Modeling Machine Learning[1]

Andrew Caplin, Daniel Martin, and Philip Marx

May 2022

[1]We thank the NOMIS Foundation and the Alfred. P. Sloan Foundation for their generous support for research on the Cognitive Foundations of Economic Behavior.

Apologia

- Unfamiliar question and non-standard notation
- Conceptual as much as technical
- Not enough time to go deeply into technical details
- Builds on prior work on Bayesian learning: Caplin and Martin (2015), Caplin and Dean (2015), and Caplin, Martin, and Marx (2022a,b)
- Diverse applications underway, but only present the first step today

Motivation (1/4)

- Machine learning is increasingly important
  - Has become central to the modern economy
  - Is improving prediction across a range of economic problems

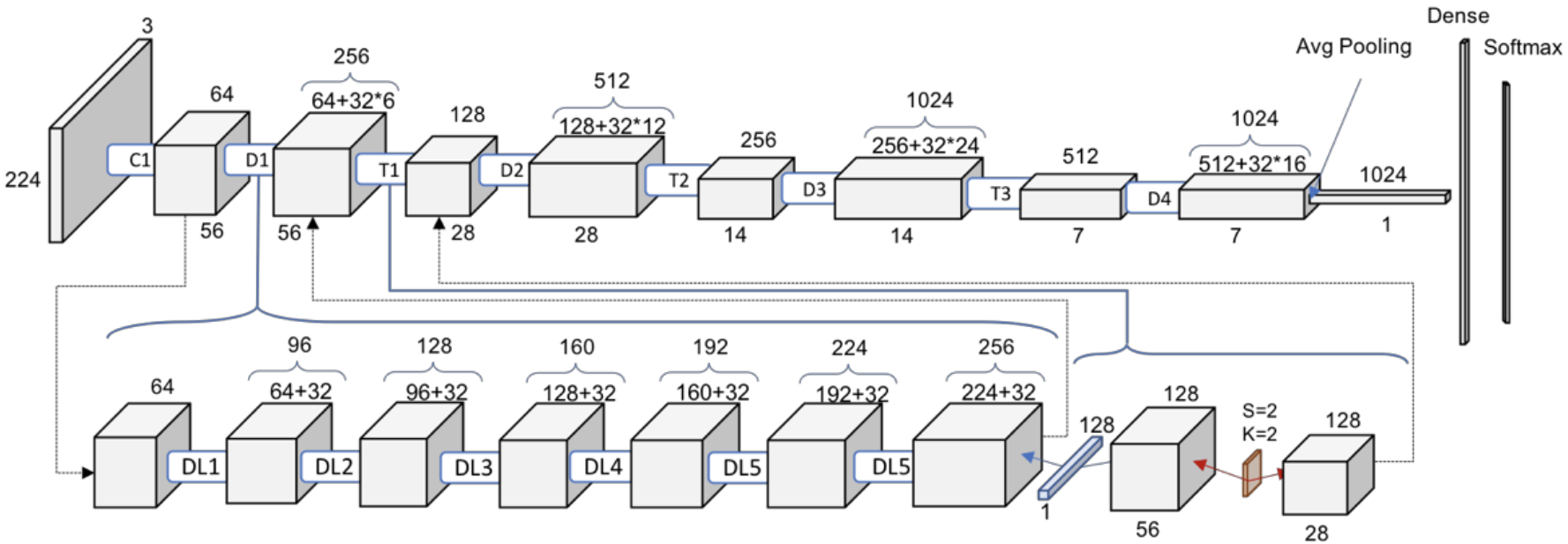
Motivation (1/4)

- Machine learning is increasingly important
  - Has become central to the modern economy
  - Is improving prediction across a range of economic problems
- But machine learning algorithms are also increasingly opaque
  - Training protocols have become more complex, the number of parameters has exploded, protocols more and more nested, etc.
  - Even if an analyst knows all of the code behind an algorithm, it is nearly impossible to fully grasp its inner-workings ("black box")

Motivation (1/4)

- Machine learning is increasingly important
  - Has become central to the modern economy
  - Is improving prediction across a range of economic problems
- But machine learning algorithms are also increasingly opaque
  - Training protocols have become more complex, the number of parameters has exploded, protocols more and more nested, etc.
  - Even if an analyst knows all of the code behind an algorithm, it is nearly impossible to fully grasp its inner-workings ("black box")
- Because of this, most modern machine learning algorithms are analytically intractable and accurately predicting them is impossible
  - e.g., CheXNeXt (Rajpurkar et al. 2018) a 121-layer convolutional neural net based on a DenseNet architecture

e.g., DenseNet Architecture



Motivation (2/4)

- For analytical tractability, we need a parsimonious "as if" model of machine learning

Motivation (2/4)

- For analytical tractability, we need a parsimonious "as if" model of machine learning
- For accurate predictions, we need this model to match behavior of machine learning algorithms

Motivation (2/4)

- For analytical tractability, we need a parsimonious "as if" model of machine learning
- For accurate predictions, we need this model to match behavior of machine learning algorithms
- To help bridge this gap, we offer two possible models of machine learning:
  1. Feasibility-based machine learning: the algorithm chooses what it learns in a way that suits its loss function, but is restricted to a fixed set of ways to learn
  2. Cost-based machine learning: the algorithm adjusts its learning in response to unobservable costs of learning

Motivation (3/4)

- We show how to test if machine learning predictions are consistent with these models
- We also show how to recover the parameters of these models
- We apply our methods to a 121-layer convolutional neural net that predicts pneumonia from chest x-rays (Rajpurkar et al. 2018)
- We find this algorithm is consistent with cost-based machine learning, but not feasibility-based machine learning
- We also recover the costs of learning for this algorithm

