An instance of inference: treatment effect estimation

Some general goals

- Predict the health state of an individual after treatment.
- Develop a strategy for steering the treatment over time.
- Understand the mechanisms underlying illness and recovery.

Variables

- Holistic: health state, treatment, individual, environment.
- x: quantifiers of the health state.
- z: factors characterizing the patient and the current conditions, including current or past values of x. Some may be known or measurable, some latent.
- a: features of the treatment under consideration.

Observations o from past and the current patient: values of x, z and a, or of quantities related to them.

General tasks

- Build a model for the effect of action a on the outcome x, qualified by factors z:

  $a, z \to x$: regression, classification

  $\rho(x|a, z)$: conditional probability estimation

  $x^i \sim \rho(:|a, z)$: simulation
- Uncover hidden parameters: factor discovery, clustering, dimensional reduction: $x, a \to z$
- Figure out the current state and model parameters: filtering, data assimilation, diagnosis: $o^{1,\dots,n} \to z^n, \alpha^{(n)}, x$
- Steer the treatment: optimal control, reinforcement learning: $z, x \to a$

Tensions and challenges 1

Interpretability

Do we care more about predicting or understanding? Model simplicity versus accuracy and detail.

Model versus data-driven inference: a full palette between field knowledge-based models and black boxes.

Tensions and challenges 2

Identifiability

A model detailed enough to be deemed realistic by a practitioner may include parameters that the data cannot robustly pin down.

Big data.

Lacking the right tools for analysis, excessive information may, somewhat paradoxically, deteriorate the quality of a prediction.

Some partial solutions

Regularizers: for robustness, to enforce regularity, to promote interpretability and to mitigate overfitting. Examples: penalizers –such as ridge regression– and priors on the parameters.

Cross-validation: training and testing populations.

Tensions and challenges 3

Observational studies vs. randomized experiments

Individualization of prediction vs. aggregation of data

Variability of data type

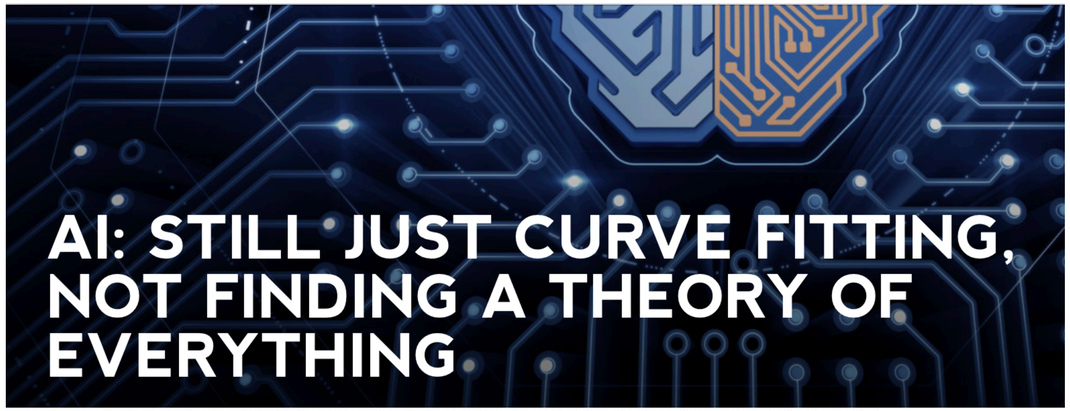
Reliability of data, robustness to outliers

Fairness

Demystifying Machine Learning

It's just…

- *Interpolation*
- *Curve fitting*
- *Regression*
- *etc.*



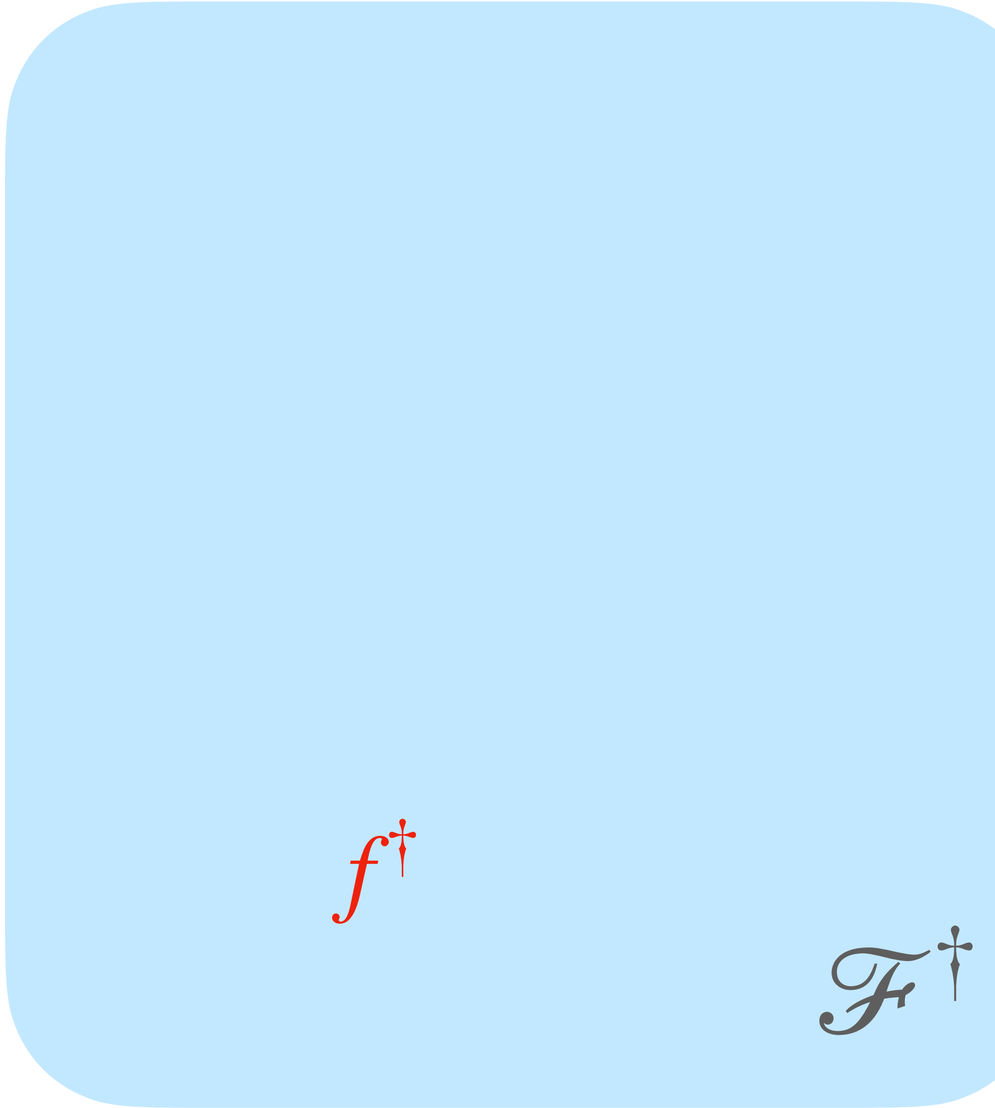
Inferring a predictive function

machine learning

Inferring a predictive function

machine learning

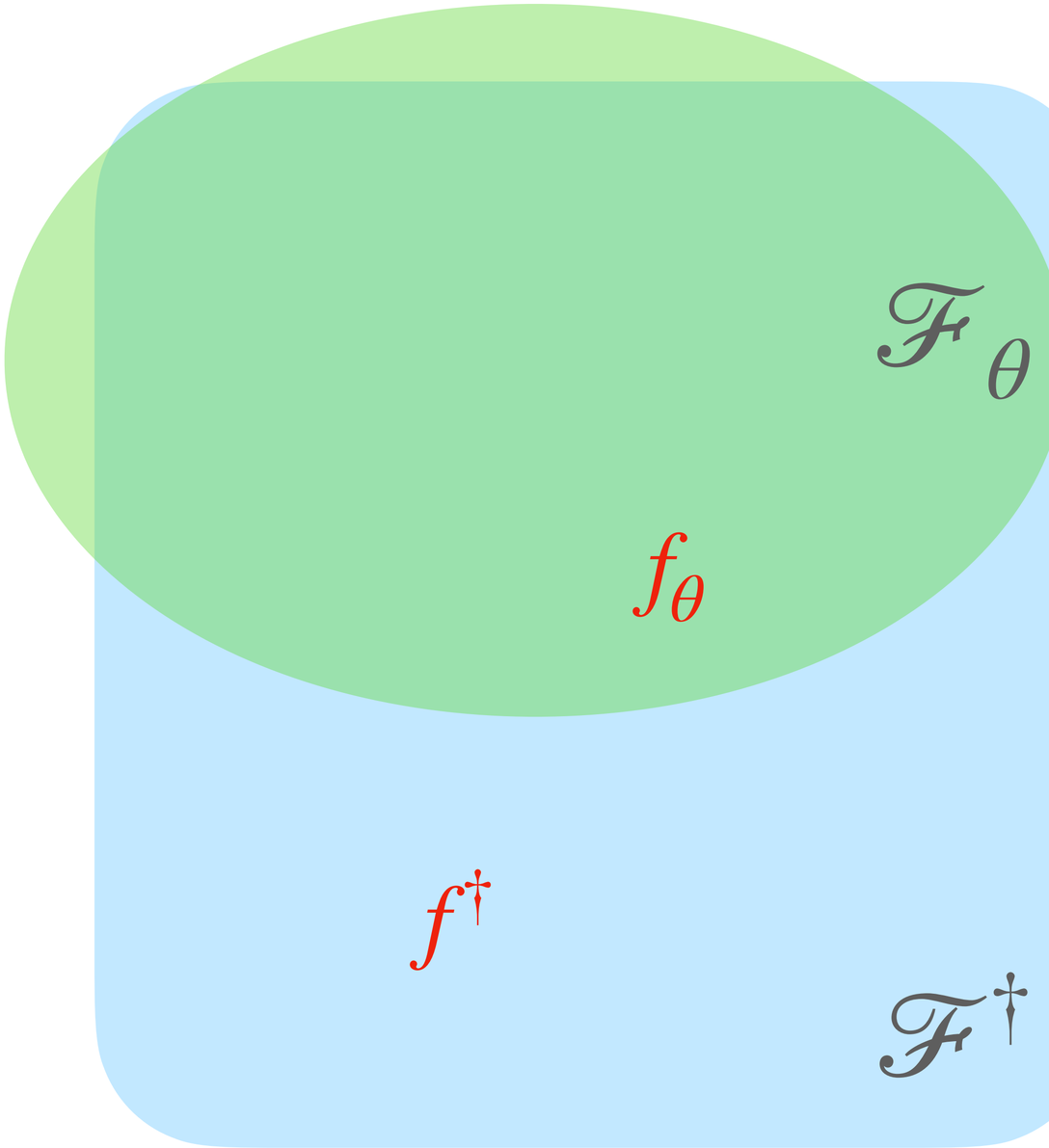
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$



