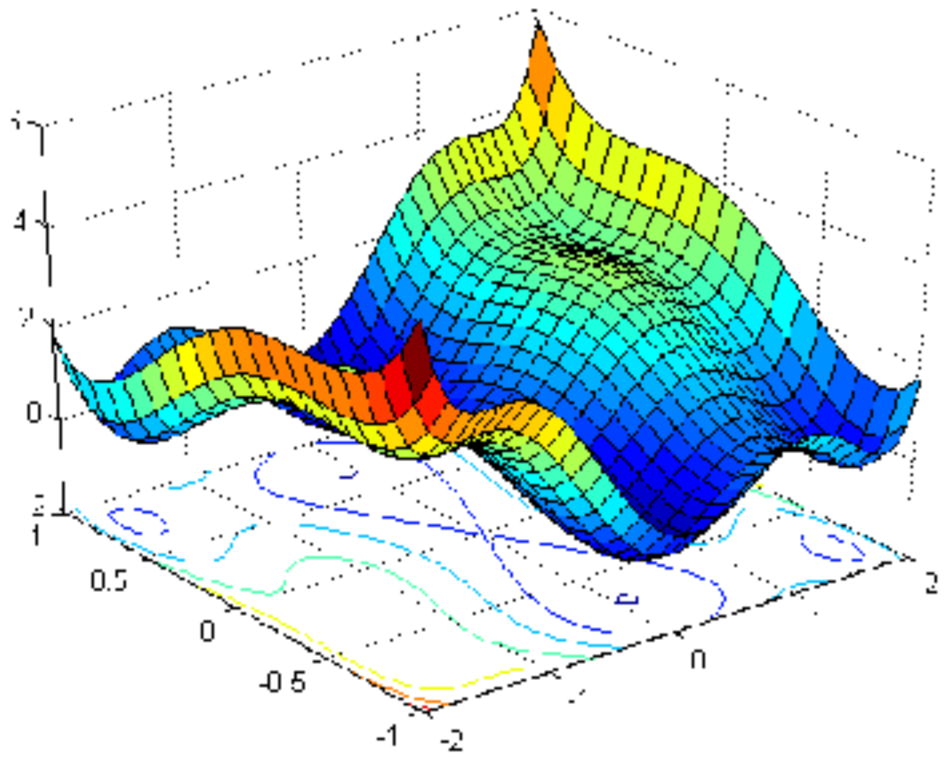
Note that the influence function estimator we looked at thus far is **linear**, i.e.,

$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

But neural network training is very complex!



$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$



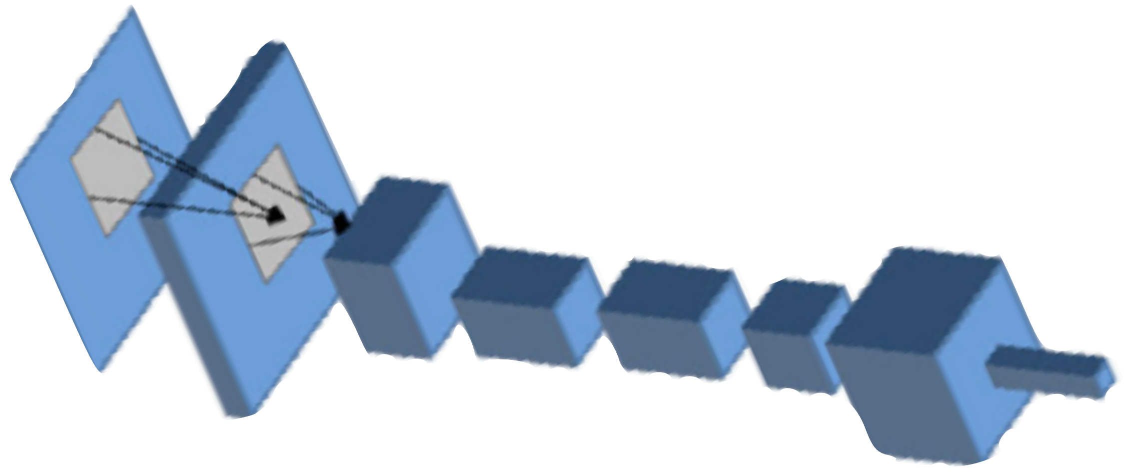
Is there a good linear datamodel?

Note that the influence function estimator we looked at thus far is **linear**, i.e.,

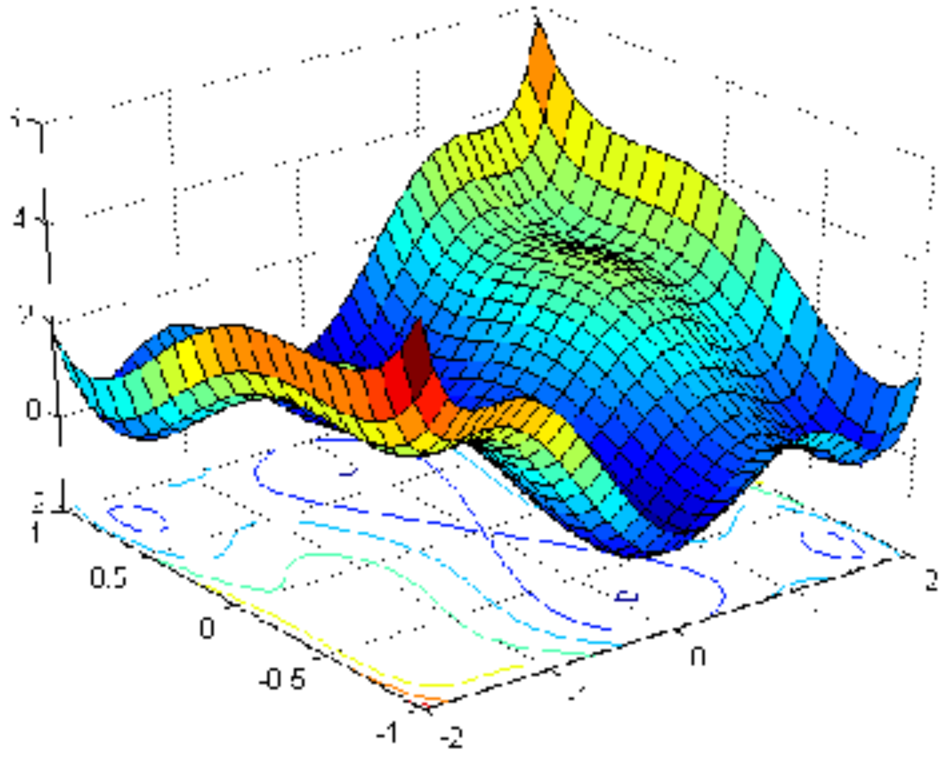
$$\hat{f}(S') = \sum_{i \in S'} \tau_i$$

But neural network training is very complex!

What if $S' \mapsto f(S')$ is too complicated to model with a linear datamodel?



$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$



Estimating datamodels directly

[Feldman Zhang ’20; Ilyas Park Engstrom Leclerc Mądry ’22; Lin Zhang Lecuyer Li Panda Sen ’22]

How to estimate? Use supervised learning!

Estimating datamodels directly

[Feldman Zhang ’20; Ilyas Park Engstrom Leclerc Mądry ’22; Lin Zhang Lecuyer Li Panda Sen ’22]

How to estimate? Use supervised learning!

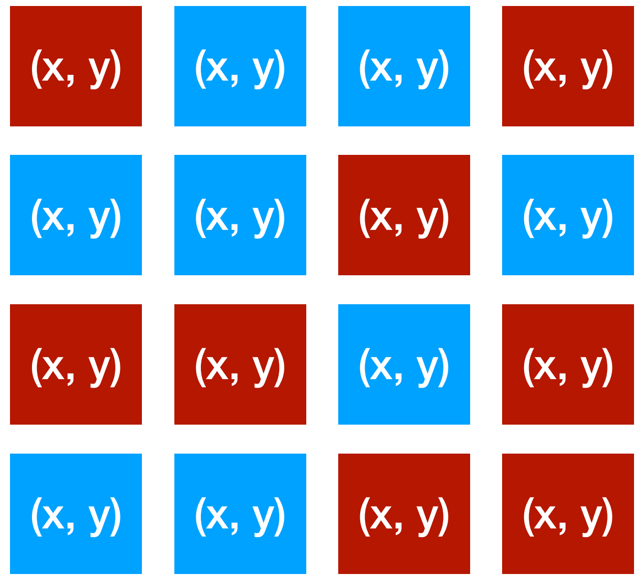
$$S' \mapsto f(S')$$

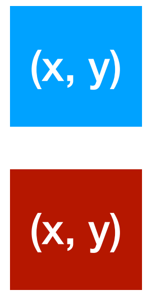
Estimating datamodels directly

[Feldman Zhang '20; Ilyas Park Engstrom Leclerc Mądry '22; Lin Zhang Lecuyer Li Panda Sen '22]

How to estimate? Use supervised learning!

$$S' \mapsto f(S')$$





= **included**

= **excluded**

Sample a random 50% subset. Then, **re-train** model

Estimating datamodels directly

[Feldman Zhang ’20; Ilyas Park Engstrom Leclerc Mądry ’22; Lin Zhang Lecuyer Li Panda Sen ’22]

How to estimate? Use supervised learning!

$$S' \mapsto f(S')$$

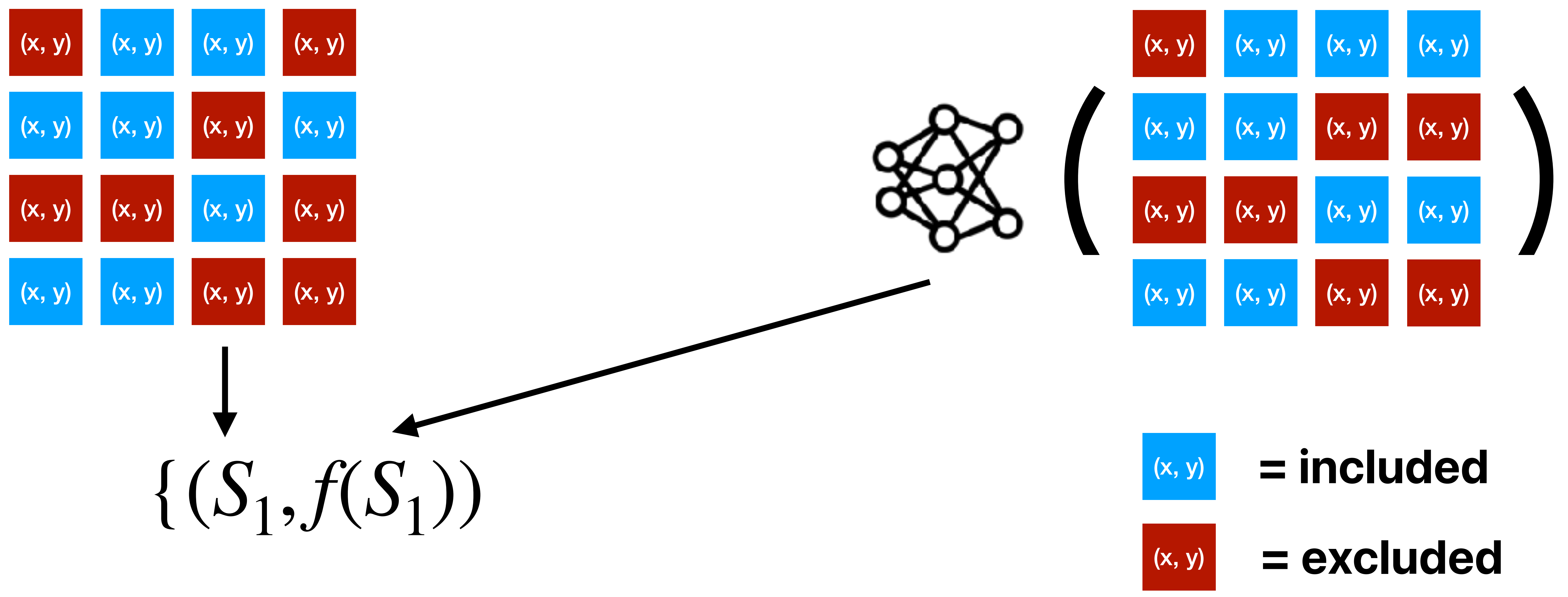
Sample a random 50% subset. Then, **re-train** model

Estimating datamodels directly

[Feldman Zhang '20; Ilyas Park Engstrom Leclerc Mądry '22; Lin Zhang Lecuyer Li Panda Sen '22]

How to estimate? Use supervised learning!

$$S' \mapsto f(S')$$



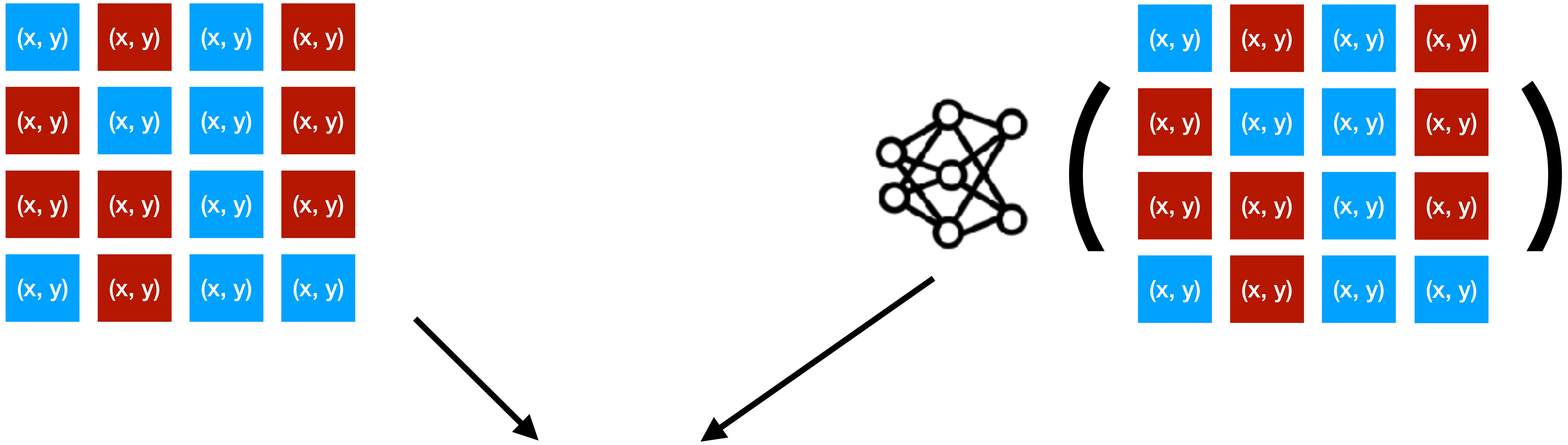
Sample a random 50% subset. Then, **re-train** model

Estimating datamodels directly

[Feldman Zhang '20; Ilyas Park Engstrom Leclerc Mądry '22; Lin Zhang Lecuyer Li Panda Sen '22]

How to estimate? Use supervised learning!

$$S' \mapsto f(S')$$



$$\{(S_1, f(S_1)), (S_2, f(S_2)),$$

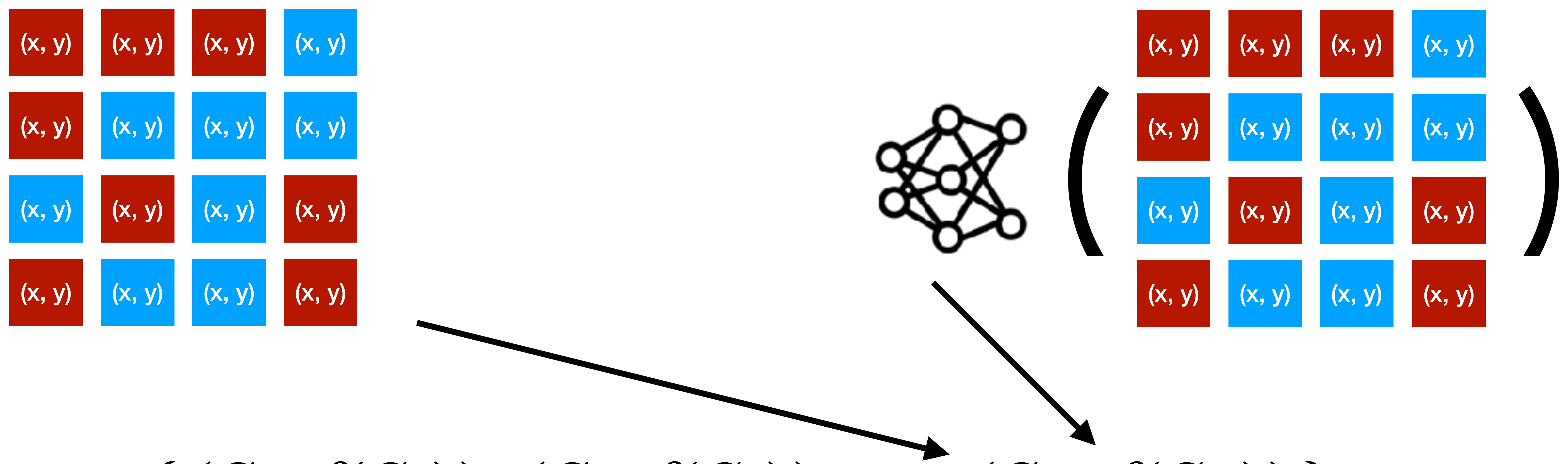
Sample a random 50% subset. Then, **re-train** model

Estimating datamodels directly

[Feldman Zhang ’20; Ilyas Park Engstrom Leclerc Mądry ’22; Lin Zhang Lecuyer Li Panda Sen ’22]

How to estimate? Use supervised learning!

$$S' \mapsto f(S')$$



$$\{(S_1, f(S_1)), (S_2, f(S_2)), \ldots, (S_m, f(S_m))\}$$

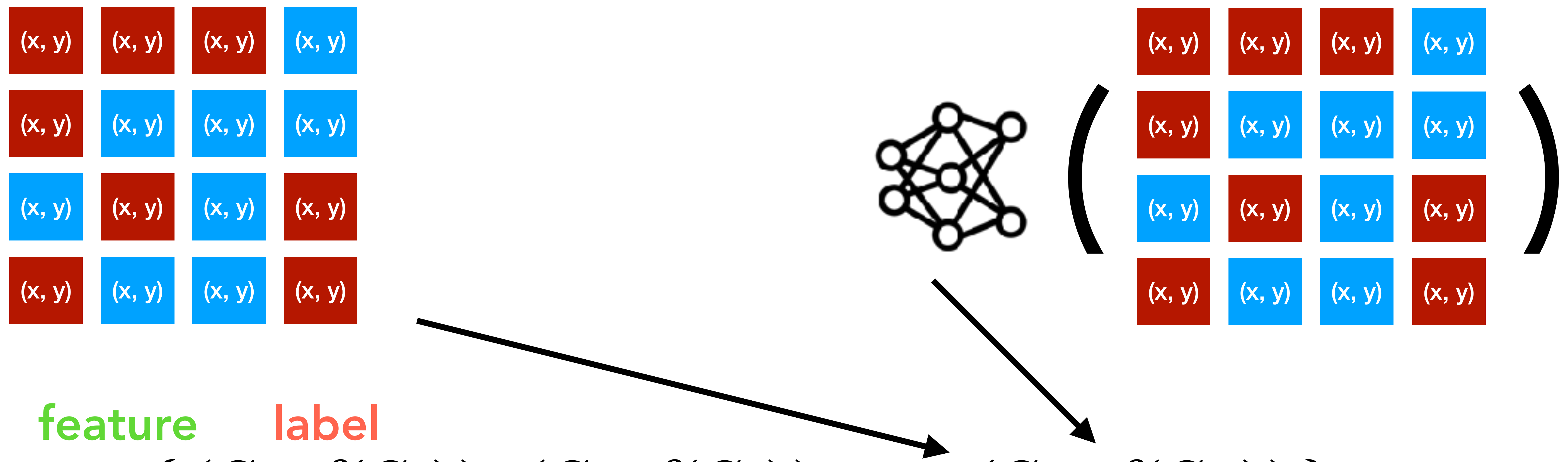
Sample a random 50% subset. Then, **re-train** model

Estimating datamodels directly

[Feldman Zhang '20; Ilyas Park Engstrom Leclerc Mądry '22; Lin Zhang Lecuyer Li Panda Sen '22]

How to estimate? Use supervised learning!

$$S' \mapsto f(S')$$



feature label

$$\{(S_1, f(S_1)), (S_2, f(S_2)), \ldots, (S_m, f(S_m))\}$$

Sample a random 50% subset. Then, **re-train** model

Estimating datamodels directly

[Feldman Zhang ’20; Ilyas Park Engstrom Leclerc Mądry ’22; Lin Zhang Lecuyer Li Panda Sen ’22]

$$\{(S_1, f(S_1)), (S_2, f(S_2)), \ldots, (S_m, f(S_m))\}$$

Estimating datamodels directly

[Feldman Zhang ’20; Ilyas Park Engstrom Leclerc Mądry ’22; Lin Zhang Lecuyer Li Panda Sen ’22]

$$\{(S_1, f(S_1)), (S_2, f(S_2)), \ldots, (S_m, f(S_m))\}$$

Estimating datamodels directly

[Feldman Zhang '20; Ilyas Park Engstrom Leclerc Mądry '22; Lin Zhang Lecuyer Li Panda Sen '22]

$$\{(S_1, f(S_1)), (S_2, f(S_2)), \ldots, (S_m, f(S_m))\}$$

$$\downarrow$$

$$\beta^* = \arg\min_{\beta} \sum_{S_i} \left(f(S_i) - \hat{f}_\beta(S_i) \right)^2 + \lambda \|\beta\|_1$$

where $\hat{f}_\beta(S) = \sum_{i \in S} \beta_i = \beta^\top \mathbf{1}_S$

Estimating datamodels directly

[Feldman Zhang '20; Ilyas Park Engstrom Leclerc Mądry '22; Lin Zhang Lecuyer Li Panda Sen '22]

$$\{(S_1, f(S_1)), (S_2, f(S_2)), \ldots, (S_m, f(S_m))\}$$

↓

$$\beta^* = \arg\min_{\beta} \sum_{S_i} \left(f(S_i) - \hat{f}_\beta(S_i)\right)^2 + \lambda \|\beta\|_1$$

where $\hat{f}_\beta(S) = \sum_{i \in S} \beta_i = \beta^\top \mathbf{1}_S$

Solve (regularized) linear regression to estimate β^*!

Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry ’22]

Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry '22]

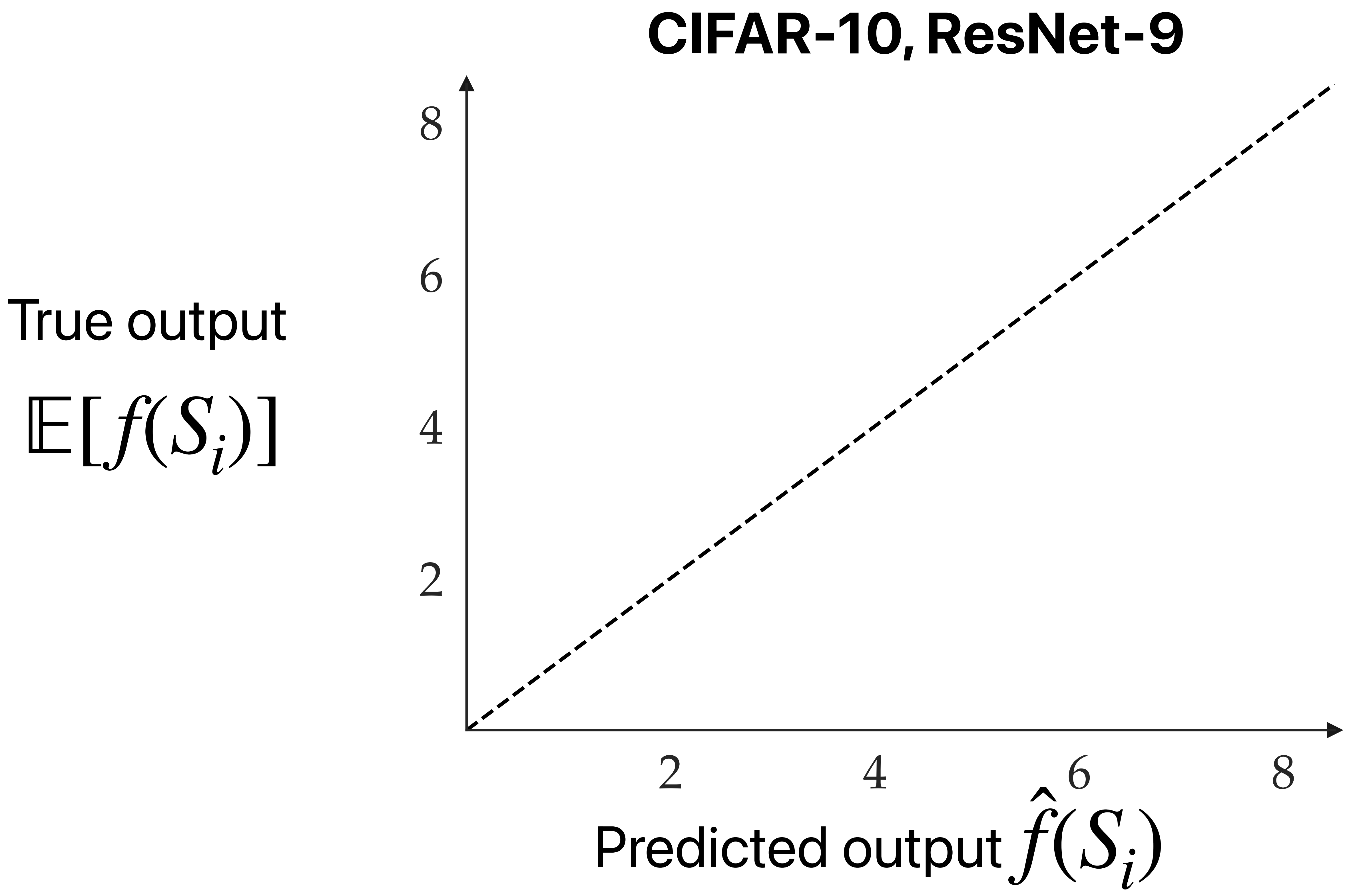
Sample **new** random *subsets* S_i, compare predictions and ground-truth

CIFAR-10, ResNet-9

Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry '22]

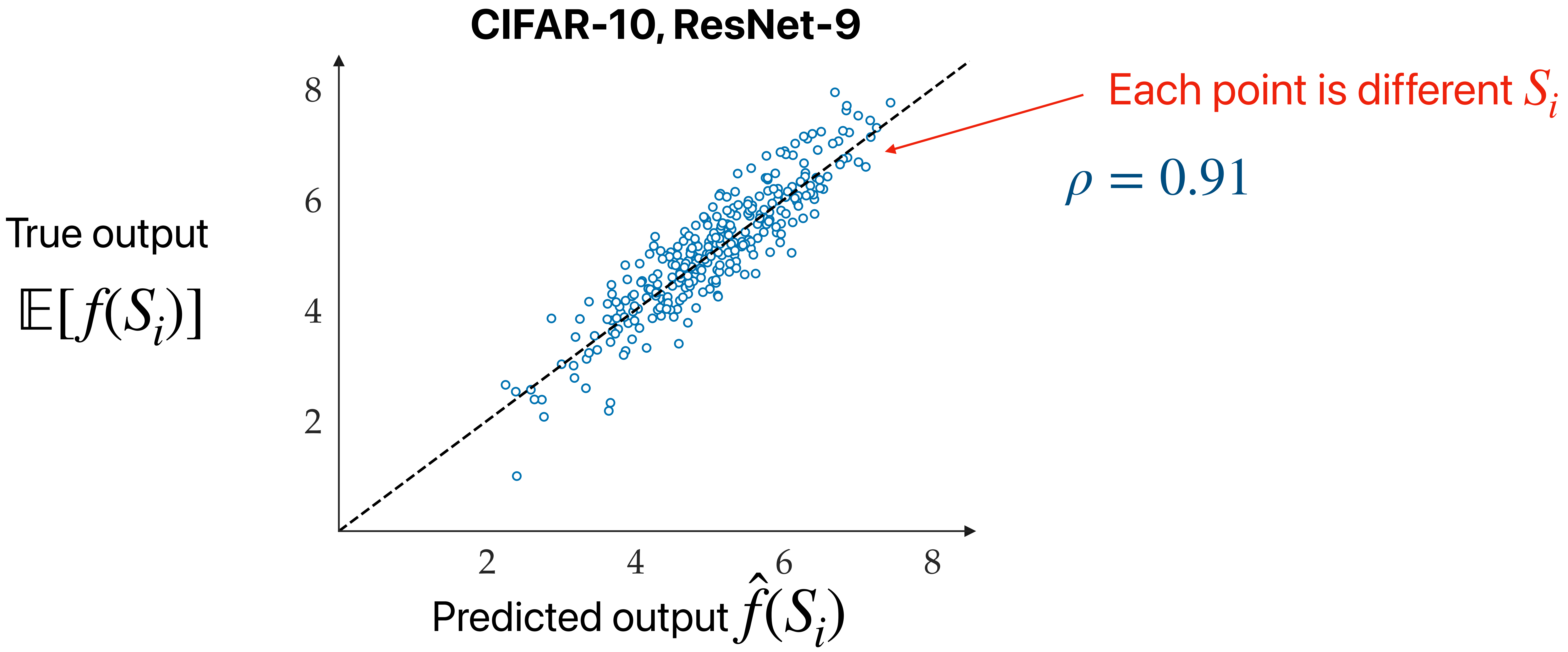
Sample **new** random *subsets* S_i, compare predictions and ground-truth



Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry '22]

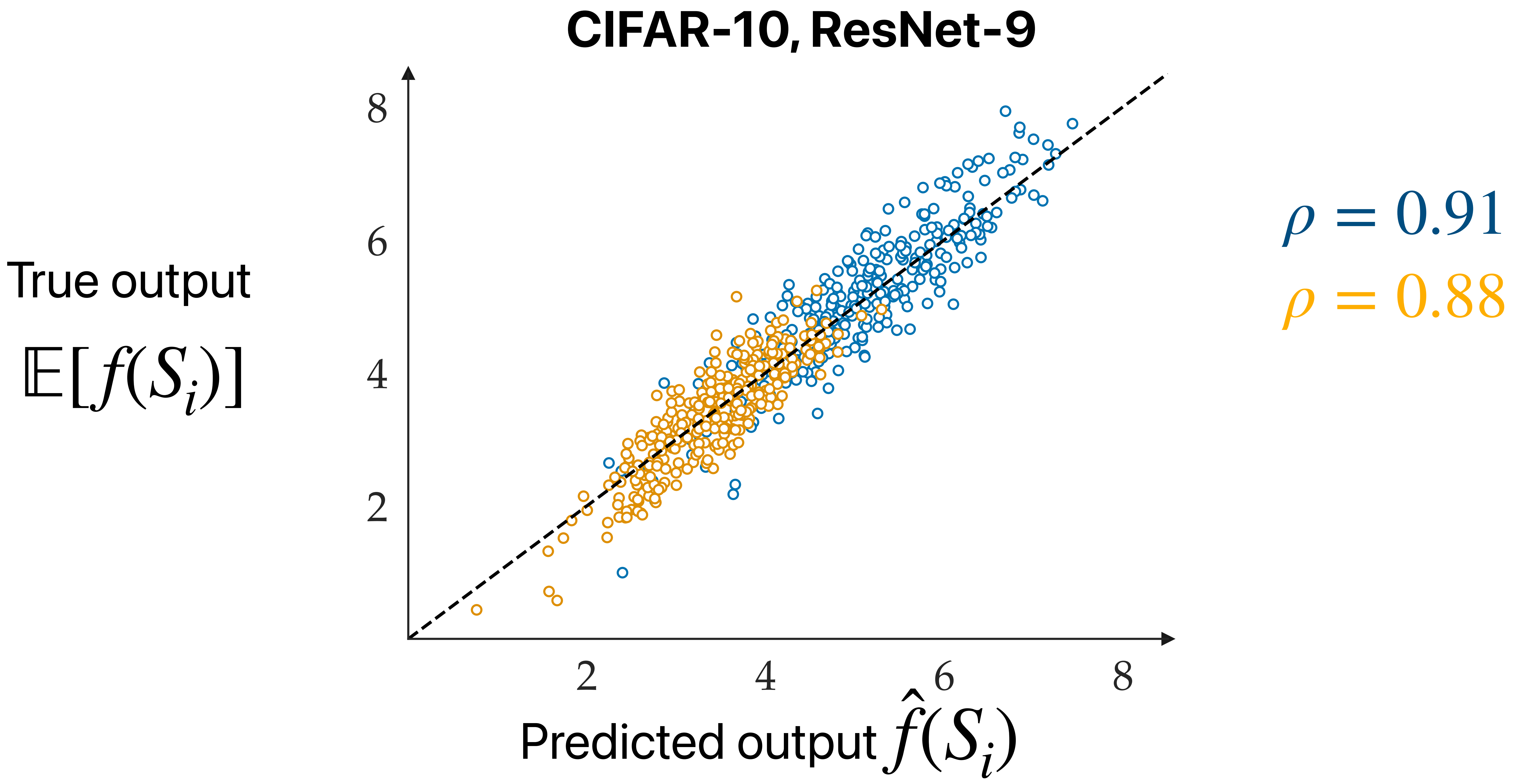
Sample **new** random *subsets* S_i, compare predictions and ground-truth



Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry '22]

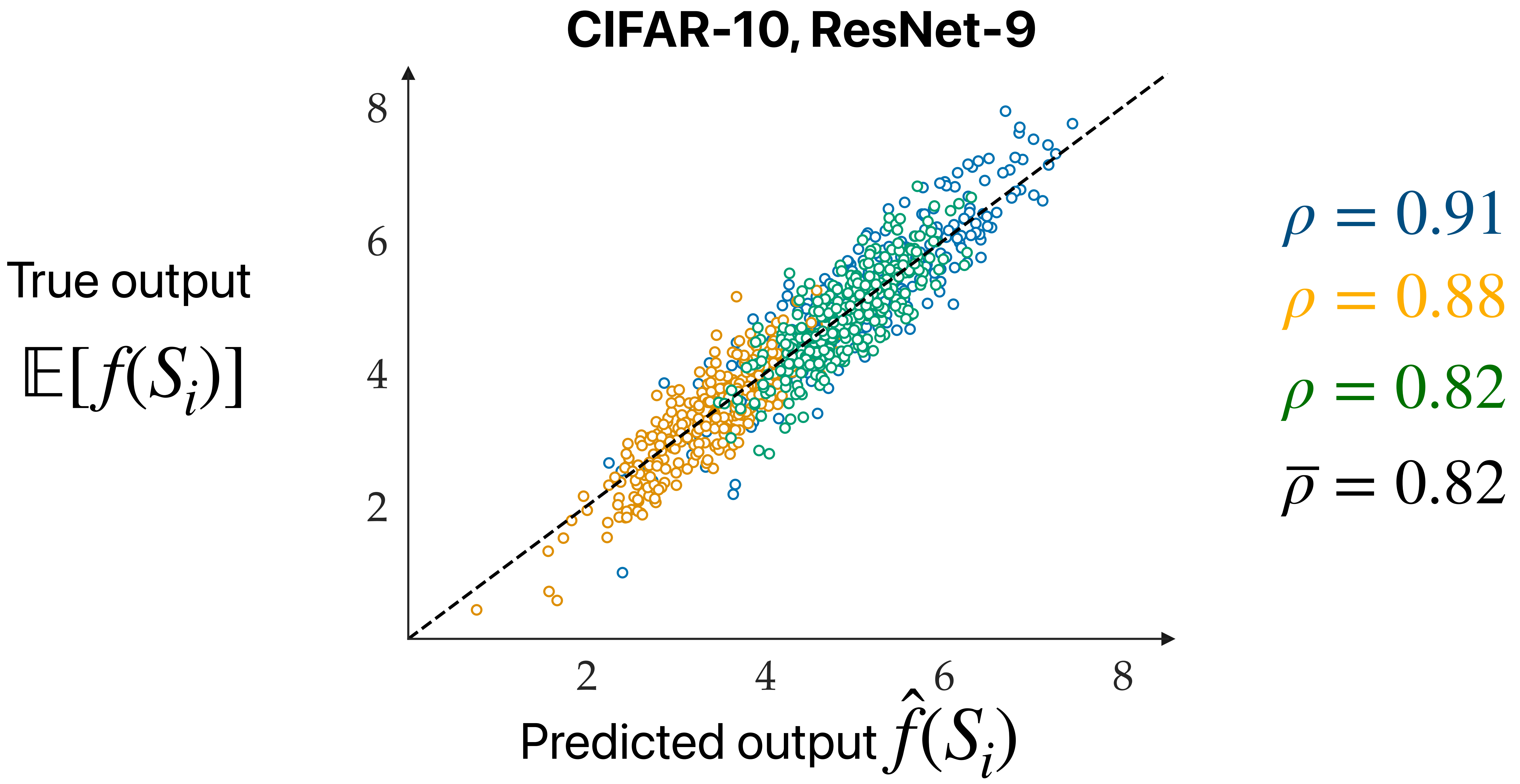
Sample **new** random *subsets* S_i, compare predictions and ground-truth



Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry '22]

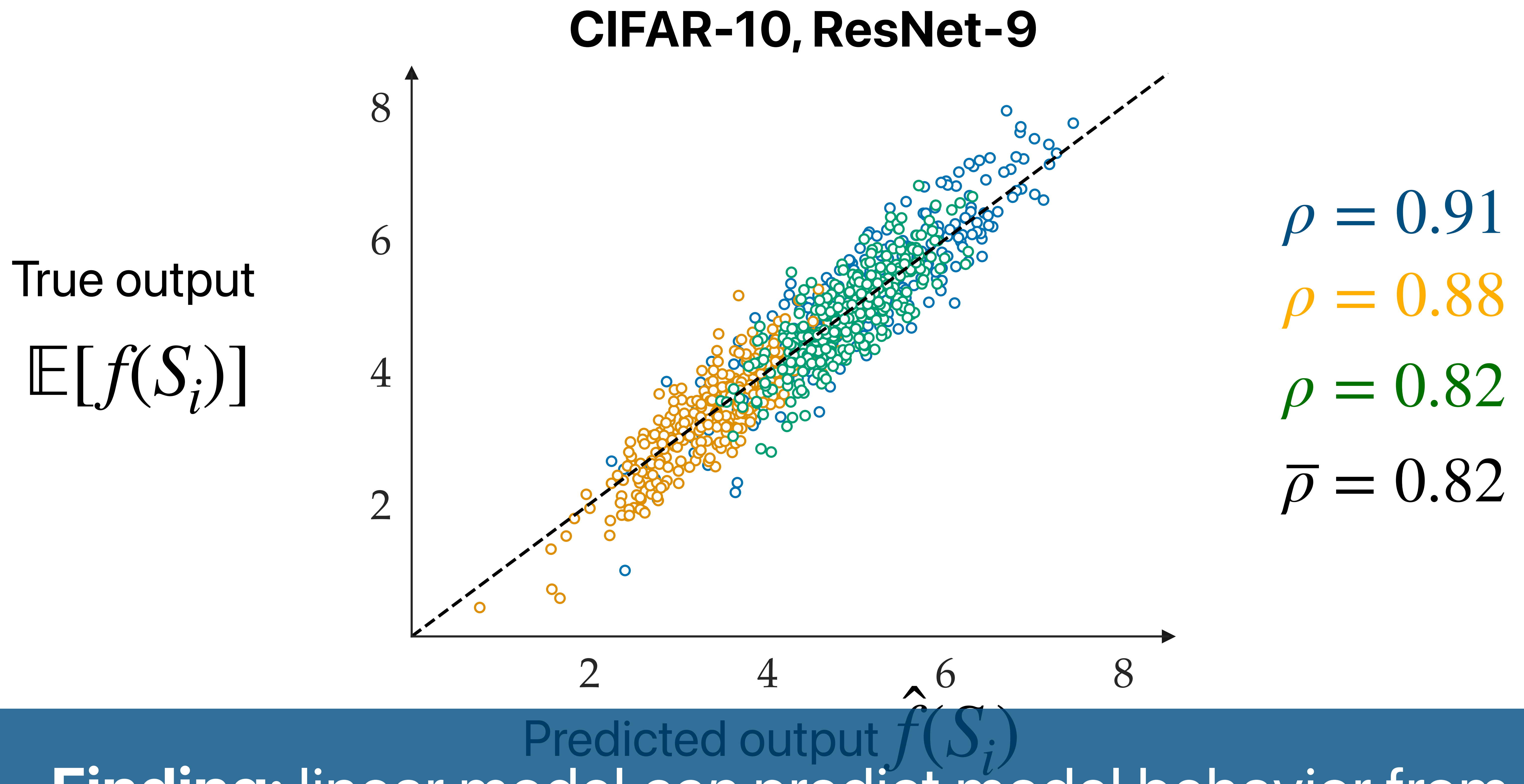
Sample **new** random *subsets* S_i, compare predictions and ground-truth



Evaluating linear datamodels

[Ilyas Park Engstrom Leclerc Mądry '22]

Sample **new** random *subsets* S_i, compare predictions and ground-truth



Finding: linear model *can* predict model behavior from data!

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

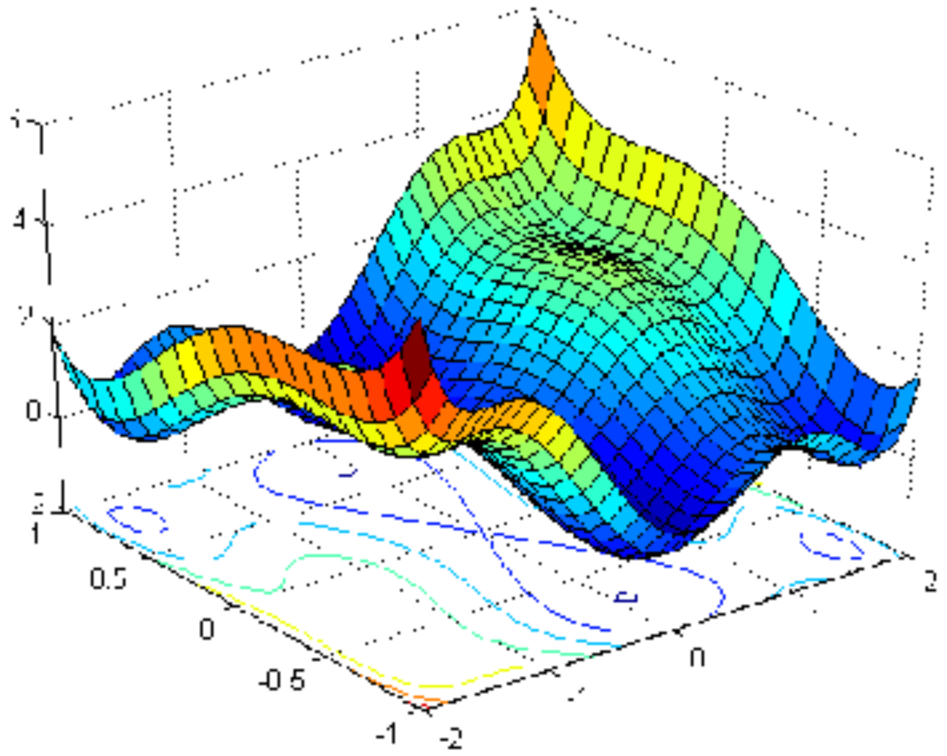
Non-convexity $\nabla_\theta^2 L \nsucceq 0$

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

Non-convexity $\nabla_\theta^2 L \not\succeq 0$

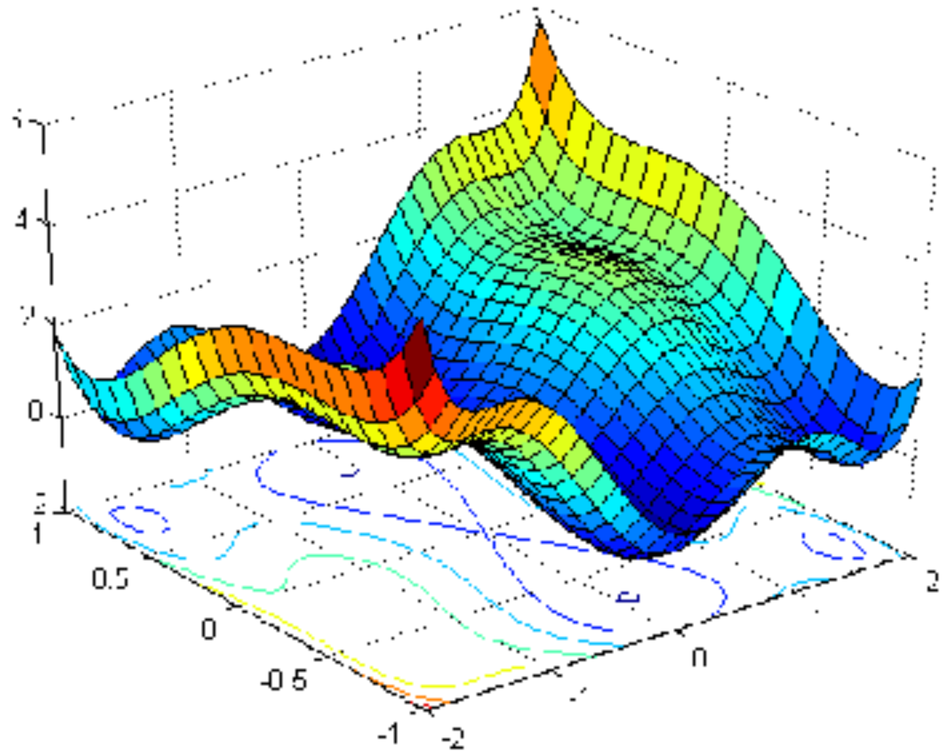


Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

Non-convexity $\nabla_\theta^2 L \not\succeq 0$



Hessian **non-invertible** $\Rightarrow$ Approximations like IFs are not well-defined

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

Non-convexity $\nabla_\theta^2 L \not\succeq 0$

Randomness $\theta(S)$ is a random variable

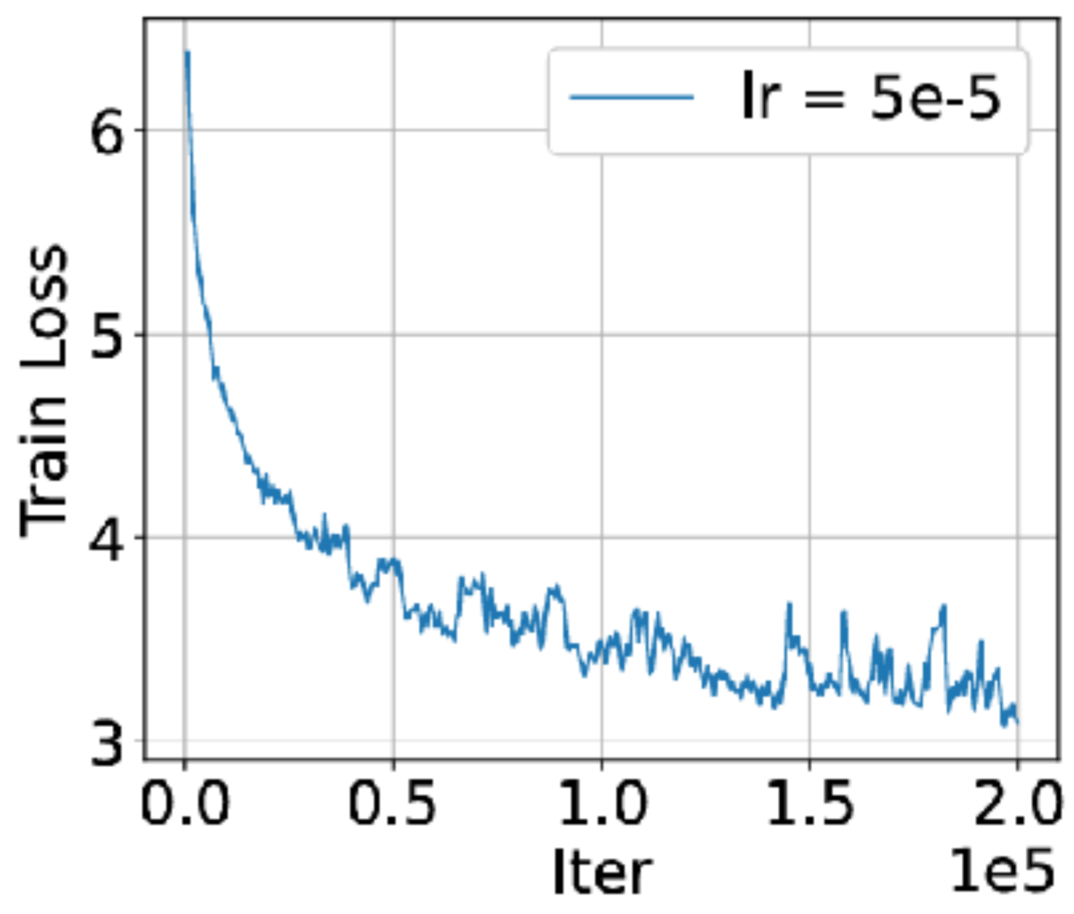
Solution is no longer unique; counterfactual (re-training) is not even well-defined!

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

Non-convexity	$\nabla^2_\theta L \not\succeq 0$
Randomness	$\theta(S)$ is a random variable
Non-convergence	Not trained to convergence



Questionable to rely on convergence conditions

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

Non-convexity	$\nabla_\theta^2 L \not\succeq 0$
Randomness	$\theta(S)$ is a random variable
Non-convergence	Not trained to convergence

Large-scale / high-dimensionality

Everything is much harder to compute (e.g., impossible to actually store Hessian)

Why do old methods fail?

Why do classical methods (Chapter 2) no longer work for DNNs?

New challenges:

Non-convexity	$\nabla^2_\theta L \not\succeq 0$
Randomness	$\theta(S)$ is a random variable
Non-convergence	Not trained to convergence
Large-scale / high-dimensionality	

Can we bridge the gap between efficient estimators and direct estimators?

A quick tour of the landscape

A quick tour of the landscape

Many works have tried to circumvent these challenges

A quick tour of the landscape

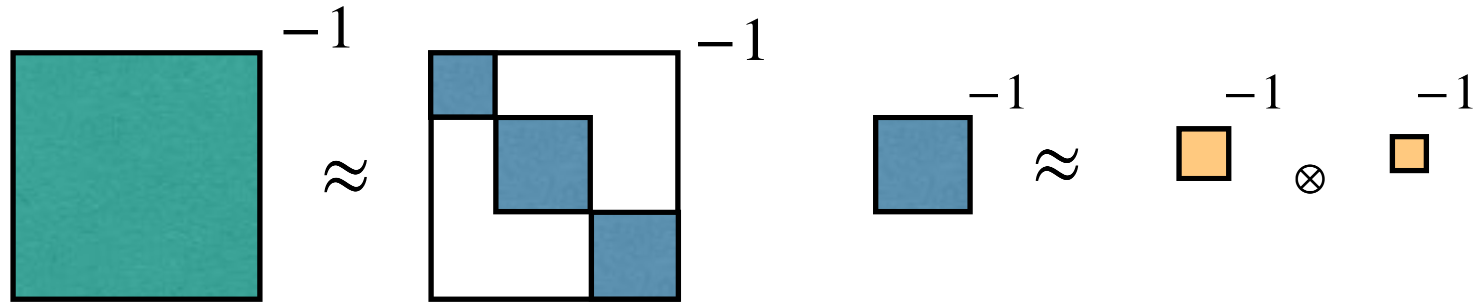
Many works have tried to circumvent these challenges

Themes:

A quick tour of the landscape

Many works have tried to circumvent these challenges

Themes:



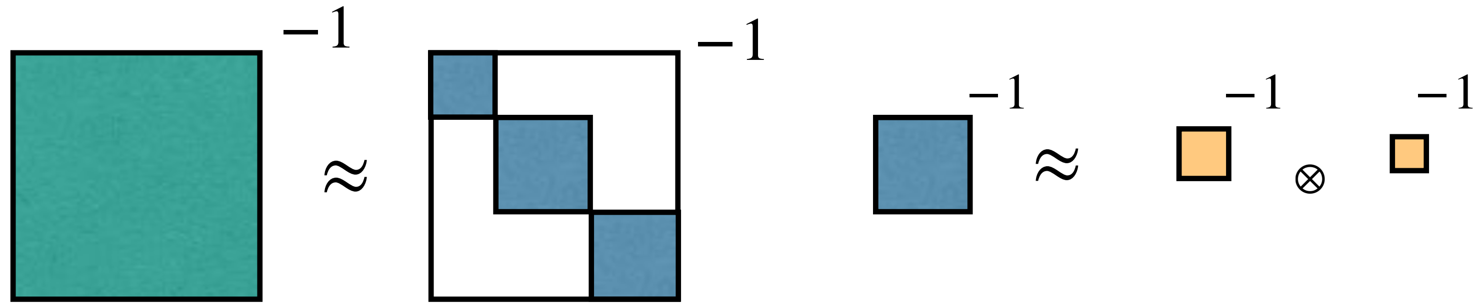
Better IF/Hessian approximations

A quick tour of the landscape

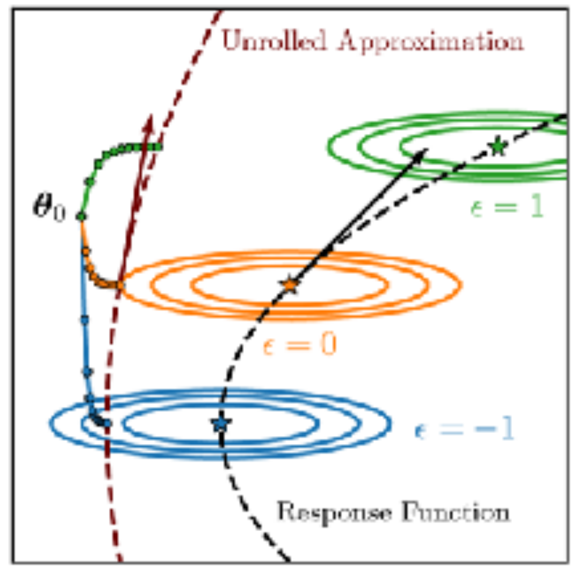
Many works have tried to circumvent these challenges

Themes:

Better IF/Hessian approximations



Approximating training dynamics ("unrolling")



[BLLG '24]

A quick tour of the landscape

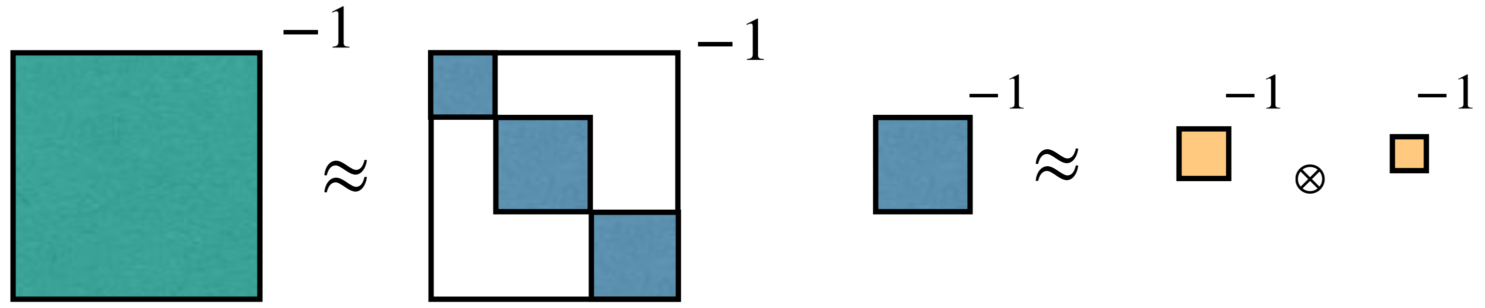
Many works have tried to circumvent these challenges

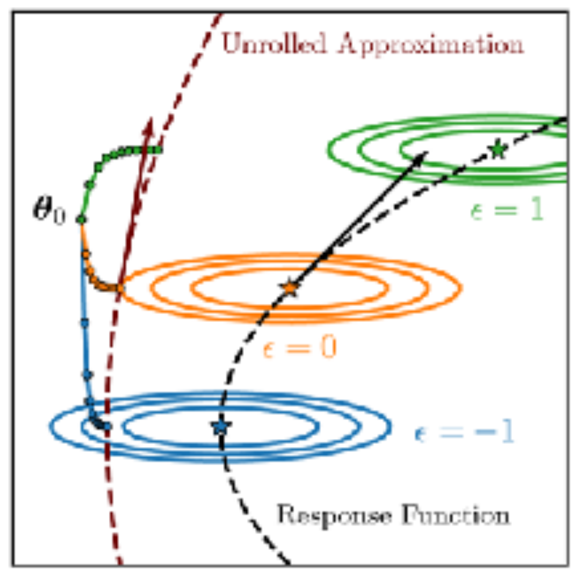
Themes:

Better IF/Hessian approximations

Approximating training dynamics (“unrolling”)

Simple surrogate models





[BLLG ’24]

A quick tour of the landscape

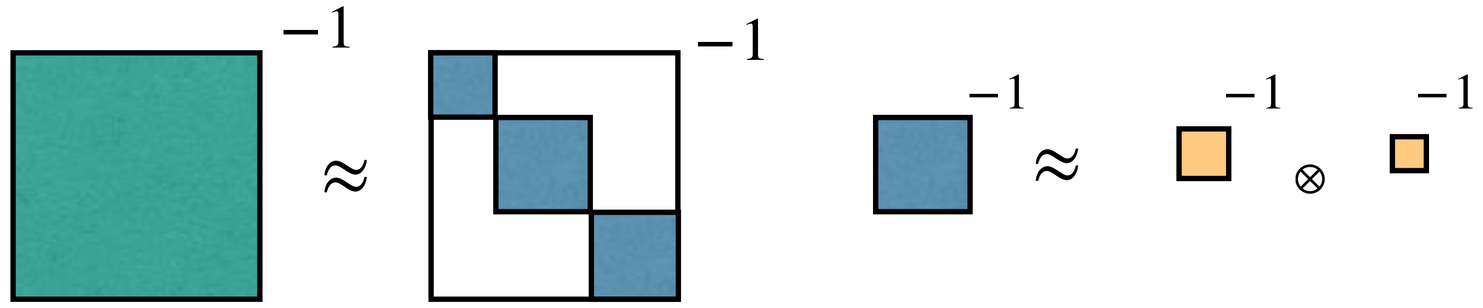
Many works have tried to circumvent these challenges

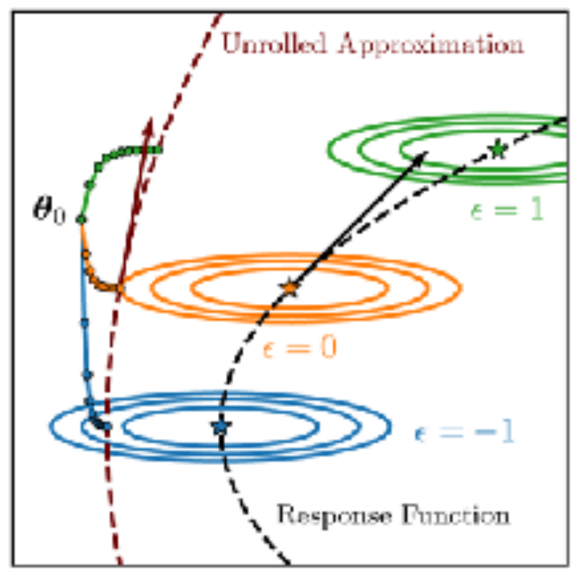
Themes:

Better IF/Hessian approximations

Approximating training dynamics (“unrolling”)

Simple surrogate models





[BLLG ’24]

Next, we’ll look at how each approach works at a high-level

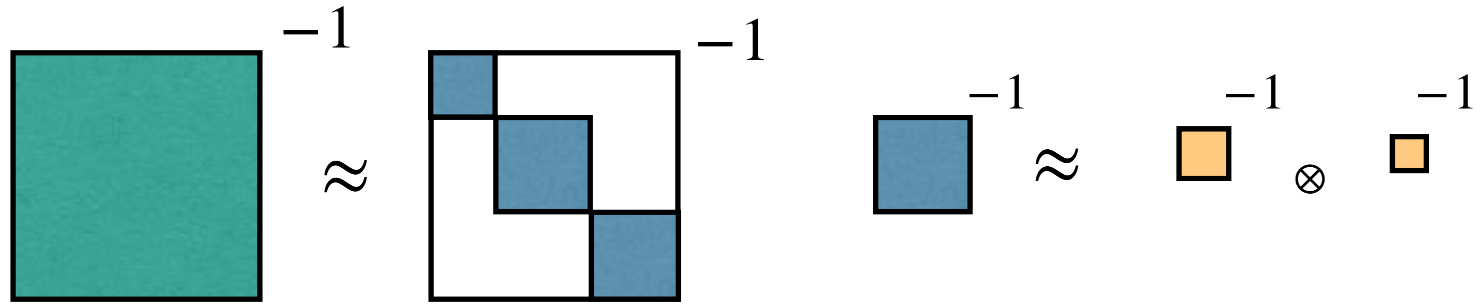
A quick tour of the landscape

Themes:

Better IF/Hessian approximations

Approximating training dynamics (“unrolling”)

Surrogate models



[BLLG '24]

Better IF approximations

Themes:

Better IF/Hessian approximations

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation
[Teso et al. '21] [Bae et al. '22] + many others

Structural approximations based on NN architecture [Kwon Wu Wu Zou '24]
[Choe et al. '24]

Approximating training dynamics ("unrolling")

Surrogate models

Better IF approximations

Themes:

- **Better IF/Hessian approximations**
  - **Replace Hessian with the “Gauss Newton Hessian” (GNH) approximation** **[Teso et al. ’21] [Bae et al. ’22] + many others**
  - Structural approximations based on NN architecture [Kwon Wu Wu Zou ’24] [Choe et al. ’24]
- Approximating training dynamics (“unrolling”)
- Surrogate models

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini ’21] [Bae Ng Lo Ghassemi Grosse ’22]

Replace Hessian with the “Gauss Newton Hessian” (GNH) approximation

[Schraudolph ’02]

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini '21] [Bae Ng Lo Ghassemi Grosse '22]

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation

[Schraudolph '02]

Assume loss $\ell(h(x, \theta), y)$

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini '21] [Bae Ng Lo Ghassemi Grosse '22]

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation

[Schraudolph '02]

Assume loss $\ell(h(x, \theta), y)$

Model outputs (e.g. "logits")

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini '21] [Bae Ng Lo Ghassemi Grosse '22]

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation

[Schraudolph '02]

Assume loss $\ell(h(x, \theta), y)$

Model outputs (e.g. "logits")

$$\tilde{H} = J^\top D J$$

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini '21] [Bae Ng Lo Ghassemi Grosse '22]

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation

[Schraudolph '02]

Assume loss $\ell(h(x, \theta), y)$

Model outputs (e.g. "logits")

$$\tilde{H} = J^\top D J$$

Jacobian of h w.r.t. θ

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini '21] [Bae Ng Lo Ghassemi Grosse '22]

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation

[Schraudolph '02]

Assume loss $\ell(h(x, \theta), y)$

Model outputs (e.g. "logits")

$$\tilde{H} = J^\top D J$$

Jacobian of h w.r.t. θ

Hessian of ℓ w.r.t. h

Better IF approximations

[Teso Bontempelli Giunchiglia Passerini '21] [Bae Ng Lo Ghassemi Grosse '22]

Replace Hessian with the "Gauss Newton Hessian" (GNH) approximation

[Schraudolph '02]

Assume loss $\ell(h(x, \theta), y)$

Model outputs (e.g. "logits")

$$\tilde{H} = J^\top D J \succeq 0$$

Jacobian of h w.r.t. θ **Hessian of ℓ w.r.t. h**

$\tilde{H}$ is guaranteed to be p.s.d. $\Rightarrow (\tilde{H} + \lambda I)^{-1}$ well-defined for any $\lambda > 0$

$\Rightarrow$ IF estimates are well-defined

Better IF approximations

Themes:

- **Better IF/Hessian approximations**
  - Replace Hessian with the “Gauss Newton Hessian” (GNH) approximation [Teso et al. ’21] [Bae et al. ’22] + many others
  - **Structural approximations based on NN architecture [Kwon Wu Wu Zou ’24] [Choe et al. ’24]**
- Approximating training dynamics (“unrolling”)
- Surrogate models

Better IF approximations

[Martens Grosse ’15] [George Laurent Bouthillier Ballas Vincent ’18]

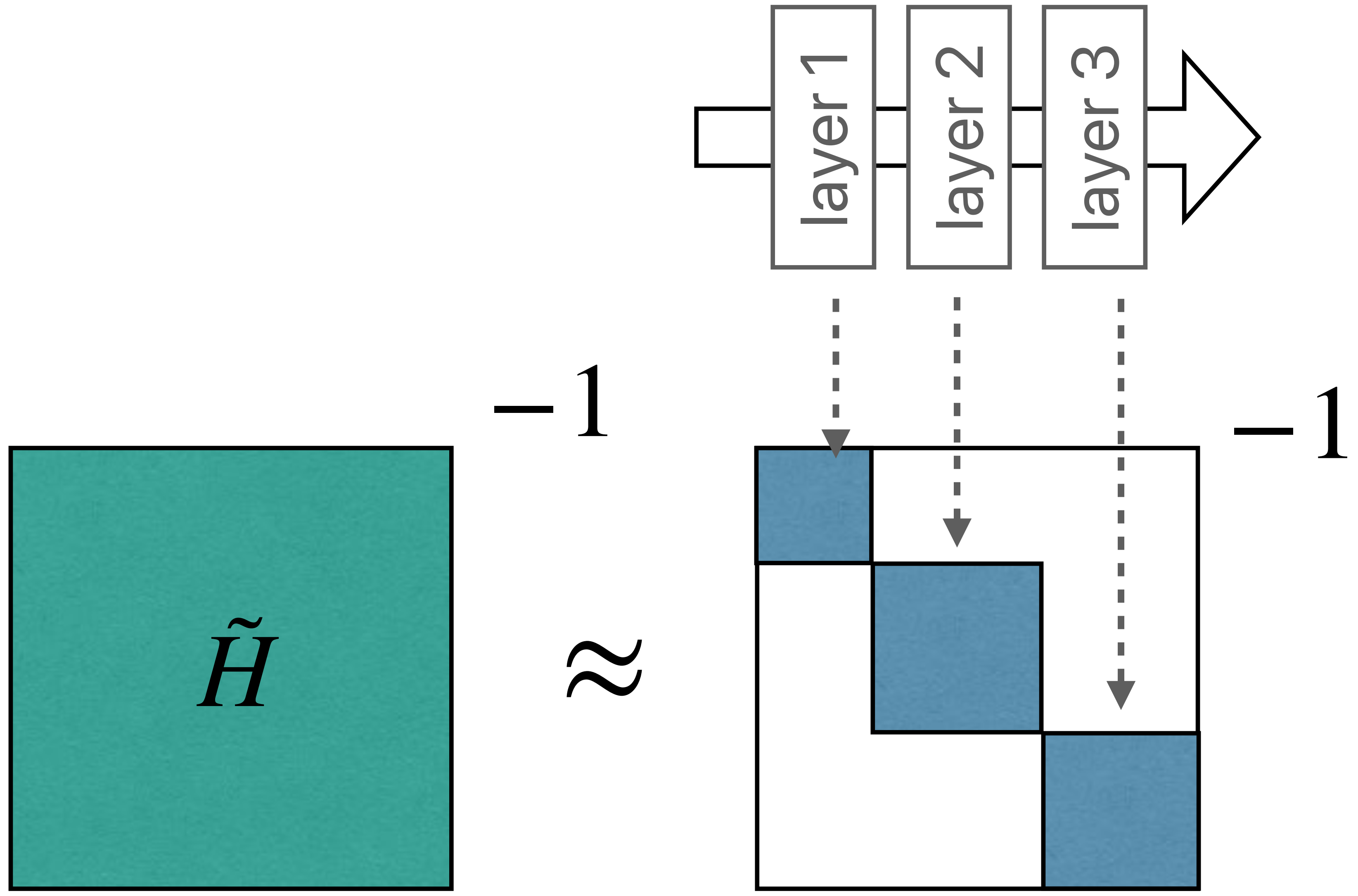
Structural approximations based on NN architecture (EK-FAC, etc.)

Better IF approximations

[Martens Grosse '15] [George Laurent Bouthillier Ballas Vincent '18]

Structural approximations based on NN architecture (EK-FAC, etc.)

Can factor $\tilde{H}$ as block-diagonal across layers



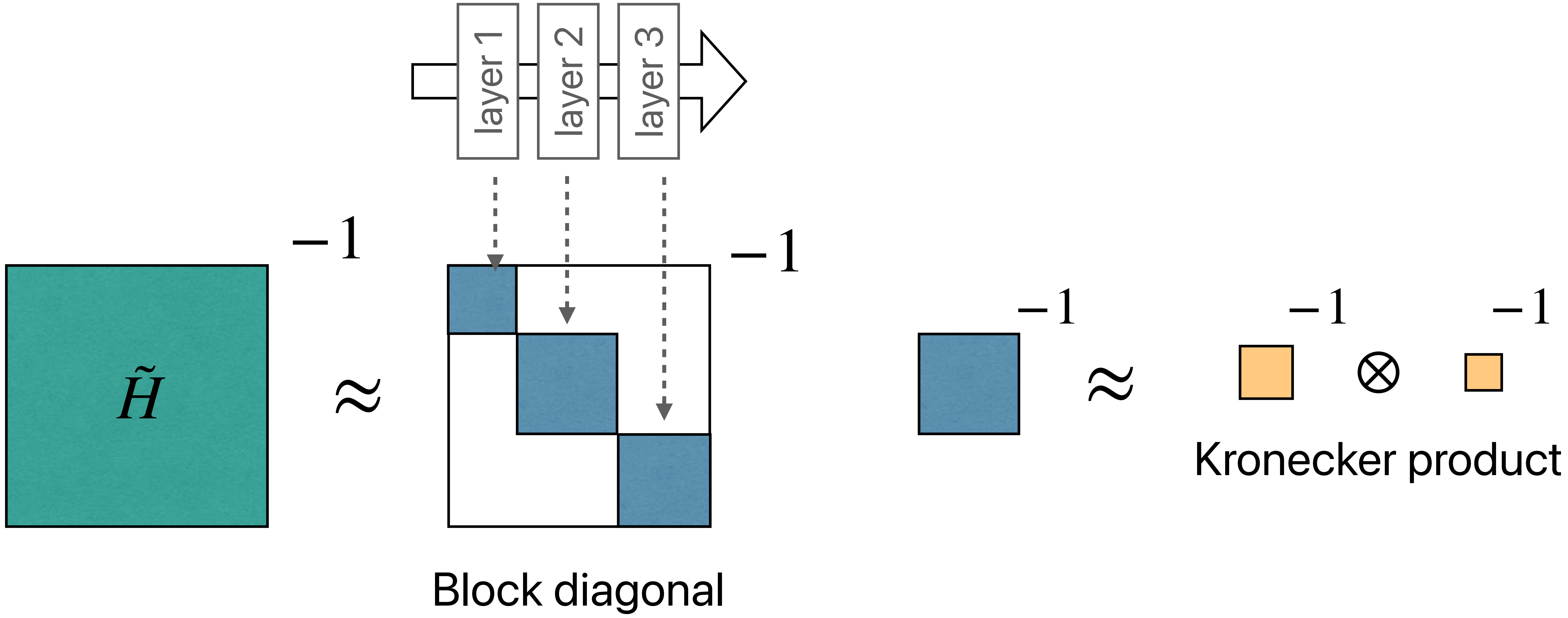
Block diagonal

Better IF approximations

[Martens Grosse '15] [George Laurent Bouthillier Ballas Vincent '18]

Structural approximations based on NN architecture (EK-FAC, etc.)