Motivation (4/4)

- Why do we want to model machine learning?
- Modeling machine learning as we do for human decisions opens up the tools of economics, psychology, and neuroscience to this setting
- Much potential value in using these tools to help understand what algorithms will do in "unknown territory"
  - Applications: robustness, counterfactuals, optimization, etc.
- Also value in using these tools to help interpret machine learning algorithms
- Many applications underway, including a finished companion paper
- But the goal today is to provide the necessary first step

Talk Structure

1. Provide modeling foundation
2. Define the machine learning models
3. Show how to test these models
4. Show how to recover costs
5. Present empirical implementation

1. Provide modeling foundation

Notation

- Employ very general setup to capture vast majority of machine learning algorithms
- Finite set of *instances* X (digital photographs, text excerpts, etc.)
- Finite set of *outcomes* Y (image types, numeric rating, etc.)
- Deterministic map between instances and outcomes given by $f : X \to Y$, called the *ground truth*
- Set of possible *predictions* A (confidence scores, outcomes, etc.)
  - Confidence scores become outcome predictions via threshold rule
  - Analyst can model either scores or downstream outcome predictions
- Application: X is over 100,000 chest x-ray images, Y is pneumonia ($y = 1$) or no pneumonia ($y = 0$), and A is a confidence score $[0, 1]$
  - We use scores because we want to model raw output of algorithm

Loss Functions

- An important input to an algorithm is a *loss function* $L : A \times Y \to \mathbb{R}$
- Application: Loss function is weighted cross-entropy with $\beta \approx .99$

$$L(a, y) = \begin{cases} -\beta log(a) \text{ if } y = 1 \\ -(1-\beta) log(1-a) \text{ if } y = 0 \end{cases}$$

- Using L in training generates a *trained model* $g^L : X \to A$
- Typically assess performance at aggregate level (by outcome)
  - e.g., how often an algorithm correctly predicts pneumonia instead of whether it correctly predicts pneumonia for a particular x-ray
- Aggregate level summarized by *performance data* $P^L : A \times Y \to [0, 1]$

$$P^L(a, y) = \frac{|\{x \in X | g^L(x) = a \ \& \ f(x) = y\}|}{|X|}$$

Foundation

- Our models build on information theoretic foundation in which we conceive of algorithms as decision-makers that follow the Blackwell (1953) model of experimentation, signal processing, and choice
- We assume that an algorithm is an optimizing agent that:
  1. Starts off with prior beliefs about the outcome
  2. Gets signals that provide information about the outcome
  3. Forms posterior beliefs via Bayesian updating
  4. Makes predictions based on these posteriors to minimize expected losses
- This foundation is testable, normatively-grounded, easy to impose, and has empirical support (Caplin, Martin, and Marx 2022a)

Foundation: Model

Definition

For a given loss function L, P^L has a **signal-based representation** (SBR) if there exist prior beliefs μ, a Bayes consistent distribution of posteriors $Q \in \mathcal{Q}$, and a prediction function $q : supp(Q) \to \Delta(A)$ such that:

1. Prior beliefs are correct: $\mu(y) = \sum_{a \in supp(P^L_A)} P^L(a, y)$
2. Predictions are optimal at all possible posteriors:

$$q \in \arg\inf_{q} \sum_{\gamma \in supp(Q)} Q(\gamma) \sum_{a \in A} q(a|\gamma) \sum_{y \in Y} \gamma(y) L(a, y)$$