Inferring a predictive function

machine learning

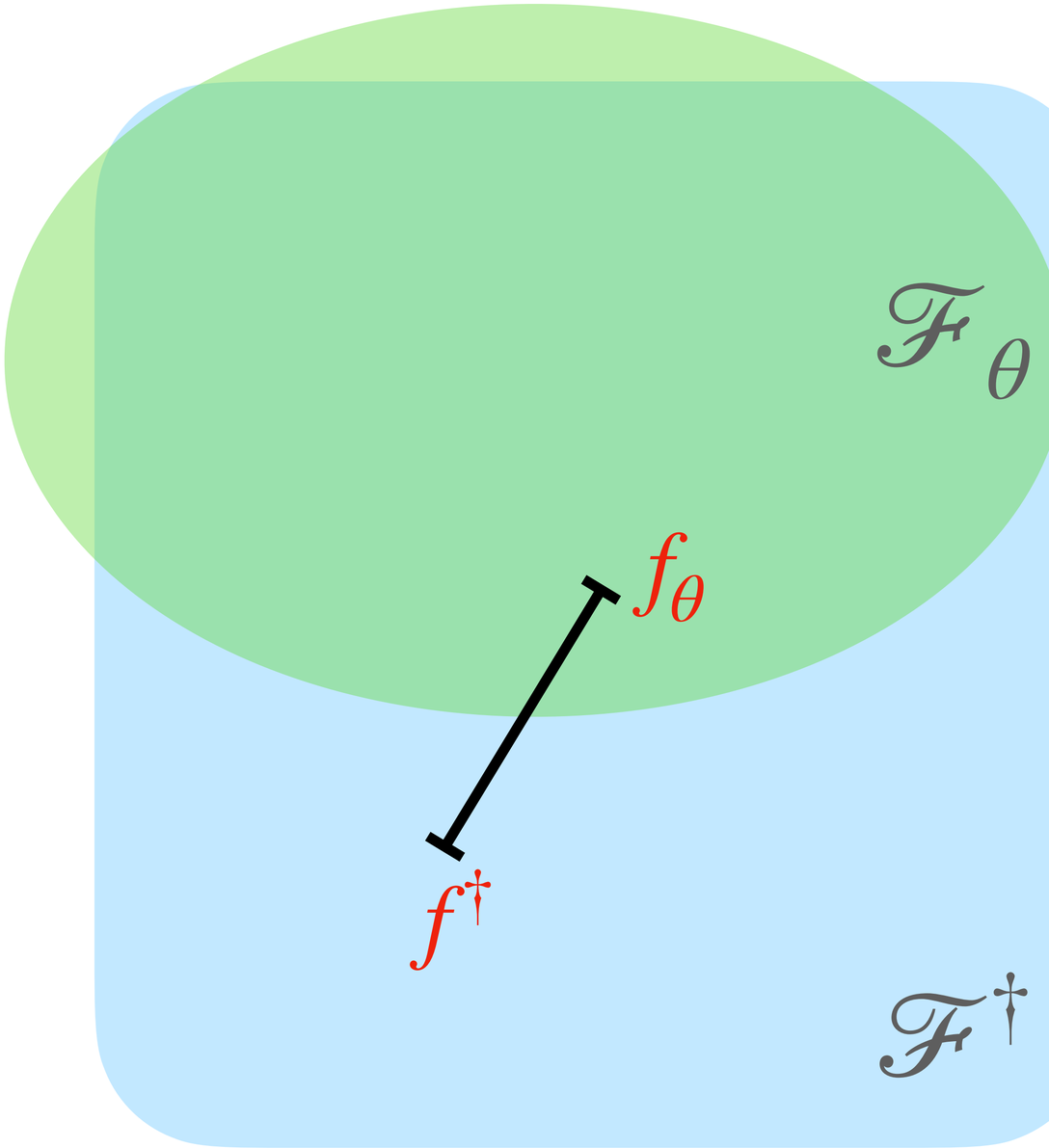
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$



Inferring a predictive function

machine learning

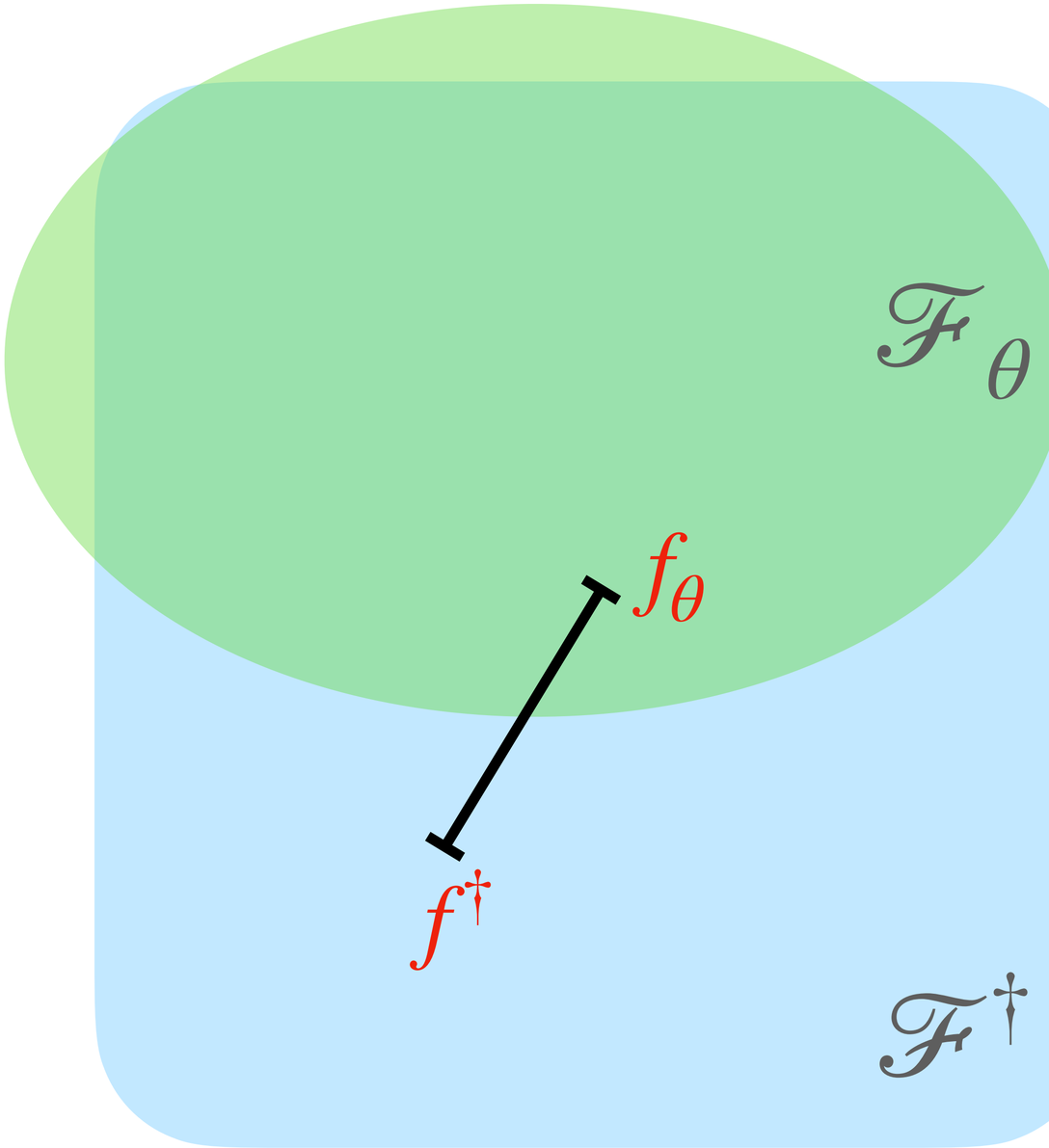
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$