For certain layers (e.g., linear), can further factor as a **kronecker product**

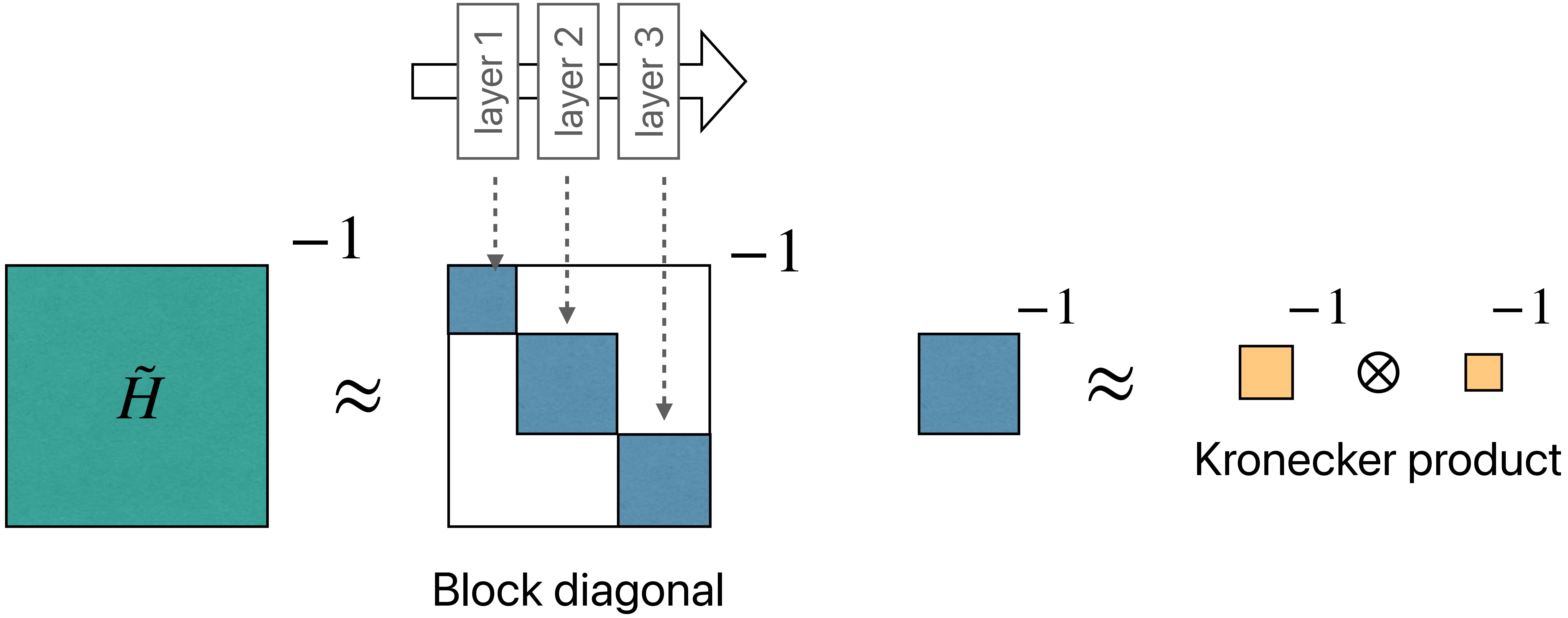


Better IF approximations

[Martens Grosse '15] [George Laurent Bouthillier Ballas Vincent '18]

Structural approximations based on NN architecture (EK-FAC, etc.)

Allows *much* faster inversion and IF computation ⇒ can scale more reliably

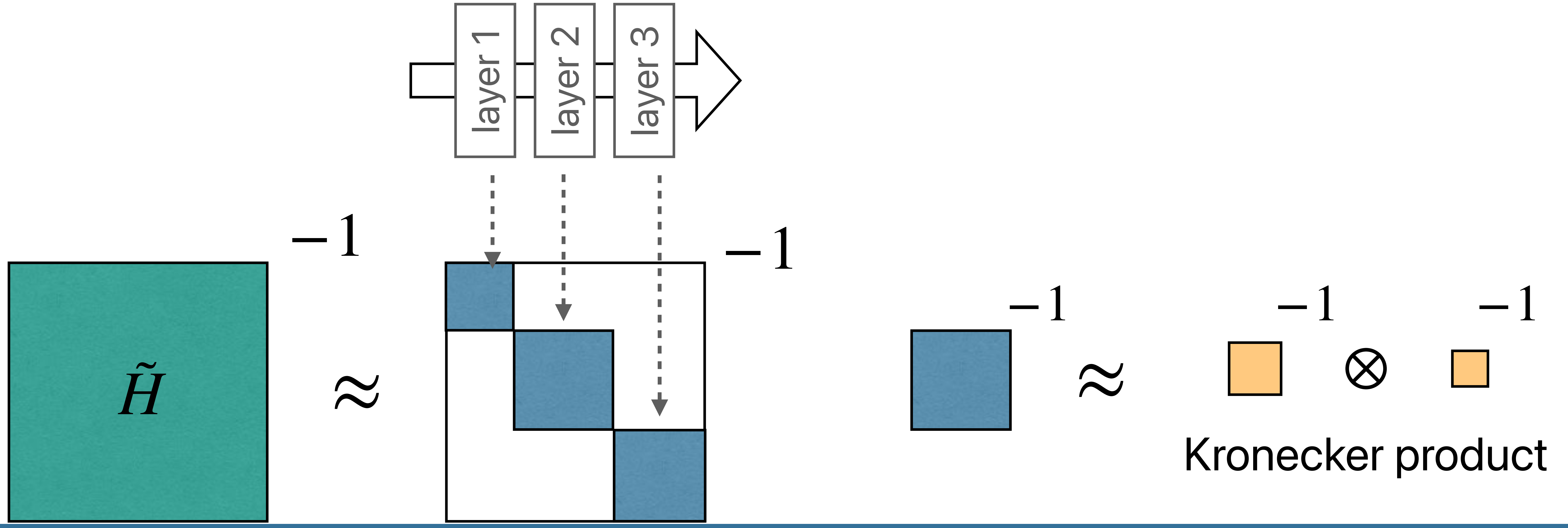


Better IF approximations

[Martens Grosse '15] [George Laurent Bouthillier Ballas Vincent '18]

Structural approximations based on NN architecture (EK-FAC, etc.)

Allows *much* faster inversion and IF computation ⇒ can scale more reliably



By leveraging structure of DNNs, we can make IFs work much better

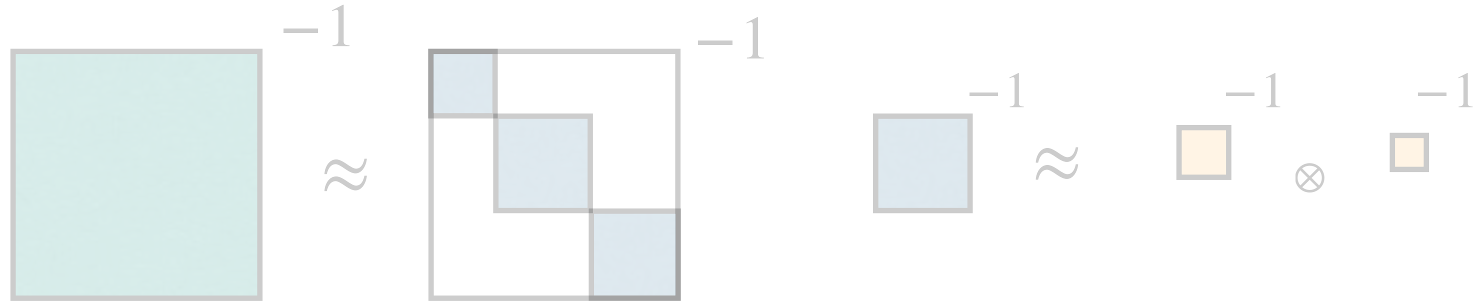
A quick tour of the landscape

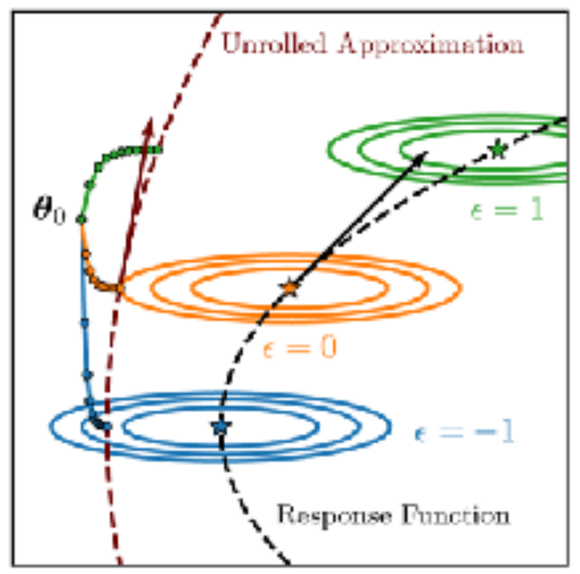
Themes:

Better IF/Hessian approximations

Approximating training dynamics ("unrolling")

Surrogate models





[BLLG '24]

Approximating training dynamics (“unrolling”)

[Hara Nitanda Maehara ’19] [Bae Lin Lorraine Grosse ’24]

What the influence function was trying to compute was:

$$\frac{d\theta^{(T)}}{dw_j}$$

(the infinitesimal effect of upweighting example j by ε on final parameters $\theta^{(T)}$)

Approximating training dynamics (“unrolling”)

[Hara Nitanda Maehara ’19] [Bae Lin Lorraine Grosse ’24]

What the influence function was trying to compute was:

$$\frac{d\theta^{(T)}}{dw_j}$$

(the infinitesimal effect of upweighting example j by ε on final parameters $\theta^{(T)}$)

Idea: instead of making assumptions about the final model,

…let’s look at the full training trajectory!

$$\rightarrow \theta^{(t)} \rightarrow \theta^{(t+1)} \rightarrow \theta^{(t+2)} \rightarrow \cdots \rightarrow \theta^{(T)}$$

Approximating training dynamics (“unrolling”)

[Hara Nitanda Maehara ’19] [Bae Lin Lorraine Grosse ’24]

Example: Let’s consider full-batch gradient descent

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1,\ldots,T$$

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1,\ldots,T$$

Step 1: Consider up-weighting example j **at time** t **only:**

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1,\ldots,T$$

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1,\ldots,T$$

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1,\ldots,T$$

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1, \ldots, T$$

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

$\nabla \ell_j(\theta^{(t)})$

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Example: Let's consider full-batch gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \cdot \sum_{i=1}^{n} \nabla \ell_i(\theta^{(t)}) \qquad t = 1, \ldots, T$$

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

$$\frac{d\theta^{(t+1)}}{d\theta^{(t)}}: \quad \mathbf{I} - \eta_t \cdot \sum_{i=1}^{n} \nabla^2 \ell_i(\theta^{(t)})$$

$$\frac{d\theta^{(t)}}{dw_j^{(t)}}: \quad \nabla \ell_j(\theta^{(t)})$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Hara Nitanda Maehara '19] [Bae Lin Lorraine Grosse '24]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics (“unrolling”)

[Pruthi Liu Sundararajan Kale ‘20]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

TracIn approximation:
ignore second-order effects

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Pruthi Liu Sundararajan Kale '20]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

TracIn approximation:
ignore second-order effects

Step 1: Consider up-weighting example j **at time t only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Bae Lin Lorraine Grosse '24]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Bae Lin Lorraine Grosse '24]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

SOURCE approximation: split sum into segments with "constant" Hessian

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Bae Lin Lorraine Grosse '24]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

SOURCE approximation: split sum into segments with "constant" Hessian

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

$$\frac{d\theta^{(T)}}{dw_j} = \sum_{t=1}^{T} \frac{d\theta^{(T)}}{dw_j^{(t)}}$$

(total derivative rule)

Approximating training dynamics ("unrolling")

[Bae Lin Lorraine Grosse '24]

Problem: This is very expensive to compute! (T Hessians 😬)

Solutions: Approximate unrolling

SOURCE approximation:
split sum into segments with
"constant" Hessian

Step 1: Consider up-weighting example j **at time** t **only:**

$$\frac{d\theta^{(T)}}{dw_j^{(t)}} = \frac{d\theta^{(T)}}{d\theta^{(T-1)}} \cdot \frac{d\theta^{(T-1)}}{d\theta^{(T-2)}} \cdots \frac{d\theta^{(t+1)}}{d\theta^{(t)}} \cdot \frac{d\theta^{(t)}}{dw_j^{(t)}}$$

(chain rule)

Step 2: Compute **total effect** of example j by summing:

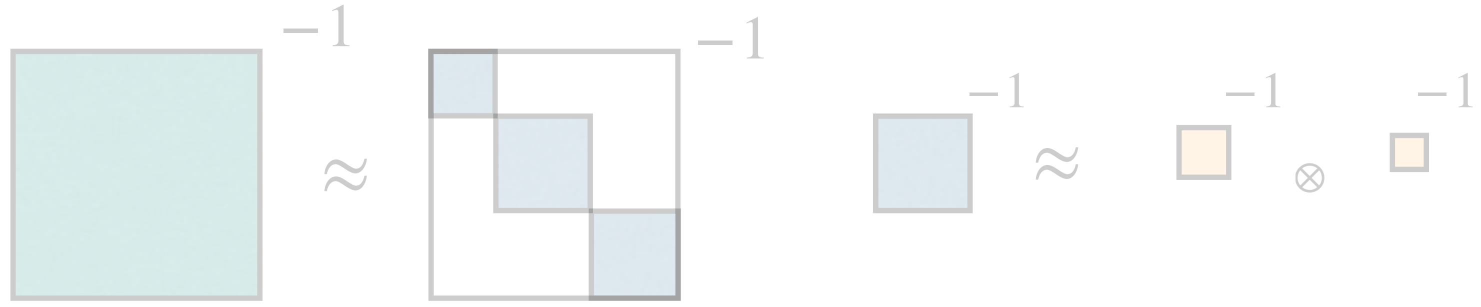
rule)

It is possible to approximately "retrace" the GD trajectory

A quick tour of the landscape

Themes:

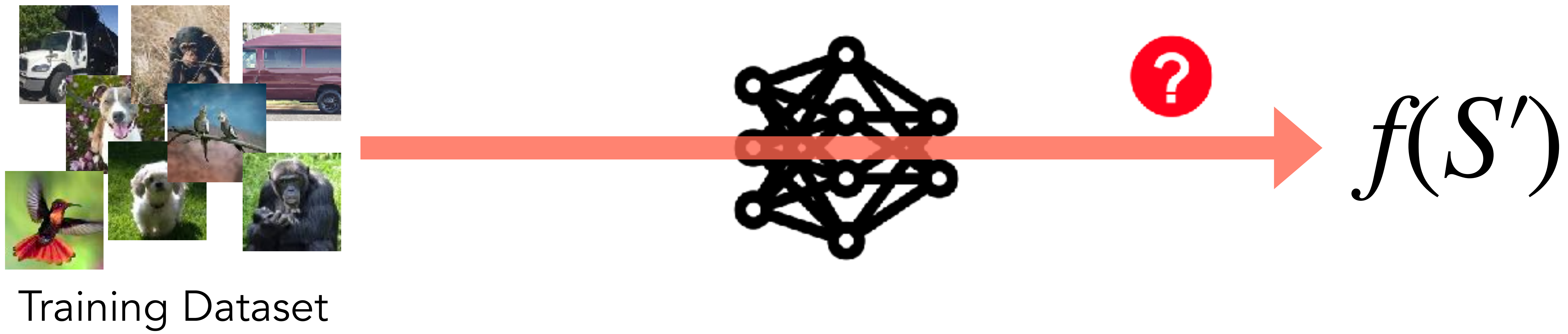
- Better IF/Hessian approximations
- Approximating training dynamics (“unrolling”)
- **Surrogate models**



[BLLG '24]

Simple surrogate models

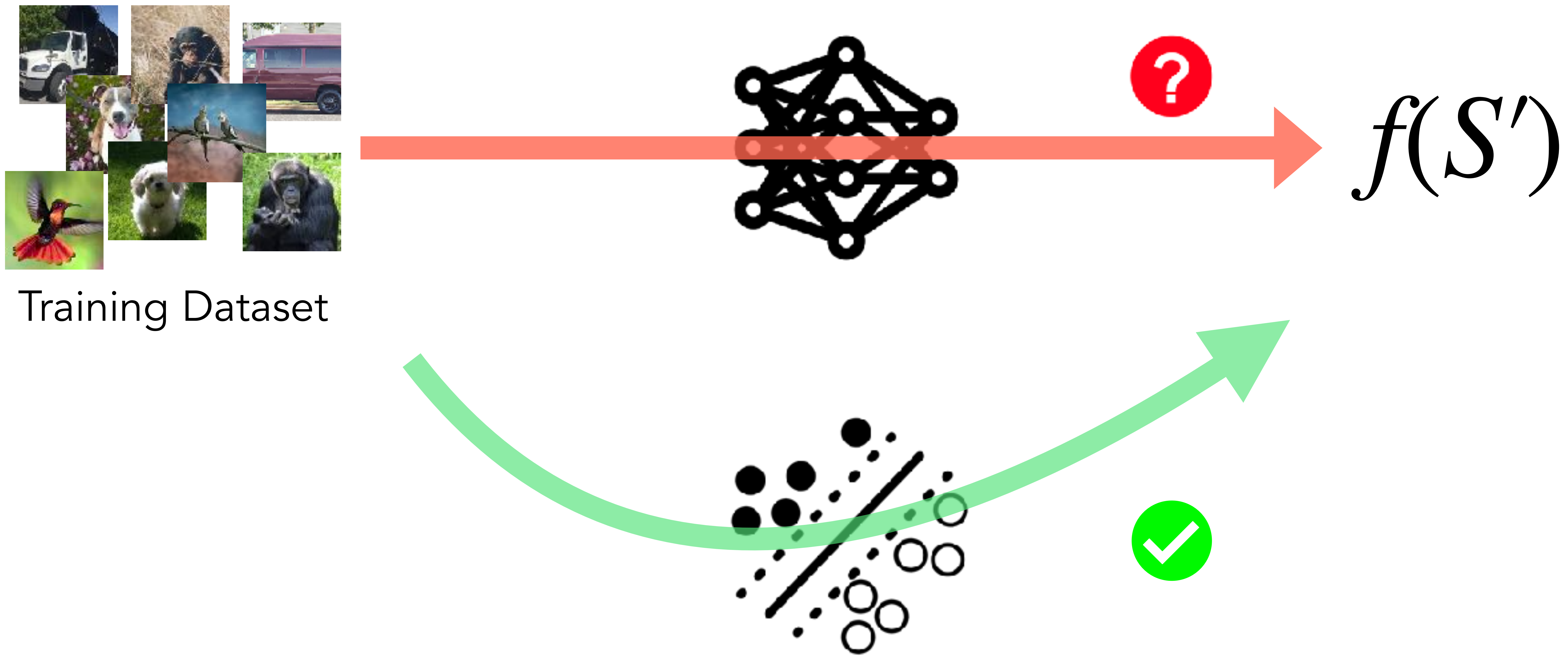
Original model might be too hard to attribute directly



Simple surrogate models

Observation: we **can** attribute simple model classes (e.g., linear, k-NNs) well

(Recall: Chapter 2!)

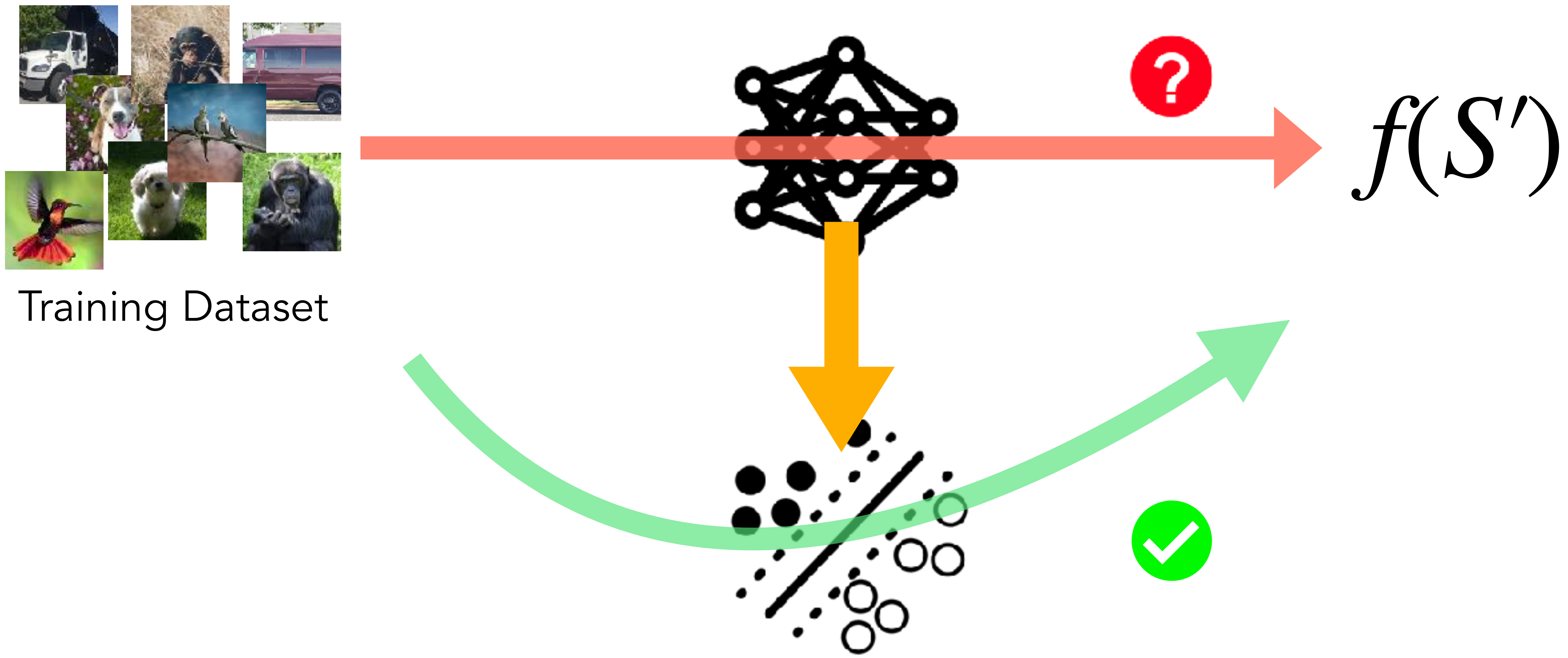


Simple surrogate models

Observation: we **can** attribute simple model classes (e.g., linear, k-NNs) well

Idea: use a **simple surrogate model** that approximates the original model

Then, compute attributions based on the surrogate model



Simple surrogate models

Observation: we **can** attribute simple model classes (e.g., linear, k-NNs) well

Idea: use a **simple surrogate model** that approximates the original model

Then, compute attributions based on the surrogate model

Examples:

- k-NN classifiers [Jia et al. '19]
- linear classifier on last layer representations [Koh Liang '17]
- NTK kernel approximation [PGI+'23]

Simple surrogate models

Observation: we **can** attribute simple model classes (e.g., linear, k-NNs) well

Idea: use a **simple surrogate model** that approximates the original model

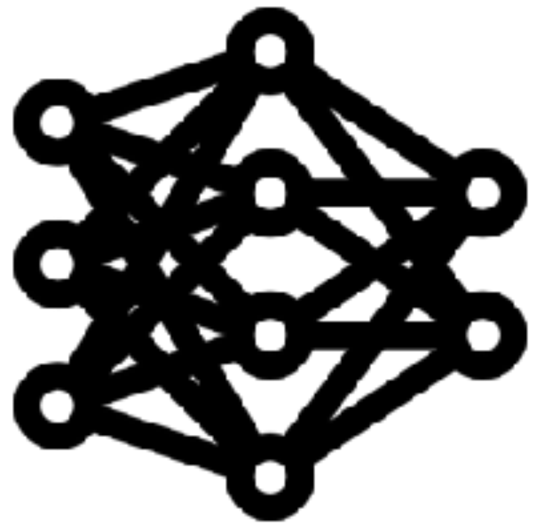
Then, compute attributions based on the surrogate model

Examples:

- k-NN classifiers [Jia et al. '19]
- linear classifier on last layer representations [Koh Liang '17]
- **NTK kernel approximation** [PGI+'23]

Simple surrogate models: TRAK

[Park Georgiev Ilyas Leclerc Madry ’23]



Differentiable model $\theta^\star$

Simple surrogate models: TRAK

[Park Georgiev Ilyas Leclerc Madry '23]

Differentiable model $\theta^\star$

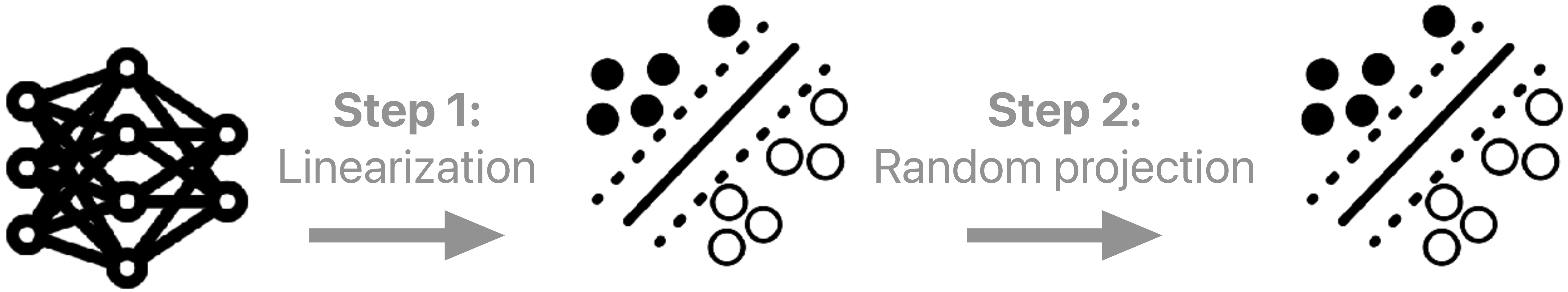
High-dim linear model $\theta^{(L)}$

Idea 1: *Empirical* Neural Tangent Kernel is a good proxy for final model

e.g., [Malladi Wettig Yu Chen Arora '23]

Simple surrogate models: TRAK

[Park Georgiev Ilyas Leclerc Madry '23]



Differentiable model $\theta^{\star}$

High-dim linear model $\theta^{(L)}$

Low-dim linear model $\theta^{(P)}$

Idea 1: *Empirical* Neural Tangent Kernel is a good proxy for final model

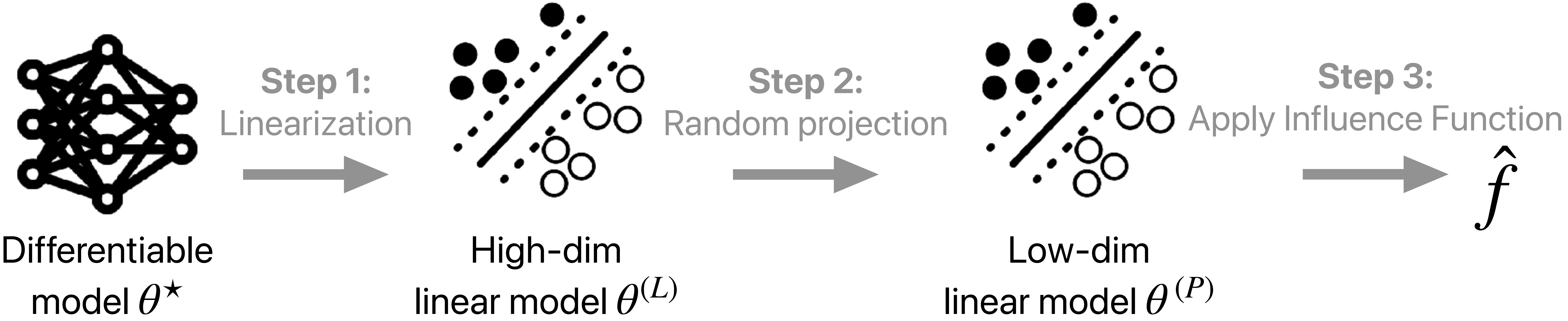
e.g., [Malladi Wettig Yu Chen Arora '23]

Idea 2: Random projections preserve relevant linear structure

e.g., [LeJeune Patil Javadi Baraniuk Tibshirani '24]

Simple surrogate models: TRAK

[Park Georgiev Ilyas Leclerc Madry '23]



Idea 1: *Empirical* Neural Tangent Kernel is a good proxy for final model

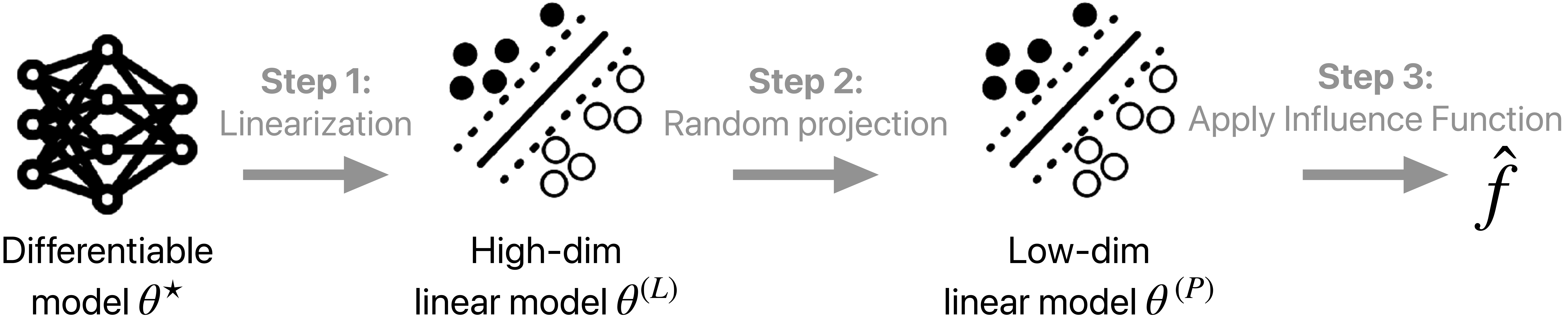
e.g., [Malladi Wettig Yu Chen Arora '23]

Idea 2: Random projections preserve relevant linear structure

e.g., [LeJeune Patil Javadi Baraniuk Tibshirani '24]

Simple surrogate models: TRAK

[Park Georgiev Ilyas Leclerc Madry ’23]



Well-designed surrogate models can be a good proxy for attributing original NN

Evaluating the landscape

Evaluating the landscape

★ TRAK [PGI+23]
□ EK-FAC [GBA+23]
○ Datamodel [IPE+22]
◇ Emp. Influence [FZ20]
□ IF [KL17]
⬠ Representation Sim.
▷ GAS [HL22]
◁ TracIn [PLS+20]

Correlation
(more accurate →)

Computation time (mins) on 1xA100
(← more efficient)

Evaluating the landscape

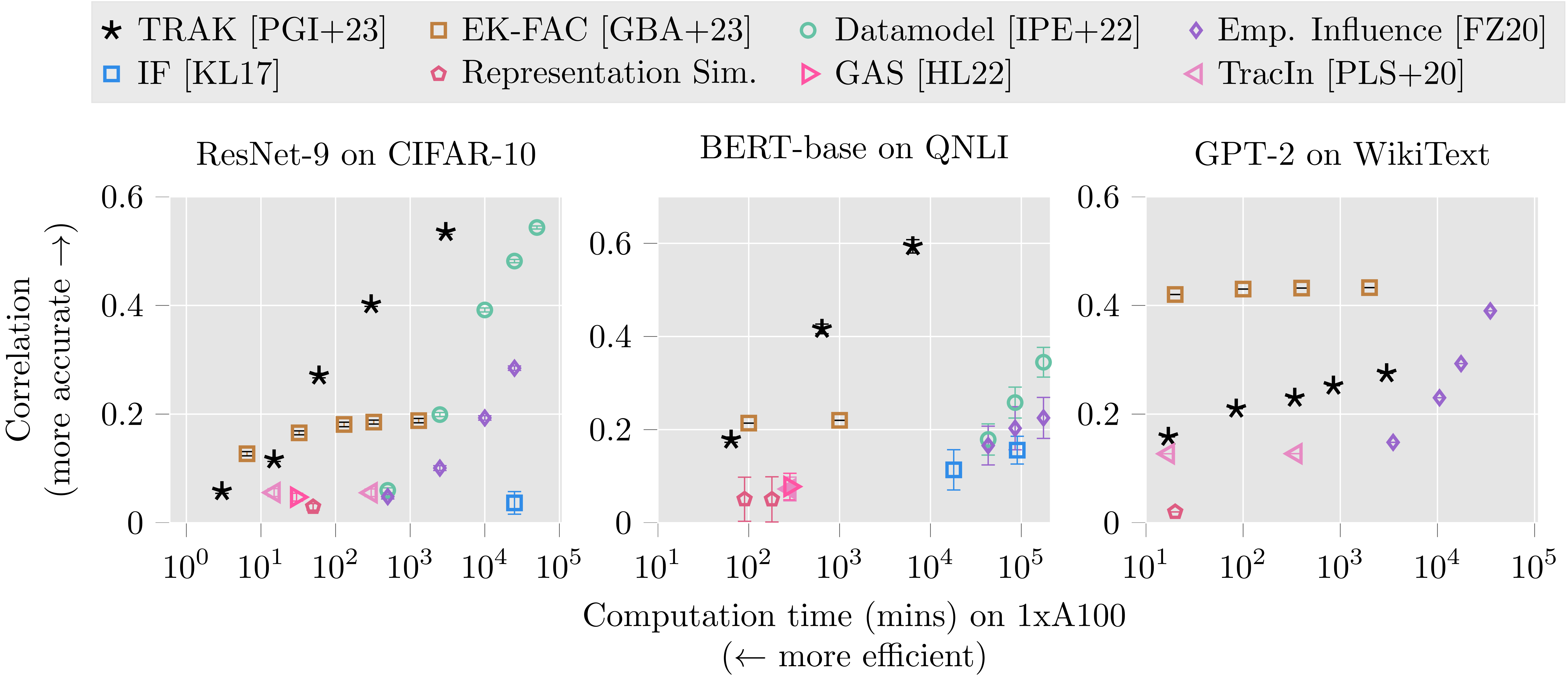
⋆ TRAK [PGI+23]
□ EK-FAC [GBA+23]
○ Datamodel [IPE+22]
◇ Emp. Influence [FZ20]
□ IF [KL17]
⬠ Representation Sim.
▷ GAS [HL22]
◁ TracIn [PLS+20]