3. Predictions are generated by the model:

$$P^L(a, y) = \sum_{\gamma \in supp(Q)} Q(\gamma) q(a|\gamma) \gamma(y)$$

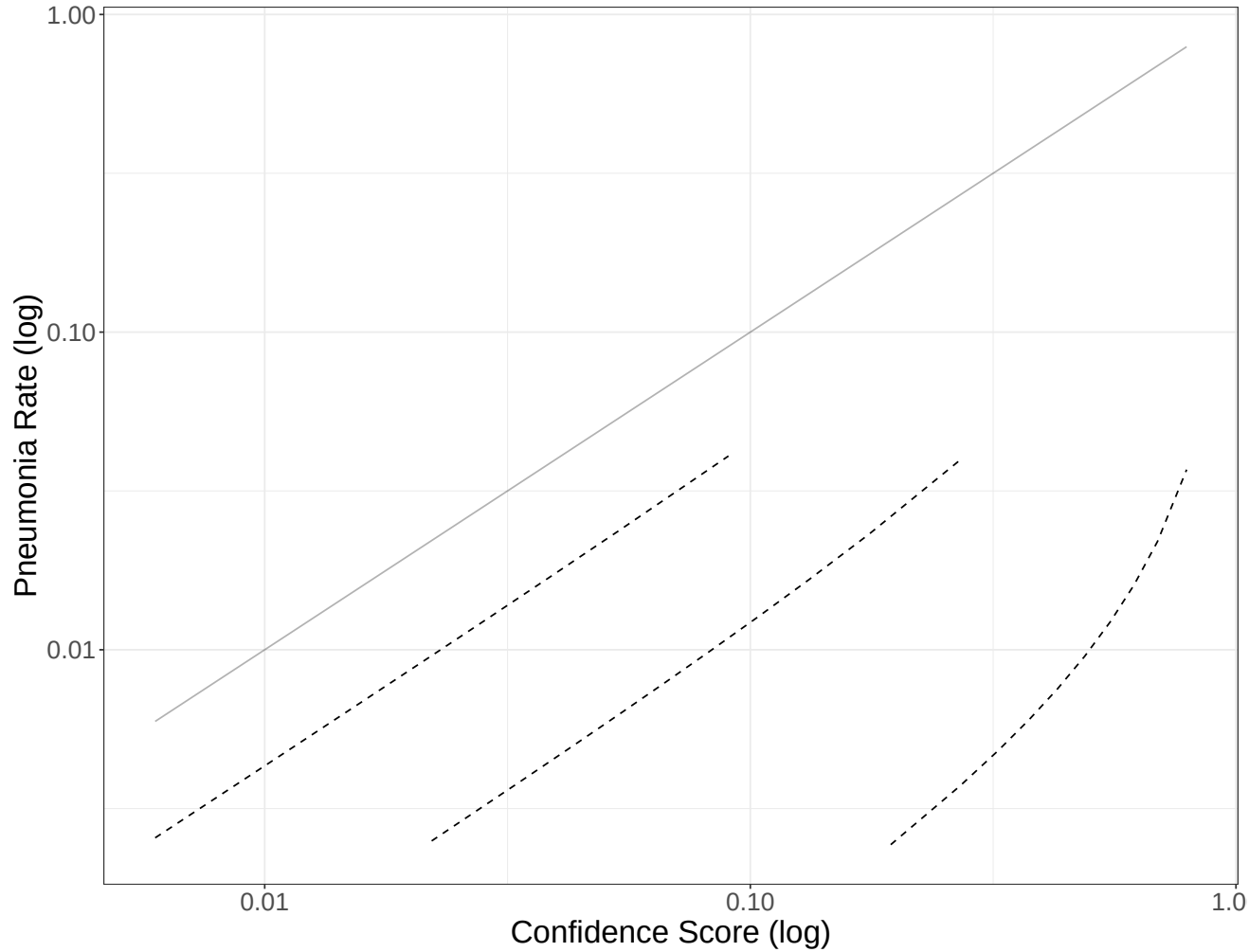
Foundation: Model

- Testable: Algorithm **loss-calibrated** to loss function L if a wholesale switch of predictions does not reduce losses according to L

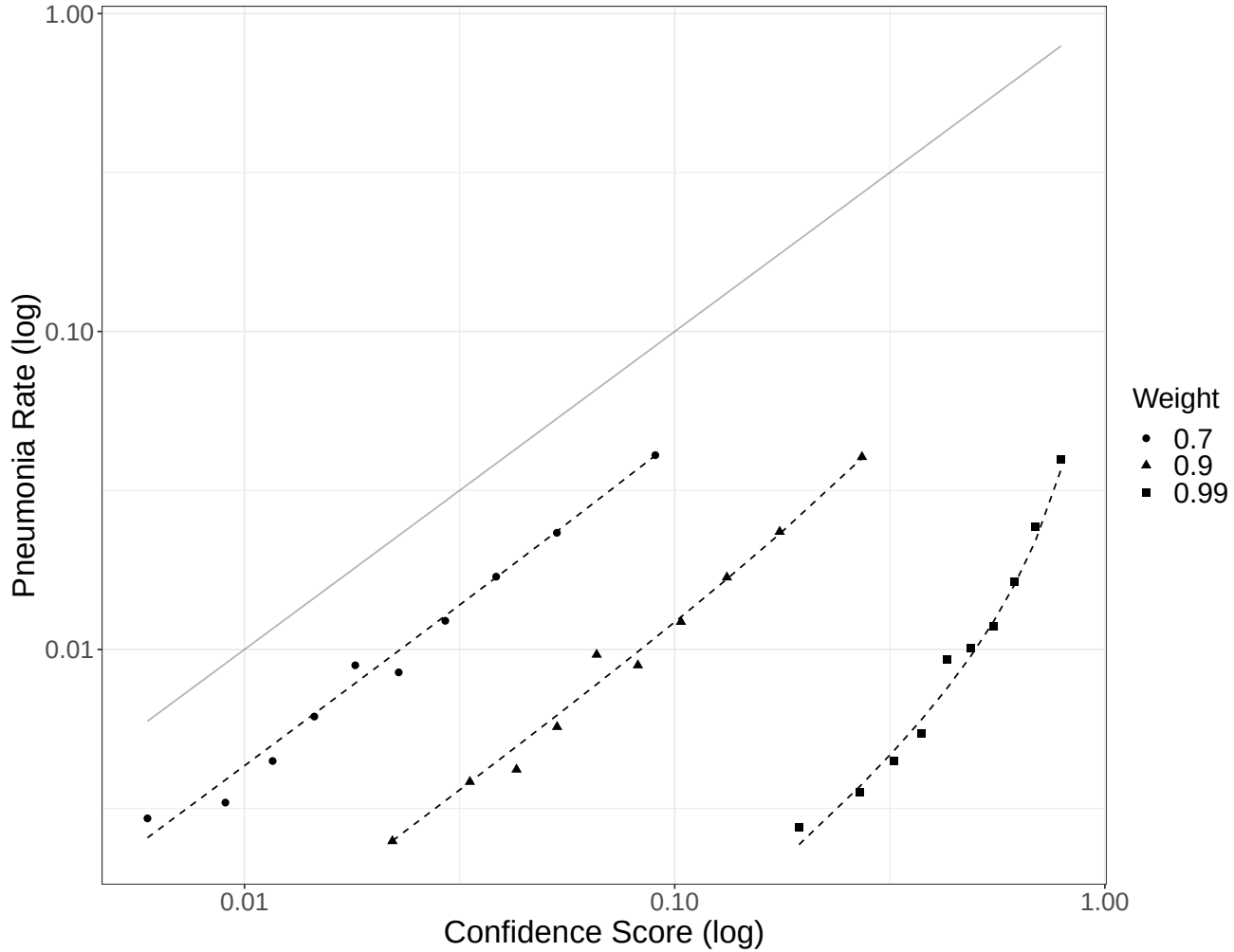
$$a \in \underset{a' \in A}{\operatorname{argmin}} \sum_{y \in Y} P^L(a, y) L(a', y) \text{ for all } a \in supp(P_A^L)$$