Inferring a predictive function

machine learning

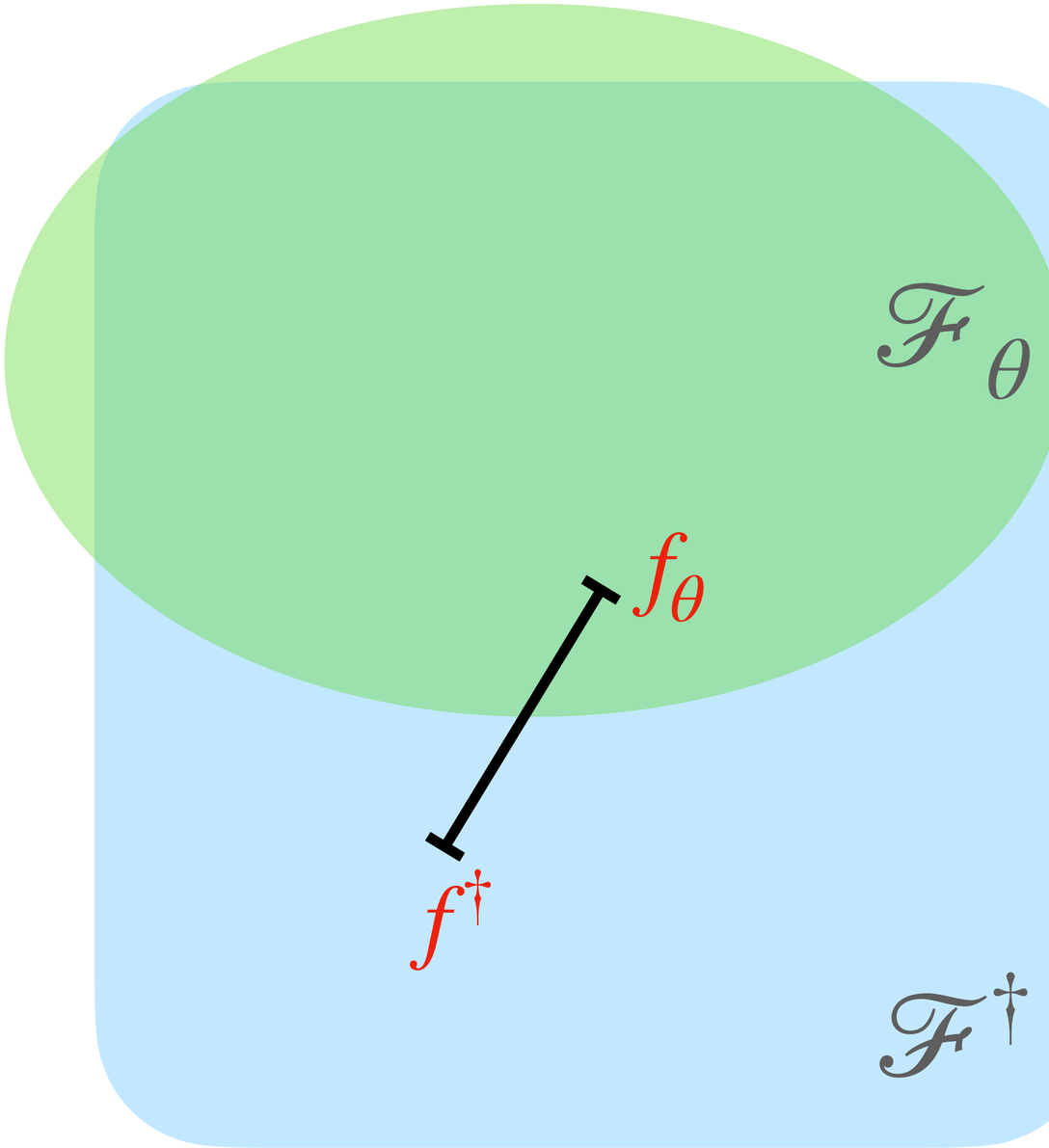
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$



Inferring a predictive function

machine learning

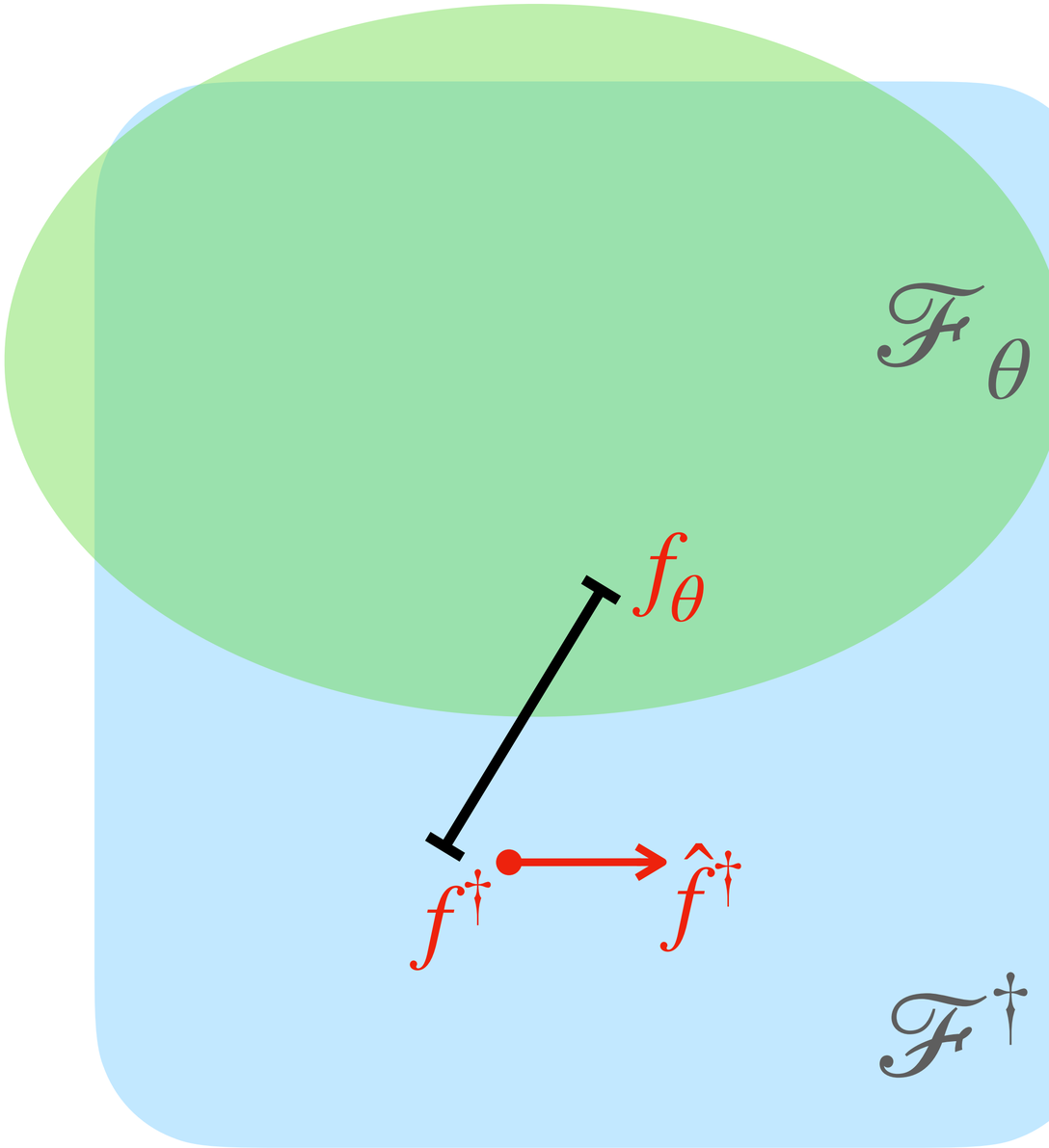
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$
5. Define optimal functions
    1. $f^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\mathscr{L}(f)$