LDS

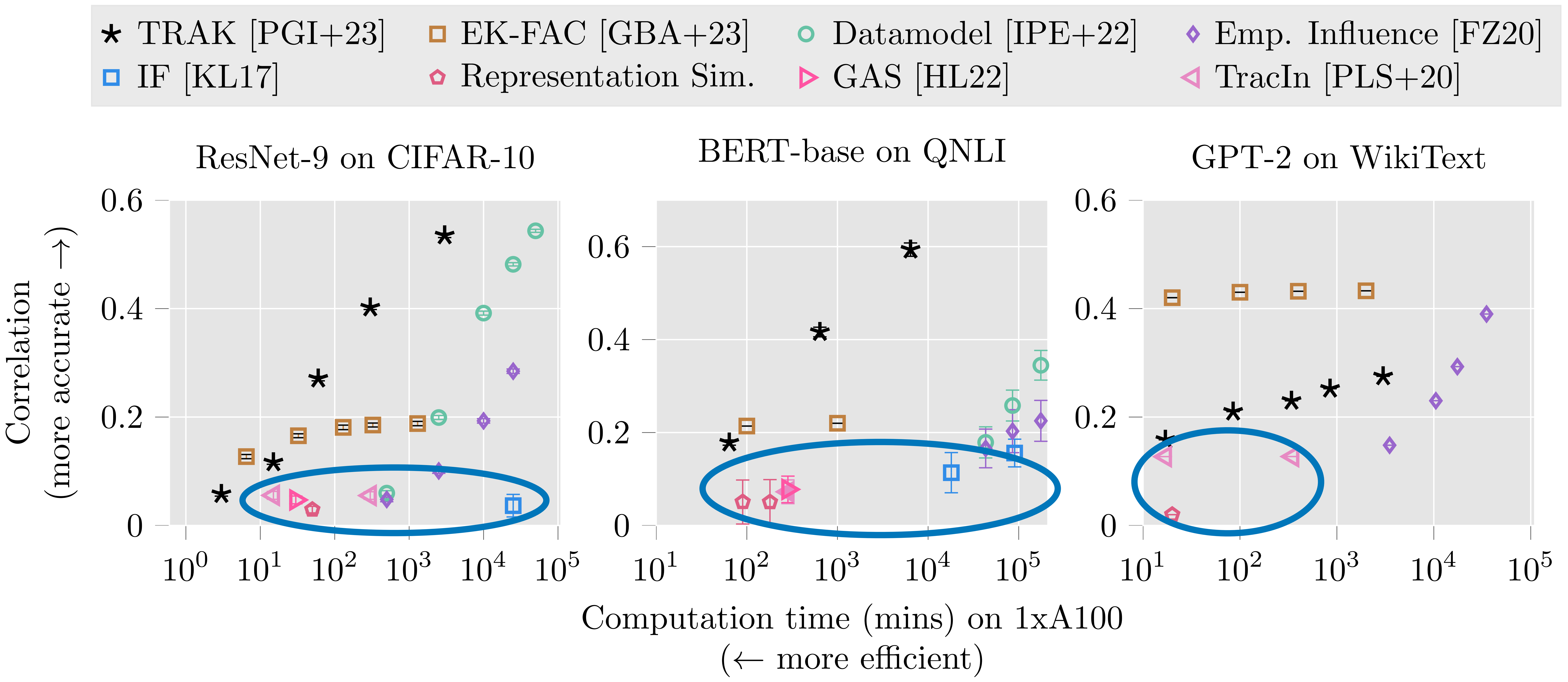
Correlation
(more accurate →)

Computation time (mins) on 1xA100
(← more efficient)

Evaluating the landscape

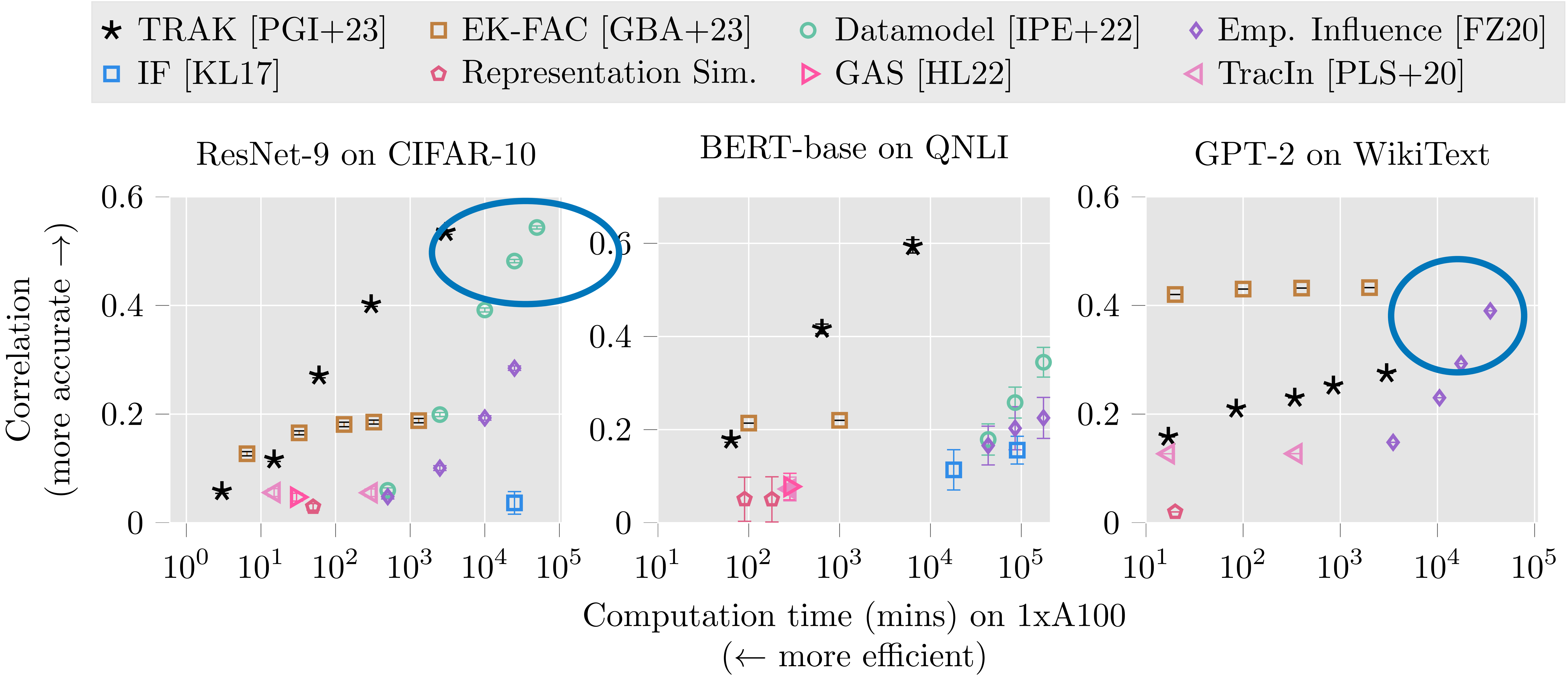


Evaluating the landscape



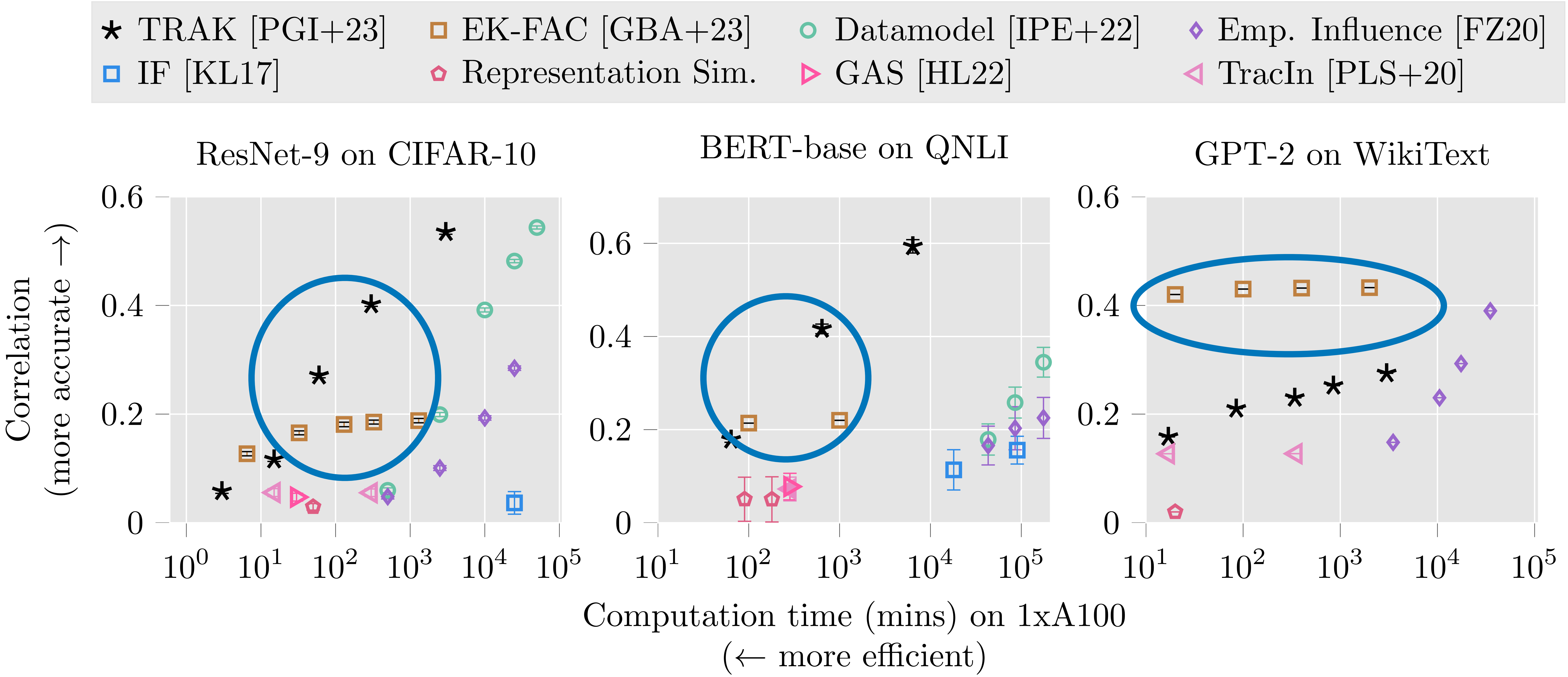
Popular baselines (original IF, rep. similarity, TracIn) **not** very predictive

Evaluating the landscape



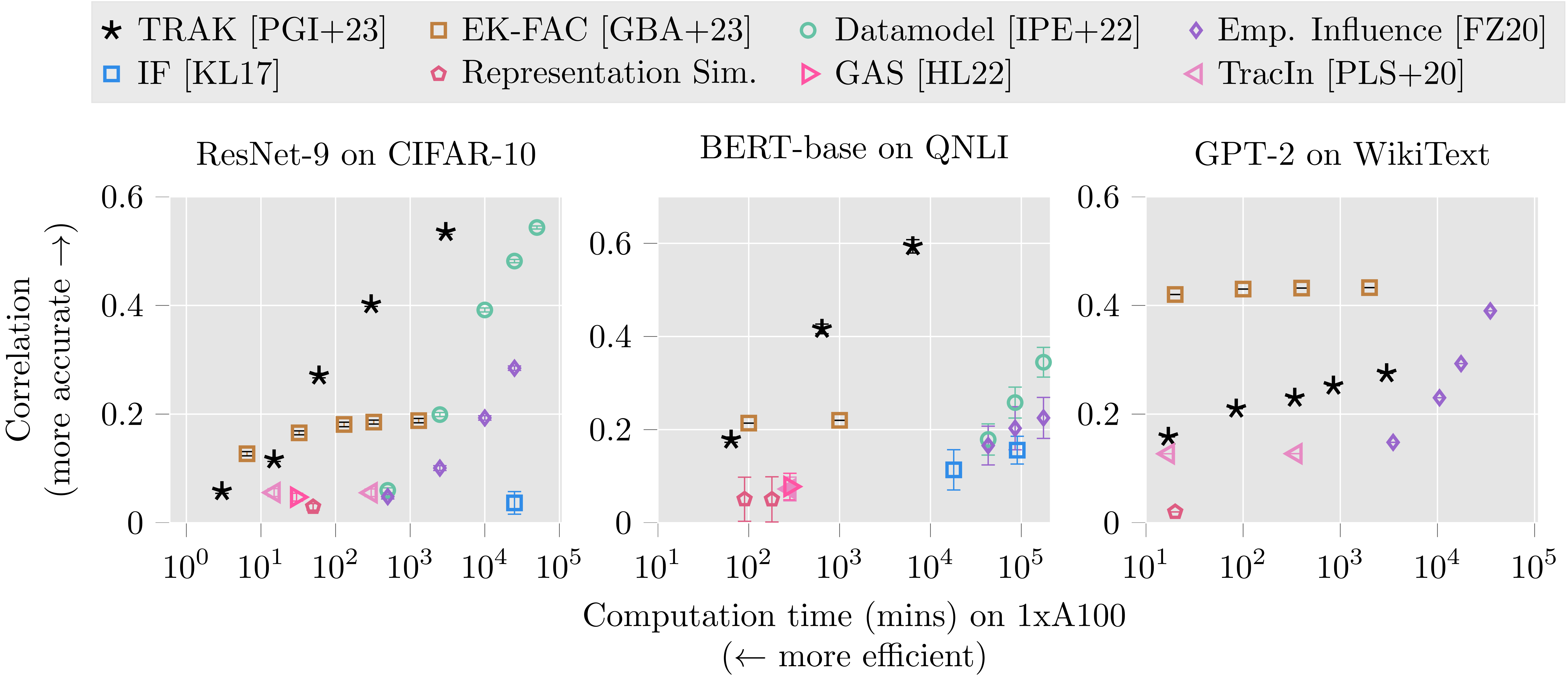
Direct estimators (regression) perform best with enough compute

Evaluating the landscape



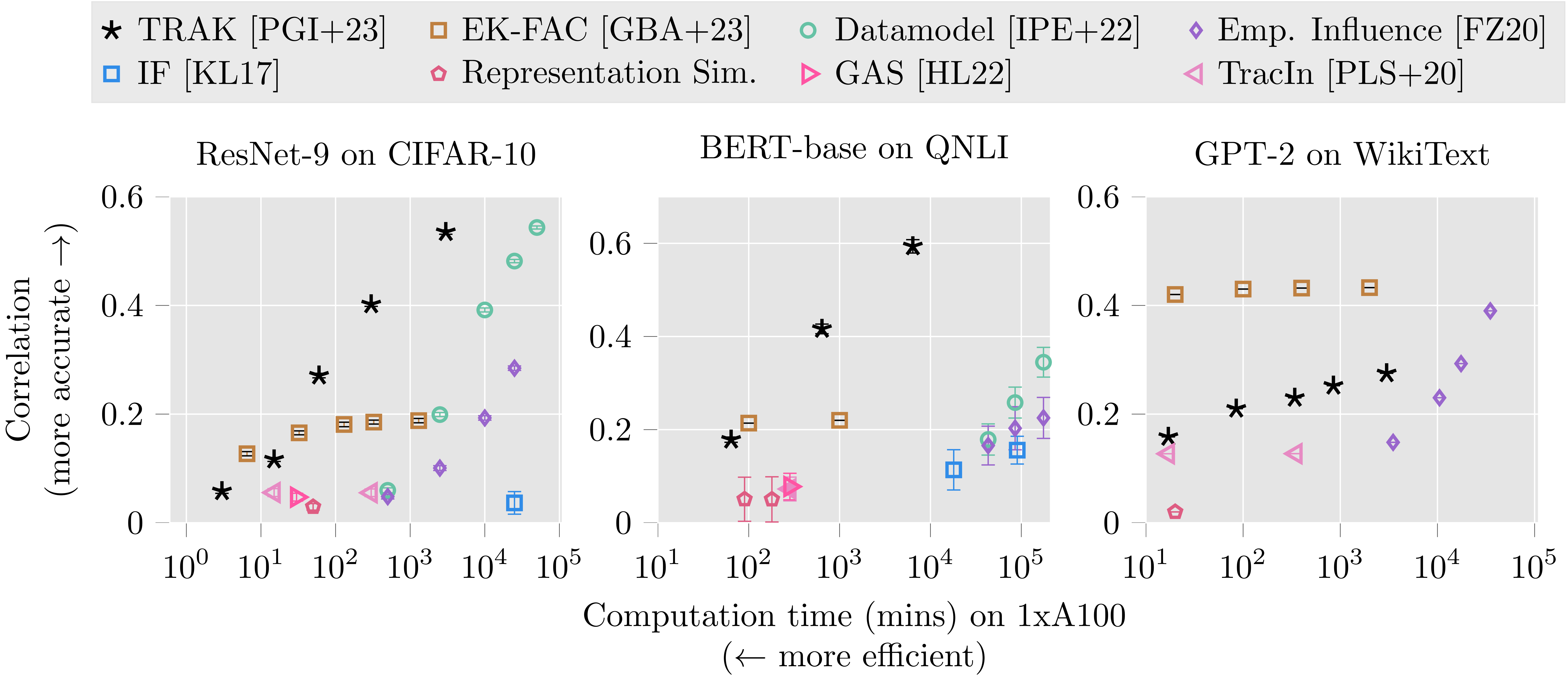
Now have efficient methods (e.g., TRAK, EK-FAC) that approach similar LDS

Evaluating the landscape



Trade-offs depending on target task:
e.g., TRAK better on vision, EK-FAC better for language modeling

Evaluating the landscape



Main takeaway: we have fast, predictive methods now! Use them

Takeaways

If you care about predictive attribution, evaluate **counterfactuals**

Use LDS or similar

Use **good** attribution methods!

Many popular baselines are not predictive, but some are quite reliable now

Some key themes (better Hessian approximation; unrolling; surrogate models)

Choose method appropriate to **modality** and **costs**

Future work

Better methods

- Beyond linear methods
- Multiple training stages
- Are there better “surrogate” models for DNNs?

“Single model counterfactual”

More efficient evaluation proxies

Applying data attribution

Current and future applications of data attribution

Chapter 4 of 4

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Predictive data attribution

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Predictive data attribution

Model debugging: motivation

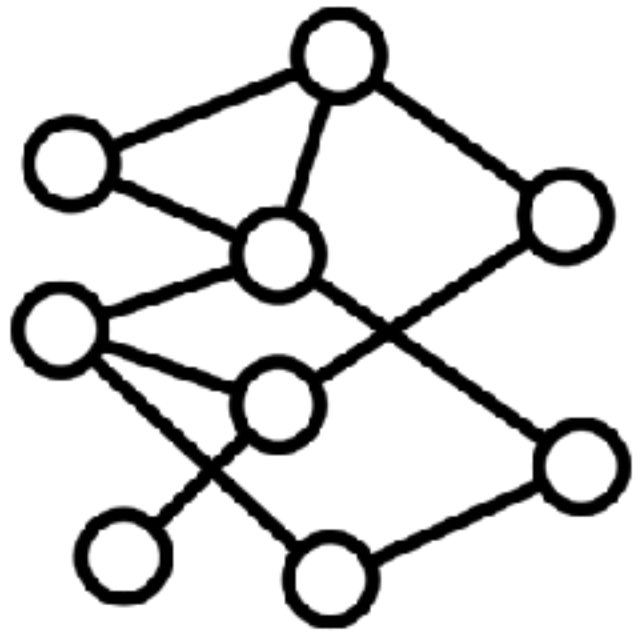
Model debugging: motivation

Goal: "understand" model behavior

Model debugging: motivation

Goal: "understand" model behavior

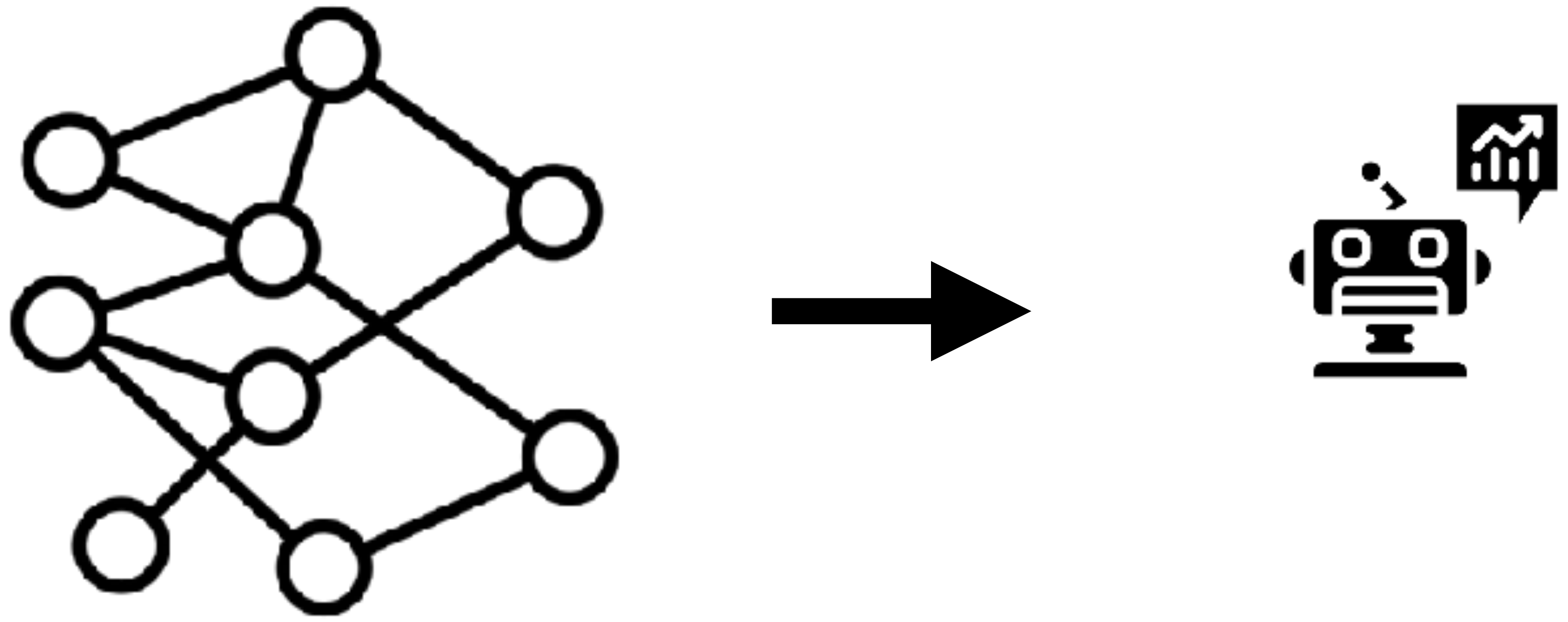
Model



Model debugging: motivation

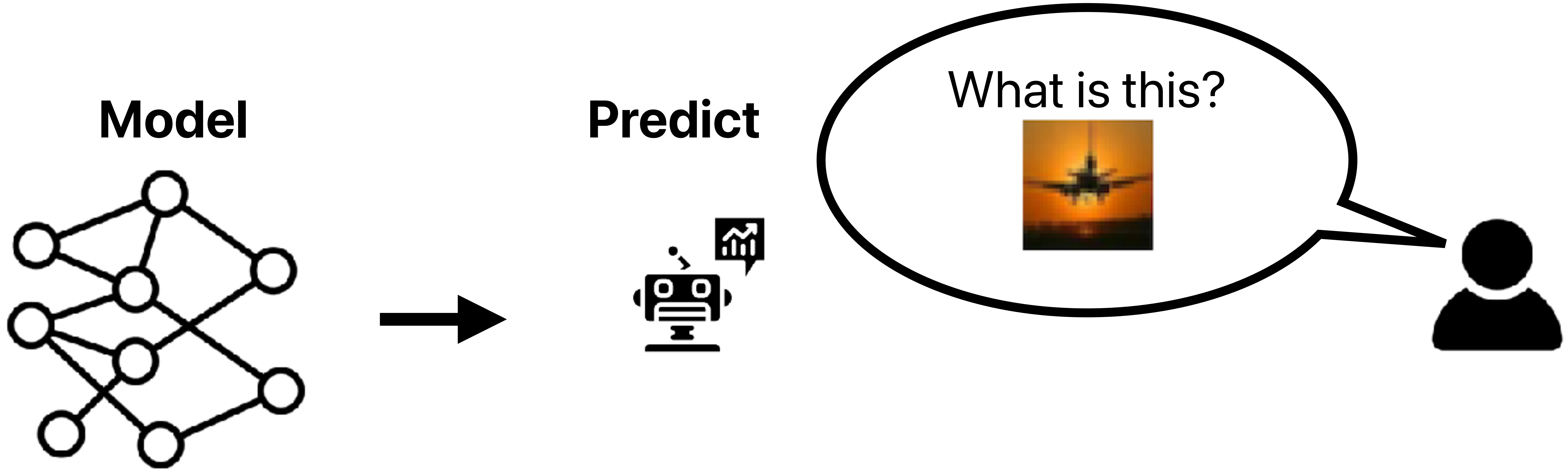
Goal: "understand" model behavior

Model **Predict**



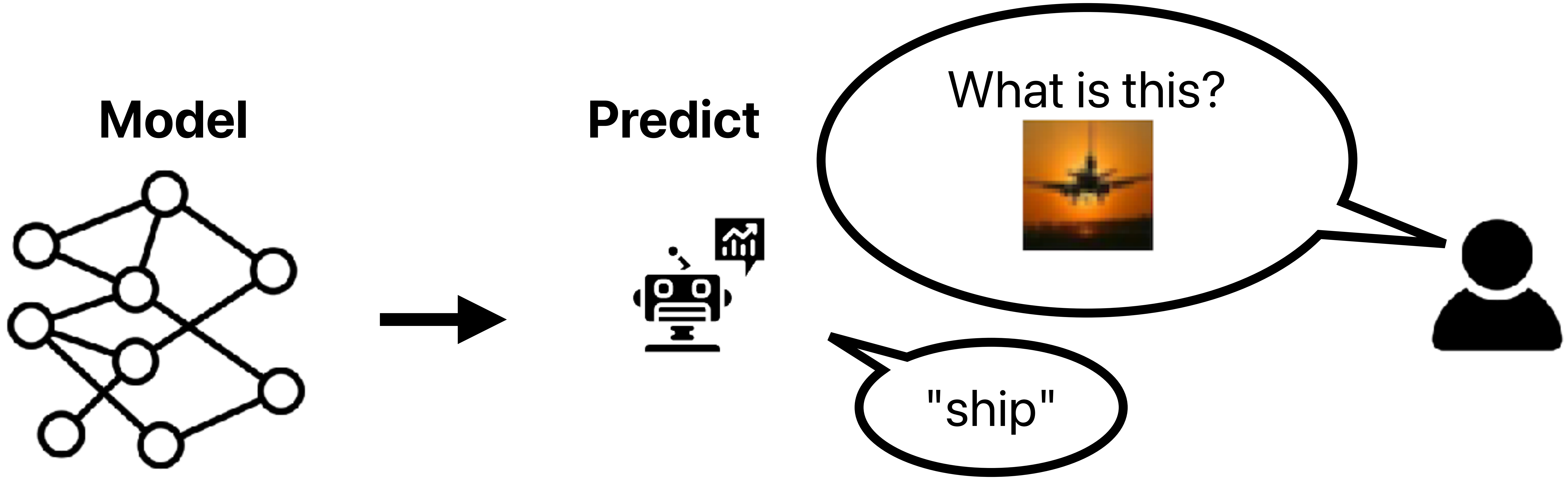
Model debugging: motivation

Goal: "understand" model behavior



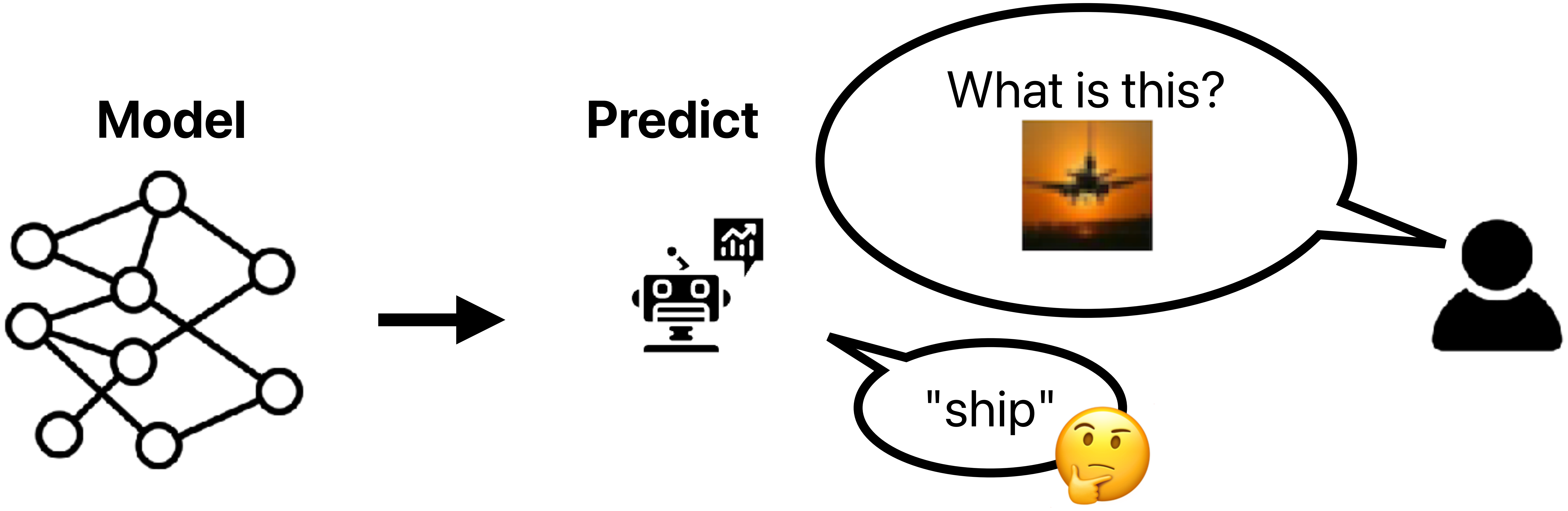
Model debugging: motivation

Goal: "understand" model behavior



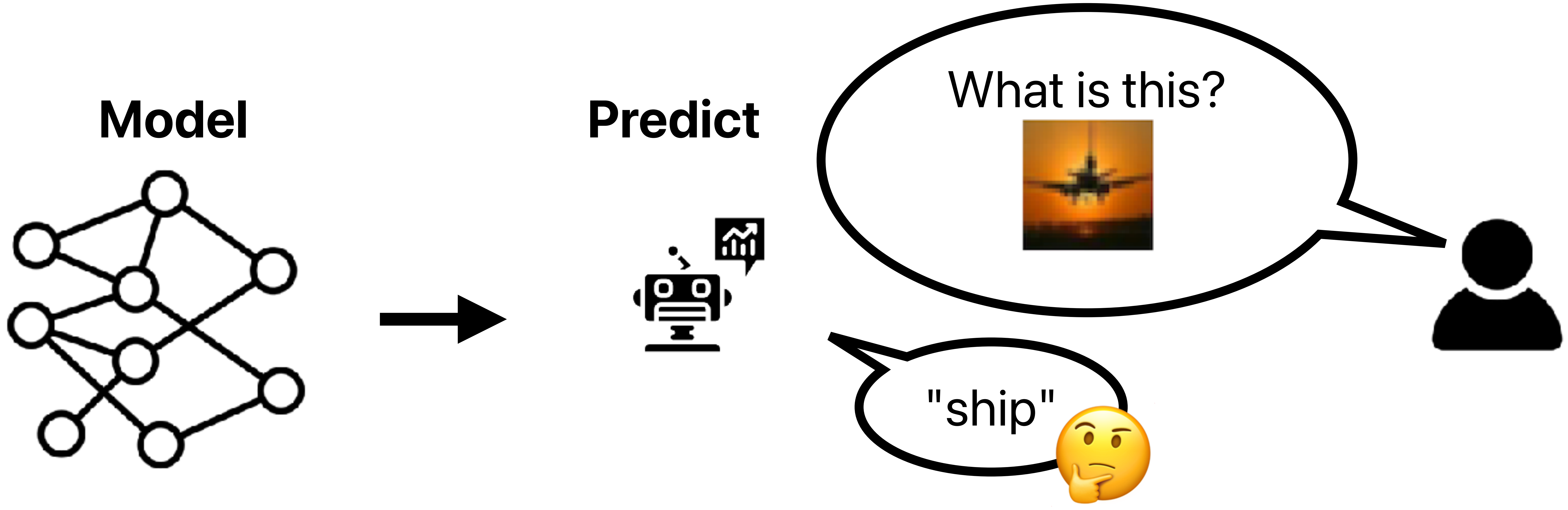
Model debugging: motivation

Goal: "understand" model behavior



Model debugging: motivation

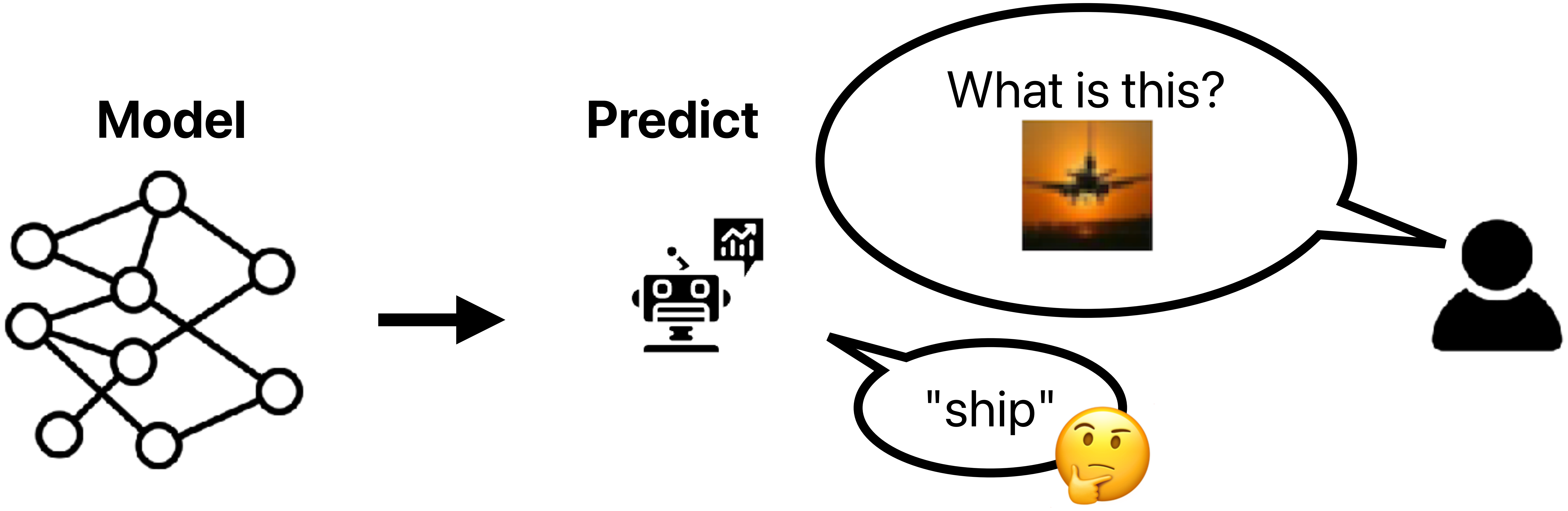
Goal: "understand" model behavior



Example: why is my model wrong?