  - Combines loss function variation with NIAS (Caplin and Martin 2015) and Obedience (Bergemann and Morris 2016)
- Normatively-grounded: Rules out trivial prediction errors
- Easy to impose: Algorithm *loss re-calibrated* when make loss-calibrated through wholesale switches
- Empirical support? Caplin, Martin, and Marx (2022a) varied the class weights in a pneumonia detection deep learning neural net (Rajpurkar et al. 2018) for ChestX-ray14 data (Wang et al. 2017)

Here are the predicted relationships between scores and rates for a loss-calibrated algorithm for class weights $\beta = 0.7, 0.9, 0.99$



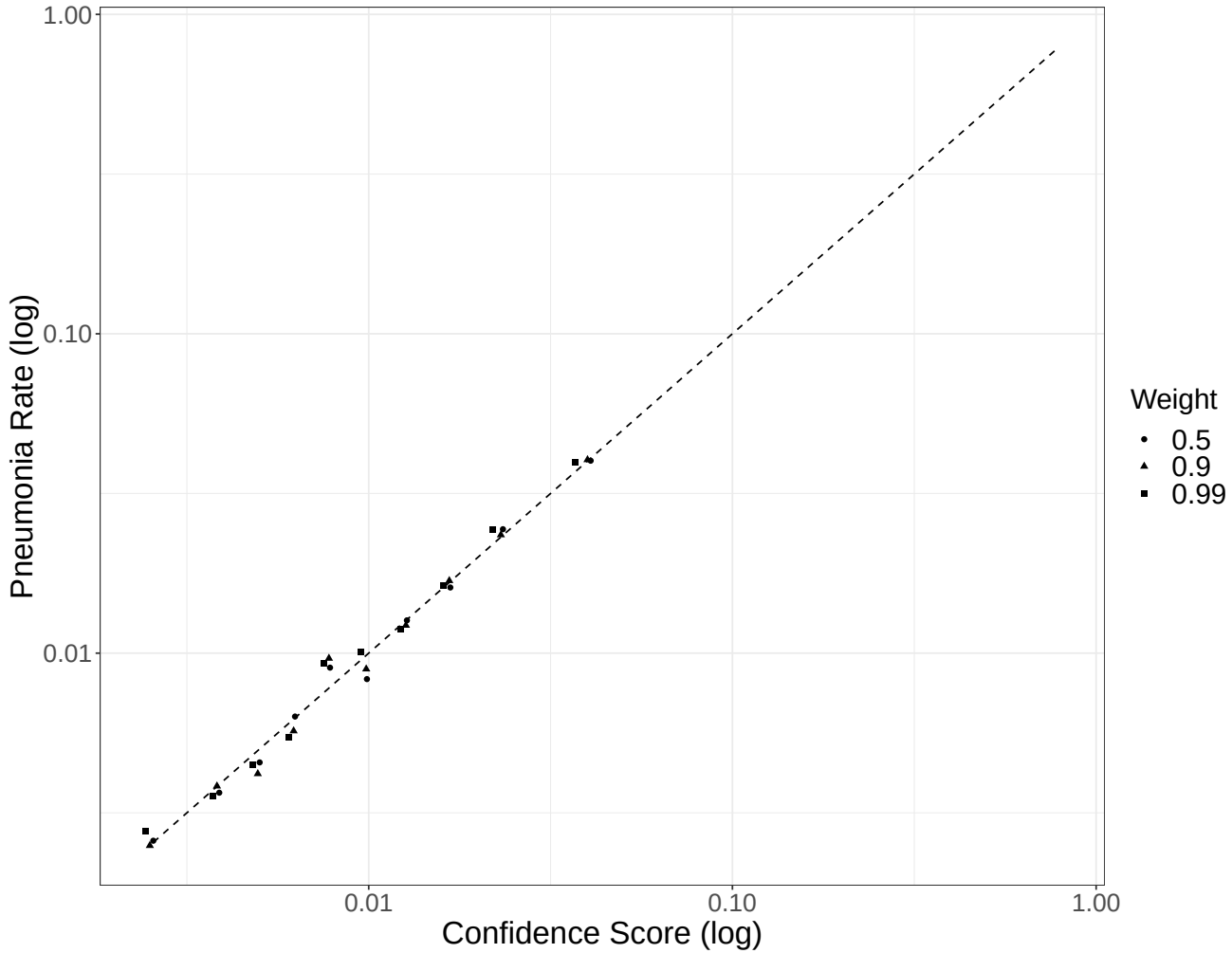
And here are the actual relationships: The algorithm becomes increasingly miscalibrated with higher weights, but is always loss-calibrated