Inferring a predictive function

machine learning

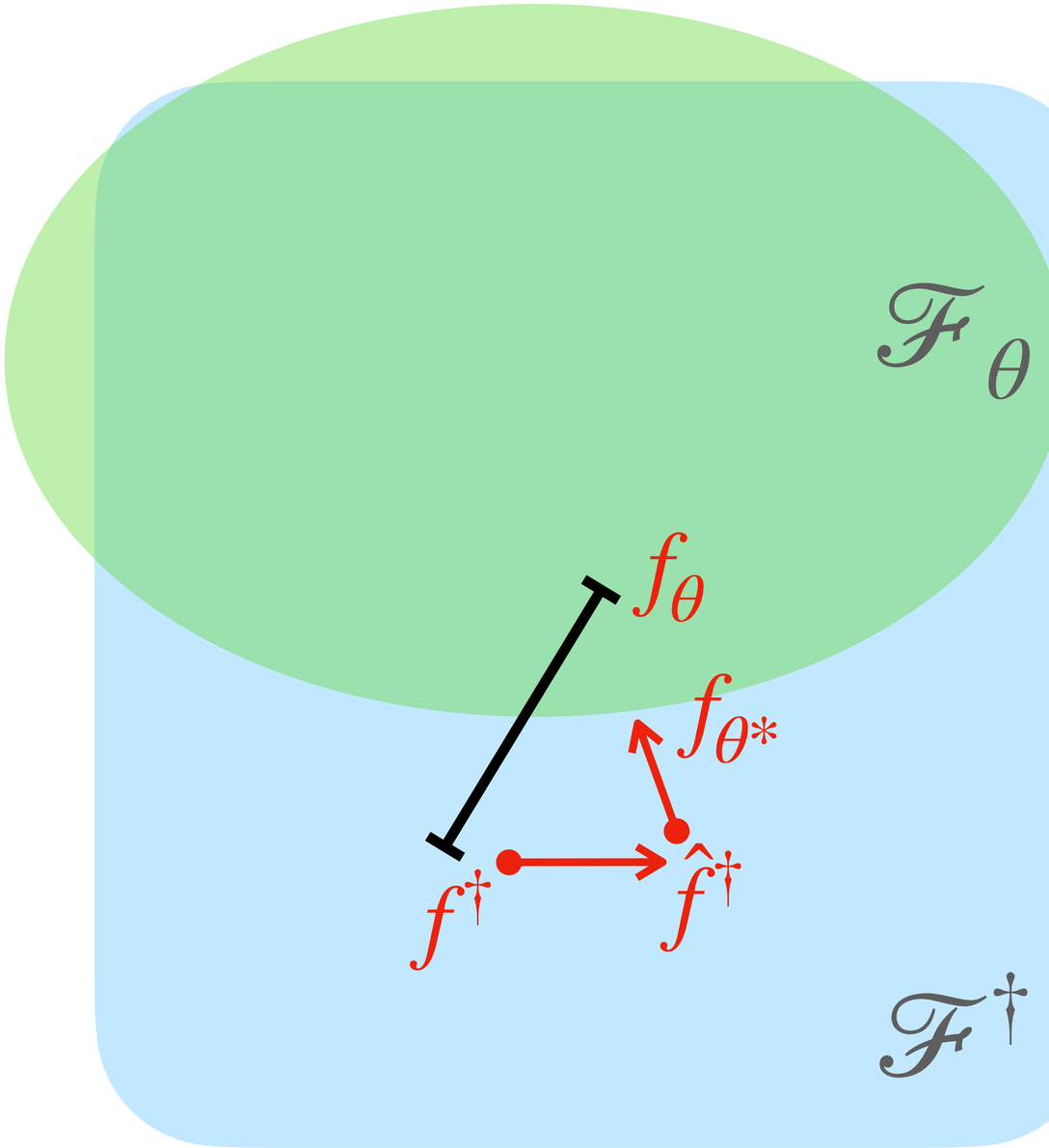
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$
5. Define optimal functions
    1. $f^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\mathscr{L}(f)$
    2. $\hat{f}^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\widehat{\mathscr{L}}(f)$



Inferring a predictive function

machine learning

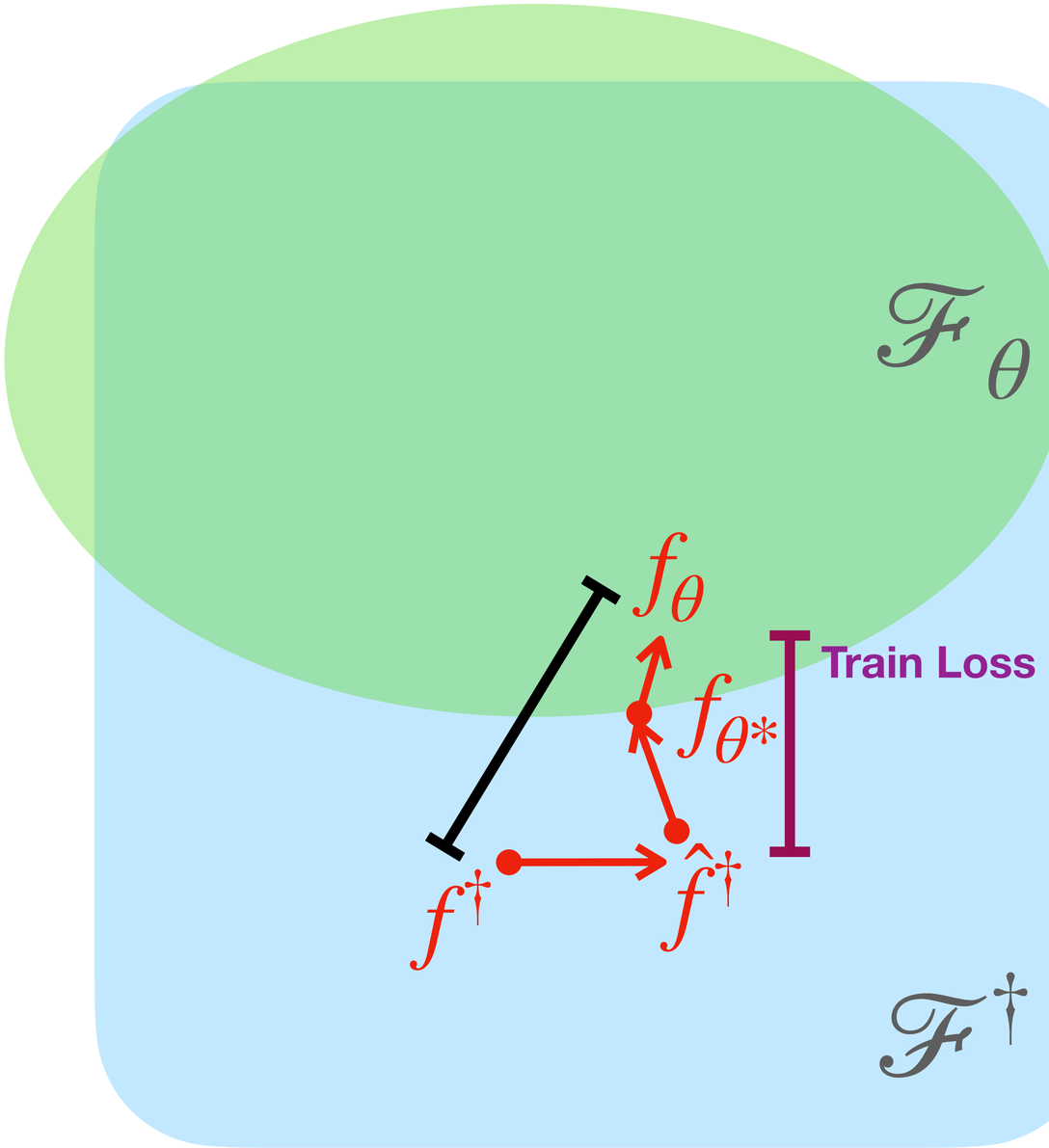
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$
5. Define optimal functions
    1. $f^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\mathscr{L}(f)$
    2. $\hat{f}^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\widehat{\mathscr{L}}(f)$
    3. $f_{\theta^*} = \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$



Inferring a predictive function

machine learning

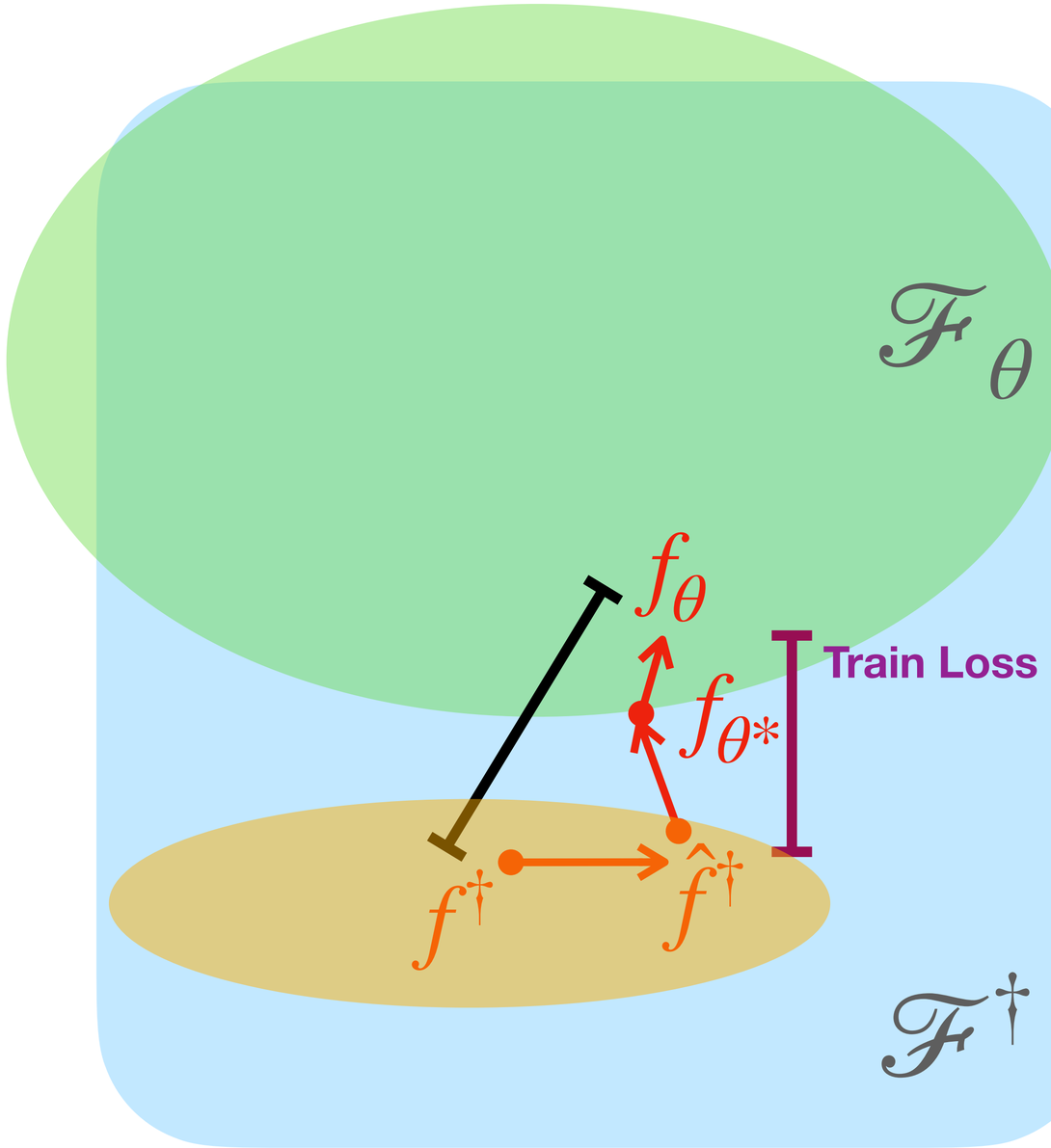
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$
5. Define optimal functions
    1. $f^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\mathscr{L}(f)$
    2. $\hat{f}^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\widehat{\mathscr{L}}(f)$
    3. $f_{\theta^*} = \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$
6. Minimize a possibly non-convex function (SGD, particle methods, etc.)
    1. $f_\theta \approx \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$