Model debugging: motivation

Goal: "understand" model behavior

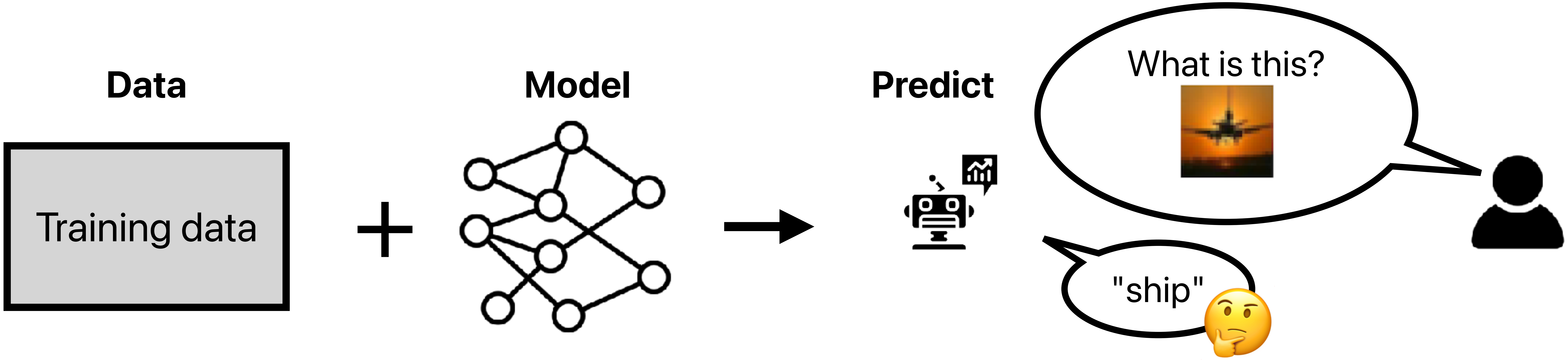


Example: why is my model wrong?

Data-centric answer: look at data to find potential explanations

Model debugging: motivation

Goal: "understand" model behavior



Example: why is my model wrong?

Data-centric answer: look at data to find potential explanations

Model debugging with data attribution

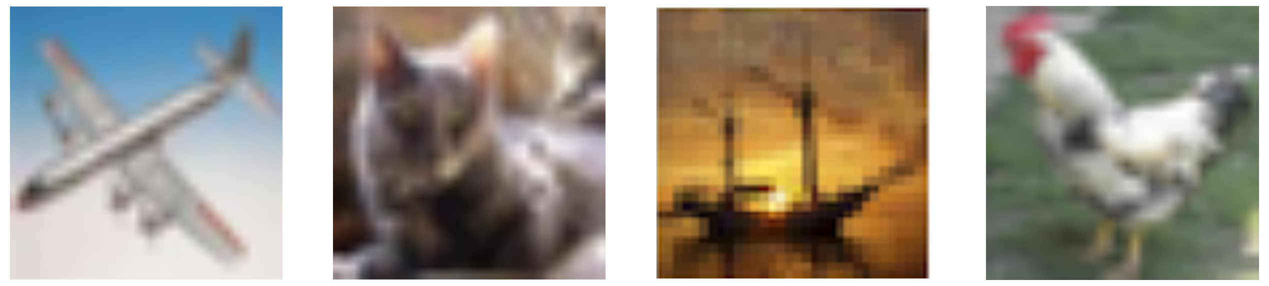
Model debugging with data attribution

Data lens: find potential explanations in the training data

Model debugging with data attribution

Data lens: find potential explanations in the training data

Data



Model debugging with data attribution

Data lens: find potential explanations in the training data

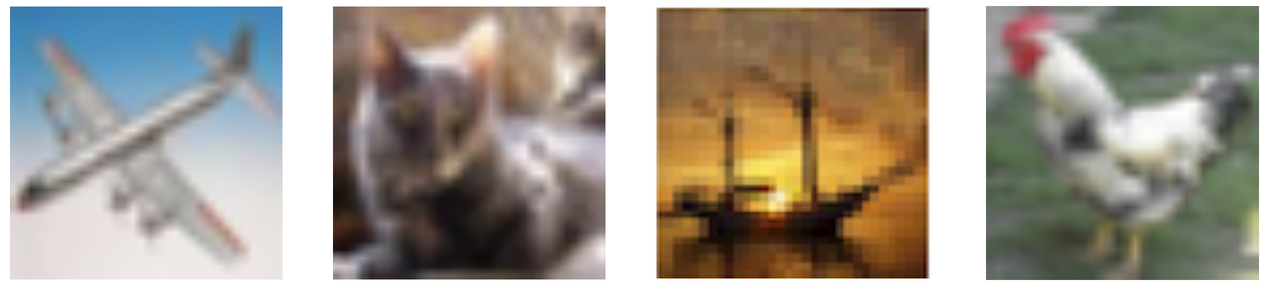
Data

Model training

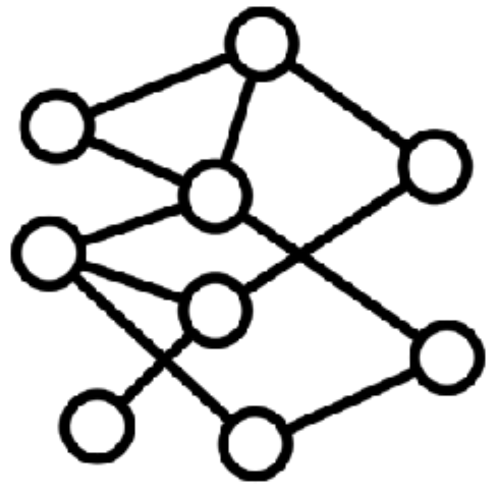
Model debugging with data attribution

Data lens: find potential explanations in the training data

Data



Model training

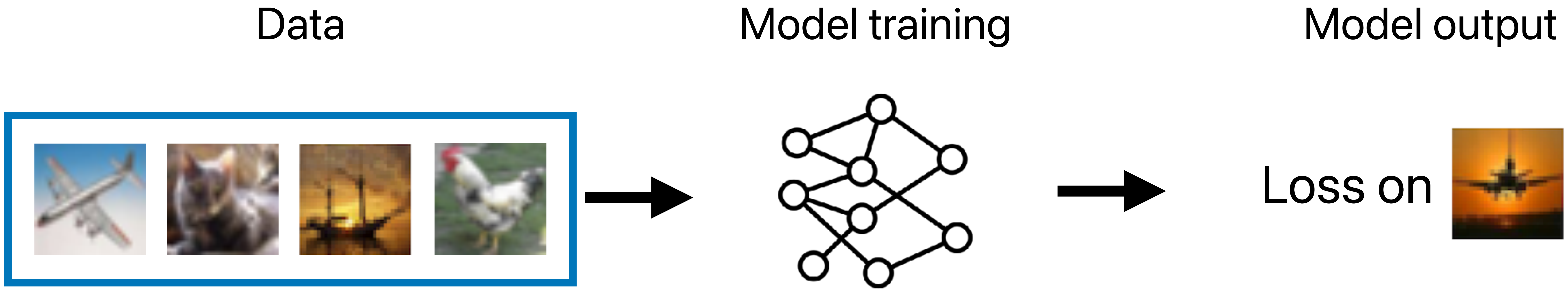


Model output

Loss on 

Model debugging with data attribution

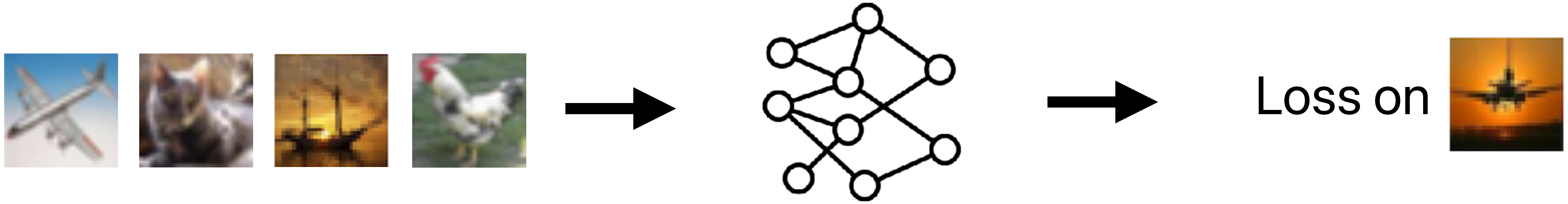
Data lens: find potential explanations in the training data



Model debugging with data attribution

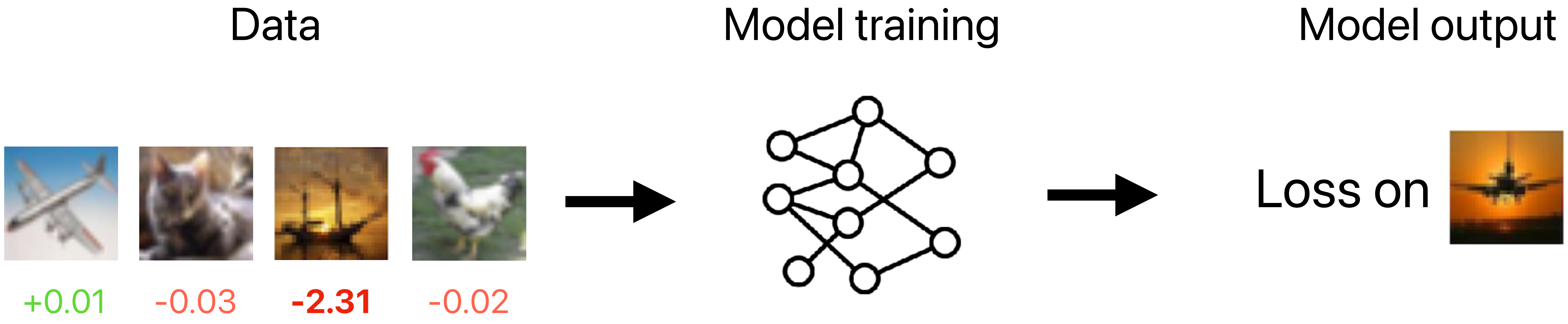
Data lens: find potential explanations in the training data

Data | Model training | Model output



Model debugging with data attribution

Data lens: find potential explanations in the training data

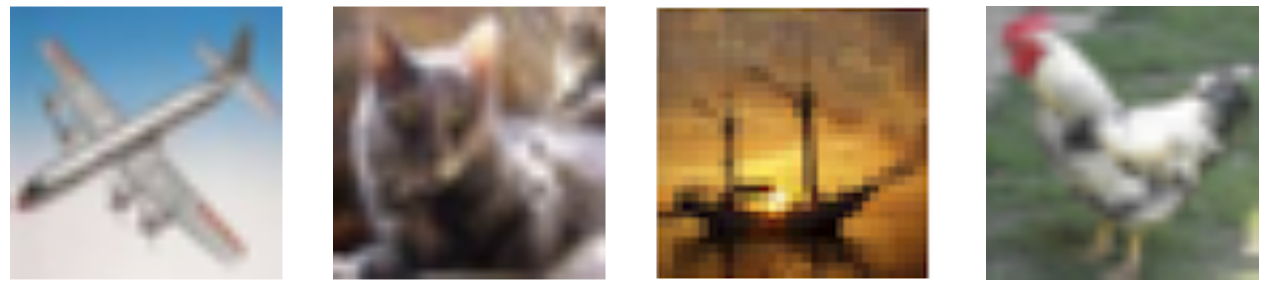


Common approach: give each training sample an "importance," then *inspect*

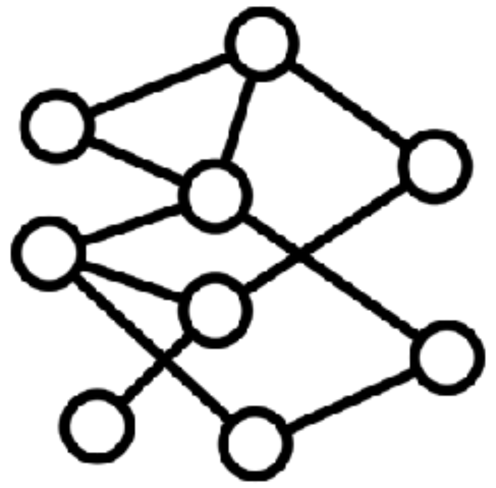
Model debugging with data attribution

Data lens: find potential explanations in the training data

Data | Model training | Model output



+0.01 -0.03 **-2.31** -0.02



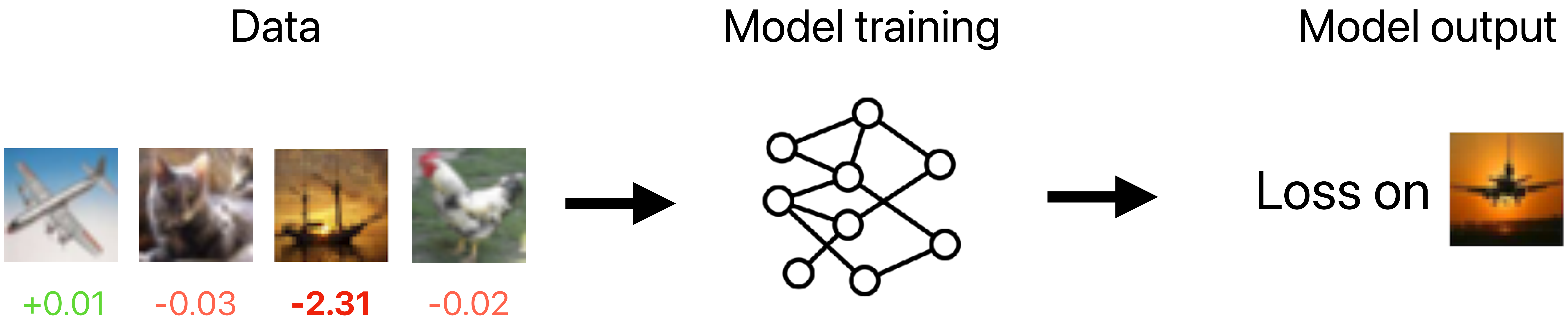
Loss on 

Common approach: give each training sample an "importance," then *inspect*

"Importance" means something different in each data attribution framework!

Model debugging with data attribution

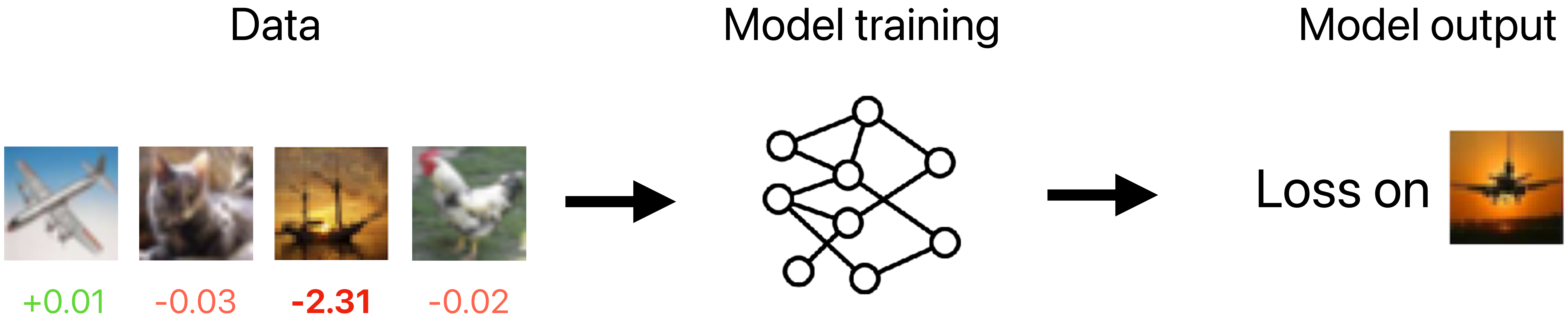
Data lens: find potential explanations in the training data



Common approach: give each training sample an "importance," then *inspect*

Model debugging with data attribution

Data lens: find potential explanations in the training data



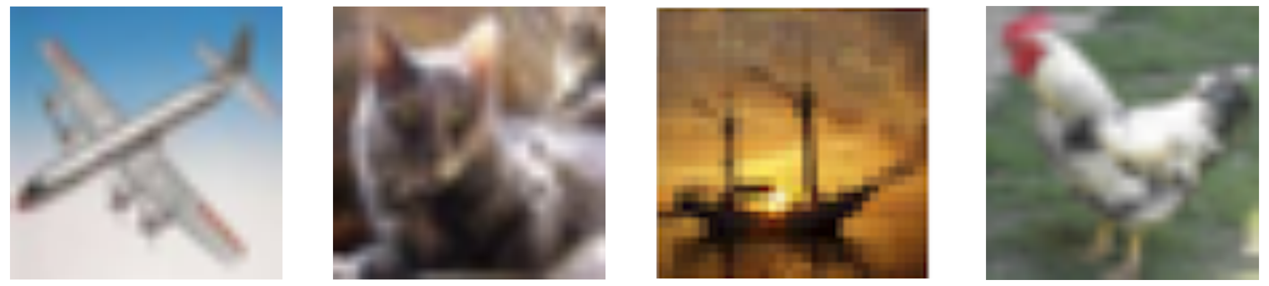
Common approach: give each training sample an "importance," then *inspect*

Train Sample

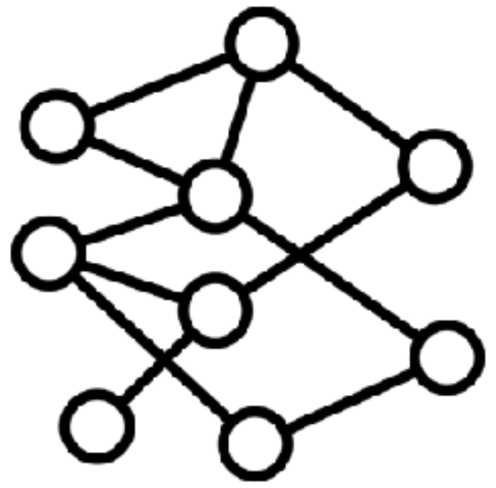
Model debugging with data attribution

Data lens: find potential explanations in the training data

Data | Model training | Model output

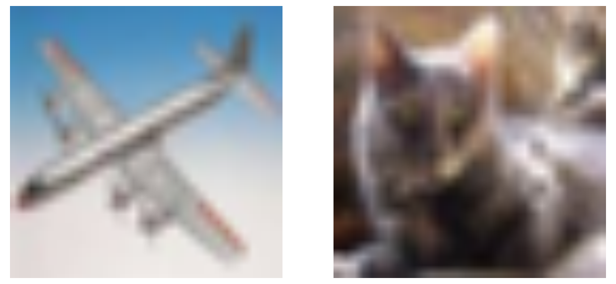
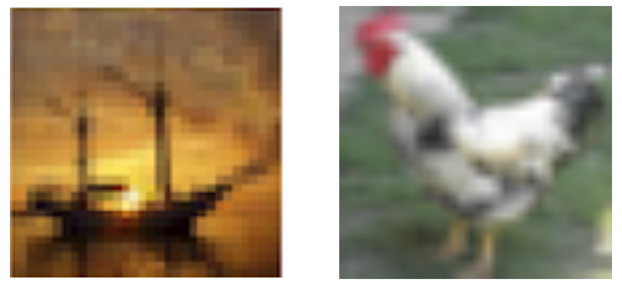


+0.01 -0.03 **-2.31** -0.02



Loss on 

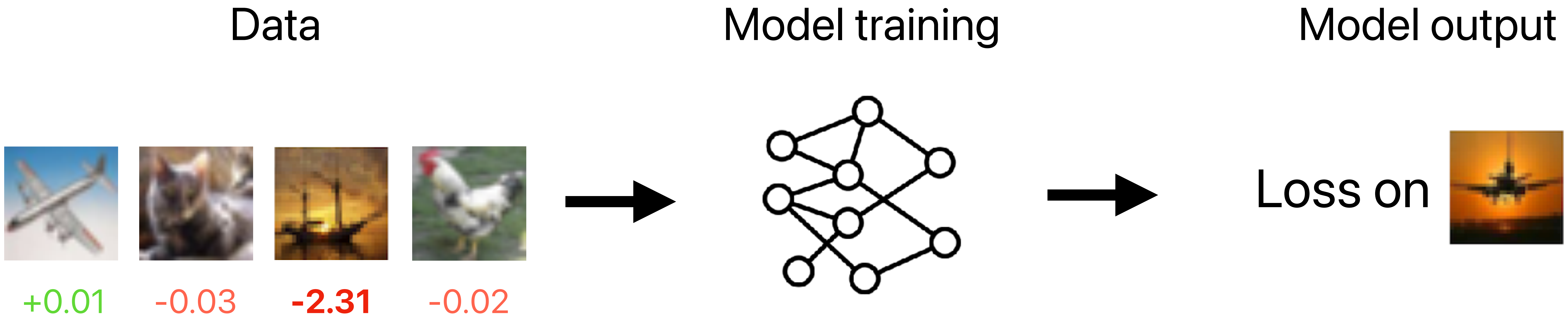
Common approach: give each training sample an "importance," then *inspect*

Train Sample  … … 

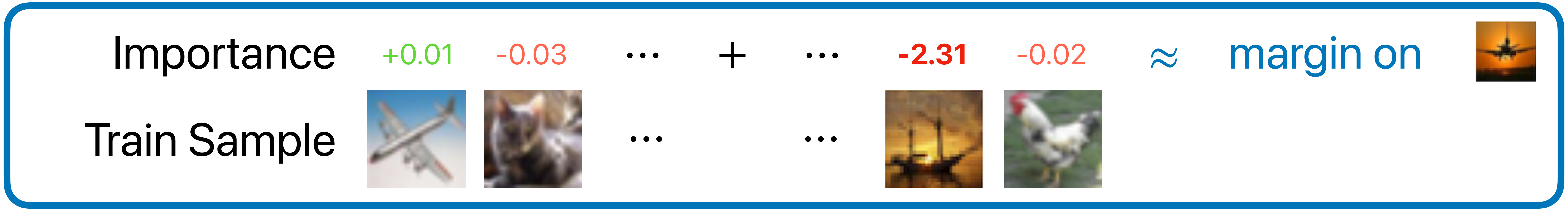
Predictive view: each weight is effect of "training on sample" on margin

Model debugging with data attribution

Data lens: find potential explanations in the training data



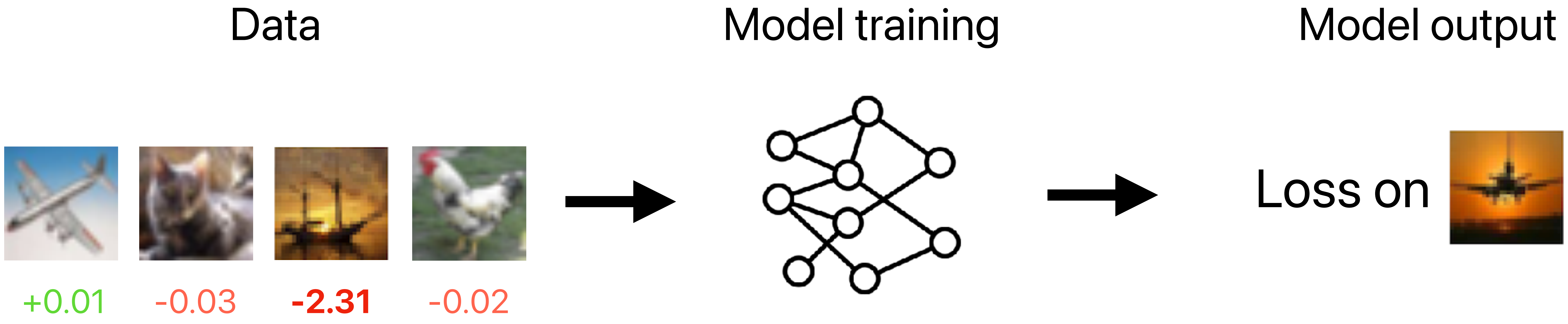
Common approach: give each training sample an "importance," then *inspect*



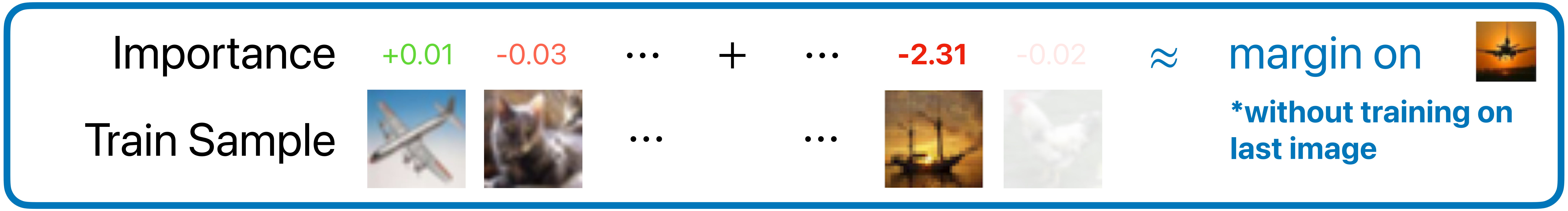
Predictive view: each weight is effect of "training on sample" on margin

Model debugging with data attribution

Data lens: find potential explanations in the training data



Common approach: give each training sample an "importance," then *inspect*

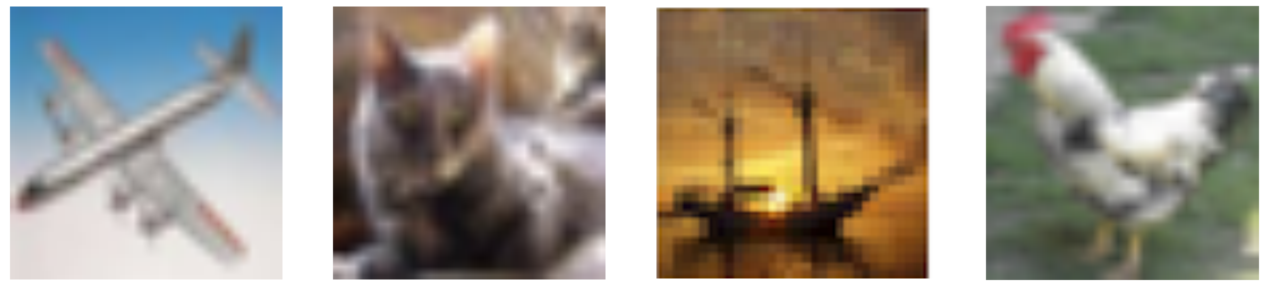


Predictive view: each weight is effect of "training on sample" on margin

Model debugging with data attribution

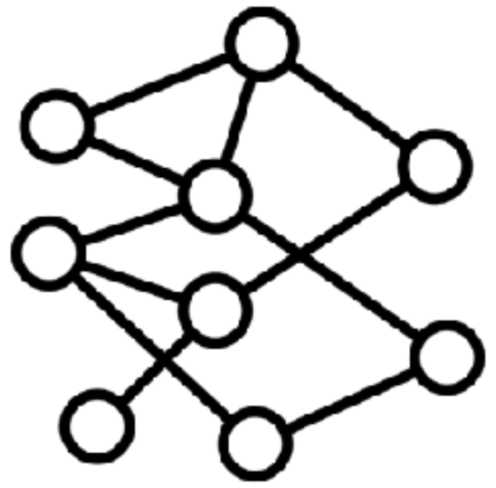
Data lens: find potential explanations in the training data

Data — Model training — Model output



+0.01 -0.03 **-2.31** -0.02

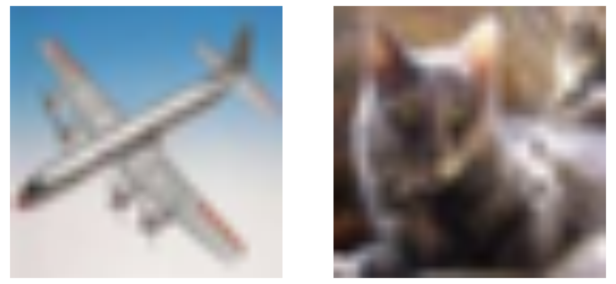
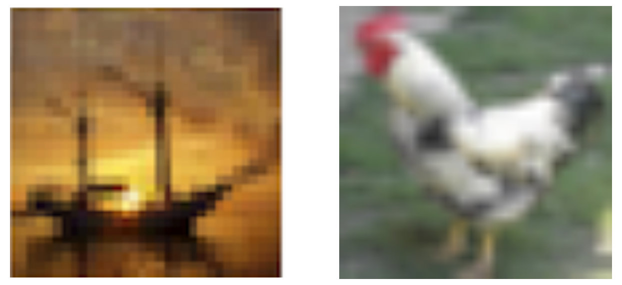
→



→

Loss on 

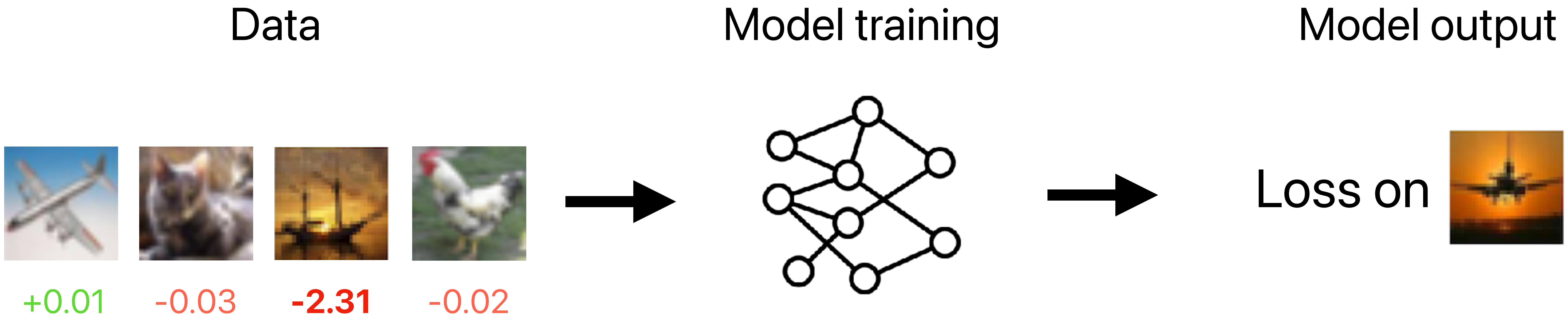
Common approach: give each training sample an "importance," then *inspect*

Importance	+0.03	+0.02	⋯	+	⋯	**-1.31**	-1.22
Train Sample			⋯		⋯		

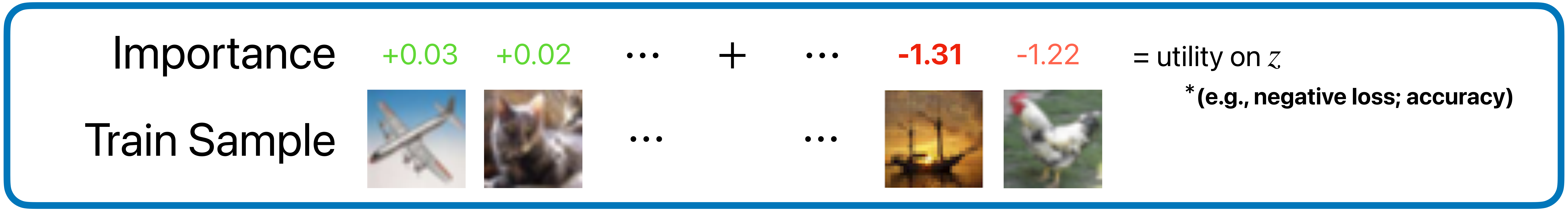
Game theoretic view: each weight is "fair share" of utility in a cooperative game

Model debugging with data attribution

Data lens: find potential explanations in the training data



Common approach: give each training sample an "importance," then *inspect*

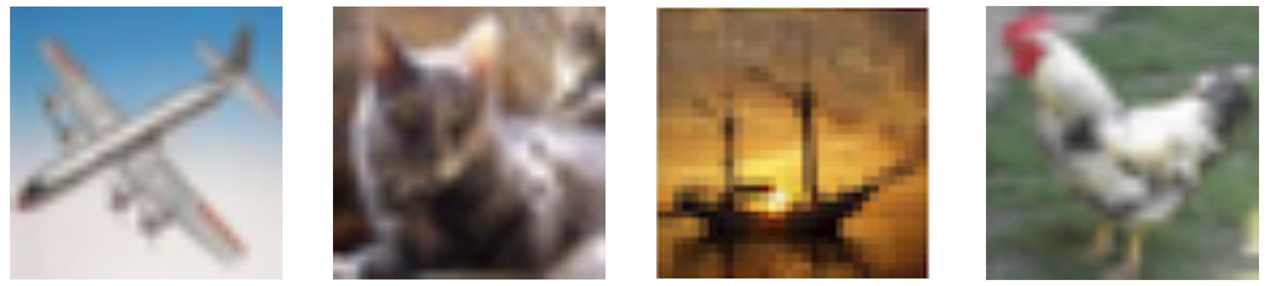


Game theoretic view: each weight is "fair share" of utility in a cooperative game

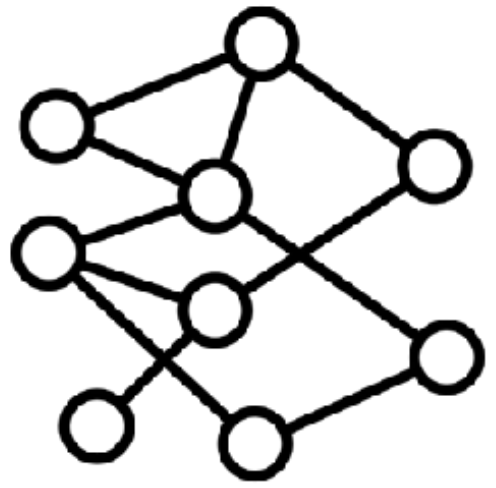
Model debugging with data attribution

Data lens: find potential explanations in the training data

Data — Model training — Model output

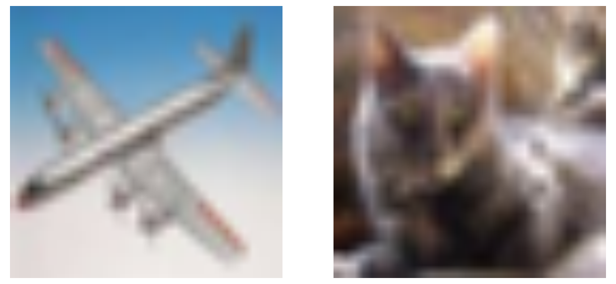
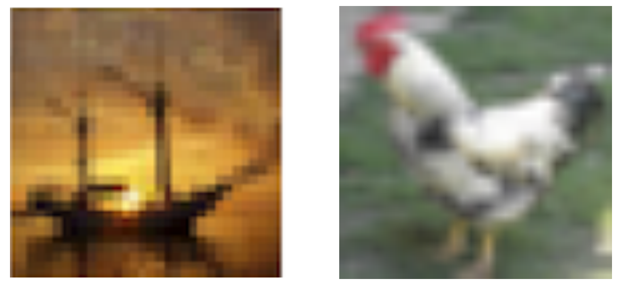


+0.01 -0.03 **-2.31** -0.02



Loss on 

Common approach: give each training sample an "importance," then *inspect*

Importance	0.67	0.5	···	+	···	**0.05**	0.82
Train Sample			···		···		

Corroborative view: each weight is similarity to sample

Model debugging with data attribution

Model debugging with data attribution

What do these "importances" yield?