Calibration

- Caplin, Martin, and Marx (2022a) use SBR to resolve an important machine learning challenge
- Desirable for downstream purposes and interpretability that predictions **calibrated** to ground truth probabilities (Guo et al., 2017, Jiang et al., 2012, Kompa et al., 2021, Cosmides and Tooby, 1996)
- Class imbalance (e.g. pneumonia) or external objective leads to **class weighting** and asymmetric loses (Thai-Nghe, Gantner, and Schmidt-Thieme 2010, Zadrozny, Langford, and Abe 2003)
- SBR explains why class weighting generates miscalibration, and offers a solution: recover "as if" revealed posterior beliefs of the algorithm, and interpret them as *loss-corrected* confidence scores

The loss-corrected confidence scores are calibrated for all weights
Takeaway: Can treat these transformed scores as a likelihoods



2. Define the machine learning models

Models

- SBR provides a solid foundation, but leaves open the question of where the algorithm's signal structure comes from
- Our models represent central (and nested) possibilities:
  1. Feasibility-based machine learning: the algorithm chooses from a fixed set of signal structures
  2. Cost-based machine learning: the algorithm chooses from signal structures of varying costs
- Note that these will be familiar from earlier presentation of *Revealed Bayesian Learning*
- Advantages relative to human learning applications: data set is standard in ML; loss function is known in ML; loss function manipulable in ML; less concern about behavioral biases in ML

Feasibility-Based Machine Learning

- Our first model class imagines an algorithm as choosing from a fixed feasible set of signal structures
- Because the algorithm can choose its Q, we expand the algorithm's strategy space Λ to include both Q and q as in Revealed Bayesian Learning:

$$\Lambda = \{(Q, q) | Q \in \mathcal{Q}, q : supp(Q) \to \Delta(A)\}$$

- We define a fixed set of experiments $\mathcal{Q}^* \subset \mathcal{Q}$
  - No assumptions on this set beyond being non-empty
- Given loss function L and fixed set $\mathcal{Q}^*$, the set of optimal strategies is

$$\arg\inf_{(Q,q)\in\Lambda, Q\in\mathcal{Q}^*} \sum_{\gamma\in supp(Q)} Q(\gamma) \sum_{a\in A} q(a|\gamma) \sum_{y\in Y} \gamma(y) L(a, y)$$

Feasibility-Based Machine Learning

- For notational understanding, unpack optimization

$$\arg\inf_{(Q,q)\in\Lambda, Q\in\mathcal{Q}^*} \sum_{\gamma\in supp(Q)} Q(\gamma) \sum_{a\in A} q(a|\gamma) \sum_{y\in Y} \gamma(y) L(a,y)$$

- Take some possible posterior and prediction possible at that posterior, (γ, a) with $Q(\gamma) > 0, q(a|\gamma) > 0$
- Term $\sum_{y\in Y} \gamma(y) L(a,y)$ is corresponding expected loss
- Sum across possible predictions at that posterior
- Sum across possible posteriors
- Arrive at expected loss
- Minimize across strategies (restricted to fixed set)

Cost-Based Machine Learning

- Our second model class assumes the algorithm chooses from signal structures of varying costs
- Define a learning cost function $K : \mathcal{Q} \to \bar{\mathbb{R}}$
- The *resource-adjusted loss* $\hat{L}$ of strategy (Q, q) is

$$\hat{L}((Q,q)|L,K) \equiv \sum_{\gamma \in supp(Q)} \sum_{a \in A} Q(\gamma) q(a|\gamma) \sum_{y \in Y} \gamma(y) L(a,y) + K(Q)$$

- The corresponding set of optimal strategies is then

$$\arg\inf_{(Q,q) \in \Lambda} \hat{L}((Q,q)|L,K)$$

Interpretations (1/2)

- Feasibility-based learning imagines an algorithm as having a fixed set of mathematical operations it can use, and that it selects among these to best match the incentives provided by the loss function
- Cost-based machine learning imagines that there is relative difficulty in performing different mathematical operations, and that it balances the trade-offs with the incentives provided by the loss function

Interpretations (2/2)

- Why might an algorithm use more or fewer resources depending on the loss function?
  - The amount of resources spent could be driven by the algorithm's choice of a particular hyperparameter, which can be influenced by the loss function through "hyperparameter optimization"
  - Also, an algorithm might perform more or fewer epochs if the losses under a particular loss function have not converged sufficiently
- Of course, while resource use may vary with the loss function, it is not immediately clear that an algorithm can be modeled as optimally balancing losses and resource costs
- To investigate this possibility, we ask whether there are "as if" costs of learning that explain the algorithm's performance

3. Show how to test these models

Loss Function Variation

- Central to testing and recovery: what the machine learns is optimal given the loss function
- Thus, whether or not algorithm is consistent with these models has testable content as we vary the loss function across a set $\mathcal{L}$
- Because indexing will be useful, in what follows we will take as given a finite set of M loss functions, indexed by $1 \leq m \leq M$
- For notational simplicity, we denote the performance data set from training the algorithm with the m-th loss function as $P^m = P^{L^m}$

Cross-Loss Matrix $\hat{T}$

- Central to testing our models is to see if losses could be changed by counterfactually switching to the predictions from training with a different loss function
- All such comparisons are visible in the *cross-loss* matrix $\hat{T}$, which summarizes a large number of counter-factual switches and their impact on losses
- The $M \times M$ *cross-loss matrix* $\hat{T}$ has generic element $\hat{T}^{mn}$ in row m and column n that specifies the minimized expected losses when the loss function is L^m and the performance data is P^n:

$$\hat{T}^{mn} \equiv \sum_{a \in supp(P^n_A)} \min_{a' \in A} \sum_{y \in Y} L^m(a', y) P^n(a, y)$$