Inferring a predictive function

machine learning

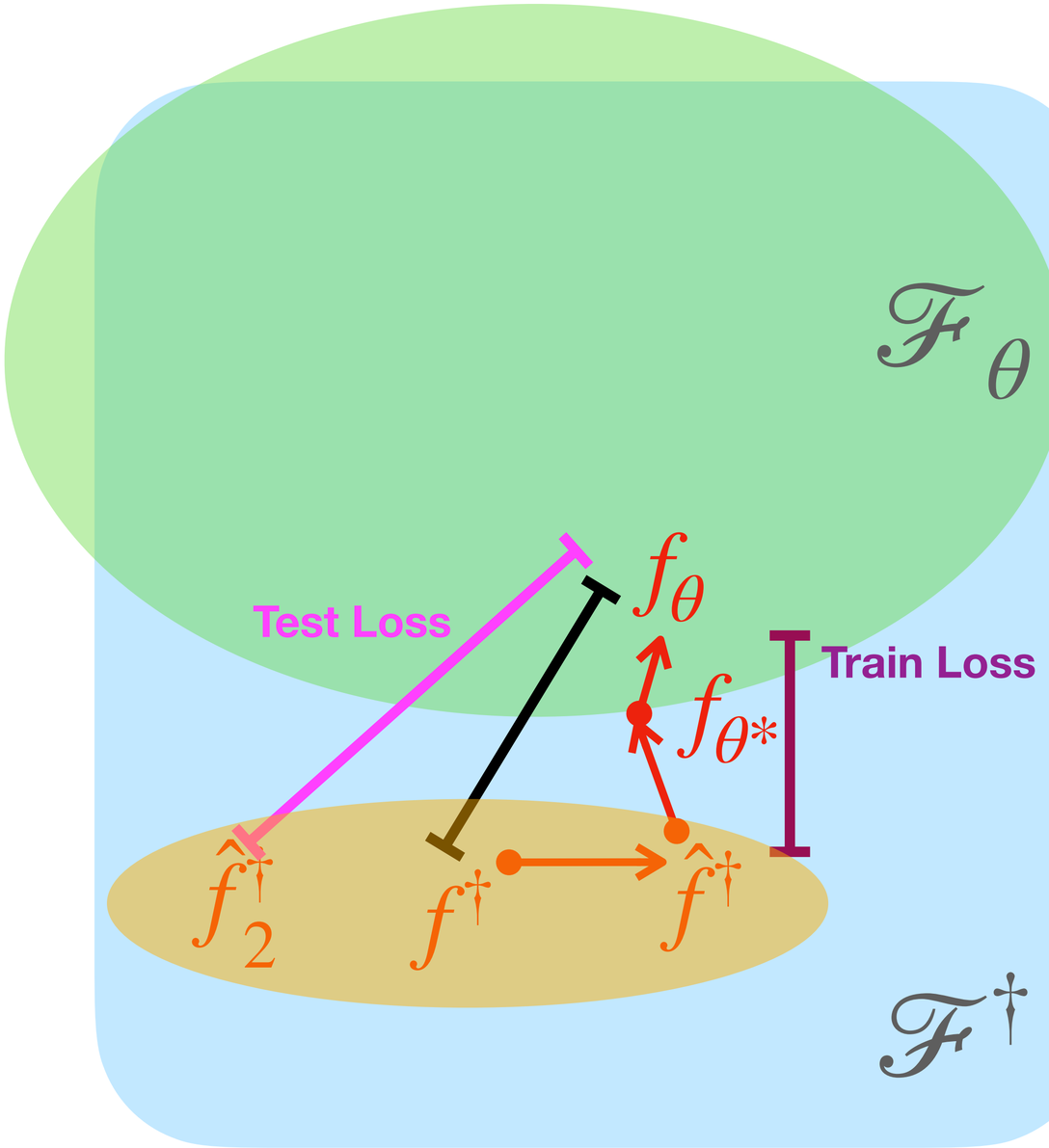
1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$
5. Define optimal functions
    1. $f^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\mathscr{L}(f)$
    2. $\hat{f}^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\widehat{\mathscr{L}}(f)$
    3. $f_{\theta^*} = \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$
6. Minimize a possibly non-convex function (SGD, particle methods, etc.)
    1. $f_\theta \approx \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$
7. Evaluate approximation fidelity with independent test set