Model debugging with data attribution

What do these "importances" yield?

One approach: inspect most influential samples.. obvious hypothesis arises! 🌅

Model debugging with data attribution

What do these "importances" yield?

One approach: inspect most influential samples.. obvious hypothesis arises! 🌅

Model debugging with data attribution

What do these "importances" yield?

One approach: inspect most influential samples.. obvious hypothesis arises!

Sample | Most "positive importance" samples | Most "negative importance" samples

Model debugging with data attribution

What do these "importances" yield?

One approach: inspect most influential samples.. obvious hypothesis arises!

Sample | Most "positive importance" samples | Most "negative importance" samples

Conceptual questions: verifying generated hypotheses, operationalizing insights

Model debugging with data attribution

What do these "importances" yield?

One approach: inspect most influential samples.. obvious hypothesis arises! 🌅

Sample | Most "positive importance" samples | Most "negative importance" samples

Conceptual questions: verifying generated hypotheses, operationalizing insights

References: [Koh Liang 2017; Ghorbani Zou 2019; Guo et al. 2020; Pruthi et al. 2020; Tang et al. 2021a,b; Basu et al. 2021; Ilyas et al. 2022; Shah et al. 2022; Karlaš et al. 2022; Grosse et al. 2023; Park et al. 2023; Rosenfeld Risteski 2023; Konz et al. 2023; Wang et al. 2023; Xia et al. 2024]

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Predictive data attribution

Recap: predictive data attribution

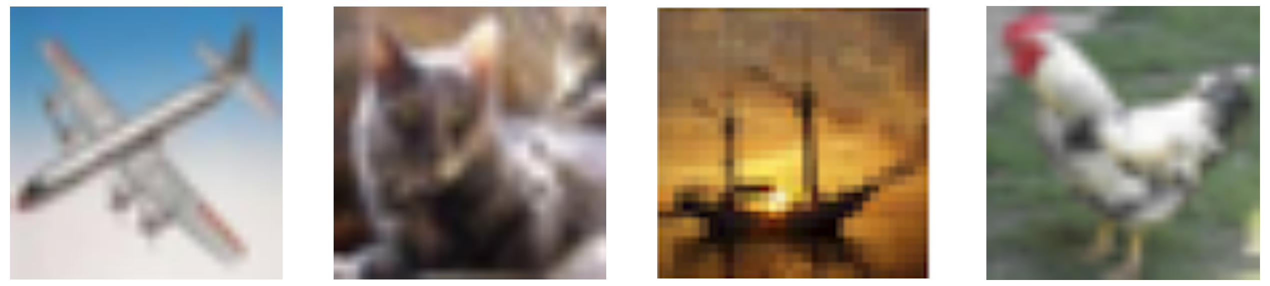
Recap: predictive data attribution

Framework:

Recap: predictive data attribution

Framework:

All data U, train subset S



Recap: predictive data attribution

Framework:

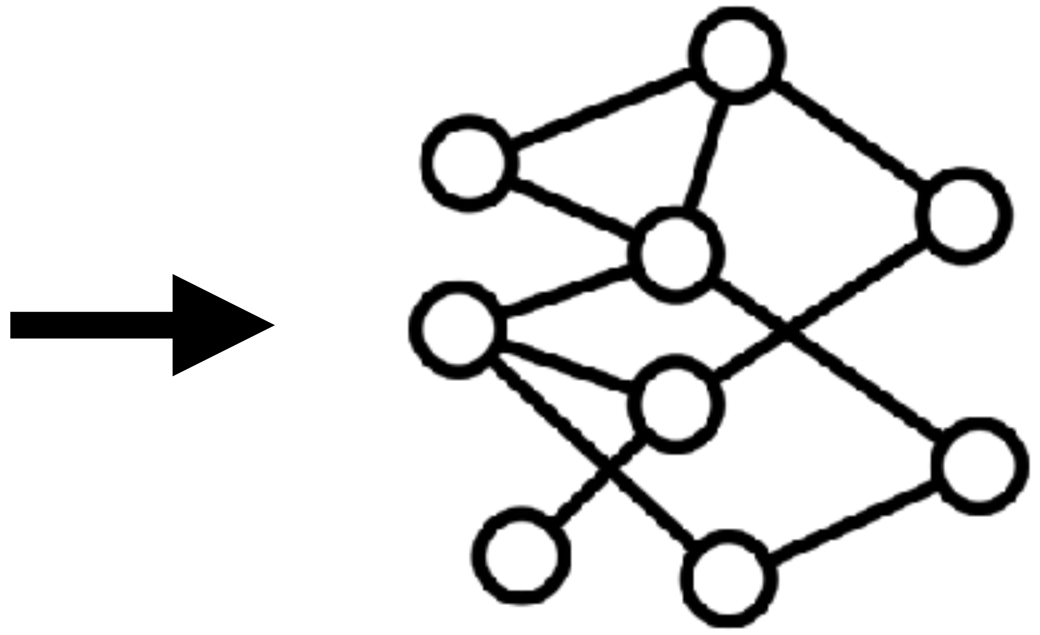
All data U, train subset S

Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$

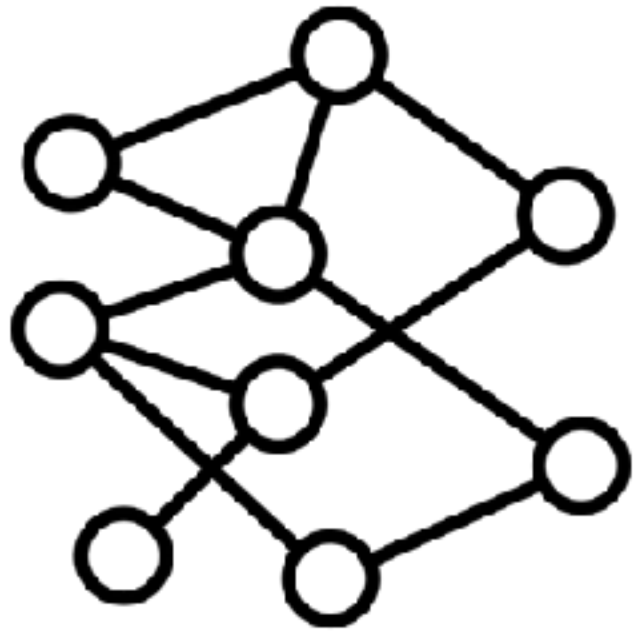


Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$



Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

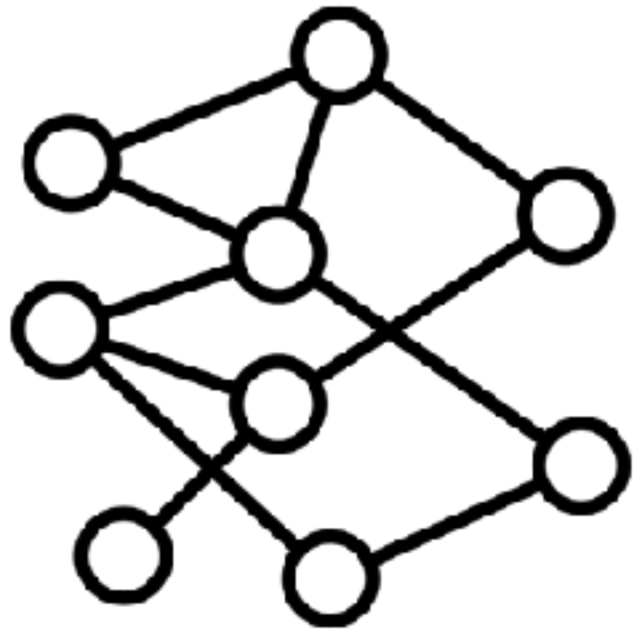
Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$

Output on sample z:



$$\ell\left(z, \theta(S)\right)$$

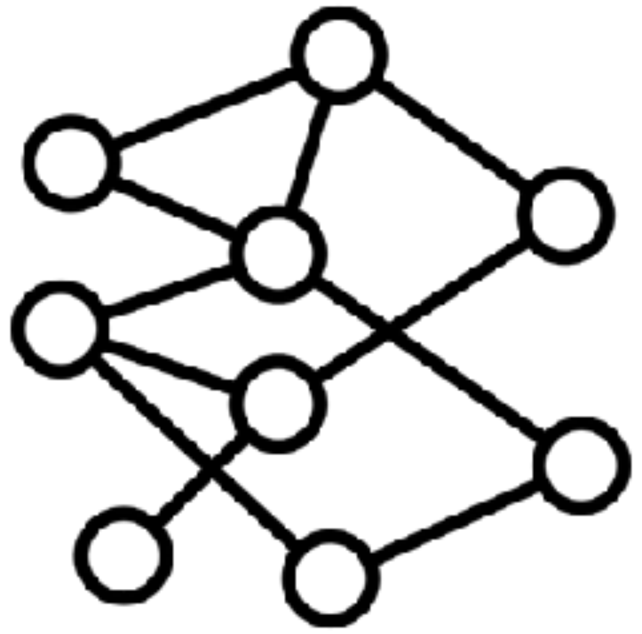
Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$

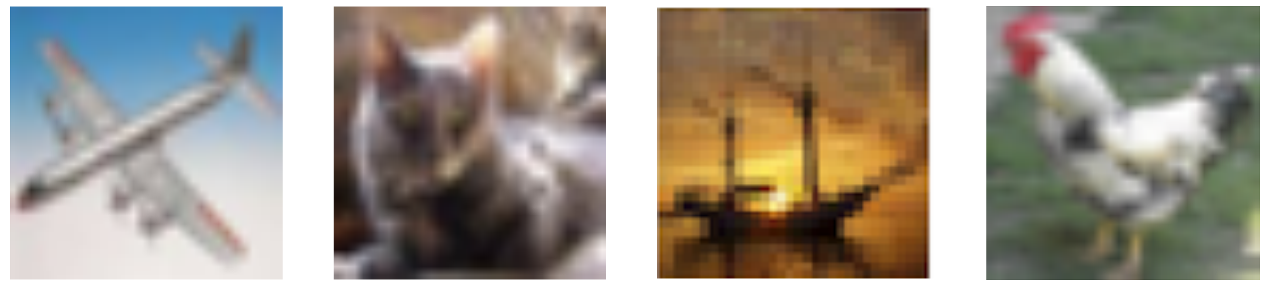


Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Data subset $S \subset U$

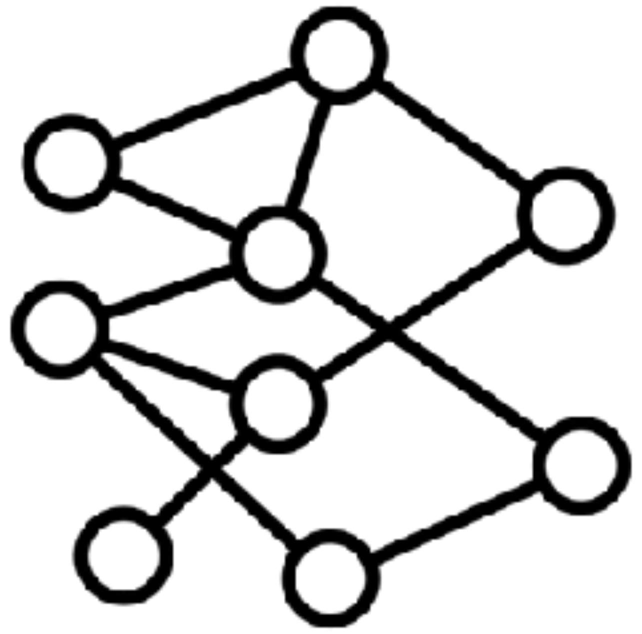


Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$

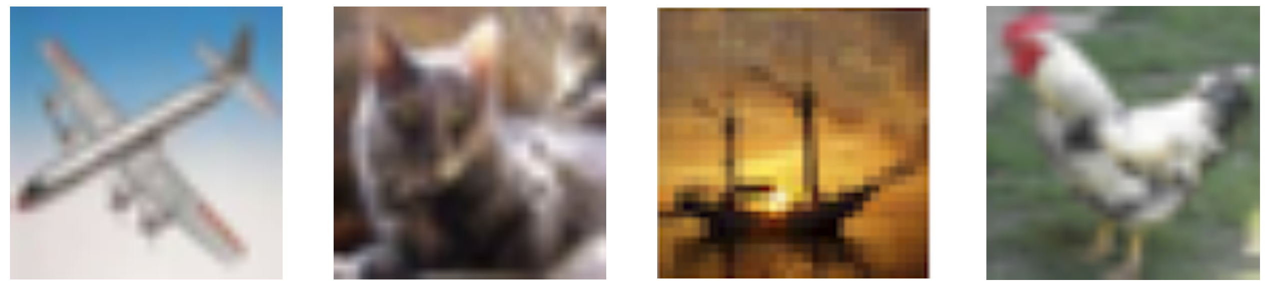


Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Data subset $S \subset U$



Output after training on S

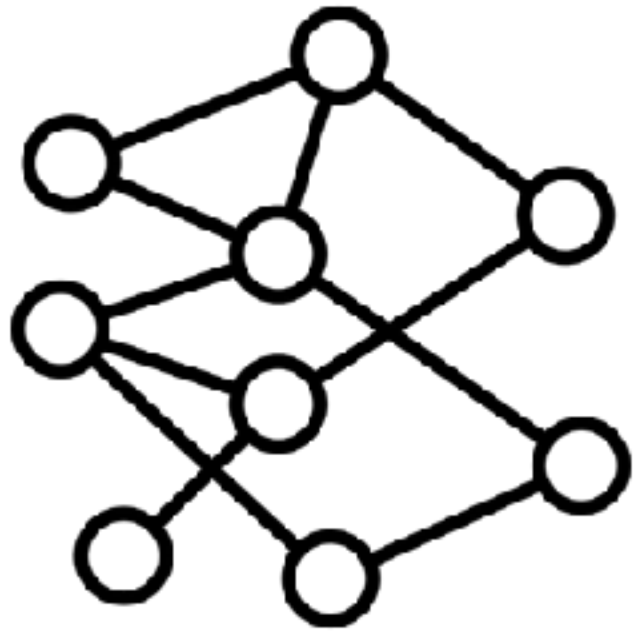
$$\ell\left(z, \theta(S)\right)$$

Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$

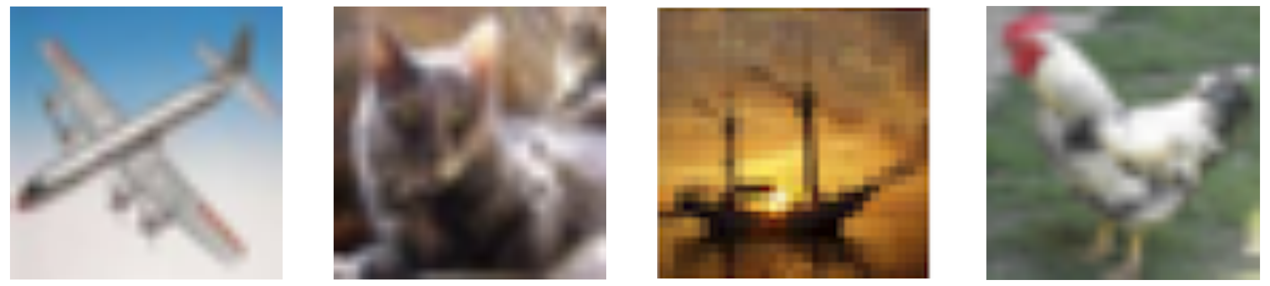


Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Data subset $S \subset U$



Output after training on S

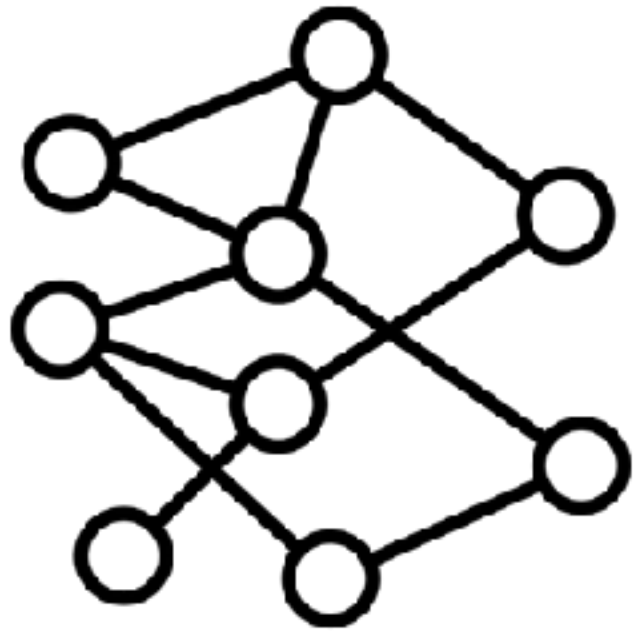
$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$



Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Data subset $S \subset U$

Output after training on S

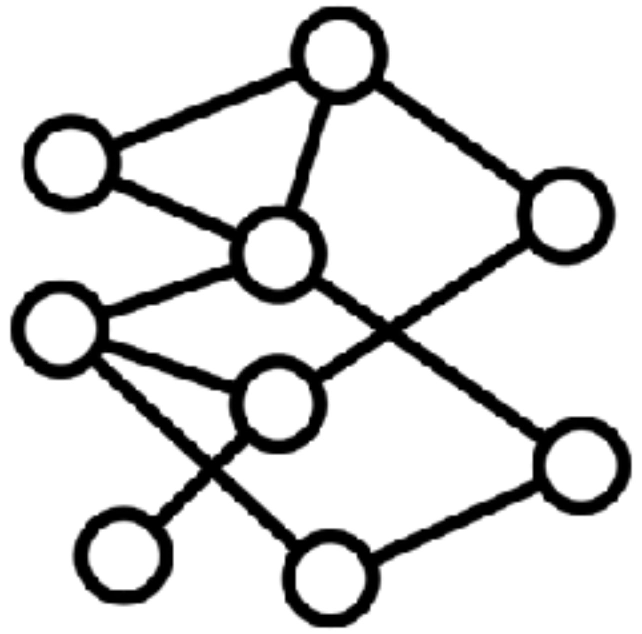
$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$

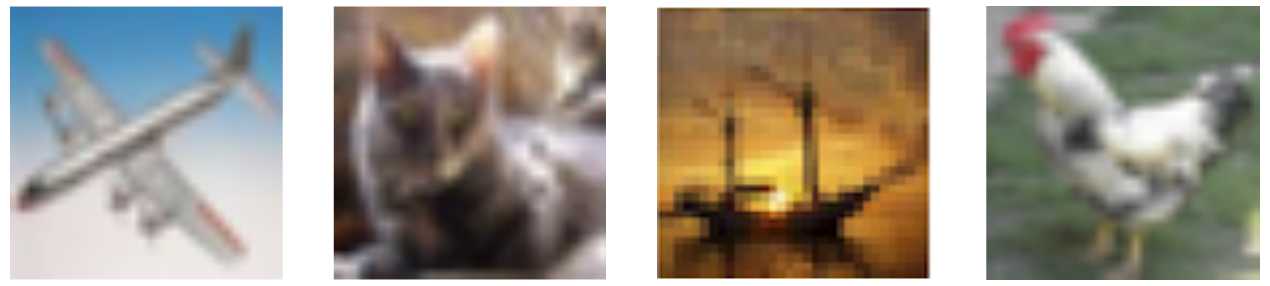


Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Data subset $S \subset U$



Output after training on S

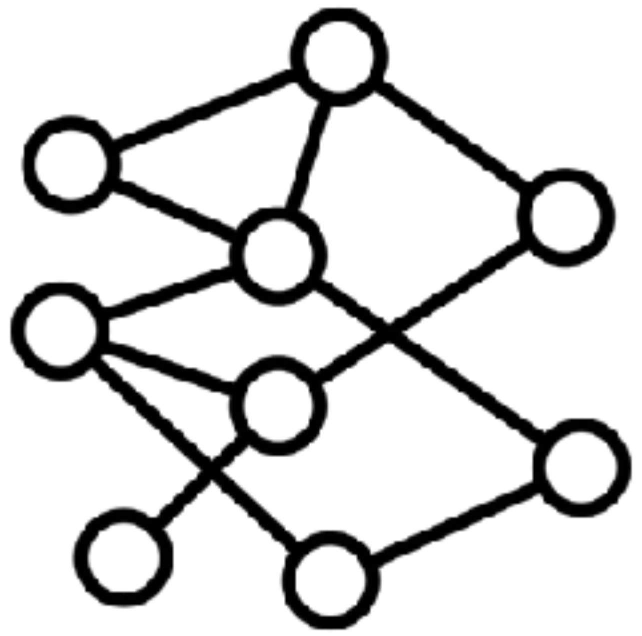
$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Recap: predictive data attribution

Framework:

All data U, train subset S

Model training $\theta(S)$



Output on sample z:

$$\ell\left(z, \theta(S)\right)$$

Predictive data attribution method $\hat{f}_z(S)$: estimates map from train subset S to model output

Data subset $S \subset U$

Output after training on S

$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Formula for applying (predictive) data attribution

Formula for applying (predictive) data attribution

Data subset $S \subset U$

→

Estimate output after training on S

$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Formula for applying (predictive) data attribution

Data subset $S \subset U$

Estimate output after training on S

$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Given a task:

Formula for applying (predictive) data attribution

Data subset $S \subset U$

Estimate output after training on S

$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Given a task:

1. **Rewrite** problem in terms of model outputs

Formula for applying (predictive) data attribution

Data subset $S \subset U$

→

Estimate output after training on S

$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Given a task:

1. **Rewrite** problem in terms of model outputs
2. **Plug-in** predictive data attribution estimate for model output (then solve)

Formula for applying (predictive) data attribution

Data subset $S \subset U$

Estimate output after training on S

$$\ell\left(z, \theta(S)\right) \approx \hat{f}_z(S)$$

Given a task:

1. **Rewrite** problem in terms of model outputs
2. **Plug-in** predictive data attribution estimate for model output (then solve)

Yields a surprisingly versatile framework

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Predictive data attribution

Dataset selection: motivation

Dataset selection: motivation

Dataset selection: choose the best possible training data to train on

Dataset selection: motivation

Dataset selection: choose the best possible training data to train on

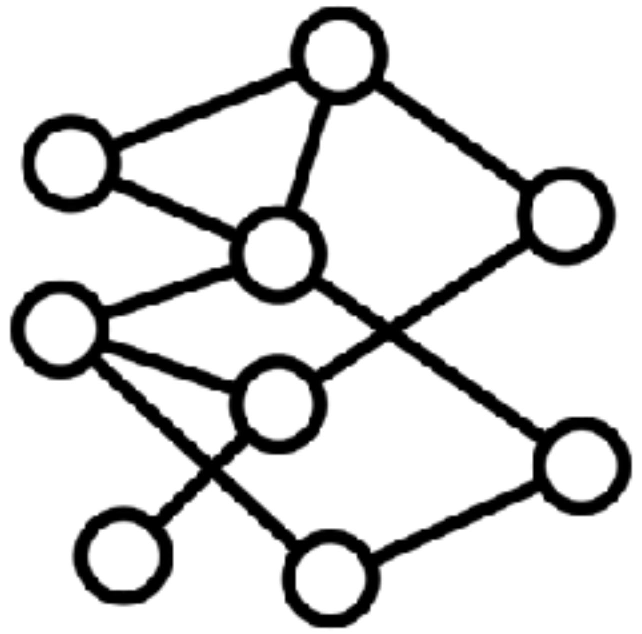
Scraped internet data

Dataset selection: motivation

Dataset selection: choose the best possible training data to train on

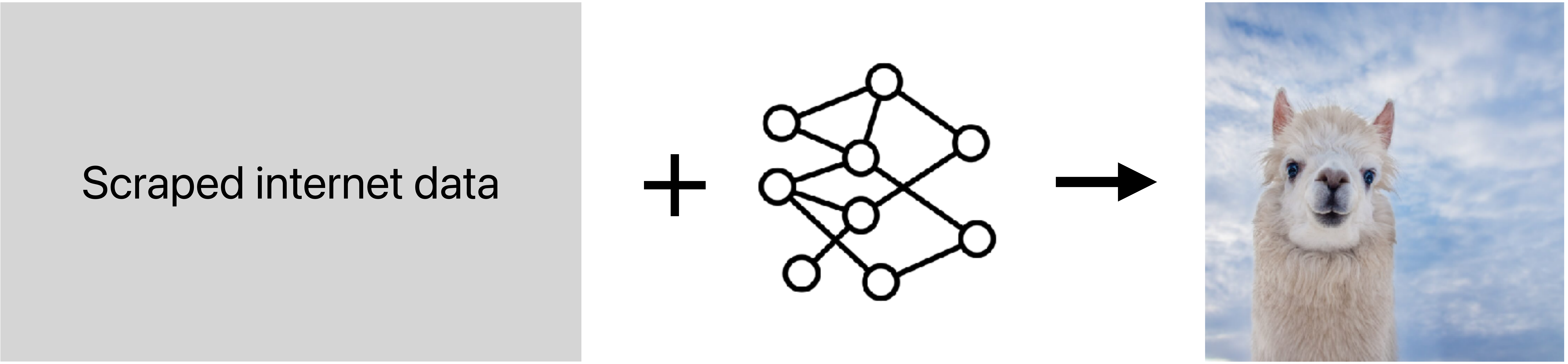
Scraped internet data

+



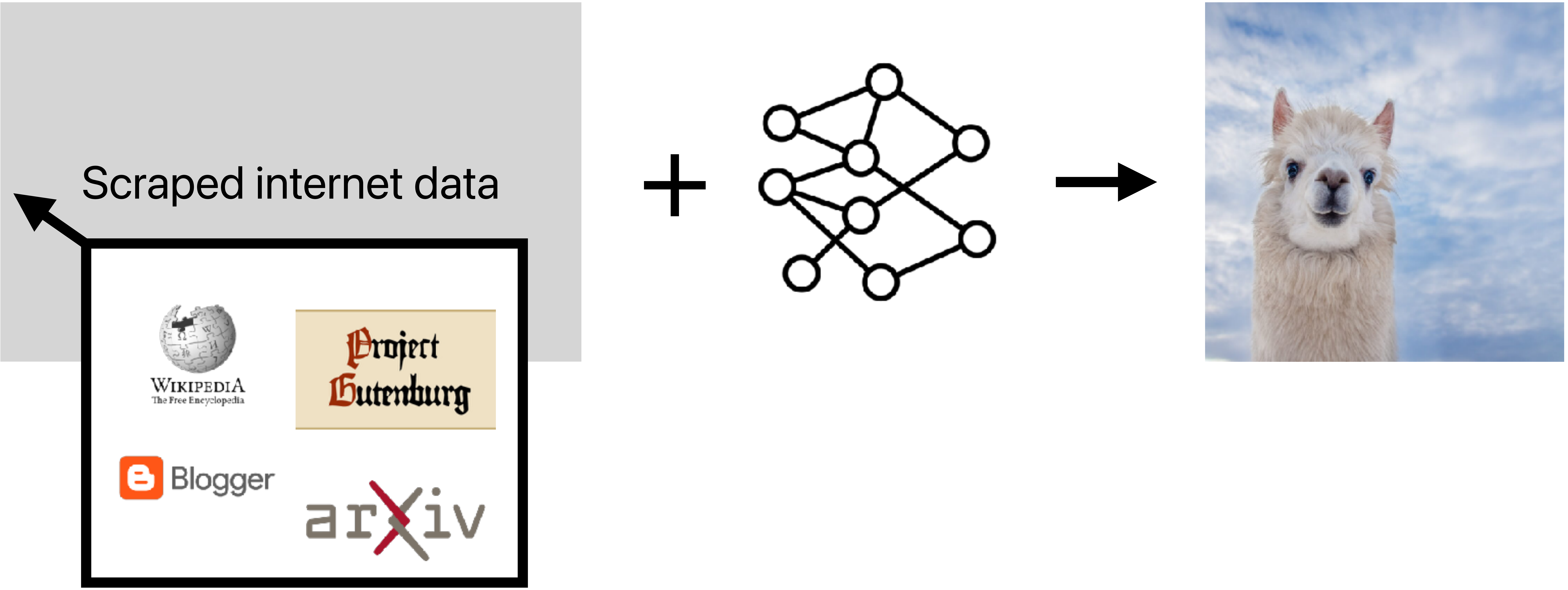
Dataset selection: motivation

Dataset selection: choose the best possible training data to train on



Dataset selection: motivation

Dataset selection: choose the best possible training data to train on

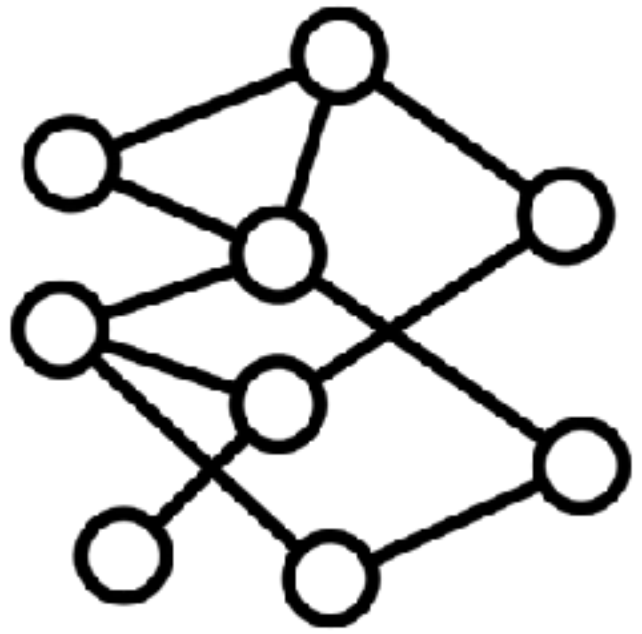


Dataset selection: motivation

Dataset selection: choose the best possible training data to train on

Scraped internet data

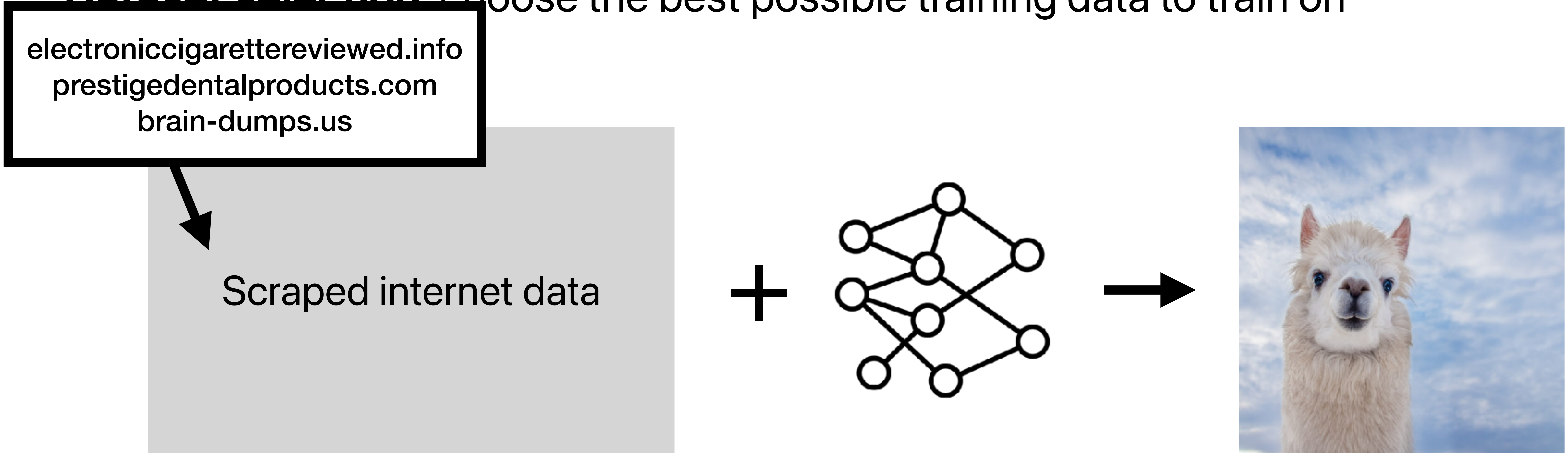
+



→

Dataset selection: motivation

Dataset selection: choose the best possible training data to train on



Dataset selectio

Dataset selection: choose th

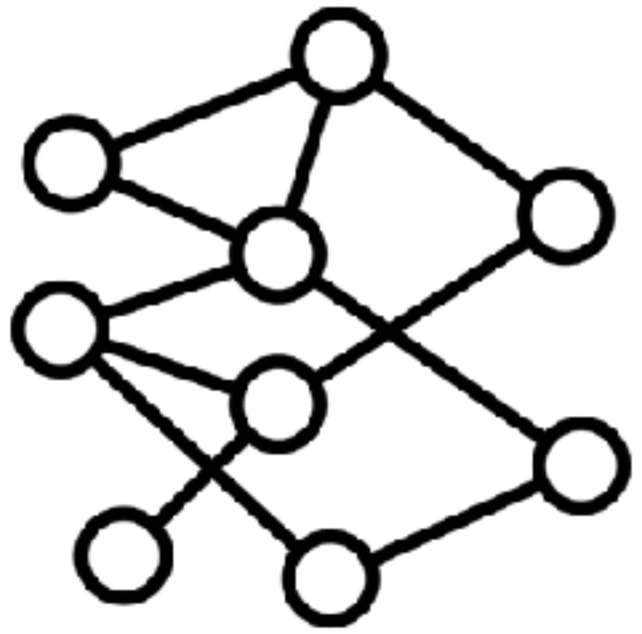
electroniccigarettereviewed.info
prestigedentalproducts.com
brain-dumps.us

http://ufdc.ufl.edu/AA00010883/00095

ac omplishmnl 'o the mi alon and welR I the men. U (SR 600-17b-I) are clear enough and cover everybody. iM-be UeIrU d at home first. IDretewr.. NatM WLWr Of Panama. B Aleol B -w s sin,ambas to V '). .. &. .... iZ'." . '- .- '."' ""5- ". ^ -*," ..^ -^:? ^~-. .... ; ,'- i ,? '. " ". 1 the eye to those Europe. like an orlontal: Mrt of the Ch-. te a ter ii wre n capw. to '. %e anti-trust sutt "I made let of money, but . mg C1 Y ay With ne shot of a Lot d Aieles mining and. petrlce Wymore is fac9 aur rtl 1 Dalton. Gr at gag tUdS 't come! Mand Jobaim AIMauccessful- M ." y uw? ie House ar&rkd, "Hold A mtllon yeas? A trillion? t Arrival." Or are they ageless?