Consider the following loss functions for binary predictions/outcomes:

$$L^1 = \begin{array}{c} \\ a=0 \\ a=1 \end{array} \begin{array}{c} \begin{array}{cc} y=0 & y=1 \end{array} \\ \begin{pmatrix} 0 & .5 \\ .5 & 0 \end{pmatrix} \end{array} \quad \& \quad L^2 = \begin{array}{c} \\ a=0 \\ a=1 \end{array} \begin{array}{c} \begin{array}{cc} y=0 & y=1 \end{array} \\ \begin{pmatrix} 0 & .6 \\ .4 & 0 \end{pmatrix} \end{array}$$

Imagine an algorithm that produces the following performance data:

$$P^1 = \begin{array}{c} \\ a=0 \\ a=1 \end{array} \begin{array}{c} \begin{array}{cc} y=0 & y=1 \end{array} \\ \begin{pmatrix} .25 & .1 \\ .25 & .4 \end{pmatrix} \end{array} \quad \& \quad P^2 = \begin{array}{c} \\ a=0 \\ a=1 \end{array} \begin{array}{c} \begin{array}{cc} y=0 & y=1 \end{array} \\ \begin{pmatrix} .3 & .2 \\ .2 & .3 \end{pmatrix} \end{array}$$

The cross-loss matrix can be computed as:

$$\hat{T} = \begin{array}{c} \begin{array}{cc} P^1 & P^2 \end{array} \\ \begin{pmatrix} .175 & .2 \\ .16 & .2 \end{pmatrix} \end{array} \begin{array}{c} L^1 \\ L^2 \end{array}$$

Lower losses under L^2 if use P^1 (performance data from training with L^1)

Loss-Difference Matrix D

- The key to testing both models is to see if any combination of switches in performance data could lower losses
- This is readily visible in the *loss-difference matrix* D, which has generic element D^{mn} defining the minimum change in the sum of losses by switching performance data along a path from m to n

$$D^{mn} \equiv \min_{\{\vec{h} \in H(m,n)\}} \sum_{i=1}^{J(\vec{h})-1} \left[\hat{T}^{h(i)h(i+1)} - \hat{T}^{h(i)h(i)} \right]$$

where:

1. $\vec{h}$ is an arbitrary length J sequence of indices $\vec{h} = (h(1), ..., h(J(\vec{h}))$ with $1 \leq h(j) \leq M$ and the first $J(\vec{h}) - 1$ entries distinct
2. H is the set of all such vectors
3. The set of paths from P^m to P^n is given by $H(m, n) = \{\vec{h} \in H | h(1) = m, h(J(\vec{h})) = n)\}$

To illustrate, consider the following $\hat{T}$ matrix:

$$\hat{T} = \begin{array}{c} \\ L^1 \\ L^2 \end{array} \begin{array}{c} \begin{array}{cc} P^1 & P^2 \end{array} \\ \begin{pmatrix} .175 & .2 \\ .16 & .2 \end{pmatrix} \end{array}$$

The corresponding loss-difference matrix is

$$D = \begin{array}{c} \\ L^1 \\ L^2 \end{array} \begin{array}{c} \begin{array}{cc} P^1 & P^2 \end{array} \\ \begin{pmatrix} -.015 & .025 \\ -.04 & -.015 \end{pmatrix} \end{array}$$

$D^{11} = -.015$ because starting from L^1 the indirect path back to L^1 involves first raising losses by .025 switching P^1 to P^2 and then lowering losses by .04 switching P^2 to P^1

Testing Feasibility-Based Machine Learning

Definition

An algorithm is **strongly loss-adapted** if for all $1 \leq m, n \leq M$,

$$D^{mn} \geq 0$$

- This condition is equivalent to the NIAAS property of Caplin, Martin, and Marx (2022b): $\hat{T}^{mm} \leq \hat{T}^{mn}$
- NIAAS has a natural interpretation in our setting: we would expect that when the loss function is L^m, training with L^m generates lower losses than can be achieved by training with any other loss function
- Caplin, Martin, and Marx (2022b) can be applied to show that being strongly loss-adapted (in addition to being loss-calibrated) is both necessary and sufficient for feasibility-based machine learning

Testing Cost-Based Machine Learning

Definition

An algorithm is **loss-adapted** if for all $1 \leq m \leq M$,

$$D^{mm} \geq 0$$

- This condition combines variation in the algorithm's loss function with the NIAC property of Caplin and Dean (2015)
- The main result in Caplin and Dean (2015) can be applied to show that being loss-adapted (in addition to being loss-calibrated) is equivalent to being consistent with cost-based machine learning
- The only adjustments necessary to apply their result in our setting are to change utility maximization to loss minimization and to map variation in actions to variation in loss functions

4. Show how to recover costs

Recovery

- We also show how to use D to recover an algorithm's learning costs
- Identifying a learning cost function boils down to identifying a suitable ("qualifying") learning cost for each observed performance data set
- For notational simplicity, we denote the learning cost for the m-th performance data set as $K^m = K(P^m)$

Theorem

For an algorithm with an SBR:

1. **Recovery:** $\{K^m\}_{m=1}^M$ *are qualifying learning costs if and only if, given* $1 \leq m, n \leq M$, $K^m - K^n \leq D^{mn}$.