Inferring a predictive function

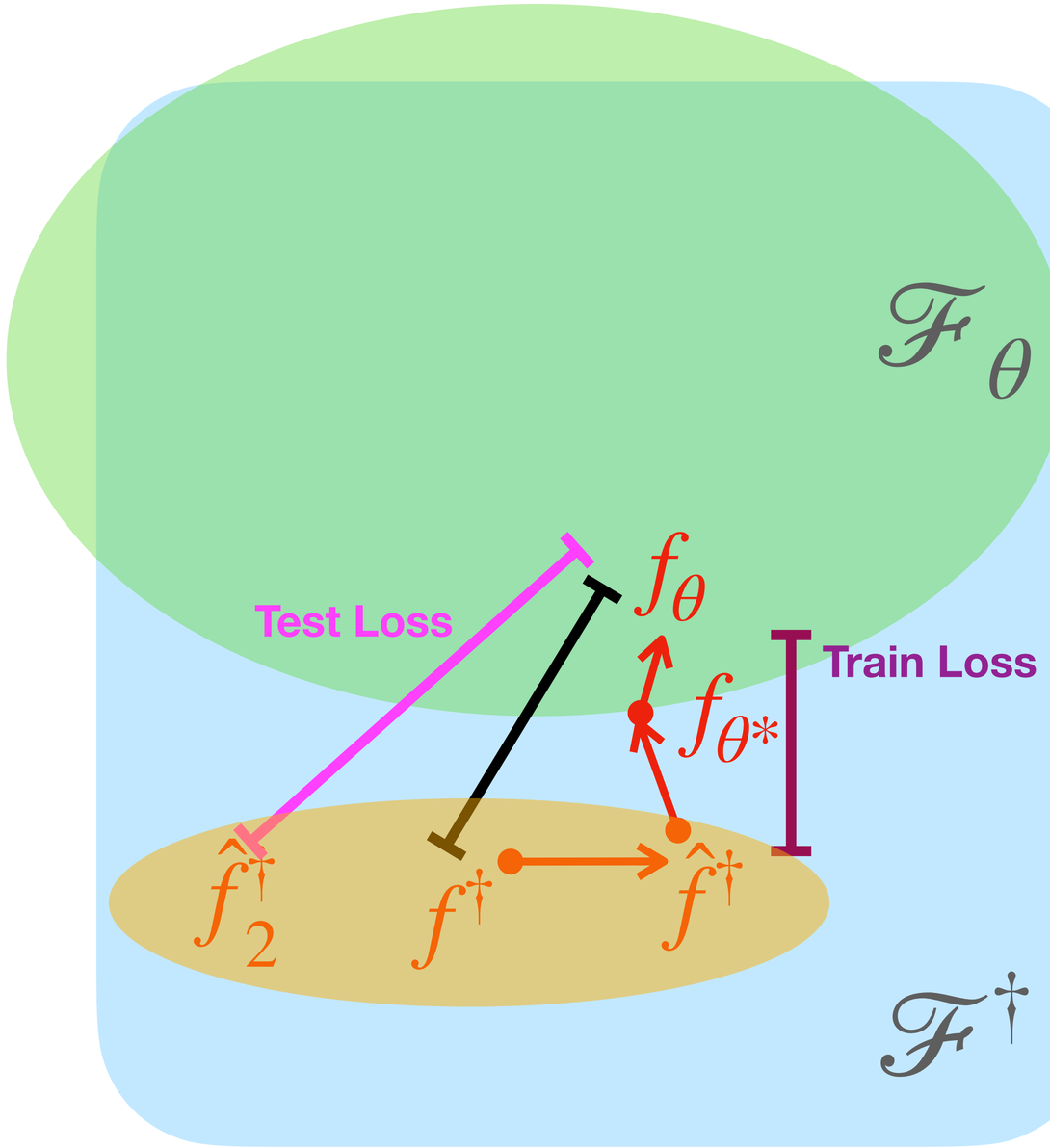
machine learning

1. Define the task: Look for $f^\dagger : X \mapsto Y$ mapping inputs $x \in X$ to outputs $y \in Y$
2. Define model class $\mathscr{F}_\theta$ that can approximate $f^\dagger \in \mathscr{F}^\dagger$
    1. Linear regression: $f(x;\ A, b) = Ax + b$
    2. Neural network: $f(x;\ A, b, C) = C\tanh(Ax + b)$
3. Define a distance (e.g. L^2_μ)
    1. $\mathscr{L}(f_\theta) = \|f_\theta - f^\dagger\| = \mathbb{E}_{x\sim\mu}\left[\| f_\theta(x) - f^\dagger(x)\|^2\right]$
4. Approximate distance empirically with data (Monte Carlo, $x_i \sim \mu$ i.i.d)
    1. $\widehat{\mathscr{L}}(f_\theta) = \sum_i \| f_\theta(x_i) - y_i)\|^2$
5. Define optimal functions
    1. $f^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\mathscr{L}(f)$
    2. $\hat{f}^\dagger = \text{argmin}_{f\in\mathscr{F}^\dagger}\widehat{\mathscr{L}}(f)$
    3. $f_{\theta^*} = \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$
6. Minimize a possibly non-convex function (SGD, particle methods, etc.)
    1. $f_\theta \approx \text{argmin}_{f_\theta\in\mathscr{F}_\theta}\widehat{\mathscr{L}}(f_\theta)$
7. Evaluate approximation fidelity with independent test set



Standard “Problems w/ Solutions” in “AI”

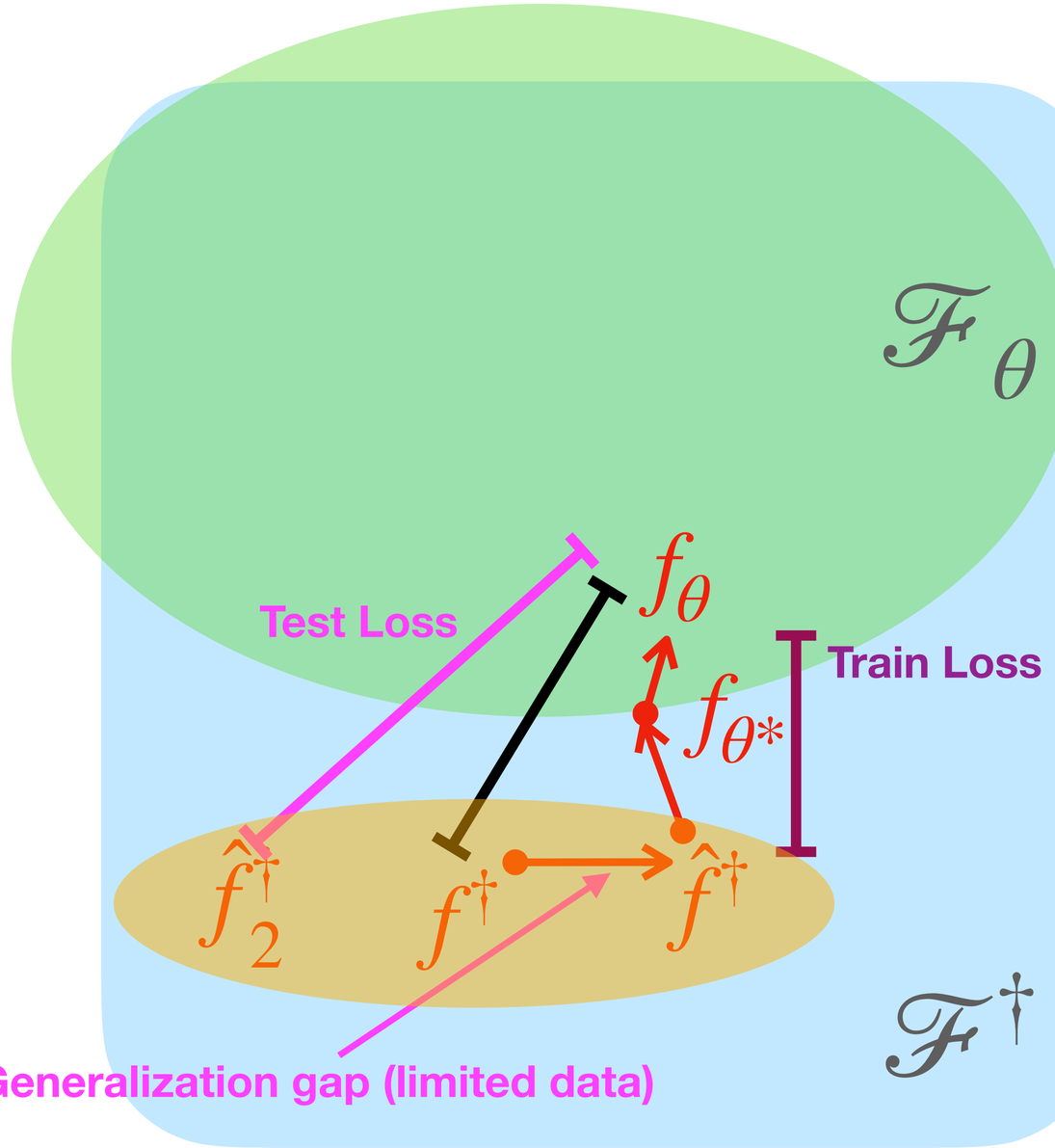
machine learning



Standard “Problems w/ Solutions” in “AI”

machine learning

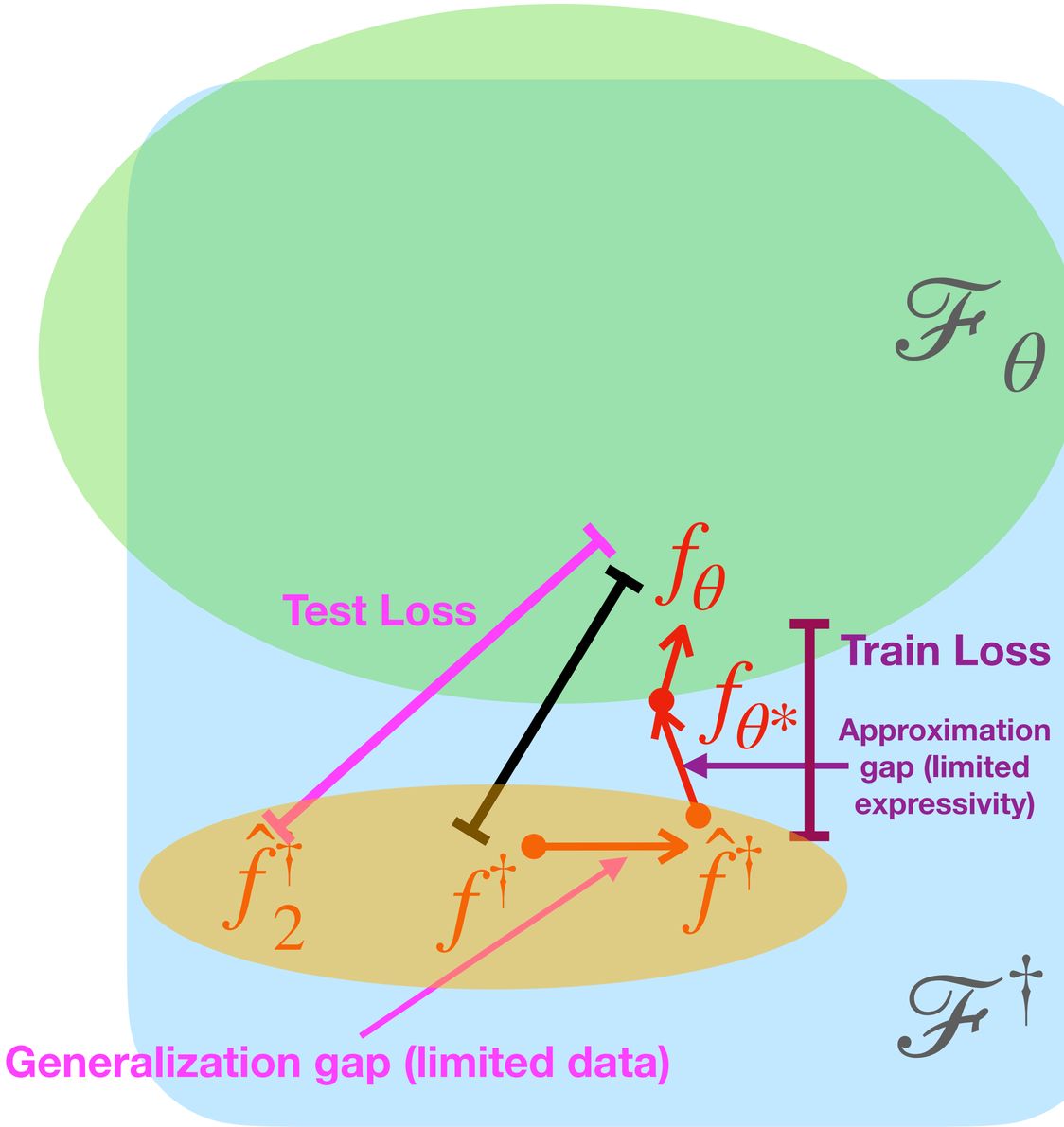
1. Generalization Gap
    1. Tons of data!