Scraped internet data

+



→

Dataset selectio

Dataset selection: choose th

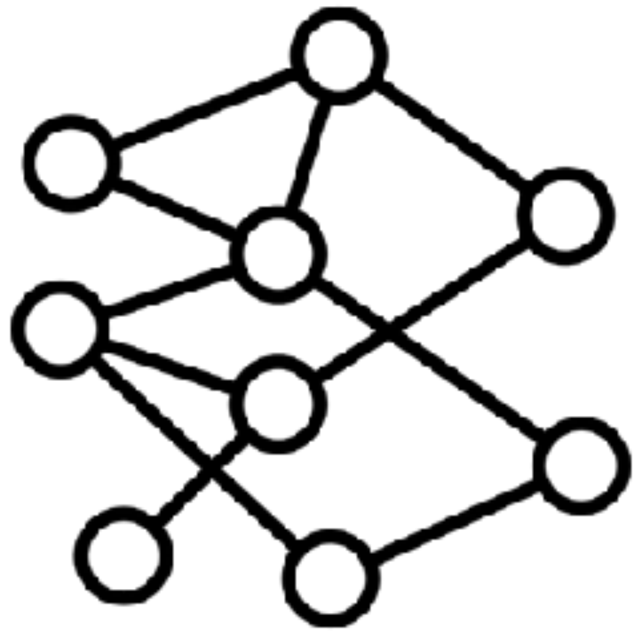
electroniccigarettereviewed.info
prestigedentalproducts.com
brain-dumps.us

http://ufdc.ufl.edu/AA00010883/00095

ac omplishmnl 'o the mi alon and welR I the men. U (SR 600-17b-I) are clear enough and cover everybody. iM-be UeIrU d at home first. IDretewr.. NatM WLWr Of Panama. B Aleol B -w s sin,ambas to V '). .. &. .... iZ'." . '- .- '."' ""5- ". ^ -*," ..^ -^:? ^~-. .... ; ,'- i ,? '. " ". 1 the eye to those Europe. like an orlontal: Mrt of the Ch-. te a ter ii wre n capw. to '. %e anti-trust sutt "I made let of money, but . mg C1 Y ay With ne shot of a Lot d Aieles mining and. petrlce Wymore is fac9 aur rtl 1 Dalton. Gr at gag tUdS 't come! Mand Jobaim AIMauccessful- M ." y uw? ie House ar&rkd, "Hold A mtllon yeas? A trillion? t Arrival." Or are they ageless?

Scraped internet data

\+



→

Problem: Much of the internet is "low quality" – what data should we train on?

Dataset selection with predictive data attribution

Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Select train set from data pool U

Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Select train set from data pool U

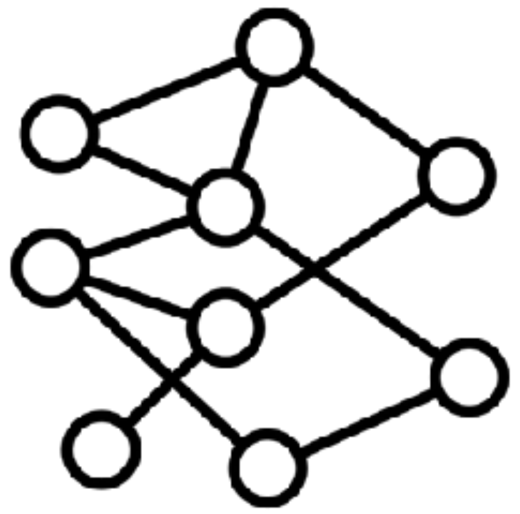
Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Select train set from data pool U

Train model $\theta(S)$



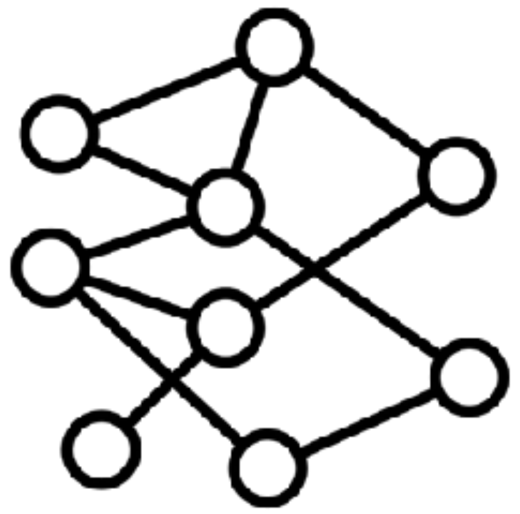
Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Select train set from data pool U

Train model $\theta(S)$



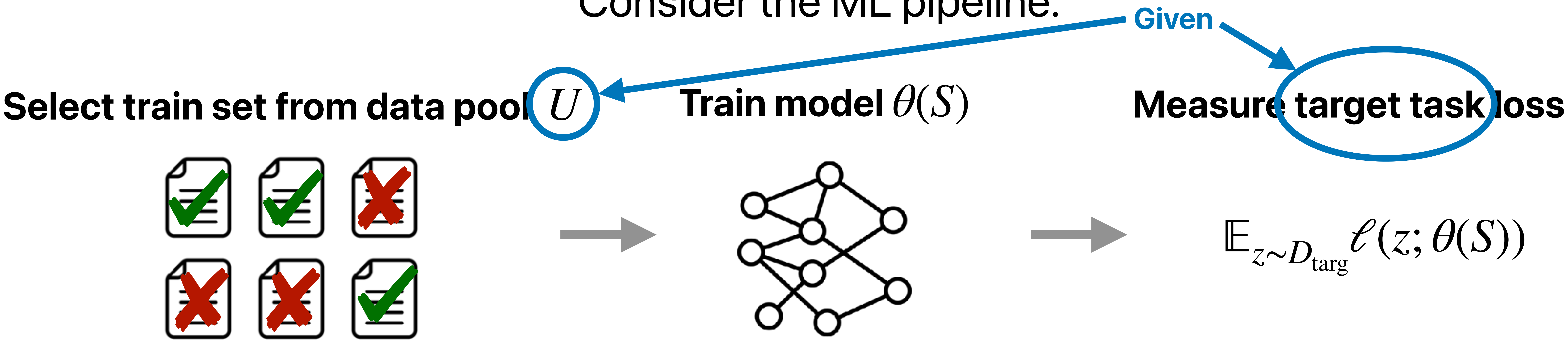
Measure target task loss

$$\mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:



Dataset selection with predictive data attribution

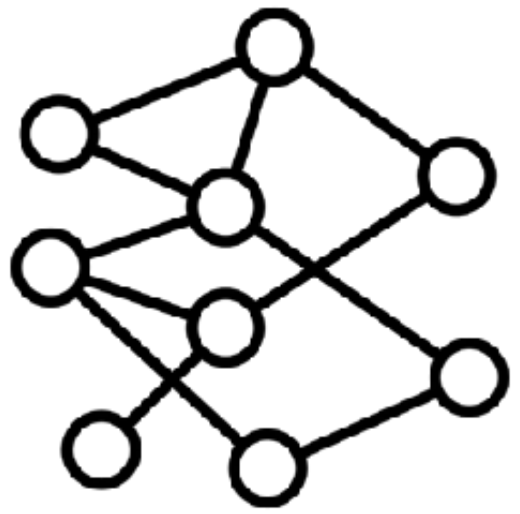
Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Given

Select train set from data pool U

Train model $\theta(S)$



Measure target task loss

$$\mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Dataset selection task: choose train set S that minimizes target task loss

Dataset selection with predictive data attribution

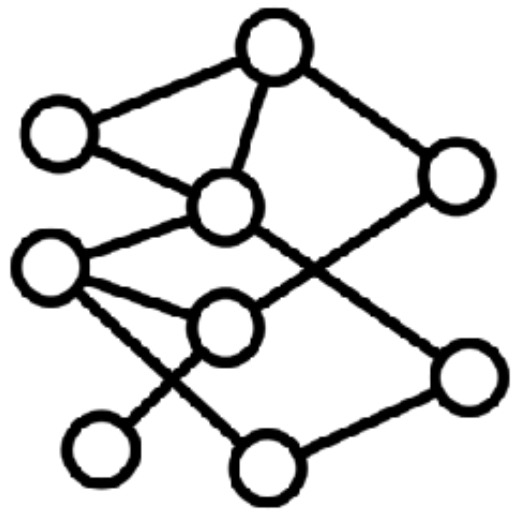
Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Given

Select train set from data pool U

Train model $\theta(S)$



Measure target task loss

$$\mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Dataset selection task: choose train set S that minimizes target task loss

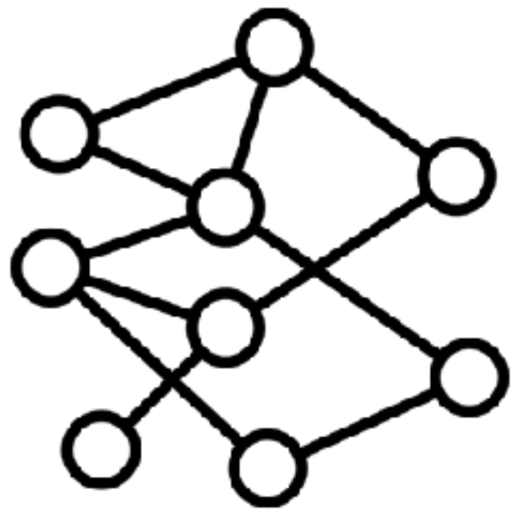
Dataset selection with predictive data attribution

Step 1: Rewrite in terms of model outputs

Consider the ML pipeline:

Given

Select train set from data pool U | **Train model $\theta(S)$** | **Measure target task loss**



$$\mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Dataset selection task: choose train set S that minimizes target task loss

$$\min_{S \subset X} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Dataset selection with predictive data attribution

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Solution sketch with data attribution:

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Solution sketch with data attribution:

1. Plug-in data attribution to estimate loss

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Solution sketch with data attribution:

1. Plug-in data attribution to estimate loss

$$\hat{f}_z(S) \approx \ell(z; \theta(S))$$

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Solution sketch with data attribution:

1. Plug-in data attribution to estimate loss

$$\hat{f}_z(S) \approx \ell(z; \theta(S))$$

2. Find train subset that minimizes the data attribution loss estimate:

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Solution sketch with data attribution:

1. Plug-in data attribution to estimate loss

$$\hat{f}_z(S) \approx \ell(z; \theta(S))$$

2. Find train subset that minimizes the data attribution loss estimate:

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{targ}} \left[\hat{f}_z(S) \right]$$

Dataset selection with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Dataset selection task: choose train set that minimizes target task loss

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{\text{targ}}} \ell(z; \theta(S))$$

Requires training a model on S

Solution sketch with data attribution:

1. Plug-in data attribution to estimate loss

$$\hat{f}_z(S) \approx \ell(z; \theta(S))$$

2. Find train subset that minimizes the data attribution loss estimate:

$$\min_{S \subset U} \mathbb{E}_{z \sim D_{targ}} \left[\hat{f}_z(S) \right]$$

Easy to optimize/evaluate

Selecting data with data attribution

Selecting data with data attribution

References: [Schoch et al. 2023; Wang et al. 2023; Wang et al. 2024; Engstrom et al. 2024; Xia et al. 2024; Jiao et al. 2024; Chhabra et al. 2024; Jain et al. 2024]

Selecting data with data attribution

References: [Schoch et al. 2023; Wang et al. 2023; Wang et al. 2024; Engstrom et al. 2024; Xia et al. 2024; Jiao et al. 2024; Chhabra et al. 2024; Jain et al. 2024]

Big conceptual question: operationalizing dataset selection

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Predictive data attribution

Data poisoning: motivation

Data poisoning: motivation

Consider the standard ML pipeline:

Data poisoning: motivation

Consider the standard ML pipeline:

Collect data

Training data

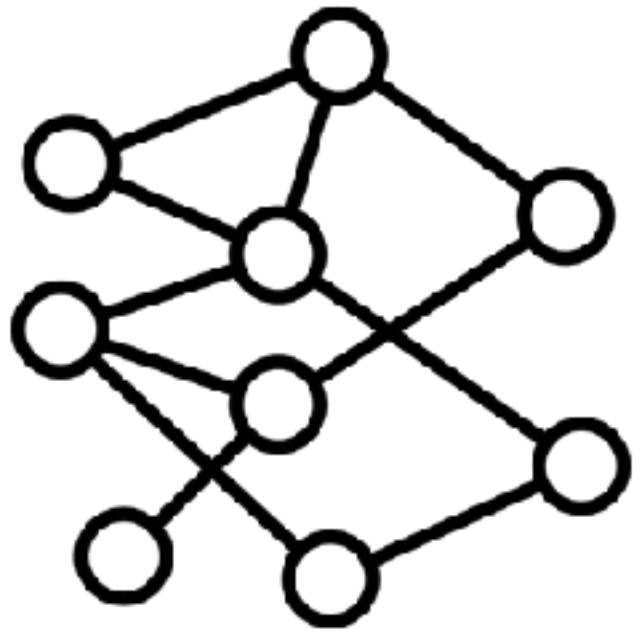
Data poisoning: motivation

Consider the standard ML pipeline:

Collect data **Train model**

Training data

+



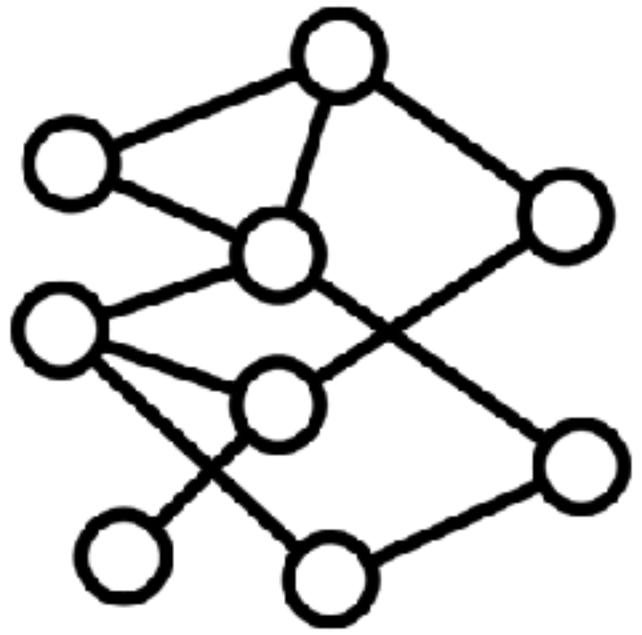
Data poisoning: motivation

Consider the standard ML pipeline:

Collect data | **Train model** | **Predict**

Training data

+

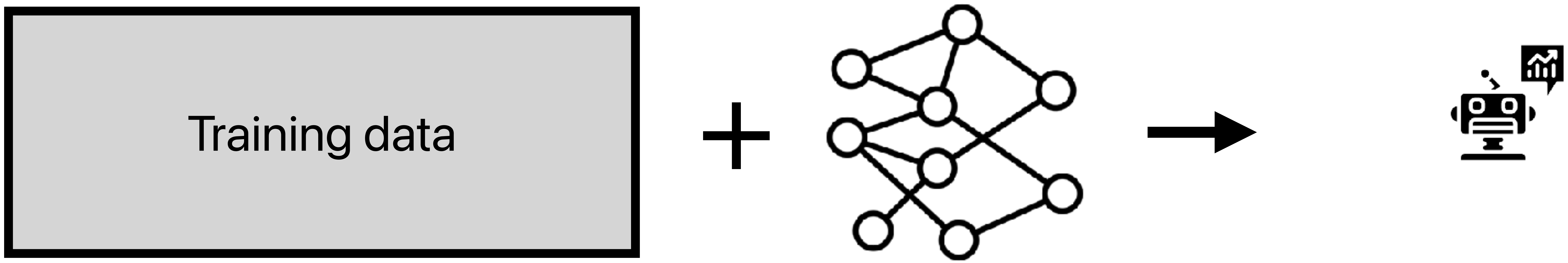


→

Data poisoning: motivation

Consider the standard ML pipeline:

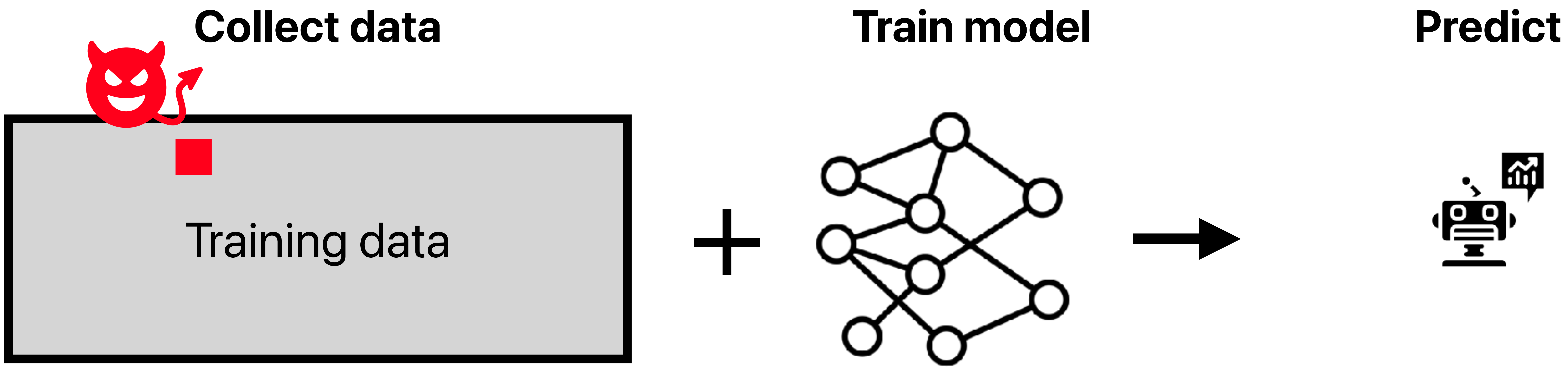
Collect data **Train model** **Predict**



Data poisoning: an adversary **changes** some training data to *hurt* model behavior

Data poisoning: motivation

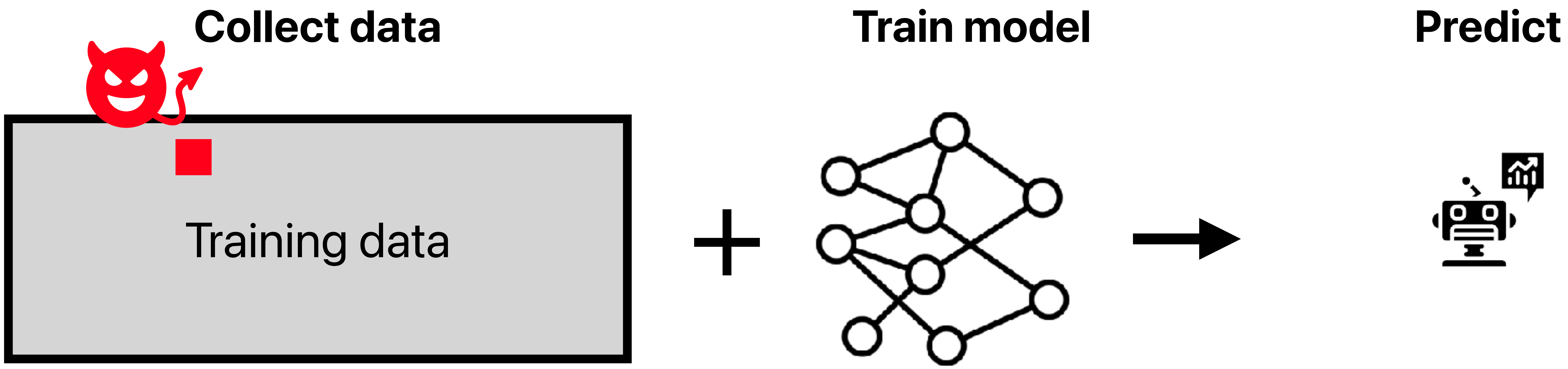
Consider the standard ML pipeline:



Data poisoning: an adversary **changes** some training data to *hurt* model behavior

Data poisoning: motivation

Consider the standard ML pipeline:

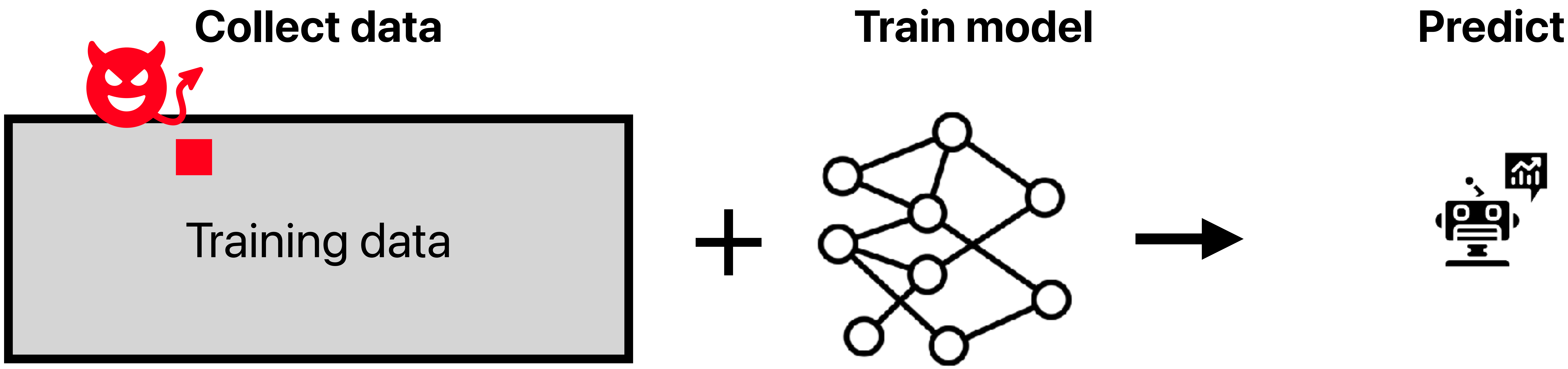


Data poisoning: an adversary **changes** some training data to *hurt* model behavior

Relevant to any task with third party (e.g., web-scraped/crowdsourced) data

Data poisoning: motivation

Consider the standard ML pipeline:



Data poisoning: an adversary **changes** some training data to *hurt* model behavior

Relevant to any task with third party (e.g., web-scraped/crowdsourced) data

Example: political candidate uploads internet text that makes LM disfavor a rival

Data poisoning with predictive data attribution

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

Training data

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

Training data

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

Training data

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

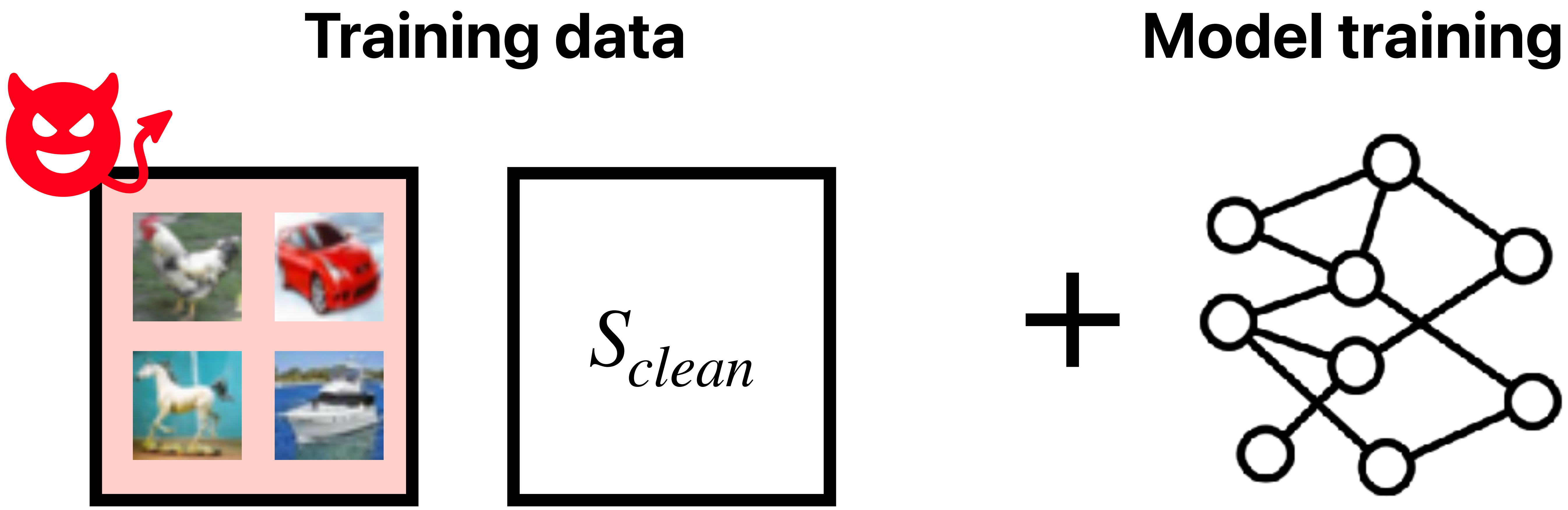
Training data

$S_{adv} \cup S_{clean}$

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data



$S_{adv} \cup S_{clean}$

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

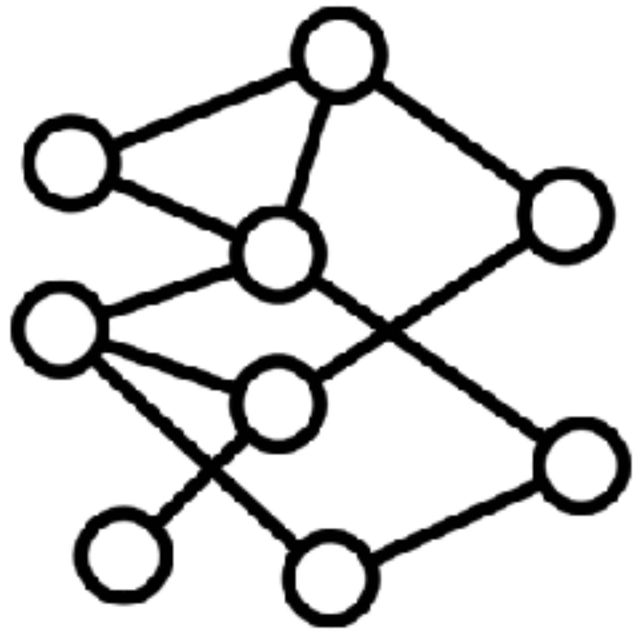
Poisoning setup: model trains on both adversarial and clean data

Training data

$S_{adv} \cup S_{clean}$

Model training

$+$

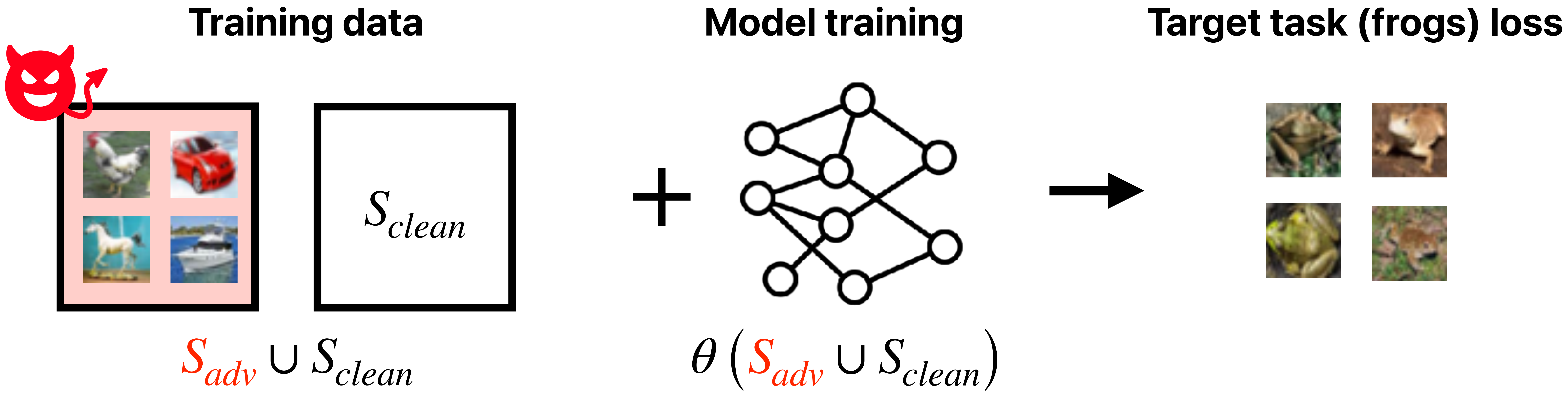


$\theta\left(S_{adv} \cup S_{clean}\right)$

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

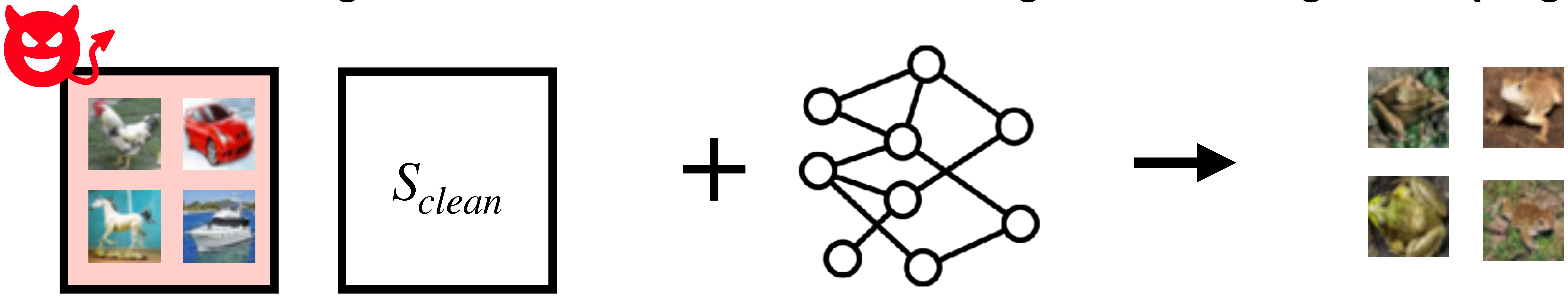


Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

Training data | **Model training** | **Target task (frogs) loss**



$S_{adv} \cup S_{clean}$

$\theta\left(S_{adv} \cup S_{clean}\right)$

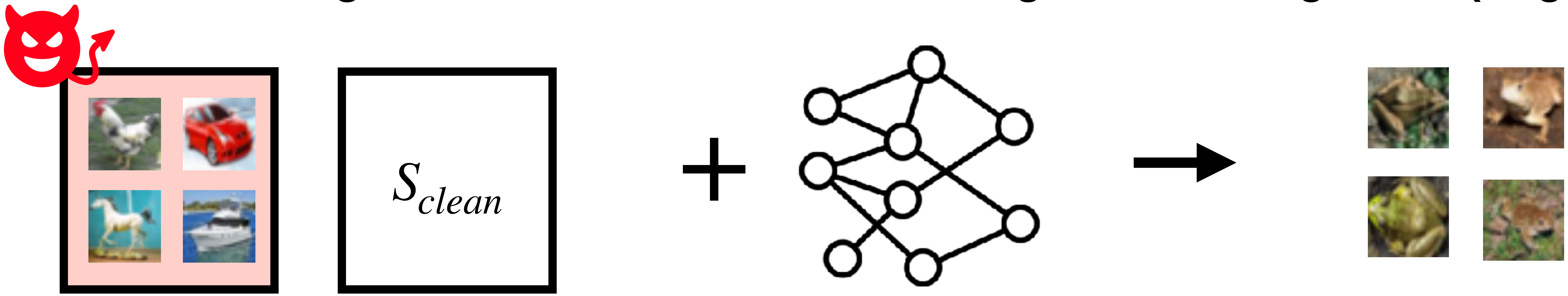
$\mathbb{E}_{z \sim D_{targ}}\left[\ell\left(z; \theta(S_{adv} \cup S_{clean}\right)\right]$

Data poisoning with predictive data attribution

Step 1: Rewrite in terms of model outputs

Poisoning setup: model trains on both adversarial and clean data

Training data | **Model training** | **Target task (frogs) loss**



$S_{adv} \cup S_{clean}$ + $\theta\left(S_{adv} \cup S_{clean}\right)$ → $\mathbb{E}_{z \sim D_{targ}}\left[\ell\left(z; \theta(S_{adv} \cup S_{clean})\right)\right]$

Adversary designs training data S_{adv} to increase loss on the (specified) target task

Optimization problem: $\max_{S_{adv}} \mathbb{E}_{z \sim D_{targ}}\left[\ell(z; \theta(S_{adv} \cup S_{clean}))\right]$

Data poisoning with predictive data attribution

Data poisoning with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Data poisoning with predictive data attribution

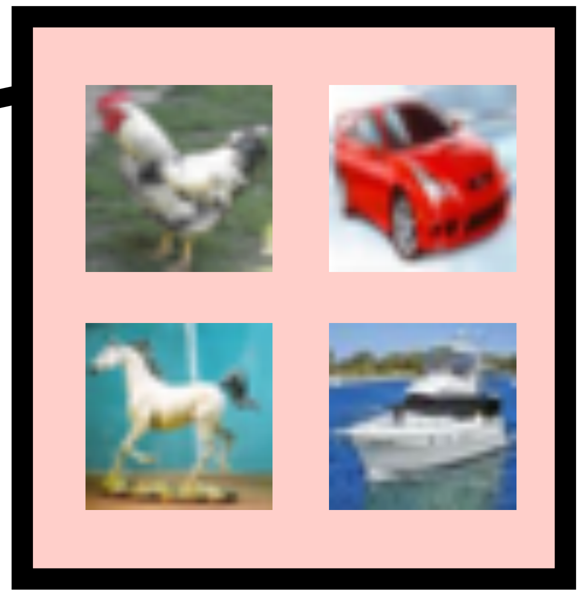
Step 2: Plug in data attribution estimate for model output

Data poisoning problem: $\max_{S_{adv}} \mathbb{E}_{z \sim D_{targ}} \left[\ell(z; \theta(S_{adv} \cup S_{clean}) \right]$

Data poisoning with predictive data attribution

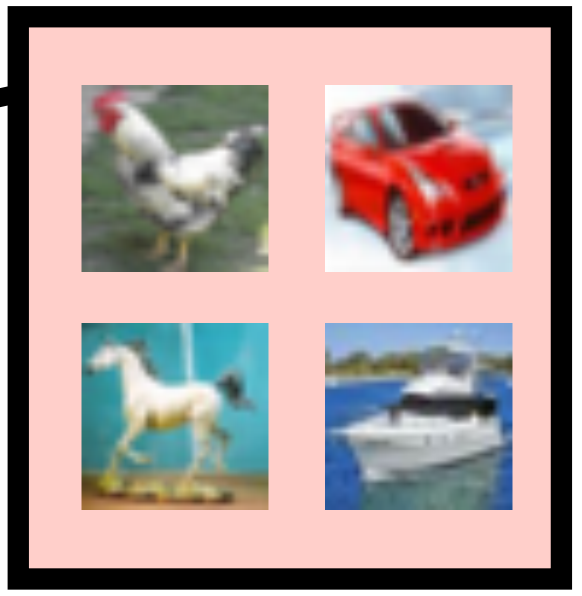
Step 2: Plug in data attribution estimate for model output

Data poisoning problem: $\max_{S_{adv}} \mathbb{E}_{z \sim D_{targ}} \left[\ell(z; \theta(S_{adv} \cup S_{clean})) \right]$

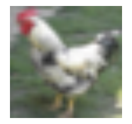


Data poisoning with predictive data attribution

Step 2: Plug in data attribution estimate for model output

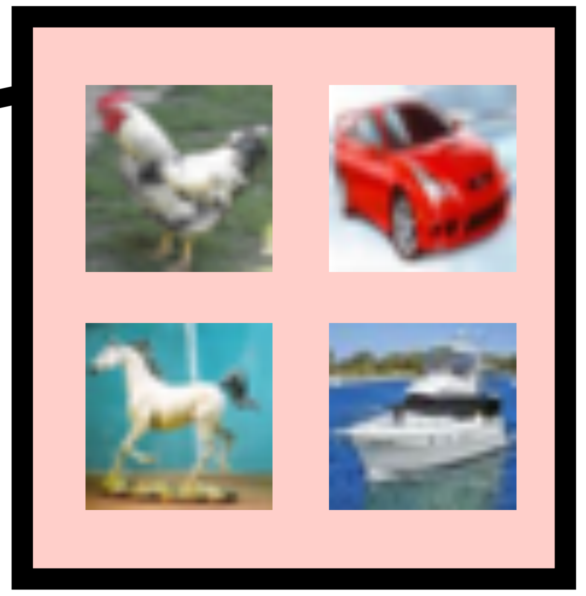


Data poisoning problem: $\max_{S_{adv}} \mathbb{E}_{z \sim D_{targ}} \left[\ell(z; \theta(S_{adv} \cup S_{clean})) \right]$

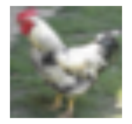
Hard to optimize *through* model training: how do we design poisonous inputs (like  ?)