Theorem

For an algorithm with an SBR:

1. **Recovery:** $\{K^m\}_{m=1}^M$ *are qualifying learning costs if and only if, given* $1 \leq m, n \leq M$, $K^m - K^n \leq D^{mn}$.
2. **Convex Geometry:** *Normalizing to* $K^M = 0$, *qualifying learning costs* $\{K^m\}_{m=1}^{M-1}$ *define a convex polyhedron in* $\mathbb{R}^{M-1}$ *with* $(-D^{M1}, \cdots, -D^{M(M-1)})$, *the sign-inverted M-th row, and the M-th column* $(D^{1M}, \cdots, D^{(M-1)M})$ *as extreme points.*

Theorem

For an algorithm with an SBR:

1. **Recovery:** $\{K^m\}_{m=1}^M$ *are qualifying learning costs if and only if, given* $1 \leq m, n \leq M$, $K^m - K^n \leq D^{mn}$.
2. **Convex Geometry:** *Normalizing to* $K^M = 0$, *qualifying learning costs* $\{K^m\}_{m=1}^{M-1}$ *define a convex polyhedron in* $\mathbb{R}^{M-1}$ *with* $(-D^{M1}, \cdots, -D^{M(M-1)})$, *the sign-inverted M-th row, and the M-th column* $(D^{1M}, \cdots, D^{(M-1)M})$ *as extreme points.*
3. **Computation:** *If an algorithm is loss-adapted, the matrix D can be computed by applying the Floyd-Warshall algorithm to the complete weighted directed graph with*

$$W^{mn} = \hat{T}^{mn} - \hat{T}^{mm}$$

on the directed edge from node $1 \leq m \leq M$ *to node* $1 \leq n \leq M$.

Illustration

- Consider a Loss Calibrated algorithm for which we specify the corresponding $\hat{T}$ matrix directly as:

$$\hat{T} = \begin{array}{c} \begin{array}{ccc} P^1 & P^2 & P^3 \end{array} \\ \begin{pmatrix} 3 & 1 & 10 \\ 5 & 2 & 5 \\ 10 & 1 & 3 \end{pmatrix} \end{array} \begin{array}{c} L^1 \\ L^2 \\ L^3 \end{array}$$

- The only elements of D that reflect lowering losses through indirect paths are D^{13} and D^{31}:

$$\begin{aligned} D^{13} &= \hat{T}^{12} - \hat{T}^{11} + \hat{T}^{23} - \hat{T}^{22} = 1; \\ D^{31} &= \hat{T}^{32} - \hat{T}^{33} + \hat{T}^{21} - \hat{T}^{22} = 1 \end{aligned}$$

Illustration

- Writing out the D matrix we therefore get:

$$D = \begin{array}{c} \begin{array}{ccc} P^1 & P^2 & P^3 \end{array} \\ \begin{pmatrix} 0 & -2 & 1 \\ 3 & 0 & 3 \\ 1 & -2 & 0 \end{pmatrix} \end{array} \begin{array}{c} L^1 \\ L^2 \\ L^3 \end{array}$$

- The negative entries reveal this algorithm to be not strongly adapted, but the zeros on the diagonal reveal this algorithm to be loss-adapted
- Thus, it is not consistent with feasibility-based machine learning, but it is consistent with cost-based machine learning

Illustration

- The theorem also implies directly that (-1,2) and (1,3) are extreme points of the cost polyhedron normalized to $K^3 = 0$
- The entire normalized polyhedron involves also the upper and lower bounds on $K^1 - K^2$,

$$\begin{aligned} K^1 - K^2 &\leq D^{12} = -2; \\ K^2 - K^1 &\leq D^{21} = 3 \end{aligned}$$

- This is marked in Figure 1

Recovery

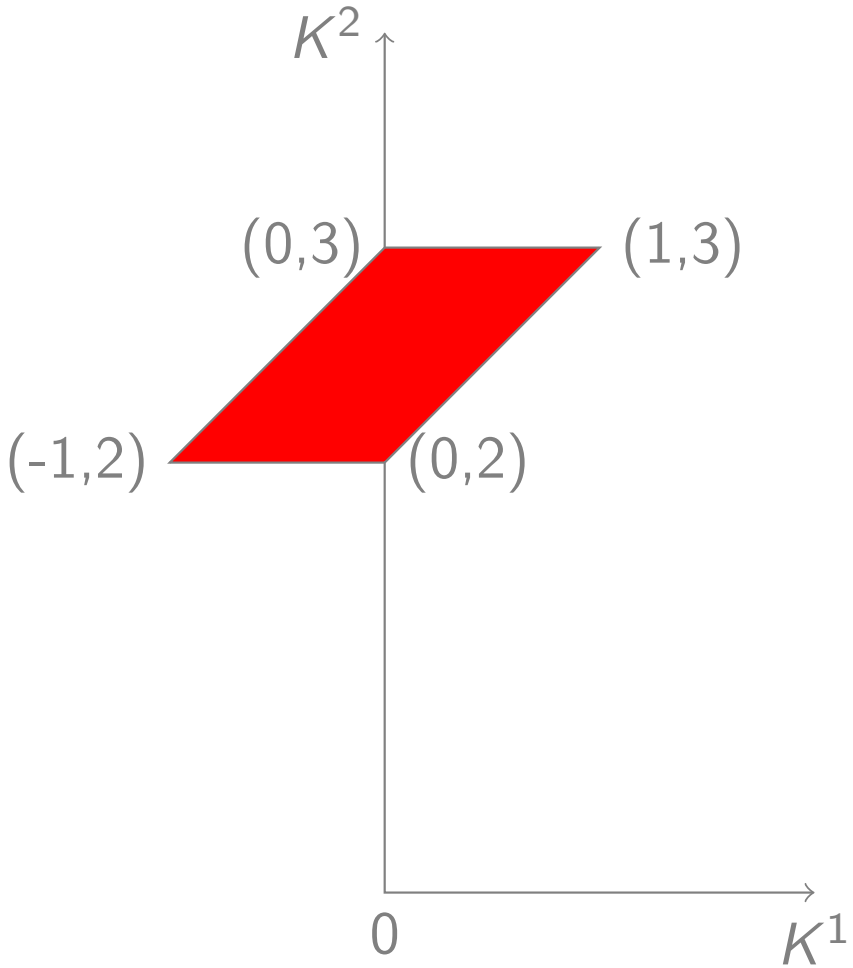


Figure 1. Example 2-dimensional learning cost polyhedron.