Standard “Problems w/ Solutions” in “AI”

machine learning

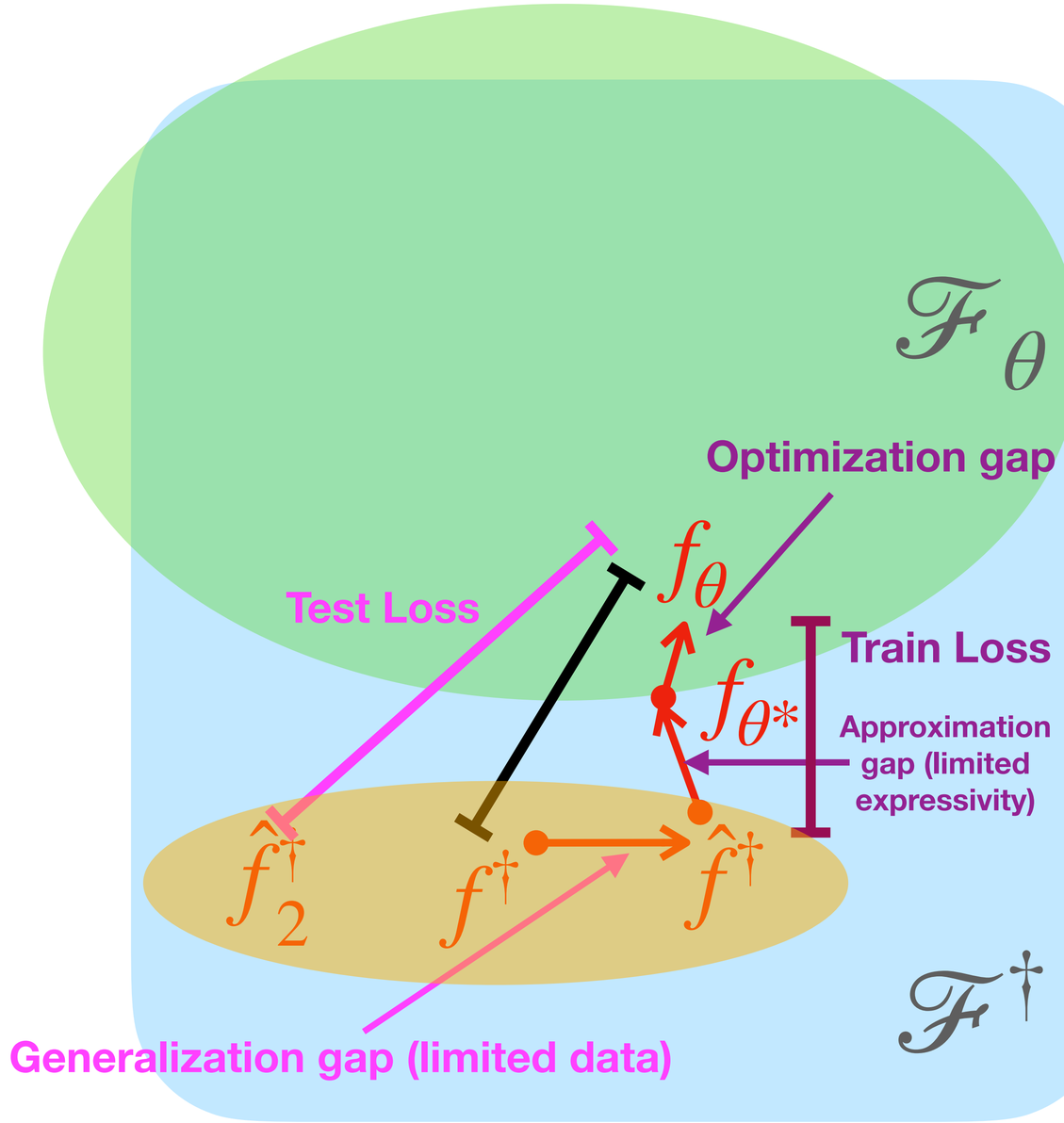
1. Generalization Gap
    1. Tons of data!
2. Approximation Gap
    1. More parameters!



Standard “Problems w/ Solutions” in “AI”

machine learning

1. Generalization Gap
    1. Tons of data!
2. Approximation Gap
    1. More parameters!
3. Optimization Gap
    1. SGD (and variants) with lots of tunings/ hacks
    2. More parameters!
        1. Over-parametrization seems to convexify loss surfaces



Small Data Challenges

machine learning

Small Data Challenges

machine learning

1. If data covers the input-output space sparsely, but somewhat uniformly
    1. Regularize / limit complexity

Small Data Challenges

machine learning

1. If data covers the input-output space sparsely, but somewhat uniformly
    1. Regularize / limit complexity
2. If data covers subset of domain, be careful about extrapolation
    1. e.g. be sure the learnt function is very boring away from data!

Small Data Challenges

machine learning

1. If data covers the input-output space sparsely, but somewhat uniformly
    1. Regularize / limit complexity
2. If data covers subset of domain, be careful about extrapolation
    1. e.g. be sure the learnt function is very boring away from data!

Don’t be afraid of expressive models—just learn how to regularize!

Other challenges

machine learning

Other challenges

machine learning

1. Latent variables
    1. Hypothesize $f^{\dagger} : X \mapsto Y$, but our input space $U_{\text{inputs}} \subset X$
    2. Need to discover or account for hidden/latent variables!

Other challenges

machine learning

1. Latent variables
    1. Hypothesize $f^{\dagger} : X \mapsto Y$, but our input space $U_{\text{inputs}} \subset X$
    2. Need to discover or account for hidden/latent variables!
2. Noisy inputs
    1. For non-linear functions, input noise does not “cancel out”
        1. $f(x) \not\approx \mathbb{E}_{\eta}[f(x+\eta)]$ even for mean-zero noise η
    2. Need to incorporate de-noising during learning

Other challenges

machine learning

1. Latent variables
    1. Hypothesize $f^\dagger : X \mapsto Y$, but our input space $U_{\text{inputs}} \subset X$
    2. Need to discover or account for hidden/latent variables!
2. Noisy inputs
    1. For non-linear functions, input noise does not “cancel out”
        1. $f(x) \not\approx \mathbb{E}_\eta[f(x+\eta)]$ even for mean-zero noise η
    2. Need to incorporate de-noising during learning
3. Non-stationarity
    1. Shifting data-distribution
    2. Shifting map $f^\dagger$

RL Introductory Discussion

Eric B. Laber

Duke University

Banff, June 2022

Acknowledgments

Thanks to

- Esteban and Matt
- Duke/UNC AI and Precision Medicine Lab
- National Science Foundation
- National Institutes of Health

Joint work with

- Alex Cloud*
- Michael Kosorok

On choices

Dad always thought laughter was the best medicine, which I guess is why several of us died of tuberculosis.
—Jack Handy

What is reinforcement learning (RL)?

- Subfield of ML focused on decision making under uncertainty
- Goal is to efficiently generate and use information to inform decision making to maximize some long term measure of utility
- Encompasses ideas from a wide range of disciplines
  - Experimental design
  - Statistical efficiency theory
  - Causal inference
  - Optimization
  - ...

RL in the news

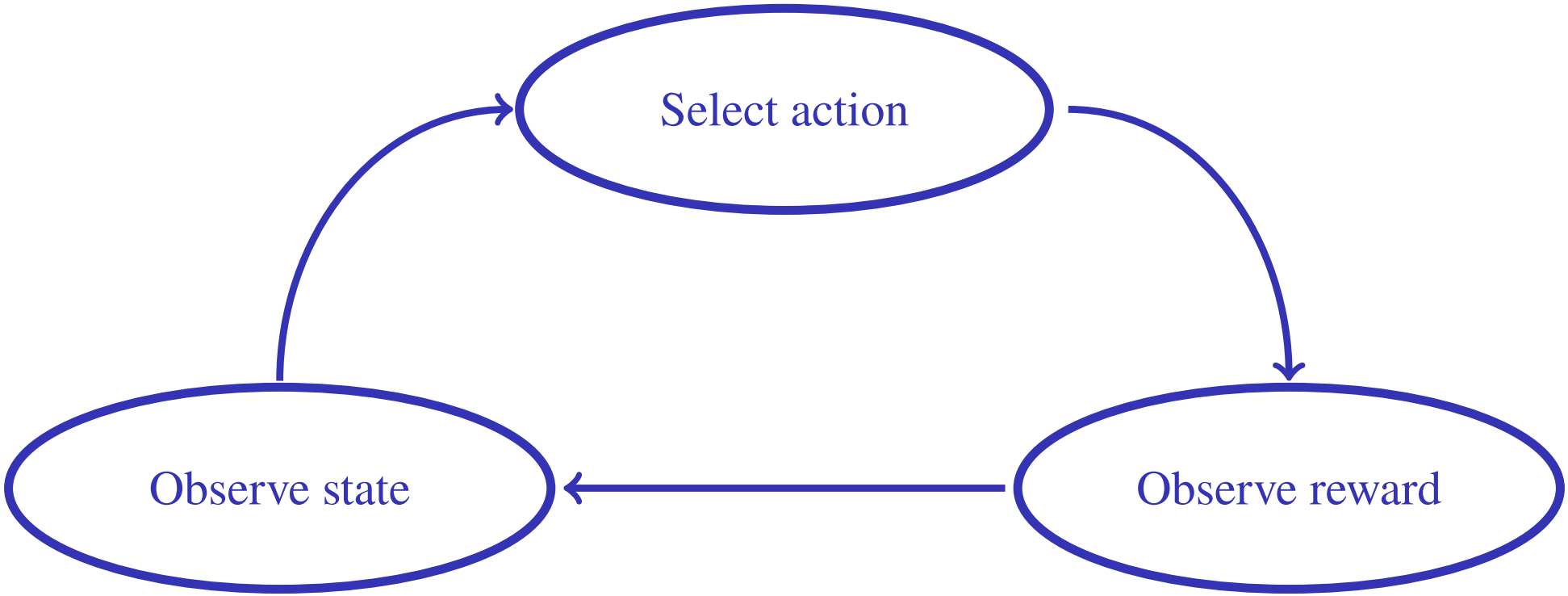
- Many recent high-profile examples of RL
  - Games
  - Robotics
  - Protein structure prediction