Data poisoning with predictive data attribution

Step 2: Plug in data attribution estimate for model output



Data poisoning problem: $\max_{S_{adv}} \mathbb{E}_{z \sim D_{targ}} \left[\ell(z; \theta(S_{adv} \cup S_{clean}) \right]$

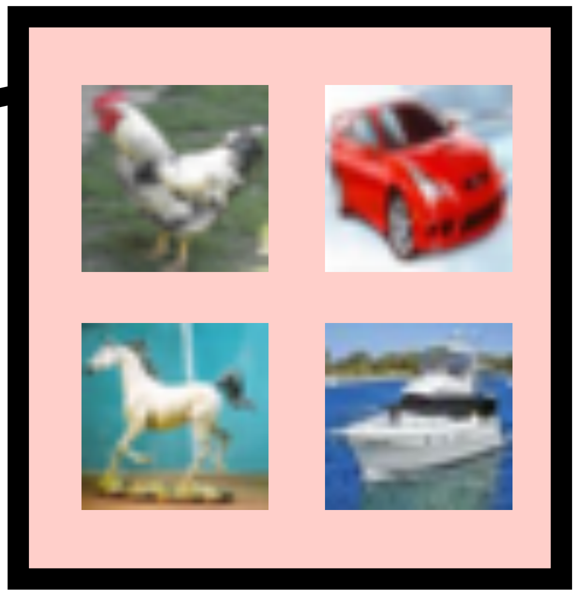
Hard to optimize *through* model training: how do we design poisonous inputs (like  ?)

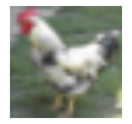
Data attribution approach:

Data poisoning with predictive data attribution

Step 2: Plug in data attribution estimate for model output

Data poisoning problem: $\max\limits_{\color{red}{S_{adv}}} \mathbb{E}_{z\sim D_{targ}}\left[\ell(z;\theta(\color{red}{S_{adv}} \color{black}{\cup S_{clean}}))\right]$



Hard to optimize *through* model training: how do we design poisonous inputs (like  ?)

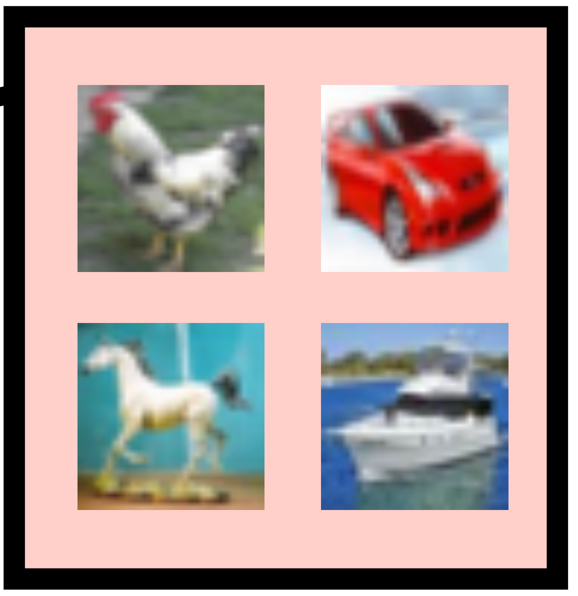
Data attribution approach:

1. Estimate sample losses as a function of $\color{red}{S_{adv}}$ via data attribution:

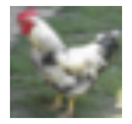
$$\hat{f}_z\left(S_{clean} \cup \color{red}{S_{adv}}\right)\left(\approx \ell(z, \color{red}{S_{adv}} \color{black}{\cup S_{clean}})\right)$$

Data poisoning with predictive data attribution

Step 2: Plug in data attribution estimate for model output



Data poisoning problem: $\max_{S_{adv}} \mathbb{E}_{z \sim D_{targ}} \left[\ell(z; \theta(S_{adv} \cup S_{clean}) \right]$

Hard to optimize *through* model training: how do we design poisonous inputs (like  ?)

Data attribution approach:

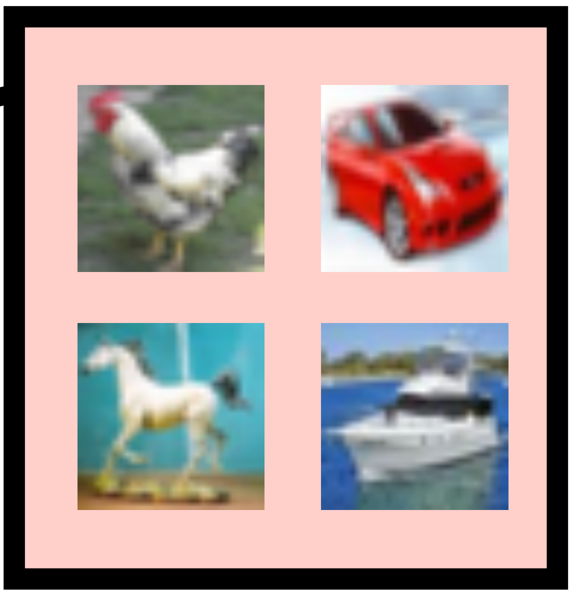
1. Estimate sample losses as a function of S_{adv} via data attribution:

$$\hat{f}_z \left(S_{clean} \cup S_{adv} \right) \left(\approx \ell(z, S_{adv} \cup S_{clean}) \right)$$

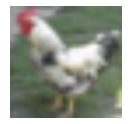
2. Maximize estimate with respect to S_{adv} through data attribution

Data poisoning with predictive data attribution

Step 2: Plug in data attribution estimate for model output



Data poisoning problem: $\max_{\color{red}{S_{adv}}} \mathbb{E}_{z \sim D_{targ}} \left[\ell(z; \theta({\color{red}{S_{adv}}} \cup S_{clean})) \right]$

Hard to optimize *through* model training: how do we design poisonous inputs (like  ?)

Data attribution approach:

1. Estimate sample losses as a function of ${\color{red}{S_{adv}}}$ via data attribution:

$$\hat{f}_z \left(S_{clean} \cup {\color{red}{S_{adv}}} \right) \left(\approx \ell(z, {\color{red}{S_{adv}}} \cup S_{clean}) \right)$$

2. Maximize estimate with respect to ${\color{red}{S_{adv}}}$ through data attribution

New objective: $\max_{\color{red}{S_{adv}}} \mathbb{E}_{z \sim D_{targ}} \left[\hat{f}_z({\color{red}{S_{adv}}} \cup S_{clean}) \right]$

Data poisoning with data attribution

Data poisoning with data attribution

Approaches: [Biggio et al. 2013; Koh Liang 17; Xiao et al. 2018; Fang et al. 2020; Koh et al. 2021; Wu et al. 2023]

Data poisoning with data attribution

Approaches: [Biggio et al. 2013; Koh Liang 17; Xiao et al. 2018; Fang et al. 2020; Koh et al. 2021; Wu et al. 2023]

Conceptual questions: scaling to large-scale learning problems, threat models

Data poisoning with data attribution

Approaches: [Biggio et al. 2013; Koh Liang 17; Xiao et al. 2018; Fang et al. 2020; Koh et al. 2021; Wu et al. 2023]

Conceptual questions: scaling to large-scale learning problems, threat models

Poison

Data poisoning with data attribution

Approaches: [Biggio et al. 2013; Koh Liang 17; Xiao et al. 2018; Fang et al. 2020; Koh et al. 2021; Wu et al. 2023]

Conceptual questions: scaling to large-scale learning problems, threat models

→

Poison

Target

Data poisoning with data attribution

Approaches: [Biggio et al. 2013; Koh Liang 17; Xiao et al. 2018; Fang et al. 2020; Koh et al. 2021; Wu et al. 2023]

Conceptual questions: scaling to large-scale learning problems, threat models

Poison

→

Target

[Koh Liang 17]

Four applications

Model debugging

Understanding model behavior

Dataset selection

Choosing the *best* training data

Data poisoning

Constructing the *worst* training data

Machine unlearning

Forgetting previously learned data

Predictive data attribution

Machine unlearning: motivation

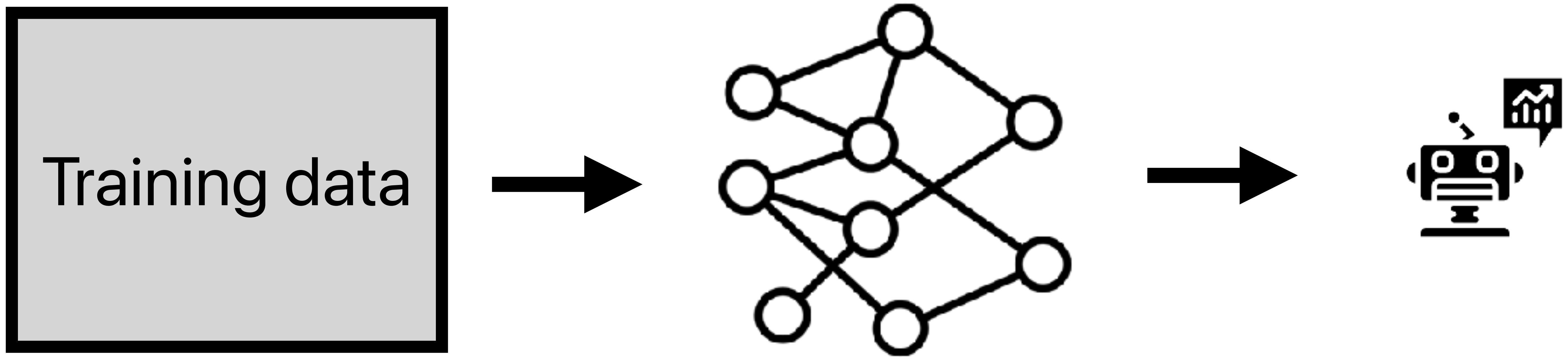
Machine unlearning: motivation

ML pipeline:

Machine unlearning: motivation

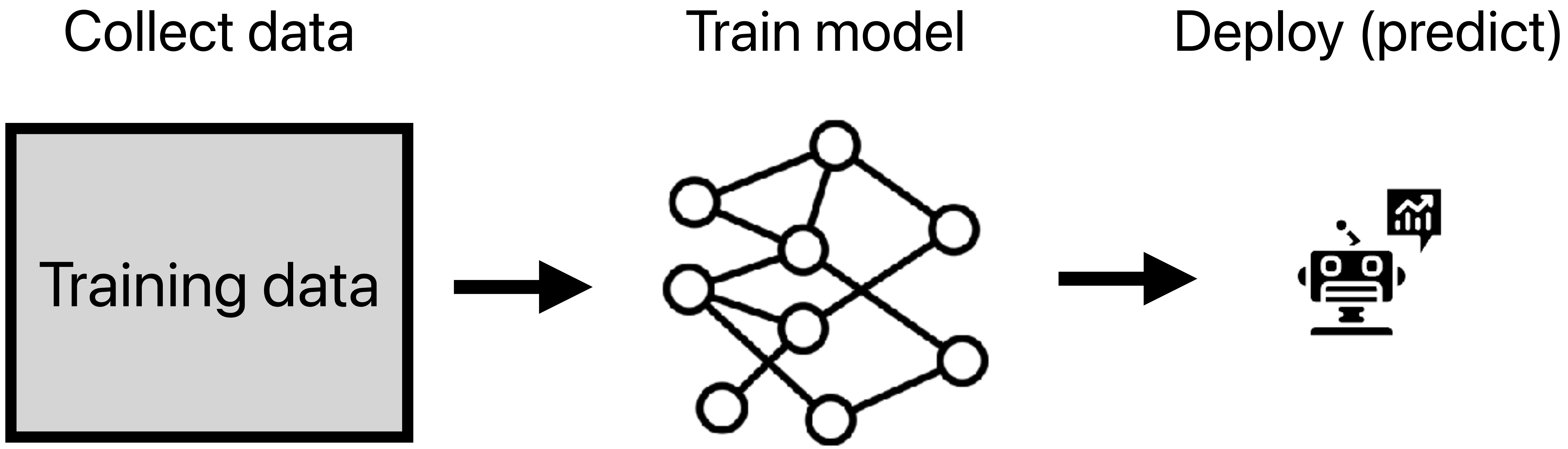
ML pipeline:

Collect data | Train model | Deploy (predict)



Machine unlearning: motivation

ML pipeline:



What if customer(s) want to "delete" data? Model retraining is expensive!

Machine unlearning: motivation

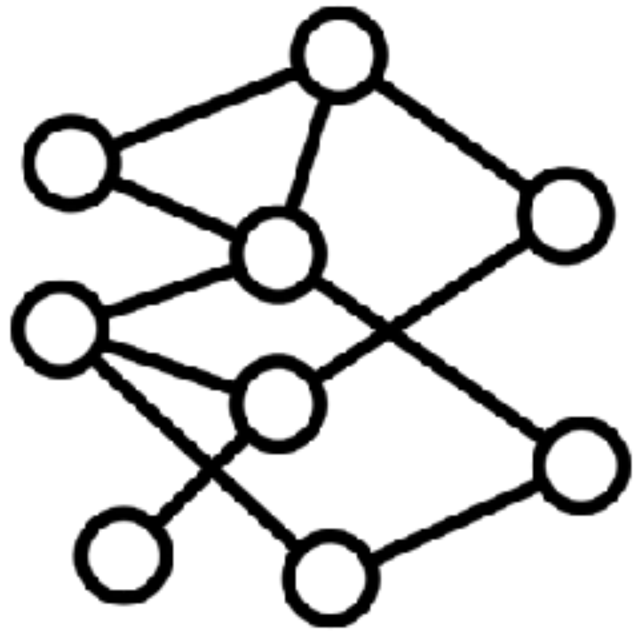
ML pipeline:

Grandma's secret cookie recipe

Collect data

Training data

Train model



Deploy (predict)

What if customer(s) want to "delete" data? Model retraining is expensive!

Machine unlearning: motivation

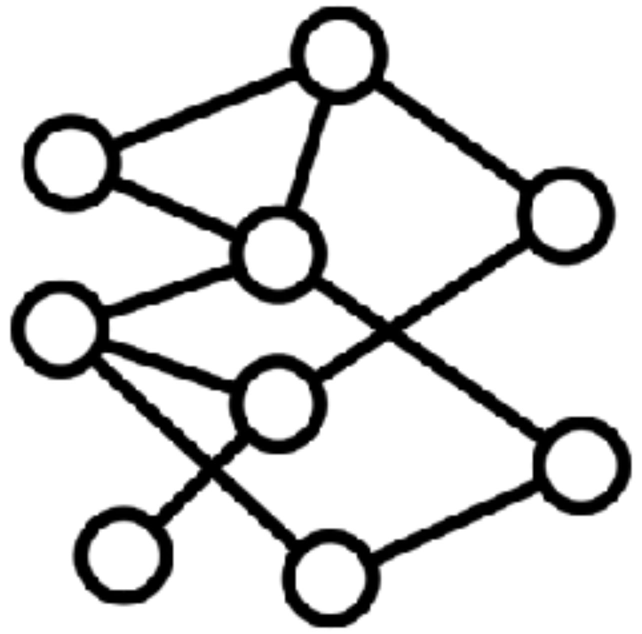
ML pipeline:

Grandma's secret cookie recipe

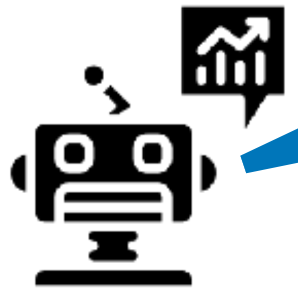
Collect data

Training data

Train model



Deploy (predict)



"Sure, here's a great cookie recipe..."

What if customer(s) want to "delete" data? Model retraining is expensive!

Machine unlearning: motivation

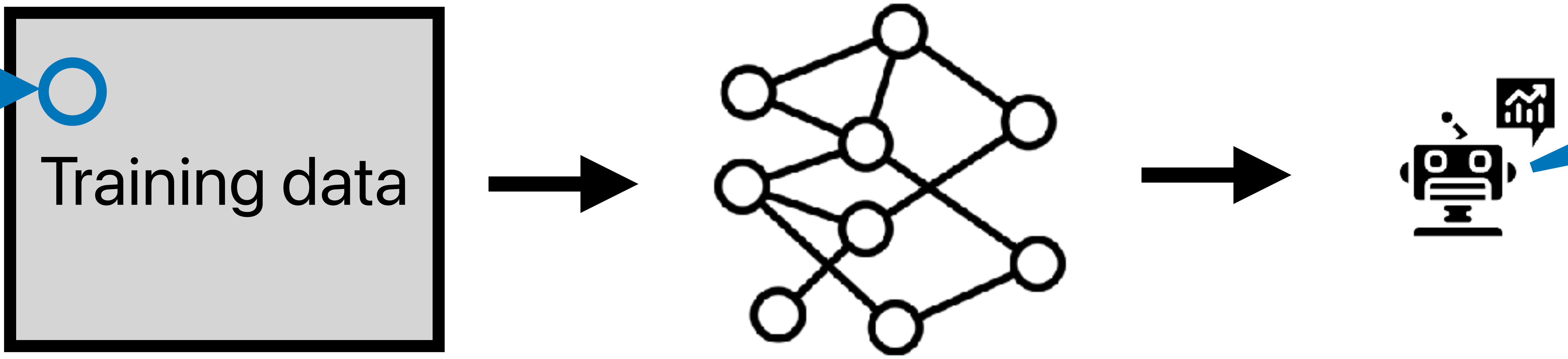
ML pipeline:

Grandma's secret cookie recipe

Collect data

Train model

Deploy (predict)



"Sure, here's a great cookie recipe..."

What if customer(s) want to "delete" data? Model retraining is expensive!

Machine unlearning goal: given "forget set" and a trained model, modify predictions to behave as if the model had never trained on the set.

Machine unlearning with data attribution

Machine unlearning with data attribution

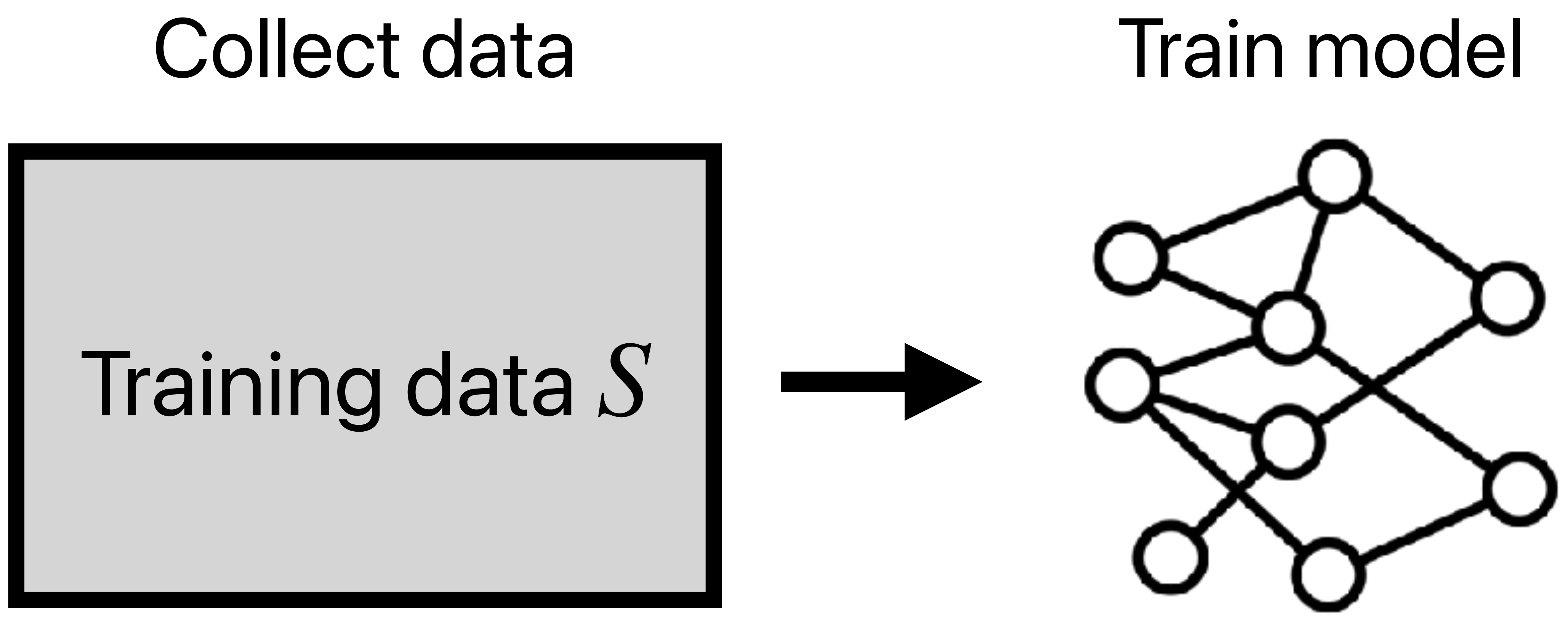
Step 1: Rewrite in terms of model outputs

Machine unlearning with data attribution

Step 1: Rewrite in terms of model outputs

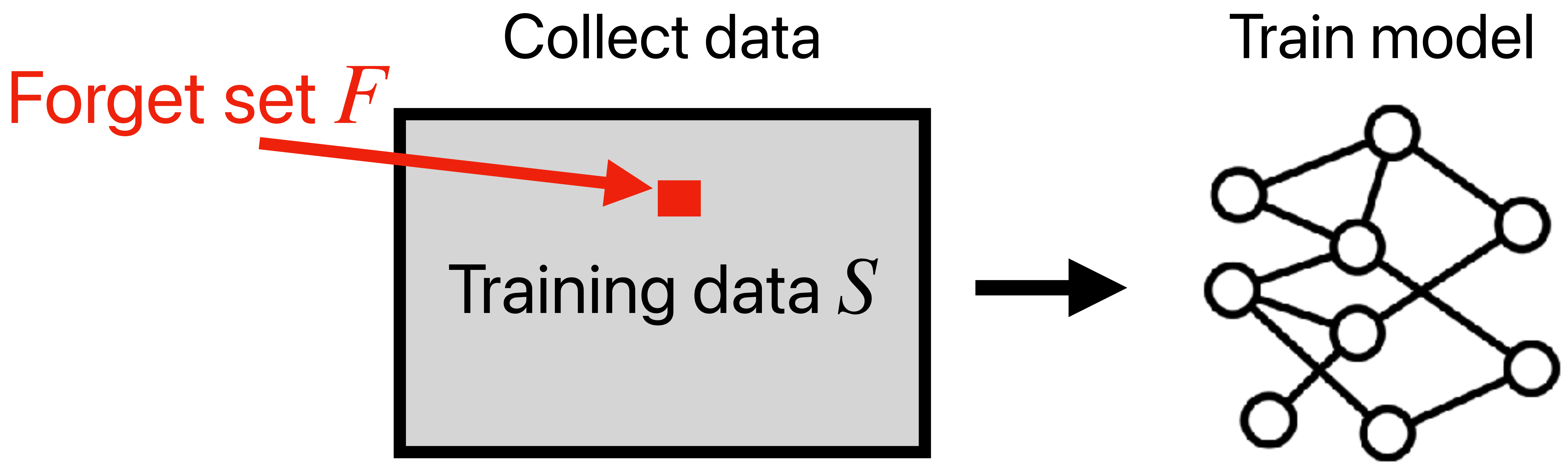
Machine unlearning with data attribution

Step 1: Rewrite in terms of model outputs



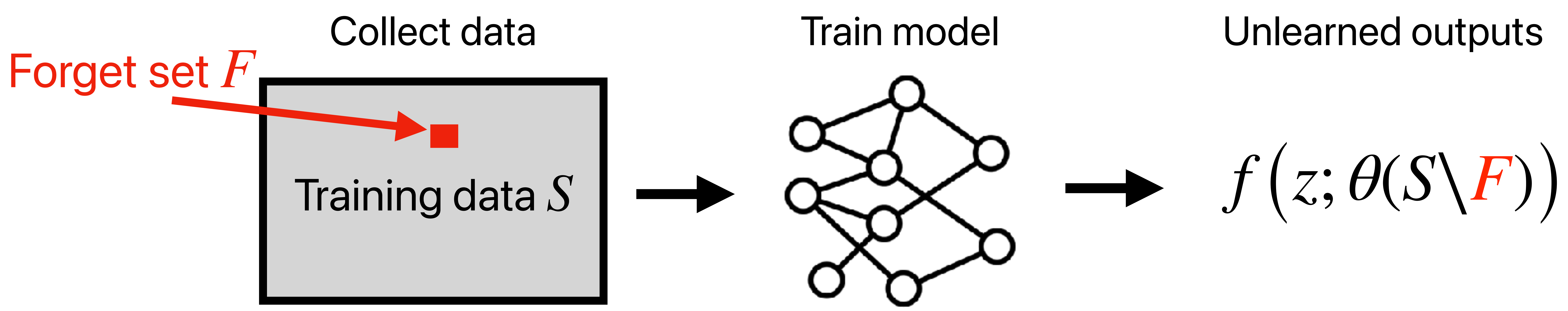
Machine unlearning with data attribution

Step 1: Rewrite in terms of model outputs



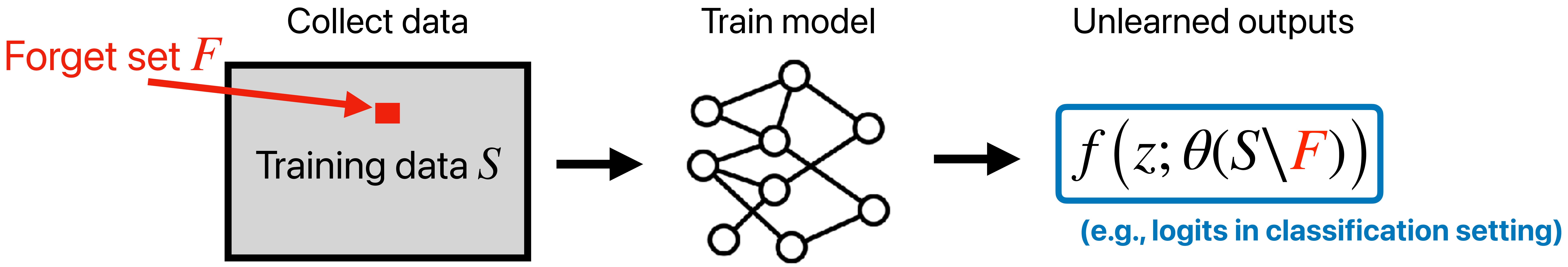
Machine unlearning with data attribution

Step 1: Rewrite in terms of model outputs



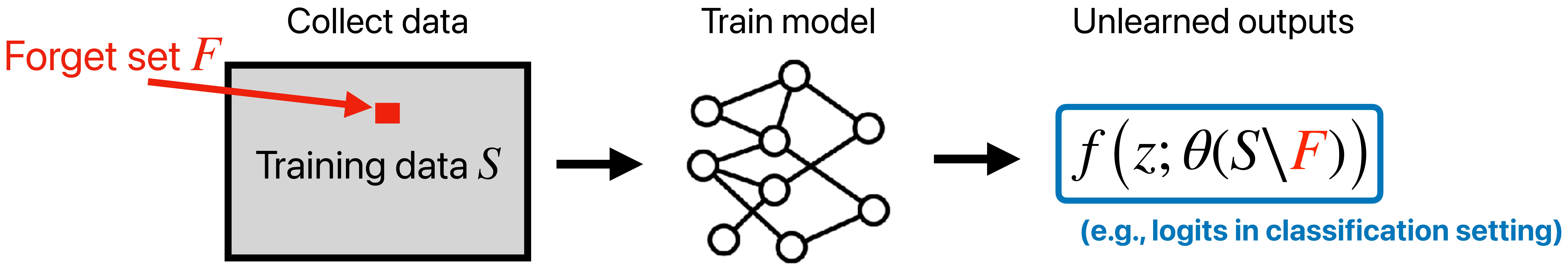
Machine unlearning with data attribution

Step 1: Rewrite in terms of model outputs



Machine unlearning with data attribution

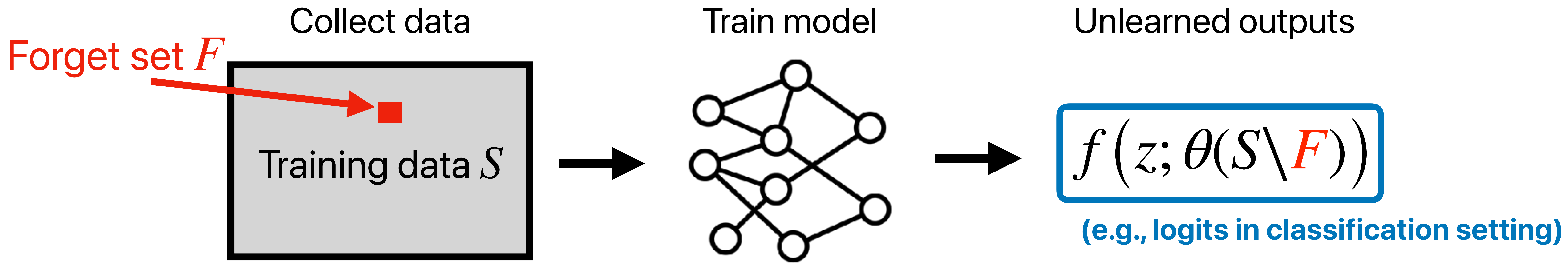
Step 1: Rewrite in terms of model outputs



Unlearning: estimate model outputs $f\left(z; \theta(S \backslash F)\right)$ as if the model had *not* trained on F

Machine unlearning with data attribution

Step 1: Rewrite in terms of model outputs



Unlearning: estimate model outputs $f\left(z;\theta(S \setminus F)\right)$ as if the model had *not* trained on F

First step ✅

Machine unlearning with data attribution

Machine unlearning with data attribution

Step 2: Plug in data attribution estimate for model output

Machine unlearning with data attribution

Step 2: Plug in data attribution estimate for model output

Given model $\theta(S)$ trained on dataset S, forget set F, and test sample z:

Machine unlearning with data attribution

Step 2: Plug in data attribution estimate for model output

Given model $\theta(S)$ trained on dataset S, forget set F, and test sample z:

Machine unlearning: estimate output $f\left(z; \theta(S \backslash F)\right)$ of a model trained without F

Machine unlearning with data attribution

Step 2: Plug in data attribution estimate for model output

Given model $\theta(S)$ trained on dataset S, forget set F, and test sample z:

Machine unlearning: estimate output $f\left(z; \theta(S \backslash F)\right)$ of a model trained without F

Data attribution solution concept: estimate "unlearned" outputs directly via

Machine unlearning with data attribution

Step 2: Plug in data attribution estimate for model output

Given model $\theta(S)$ trained on dataset S, forget set F, and test sample z:

Machine unlearning: estimate output $f\left(z; \theta(S \backslash F)\right)$ of a model trained without F

Data attribution solution concept: estimate "unlearned" outputs directly via

$$\hat{f}_z\,(S \backslash F)\,\Big(\approx f\left(z, \theta(S \backslash F)\right)\Big)$$

Machine unlearning with data attribution

Step 2: Plug in data attribution estimate for model output

Given model $\theta(S)$ trained on dataset S, forget set F, and test sample z:

Machine unlearning: estimate output $f\left(z; \theta(S \backslash F)\right)$ of a model trained without F

Data attribution solution concept: estimate "unlearned" outputs directly via

$$\hat{f}_z\,(S \backslash F)\ \left(\ \approx f\left(z, \theta(S \backslash F)\right)\right)$$

References: [Guo et al. 2020; Sekhari et al. 2021; Suriyakumar et al. 2022; Tanno et al. 2022; Warnecke et al. 2023; Georgiev et al. 2024]

Takeaways: applying data attribution

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

- Predictive data attribution uses a simple formula:

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Predictive data attribution uses a simple formula:

(1) Write in terms of model outputs, then (2) plug-in data attribution

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Predictive data attribution uses a simple formula:

(1) Write in terms of model outputs, then (2) plug-in data attribution

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Predictive data attribution uses a simple formula:

(1) Write in terms of model outputs, then (2) plug-in data attribution

Other major applications:

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Predictive data attribution uses a simple formula:

(1) Write in terms of model outputs, then (2) plug-in data attribution

Other major applications:

- Data valuation: how much is training data worth?

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Predictive data attribution uses a simple formula:

(1) Write in terms of model outputs, then (2) plug-in data attribution

Other major applications:

- Data valuation: how much is training data worth?
- Citations: how can we ground predictions in truth?

Takeaways: applying data attribution

Data attribution helps in "model understanding" tasks

But also solves a richer set of tasks

Predictive data attribution uses a simple formula:

(1) Write in terms of model outputs, then (2) plug-in data attribution

Other major applications:

- Data valuation: how much is training data worth?
- Citations: how can we ground predictions in truth?
- RAG: halfway between "dataset selection" and "citations"

Concluding notes

"All good stories must come to an end"

Recap

Part I: Data problems in ML

Corroborative, game-theoretic, and predictive data attribution

Part II: Theoretical foundations

History & theory of predictive data attribution (datamodeling)

Part III: Scaling to modern settings

Challenges & successes in predictive data attribution for large ML systems

Part IV: Scaling to modern settings

Past, present, and future applications of data attribution

Opinionated perspective: are we "there" yet?

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]
- Frameworks for SOTA methods are even more recent

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]
- Frameworks for SOTA methods are even more recent

Yet, still room for improvement: "TDA... not well known [nor] used" [NKS+ 2023]

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]
- Frameworks for SOTA methods are even more recent

Yet, still room for improvement: "TDA... not well known [nor] used" [NKS+ 2023]

- Effectiveness

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]
- Frameworks for SOTA methods are even more recent

Yet, still room for improvement: "TDA... not well known [nor] used" [NKS+ 2023]

- Effectiveness
- Compute efficiency

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]
- Frameworks for SOTA methods are even more recent

Yet, still room for improvement: "TDA... not well known [nor] used" [NKS+ 2023]

- Effectiveness
- Compute efficiency
- Translation to practice

Opinionated perspective: are we "there" yet?

Focussing on large-scale model setting: rapid progress in recent years

- Classical methods first applied to "machine learning" in 2017 [KL 2017]
- Frameworks for SOTA methods are even more recent

Yet, still room for improvement: "TDA... not well known [nor] used" [NKS+ 2023]

- Effectiveness
- Compute efficiency
- Translation to practice

Prediction: data attribution will be standard part of ML pipeline in 5 years

Data attribution at scale

Connecting ML behavior to (training) data

Andrew Ilyas (Part I & II) andrewilyas.com
Sung Min (Sam) Park (Part III) sungminpark.com
Logan Engstrom (Part IV) loganengstrom.com
Kristian Georgiev kristian-georgiev.github.io
Aleksander Mądry madry.mit.edu

🌐 **ml-data-tutorial.org | ICML 2024**