Recovery

- One can directly confirm that the other normalizations, while producing a different geometry, reconstitute precisely the same set of qualifying cost functions
- Geometry suggests a simple "representative" cost (average of the centers of the M normalized cost polyhedra) directly computable from D matrix
- Finishes theoretical material for this paper

5. Present empirical implementation

Algorithm

- We vary the loss function for a deep learning neural net used to predict pneumonia in the ChestX-ray14 dataset (Wang et al. 2017)
- Essentially a replication from the publicly available codebase of Rajpurkar et al. (2018)
- The ChextX-ray14 dataset consists of 112,120 frontal chest X-rays which were labeled with thoracic diseases
- In the binary classification task, the labels of interest are pneumonia ($y = 1$) or not ($y = 0$)
- We vary the loss function with class weights $\beta = 0.7, 0.9, 0.99$ ($\beta \approx 0.99$ in Rajpurkar et al. 2018)

Results (1/3)

- We start by calculating the D matrix

$$D = 0.1^5 \begin{array}{c} \begin{matrix} P^{0.7} & P^{0.9} & P^{0.99} \end{matrix} \\ \begin{pmatrix} 0 & 2.548 & 12.467 \\ -2.529 & 0 & 9.938 \\ -6.993 & -4.463 & 0 \end{pmatrix} \end{array} \begin{matrix} L^{0.7} \\ L^{0.9} \\ L^{0.99} \end{matrix}$$

- Negative entries, so not strongly loss-adapted
- Performed statistical test by bootstrapping covariance matrix from 10,000 samples of ensemble predictions
- Null hypothesis: $H_0 : D^{mn} \geq 0$ for all m, n
  - Bonferroni correction on set of one-sided Wald tests, $p = 0.0014$
  - Conservative test for multivariate one-sided, yet still reject

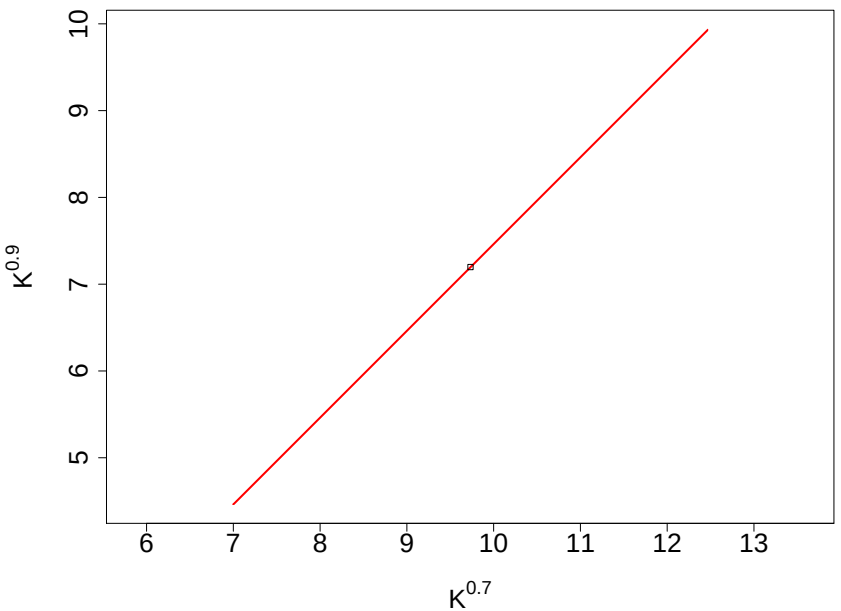
Results (2/3)

$$D = 0.1^5 \begin{array}{c} \begin{matrix} P^{0.7} & P^{0.9} & P^{0.99} \end{matrix} \\ \begin{pmatrix} 0 & 2.548 & 12.467 \\ -2.529 & 0 & 9.938 \\ -6.993 & -4.463 & 0 \end{pmatrix} \end{array} \begin{matrix} L^{0.7} \\ L^{0.9} \\ L^{0.99} \end{matrix}$$

- But $D^{mm} \geq 0$ for all m, so algorithm is loss-adapted
- Thus, while data is not consistent with feasibility-based machine learning, it is consistent with cost-based machine learning
- How strong is loss-adapted condition?
  - Generated 100m random monotonic choice probabilities in this setting, and loss-adapted condition failed 83.3% of the time

Results (3/3)

- Since data is loss-adapted, can recover set of possible learning costs



- What is a possible explanation? By telling the algorithm to care more about non-pneumonia cases, it chooses to learn more about these cases, and the large number of such cases leads to costlier learning

Motivation Recap

- Why do we want to model machine learning?
- Modeling machine learning as we do for human decisions opens up the tools of economics, psychology, and neuroscience to this setting
- Much potential value in using these tools to help understand what algorithms will do in "unknown territory"
  - Applications: robustness, counterfactuals, optimization, etc.
- Also value in using these tools to help interpret machine learning algorithms
- Many applications underway, including a finished companion paper
- But the goal today was to provide the necessary first step