RL in the wild[1]

- Precision medicine
  - Adaptive clinical trials
  - Dynamic treatment regimes and just-in-time adaptive interventions (mHealth)
  - System-level (triage, adaptive team-based care, recall, etc.)
  - Public health (messaging, resource allocation, etc.)
- Retail
  - Advertising, recommender systems, pricing, etc.
  - Assortment selection
  - Fraud detection
  - Site selection

[1] Heavily biased by my own experience.

Reinforcement learning (RL) model of decision making



- Learn from experience $\Rightarrow$ action selection balance info and reward
- Examples
  - Randomized clinical trial: explore than exploit
  - Batched learning: alternating phases of learning and optimization
  - Fully online: jointly learn and optimization

Formalizing a batch one-stage decision problem

- Observe $\{(\boldsymbol{X}_i, A_i, Y_i)\}_{i=1}^n$ iid from P
  - $\boldsymbol{X} \in \mathcal{X} \subseteq \mathbb{R}^p$ covariates (decision context)
  - $A \in \mathcal{A}$ action (treatment, intervention, decision, etc.)
  - $Y \in \mathbb{R}$ utility (outcome, output, reward, etc.) higher is better
- Goal: select actions to maximize expected utility

Policies

- $\psi : \mathcal{X} \to 2^{\mathcal{A}}$ is set of allowable actions, i.e., $\psi(\boldsymbol{x}) \subseteq \mathcal{A} \setminus \emptyset$
- Policy $\pi : \mathcal{X} \to \mathcal{A}$ such that $\pi(\boldsymbol{x}) \in \psi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$
  - Under π decision maker will select action $\pi(\boldsymbol{x})$ in context $\boldsymbol{x}$
  - Define $V(\pi) \triangleq \mathbb{E}^{\pi} Y$ to be expected utility if actions are selected according the policy π
  - Optimal policy satisfies $V(\pi^{\text{opt}}) \geq V(\pi)$ for all π

Approaches to estimation in batch one-stage setting

- Regression
  - Define $Q(\boldsymbol{x}, a) \triangleq \mathbb{E}(Y \big| \boldsymbol{X} = \boldsymbol{x}, A = a)$
  - Construct estimator $\widehat{Q}_n$ of Q using observed data
  - Estimated optimal policy $\widehat{\pi}_n(\boldsymbol{x}) = \arg\max_{a \in \psi(\boldsymbol{x})} \widehat{Q}_n(\boldsymbol{x}, a)$
- Policy search
  - For any policy π construct estimator[2] $\widehat{V}_n(\pi)$ of $V(\pi)$
  - Given class of policies Π compute $\widetilde{\pi}_n = \arg\max_{\pi \in \Pi} \widehat{V}_n(\pi)$

[2]Common approaches include augmented inverse probability weighting, marginal structural models, and G-computation. See Tsiatis et al. (2019) for review.

Considerations and open problems in one-stage batch case

- Scale and complexity
  - High-dimensional action spaces, e.g., assortment selection has $O(n^k)$ actions where n is catalog size and k is selection size
  - High-dimensional and complex feature spaces, e.g., txt recommendation from functional, imaging, and/or genetic data
- Experimental design
  - Assign actions to maximize efficiency of $\widehat{\pi}_n$
  - Ensure power for model diagnostics, secondary analyses, etc.
- Multiple outcomes and preference heterogeneity
  - Efficacy vs. side effects
  - Stated vs. revealed preferences

Formalizing an online one-stage decision problem

- Consider contextual bandit setting
- Observe $\{(\boldsymbol{X}_i, A_i, Y_i)\}_{i=1}^n$
  - Contexts $\boldsymbol{X}$ from fixed but unknown distn $P_{\mathbf{x}}$
  - Utility Y given $\boldsymbol{X} = \boldsymbol{x}, A = a$, from fixed but unknown distn $P_{y|\mathbf{x},a}$
  - Actions A_i drawn from distn which is allowed to vary over time and to depend on entire previous history $\overline{\boldsymbol{X}}_i, \overline{\boldsymbol{A}}_{i-1}, \overline{\boldsymbol{Y}}_{i-1}$
- Goal: adaptively select actions to maximize cumulative reward[3]

[3] I'm using reward here as a generic quantity which may mean utility, information, or some composite outcome.

Batch vs. online one-stage problems

- Learn from accumulating information
  - Generate information efficiently
  - Improve models $\Rightarrow$ more utility
  - Much more flexibility
- Inference more complicated
  - (Martingale) CLT may not apply directly
  - Standard tests and confidence intervals invalid with modification

Approaches to estimation in contextual bandit

- Perturbation-based methods
  - ϵ-greedy: estimate $\widehat{\pi}_n$ as in batch setting, follow $\widehat{\pi}_n$ with probability $(1 - \epsilon)$ and draw A_n from some 'exploration distribution' otherwise
  - Sample from posterior distribution of optimal posterior (Thompson Sampling)
- Deterministic approaches
  - Upper confidence bound selection: choose action with larger upper confidence bound on mean reward
  - Penalize reward with information gain (e.g., info-directed sampling)
- Many variant and nuances

Considerations and open problems in contextual bandits

- Same problems of scale and complexity as one-stage batch case
- Experimental design problem becomes much richer
  - Interim updates to estimated optimal designs
  - Sequential designs (enrichment, optimal, etc.)
- Study planning
  - Sample size and power analyses
  - Interim analyses, stopping criteria, etc.
- Indefinite deployment $\Rightarrow$ non-stationarity

Markov decision processes (MDPs)

- Ubiquitous model for RL
- Observe $\left\{\boldsymbol{S}_i^1, A_i^1, \ldots, \boldsymbol{S}_i^T, A_i^T, \boldsymbol{S}^{T+1}\right\}_{i=1}^n$ drawn i.i.d. from P
  - T observation period, e.g., trial follow-up
  - $\boldsymbol{S}^t \in \mathcal{S} \subseteq \mathbb{R}^p$ state at time t
  - $A \in \mathcal{A}$
- Utility $U^t = u(\boldsymbol{S}^t, A^t, \boldsymbol{S}^{t+1})$, higher is better
- Assume homogeneous MDP $\rightarrow$ focus on deterministic stationary policies $\pi : \mathcal{S} \rightarrow \mathcal{A}$

Notions of cumulative utility

- Discounted utility

$$V_\gamma(\boldsymbol{s};\pi) \triangleq \mathbb{E}^\pi \left\{ \sum_{k\geq 0} \gamma^k U^{t+k} \middle| \boldsymbol{S}^t = \boldsymbol{s} \right\}$$

- Average utility[4]

$$\overline{V}(\boldsymbol{s};\pi) \triangleq \lim_{T\to\infty} \mathbb{E}^\pi \left(\frac{1}{T} \sum_{k=1}^{T} U^{t+k} \middle| \boldsymbol{S}^t = \boldsymbol{s} \right)$$

- Given some measure $V(\boldsymbol{s};\pi)$ of the 'goodness' of π, optimal policy satisfies $V(\boldsymbol{s};\pi^{\text{opt}}) \geq V(\boldsymbol{s};\pi)$ for all π

[4]Under mild regularity conditions $\overline{V}$ doesn't depend on the starting state.

Estimation (of π^{opt}) in MDPs

- Model-based: system dynamics model + simulation optimization
- Model-free: construct estimating functions for optimal policy
  - Typically derived from Bellman equations
  - Only model part of dynamics, e.g., moments
- Bias-variance trade-off in model-free vs. model-based

Considerations and open problems in MDPs

- Scalability and complexity
- Experimental design is still more complicated
  - Need to ensure sufficient exploration of state space
  - Need to consider delayed effects
- Feature construction to ensure Markov (approx) holds
- Need principled methods for optimally combing model-based and model-free methods

We're excited to chat about these problems!

eric.laber@duke.edu